



i i i i i i i i

You make **possible**

A decorative graphic of vertical bars of varying heights, resembling a bar chart or a音波 spectrum, is positioned on both the left and right sides of the text. The text "i i i i i i i i" is displayed above the slogan, with each letter "i" being a single vertical bar of a different color: blue, green, orange, red, orange, blue, green, blue. Below this, the slogan "You make **possible**" is written in white, with "possible" in a larger, bold blue font.



# Cisco Network Services Orchestrator (NSO)



教主技术进化论2020第28期

PPT : 冯正宗, 现任明教教主  
主讲人 : 现任明教教主

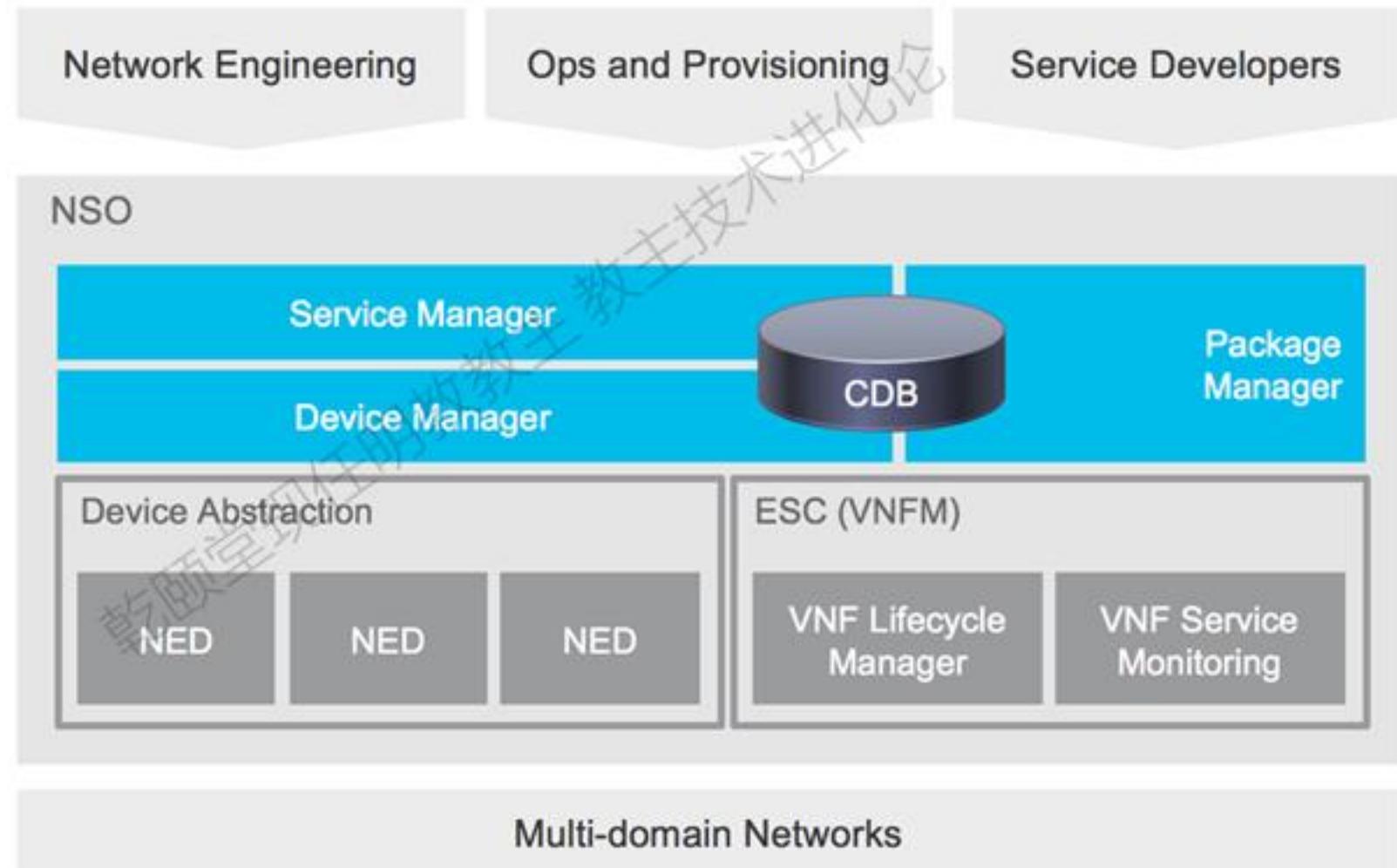


# NSO Overview



# What is Cisco NSO?

Cisco Network Services Orchestrator is a **Linux application which orchestrates the configuration** life-cycle of physical and virtual network devices.





# NSO Fundamentals

<https://www.tail-f.com/>

- NSO gathers, parses and stores the configuration state of the network devices it manages in a configuration database (CDB). **Users and other applications can then ask NSO to create, read, update or delete configuration in a programmatic way either ad hoc or through customizable network services.**
- NSO uses software packages called **Network Element Drivers (NEDs)** to facilitate telnet, SSH, or API interactions with the devices that it manages. The NED provides an abstraction layer that reads in the device's running configuration and parses it into a data-model-validated snapshot in the CDB.
- The NEDs also allow for the reverse, creating network configuration from CDB data inputs and then sending the configurations to the network devices. **There are hundreds of NEDs covering all the Cisco platforms (including IOS-XE, IOS-XR, NX-OS, and ASA) and all major non-Cisco platforms as well.**



# NETCONF/YANG

NSO uses **YANG** as the overall modeling language to manage devices and services. YANG models describe all NSO configuration, including device configuration and service configuration. This means that everything that is done in NSO is model driven, this allows all interfaces to be automatically rendered.

YANG was originally paired with **NETCONF**, but as REST became a more popular interface

**RESTCONF** was standardized as well. Both of the protocols let you define your API using YANG, giving you a model driven interface; this means that the basic working of the protocol is defined by the standards and the application specific details can be derived from the loaded YANG models. The RESTCONF protocol provides a compatible subset of the functionality of the NETCONF protocol, but does not provide the full transactional interface of the latter.

# The Configuration Database (CDB)

- At the core of NSO is the Configuration Database (CDB). This is a tree-structured database that is controlled by a YANG schema. This means that all of the information stored inside of NSO is validated against the schema.
- Every transaction towards CDB exhibits ACID properties, which among other things means either the transaction as a whole ends up on all participating devices and in the CDB master copy, or alternatively the whole transaction is aborted and all changes are automatically rolled-back.
- The CDB always contains NSOs view of the complete network configuration. To handle out-of-band changes operations are available to check if a device is in sync, write NSOs view to the device or read the device configuration into NSO.

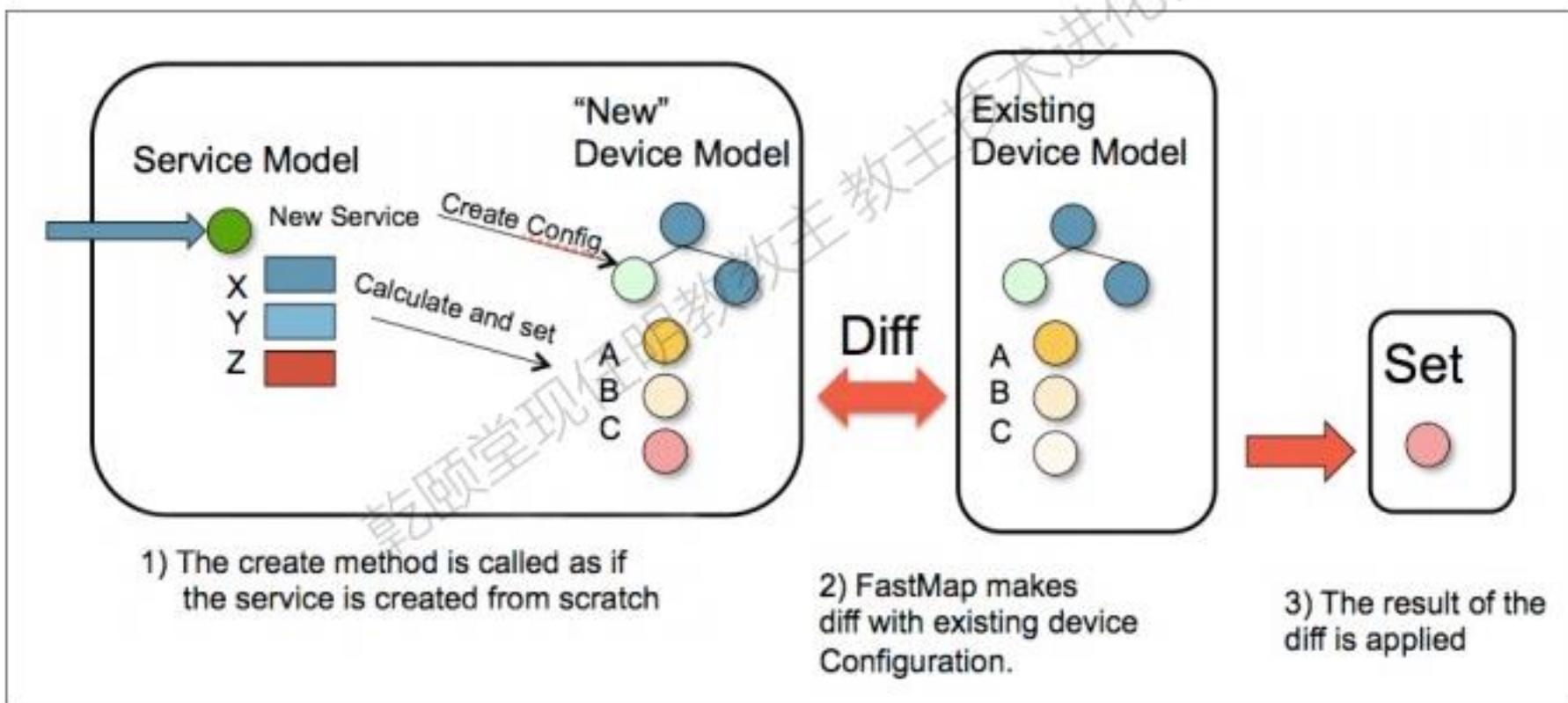


# The Service Algorithm - FastMap

FASTMAP covers the complete service life-cycle: creating, changing and deleting the service. The solution requires a minimum amount of code for mapping from a service model to a device model.

FASTMAP is based on generating changes from an initial create. When the service instance is created the reverse of the resulting device configuration is stored together with the service instance. If an NSO user later changes the service instance, NSO first applies (in a transaction) the reverse diff of the service, effectively undoing the previous results of the service creation code. Then it runs the logic to create the service again, and finally executes a diff to current configuration. This diff is then sent to the devices.

# The Service Algorithm - FastMap





# Accessing the Network (NEDs)

The NSO device manager is the centre of NSO. The device manager maintains a flat list of all managed devices. NSO keeps the master copy of the configuration for each managed device in CDB. Whenever a configuration change is done to the list of device configuration master copies, the device manager will partition this "network configuration change" into the corresponding changes for the actual managed devices.

The device manager passes on the required changes to the NEDs, Network Element Drivers. A NED needs to be installed for every type of device OS, like Cisco IOS NED, Cisco XR NED, Juniper JUNOS NED etc. The NEDs communicate through the native device protocol southbound



# Installation Operating System

Cisco NSO can run on [macOS](#) or on [Linux systems](#). If you are a Window's user (or if you don't wish to install it natively on your laptop), you can install NSO on a Linux virtual machine or in a container.

For small lab networks (less than 10 devices) you'll only need a single vCPU and 4-8 GB of RAM. NSO can be a memory intensive application, so if you are encountering issues, try increasing the RAM size. If you use Vagrant or docker to manage local Linux virtual machines, there is a NSO Vagrant file and NSO Docker file for your convenience.



# Installation Java

NSO requires Java to be installed to function. Here are some common ways to install Java:

Centos 7:

```
[root@localhost ~]# yum install java-1.8.0-openjdk //安装java软件包
```



# Installation Apache Ant

Apache Ant is a Java library and tool for building files and other dependencies that you will also need.

Centos 7:

```
[root@localhost ~]# yum install ant
```



# Local vs System Installation

Before you install NSO onto your system, you need to decide whether to do a "System" or "Local" installation. Here's a simple breakdown of the two.

- System Install is used when installing NSO for a centralized, "always-on", production grade purpose. It is configured as a system daemon that would start and end with the underlying operating system. The default users of admin and oper are not included and the file structure is more distributed.
- Local Install is used for development, lab, and evaluation purposes. It unpacks all the application components, including docs and examples, and can be used by the engineer to run multiple, unrelated, instances of NSO for different labs and demos on a single workstation



# Downloaded the installer.

1. [NSO 5.3 Mac](#)
2. [NSO 5.3 Linux](#)
3. [Cisco IOS \(XE\) NED](#)
4. [Cisco IOS XR NED](#)
5. [Cisco NXOS NED](#)
6. [Cisco ASA NED](#)

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



# 校验签名并解压缩

Use the `sh` command to "run" the `signed.bin` to verify the certificate and extract the installer binary and other files.

```
[root@docker-master -]# sh nso-5.3.linux.x86_64.signed.bin
Unpacking...
Verifying signature...
Downloading CA certificate from http://www.cisco.com/security/pki/certs/crcam2.cer ...
Successfully downloaded and verified crcam2.cer.
Downloading SubCA certificate from
http://www.cisco.com/security/pki/certs/innerspace.cer ...
Successfully downloaded 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-5.3.linux.x86_64.installer.bin using tailf.cer
```



# Performing a Local Installation

Run the installer with the argument `--local-install /nso` to install it into your home directory

# 本次试验在download目录提供解压后文件

```
[root@localhost ~]# cd /download/  
[root@localhost download]# sh nso-5.3.linux.x86_64.installer.bin --local-install /nso  
INFO Using temporary directory /tmp/ncs_installer.1680 to stage NCS installation bundle  
INFO Unpacked ncs-5.3 in /nso  
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 /nso/ncsrc  
INFO NSO installation script finished  
INFO Found and unpacked corresponding NETSIM_PACKAGE  
INFO NCS installation complete
```



# NEDs or Network Element Drivers

In order to "talk to" the network, NSO uses NEDs as device drivers for different device types. Cisco has NEDs for hundreds of different devices available for customers, and several are included in the installer in the `/nso/packages/neds` directory.

```
[root@localhost -]# cd /nso/packages/neds
[root@localhost neds]# ll
总用量 0
drwxr-xr-x 8 root wheel 195 11月 29 2019 a10-acos-cli-3.0
drwxr-xr-x 7 root wheel 184 11月 29 2019 alu-sr-cli-3.4
drwxr-xr-x 8 root wheel 207 11月 29 2019 cisco-asa-cli-6.6
drwxr-xr-x 7 root wheel 184 11月 29 2019 cisco-ios-cli-3.0
drwxr-xr-x 7 root wheel 184 11月 29 2019 cisco-ios-cli-3.8
drwxr-xr-x 8 root wheel 195 11月 29 2019 cisco-iosxr-cli-3.0
drwxr-xr-x 8 root wheel 195 11月 29 2019 cisco-iosxr-cli-3.5
drwxr-xr-x 8 root wheel 195 11月 29 2019 cisco-nx-cli-3.0
drwxr-xr-x 8 root wheel 195 11月 29 2019 dell-ftos-cli-3.0
drwxr-xr-x 5 root wheel 147 11月 29 2019 juniper-junos-nc-3.0
```



# 解压ned文件到neds目录

```
cd /nso/packages/neds/  
tar -zxvf /download/ncs-5.3-cisco-asa-6.7.7.tar.gz  
tar -zxvf /download/ncs-5.3-cisco-ios-6.42.1.tar.gz  
tar -zxvf /download/ncs-5.3-cisco-iosxr-7.18.2.tar.gz  
tar -zxvf /download/ncs-5.3-cisco-nx-5.13.1.1.tar.gz
```



# Installing New NED Versions

```
[root@localhost neds]# ll //查看新安装的NEDS
总用量 0
drwxr-xr-x 8 root wheel 195 11月 29 2019 a10-acos-cli-3.0
drwxr-xr-x 7 root wheel 184 11月 29 2019 alu-sr-cli-3.4
drwxr-xr-x 8 root wheel 207 11月 29 2019 cisco-asa-cli-6.6
drwxr-xr-x 8 9001 users 207 12月 12 2019 cisco-asa-cli-6.7
drwxr-xr-x 7 root wheel 184 11月 29 2019 cisco-ios-cli-3.0
drwxr-xr-x 7 root wheel 184 11月 29 2019 cisco-ios-cli-3.8
drwxr-xr-x 8 9001 users 207 1月 5 2020 cisco-ios-cli-6.42
drwxr-xr-x 8 root wheel 195 11月 29 2019 cisco-iosxr-cli-3.0
drwxr-xr-x 8 root wheel 195 11月 29 2019 cisco-iosxr-cli-3.5
drwxr-xr-x 9 9001 users 218 12月 27 2019 cisco-iosxr-cli-7.18
drwxr-xr-x 8 root wheel 195 11月 29 2019 cisco-nx-cli-3.0
drwxr-xr-x 9 9001 users 218 12月 18 2019 cisco-nx-cli-5.13
drwxr-xr-x 8 root wheel 195 11月 29 2019 dell-ftos-cli-3.0
drwxr-xr-x 5 root wheel 147 11月 29 2019 juniper-junos-nc-3.0
```



# NCSRC

The last thing to note are the files `ncsrc` and `ncsrc.tsch`. These are shell scripts for bash and tsch that setup your PATH and other environment variables for NSO. **Depending on your shell, you will need source this file before starting your NSO work.**

```
[root@localhost -]# source/ncso/ncsrc
[root@localhost -]# ncs
ncs          ncs_cli          ncs_conf_tool      ncs-maapi        ncs-project      ncs-start-python-vm
ncs-backup   ncs_cmd          ncs_crypto_keys  ncs-make-package ncs-setup       ncs-uninstall
ncsc         ncs-collect-tech-report ncs_load        ncs-netsim      ncs-start-java-vm
```



# Creating an Instance of NSO

A NSO Local Install unpacks and prepares your system to run NSO, but **doesn't actually start it up**. A Local Install allows the engineer to create an NSO "instance" tied to a project, which you have not done yet.



# Using ncs-setup to Create an NSO Instance

One of the included scripts with an NSO installation is `ncs-setup`, which makes it very easy to create instances of NSO from a Local Install. You can look at the `--help` for full details, but the two options we need to know are:

- \* `--dest` defines the directory where you want to setup NSO (if the directory does not exist, it will be created)
- \* `--package` defines the NEDs you want to this NSO instance to have installed. You can specify this option multiple times.



# Using ncs-setup to Create an NSO Instance

Go ahead and run this command to setup an NSO instance in the current directory with the IOS, NX-OS, IOS-XR and ASA NEDs, you only need one NED per platform that you want NSO to manage (even though you may have multiple versions in your installer neds directory).

```
cd /
```

```
ncs-setup --package /nso/packages/neds/cisco-ios-cli-6.42 \
--package /nso/packages/neds/cisco-nx-cli-5.13 \
--package /nso/packages/neds/cisco-iosxr-cli-7.18 \
--package /nso/packages/neds/cisco-asa-cli-6.7 \
--dest nso-instance
```



# Checkout the NSO Instance

If you checkout the nso-instance directory now, you'll find several new files and folders have been created. This guide won't go through them all in detail now, but a couple are handy to know about.

- `ncs.conf` is the NSO application configuration file. Used to customize aspects of the NSO instance (change ports, enable / disable features, etc.). The defaults are often perfect for projects like this.
- `packages/` is the directory that has symlinks to the NEDs that we referenced in the --package arguments at setup.
- `logs/` is the directory that contains all the logs from NSO. This directory is useful when troubleshooting.

```
[root@localhost /]# cd /nso-instance/  
[root@localhost nso-instance]# ls  
logs ncs-cdb ncs.conf packages README.ncs scripts state
```

# 查看NSO Instance

```
[root@localhost /]# cd /nso-instance/packages/
```

```
[root@localhost packages]# ls -an
```

总用量 0

```
drwxr-xr-x 2 0 0 110 11月 30 21:41 .
```

```
drwxr-xr-x 7 0 0 111 11月 30 21:41 ..
```

```
lrwxrwxrwx 1 0 0 36 11月 30 21:41 cisco-asa-cli-6.7 -> /nso/packages/neds/cisco-asa-cli-6.7
```

```
lrwxrwxrwx 1 0 0 37 11月 30 21:41 cisco-ios-cli-6.42 -> /nso/packages/neds/cisco-ios-cli-6.42
```

```
lrwxrwxrwx 1 0 0 39 11月 30 21:41 cisco-iosxr-cli-7.18 -> /nso/packages/neds/cisco-iosxr-cli-7.18
```

```
lrwxrwxrwx 1 0 0 36 11月 30 21:41 cisco-nx-cli-5.13 -> /nso/packages/neds/cisco-nx-cli-5.13
```



# Start NSO instance

Now you need to "start" your NSO instance. Navigate to the `nso-instance` directory and type the command `ncs`. It will take a few seconds to run, and you won't get any explicit output unless there is a problem.

```
[root@localhost ~]# cd /nso-instance/  
[root@localhost nso-instance]# ls  
logs ncs-cdb ncs.conf packages README.ncs scripts state  
[root@localhost nso-instance]# ncs
```



# Verify NSO status

You can verify that NSO is running by using the `ncs --status | grep status` command, which has a large amount of information, so we use grep to just search for the status:

```
[root@localhost nso-instance]# ncs --status | grep status
status: started
db=running id=31 priority=1 path=/ncs:devices/device/live-status-protocol/device-type
```



# Populating NSO Instance

Now that you have a local NSO instance created and running, you need a few more things before you can start automating:

1. An authgroup that enables device credential authentication.
2. Device information to populate the device list.
3. Device groups are helpful, but not required.



# Setting up Device Authentication

With NSO running, you now need to add your devices into its inventory. This enables NSO to read and write data. NSO uses authgroups to set up credentials for device access. You'll set up a simple authgroup for your network.

- 1、Enter your NSO instance with the `ncs_cli -C -u admin` command. This command takes several options. **This example uses -C to specify a "Cisco style" command line interface (the default is a Juniper style), and -u admin to login as the "admin" user.**

```
[root@localhost nso-instance]# ncs_cli -C -u admin
```

```
admin connected from 10.1.1.100 using ssh on localhost.localdomain  
admin@ncs#
```



# Setting up Device Authentication

- 2、Enter "config mode" with the config command.

```
admin@ncs# config  
Entering configuration mode terminal  
admin@ncs(config)#
```



# Setting up Device Authentication

3、You will set up a new authgroup called `ladmin`. This group will use a default username/password combination of `cisco / cisco` for devices, with a secondary password of `cisco`. Use the following commands:

```
admin@ncs(config)# devices authgroups group qytadmin #认证组
admin@ncs(config-group-qytadmin)# default-map remote-name admin
admin@ncs(config-group-qytadmin)# default-map remote-password Cisc0123
admin@ncs(config-group-qytadmin)# default-map remote-secondary-password Cisc0123 #enable密码
admin@ncs(config-group-qytadmin)# top
admin@ncs(config)# show configuration
devices authgroups group qytadmin
  default-map remote-name admin
  default-map remote-password $9$BPpFI9RmoU3LveUwEhaYhx8o1V6GLUypaagJ7I9oPc=
  default-map remote-secondary-password $9$YM1bmKoXsakQBylgElo5JcjSL8ojndsxyi79mXkDKGM=
!
admin@ncs(config)# commit # 提交才生效
Commit complete.
```



# IOS-XR 1000v配置

## CSR1

```
interface GigabitEthernet1
ip address 10.1.1.253 255.255.255.0
!
aaa new-model
!
aaa authentication login vty local
aaa authorization exec vty local
!
username admin privilege 15 password 0 Cisc0123
!
line vty 0 15
authorization exec vty
login authentication vty
```

## CSR17

```
interface GigabitEthernet1
ip address 10.1.1.252 255.255.255.0
!
aaa new-model
!
aaa authentication login vty local
aaa authorization exec vty local
!
username admin privilege 15 password 0 Cisc0123
!
line vty 0 15
authorization exec vty
login authentication vty
```

configure replace flash:/preconfig-aaa.cfg



# Adding devices to Cisco NSO

With your authgroup set up, let's go ahead and [add your devices](#) to the inventory. To add a device you'll need the following information:

1. The device's IP address or FQDN.
2. The protocol (ssh/telnet) and port (if non-standard) to connect to the device.
3. The authgroup to use for the device (which must be committed before adding devices).
4. The NED (device driver) you'll use to connect to the device.



# Adding devices to Cisco NSO

```
devices device CSR1  
address 10.1.1.253  
ssh host-key-verification none  
authgroup qytadmin  
device-type cli ned-id cisco-ios-cli-6.42  
device-type cli protocol ssh  
state admin-state unlocked
```

```
devices device CSR17  
address 10.1.1.252  
ssh host-key-verification none  
authgroup qytadmin  
device-type cli ned-id cisco-ios-cli-6.42  
device-type cli protocol ssh  
state admin-state unlocked
```



# Adding devices to Cisco NSO

2、Verify that you are still in the CLI context for the device. You will know this by the (`config-device-IOSXR-1000V`) prompt or by checking with `pwd`:

```
admin@ncs(config)# devices device CSR1
admin@ncs(config-device-CSR1)# pwd
Current submode path:
devices device CSR1
```



# Adding devices to Cisco NSO

- 3、Use the `connect` command to determine if you can connect to the device with NSO.

```
admin@ncs(config-device-CSR1)# connect  
result false  
info Device IOSXR-1000V is southbound locked
```

**Note:** The output shows that you are currently **locked**. NSO's default mode for devices is a locked state that prevents NSO from manipulating a device before the network admin is ready. This ability can also be used disable a device that is currently in maintenance to prevent NSO from acting on it.



# Adding devices to Cisco NSO

- 4、Unlock the device by changing its admin-state and commit the change.

```
admin@ncs(config)# devices device CSR1
admin@ncs(config-device-IOSXR-1000V)# state admin-state unlocked
admin@ncs(config-device-IOSXR-1000V)# commit
Commit complete.
```

- 5、Now try to connect again,

```
admin@ncs(config-device-CSR1)# connect
result true
info (admin) Connected to CSR1 - 10.1.1.253:22
```



# 查看设备配置

```
admin@ncs# show running-config devices device | de-select config  
devices device CSR1  
address 10.1.1.253  
ssh host-key-verification none  
authgroup labadmin  
device-type cli ned-id cisco-ios-cli-6.42  
device-type cli protocol ssh  
state admin-state unlocked  
!  
devices device CSR17  
address 10.1.1.252  
ssh host-key-verification none  
authgroup labadmin  
device-type cli ned-id cisco-ios-cli-6.42  
device-type cli protocol ssh  
state admin-state unlocked
```



# 查看设备列表

```
admin@ncs# show devices list
```

NAME	ADDRESS	DESCRIPTION	NED ID	ADMIN STATE
CSR1	10.1.1.253	-	cisco-ios-cli-6.42	unlocked
CSR17	10.1.1.252	-	cisco-ios-cli-6.42	unlocked



# 查看设备连接

```
admin@ncs# devices connect
connect-result {
    device CSR1
    result true
    info (admin) Connected to CSR1 - 10.1.1.253:22
}
connect-result {
    device CSR17
    result true
    info (admin) Connected to CSR17 - 10.1.1.252:22
}
```



# 查看已连接设备IOS-XR 1000v配置

```
admin@ncs# show running-config devices device CSR17 config  
devices device CSR17  
config  
no service password-encryption  
no cable admission-control preempt priority-voice  
no cable qos permission create  
no cable qos permission update  
no cable qos permission modems  
ip source-route  
no ip cef  
no ip forward-protocol nd  
no ipv6 cef  
no dot11 syslog  
!  
!
```



# "Learning" Current Network State

Now NSO can successfully connect to the network, but it hasn't "learned" the current network configuration deployed to the devices.

The NSO sync-from command enables the network devices to "learn" the current configuration.

**Note:** Be sure you do NOT enter devices **sync-to** by mistake. NSO will attempt to overwrite the configuration on the devices with the "**blank**" configuration that NSO currently has for the devices. This would be bad here (though there are scenarios where you may want to do this).



# "Learning" Current Network State

```
admin@ncs# devices sync-from device CSR17
sync-result {
    device CSR17
    result true
}
```



# "Learning" Current Network State

```
admin@ncs# show running-config devices device CSR17 config  
devices device CSR17  
config  
    tailfned police cirmode  
    version 17.1  
    service timestamps debug datetime msec  
    service timestamps log datetime msec  
    no service password-encryption  
    service call-home  
    login on-success log  
    platform console virtual  
    platform punt-keepalive disable-kernel-core  
    platform qfp utilization monitor load 80  
    hostname CSR17  
    call-home  
        contact-email-addr sch-smart-licensing@cisco.com  
        profile CiscoTAC-1  
            active  
            reporting smart-licensing-data  
---忽略大量输出---
```



# NSO CLI

```
admin@ncs# show running-config devices device CSR17 config interface
devices device CSR17
config
    interface GigabitEthernet1
        no switchport
        negotiation auto
        no mop enabled
        no mop sysid
        ip address 10.1.1.252 255.255.255.0
        no shutdown
    exit
!
!
```



# NSO CLI

```
admin@ncs# show running-config devices device CSR17 config interface | display json  
//接口配置 使用json方式显示
```

```
admin@ncs# show running-config devices device CSR17 config interface | display xml | save xmlinterface.xml
```

```
admin@ncs# show running-config devices device CSR17 config interface GigabitEthernet* ip address  
devices device CSR17  
config  
interface GigabitEthernet1  
ip address 10.1.1.252 255.255.255.0  
exit  
!
```

```
admin@ncs# show running-config devices device CSR17 config interface | display xpath
```



# 查看配置修改

# 修改路由器配置! 即可查看配置变化

```
admin@ncs# devices device CSR1 compare-config  
diff  
devices {  
    device CSR1 {  
        config {  
            -      hostname CSR1;  
            +      hostname CSR2;  
        }  
    }  
}
```



# Updating device configuration

```
admin@ncs(config)# devices device CSR1
admin@ncs(config-device-CSR1)# config
admin@ncs(config-config)# pwd
Current submode path:
    devices device CSR1 \ config
admin@ncs(config-config)# interface Loopback 0
admin@ncs(config-if)# ip address 1.1.1.1 255.255.255.0
admin@ncs(config-if)# top
admin@ncs(config)#
```



# Updating device configuration

```
admin@ncs(config)# show configuration  
devices device CSR1  
config  
    interface Loopback0  
        ip address 1.1.1.1 255.255.255.0  
        no shutdown  
    exit  
!  
!
```



# Updating device configuration

```
admin@ncs(config)# show configuration | display xml
<devices xmlns="http://tail-f.com/ns/ncs">
  <device>
    <name>CSR1</name>
    <config>
      <interface xmlns="urn:ios">
        <Loopback>
          <name>0</name>
          <ip>
            <address>
              <primary>
                <address>1.1.1.1</address>
                <mask>255.255.255.0</mask>
              </primary>
            </address>
          </ip>
        </Loopback>
      </interface>
    </config>
  </device>
</devices>
```



# Updating device configuration

Before you're ready to commit this to the network, you may want to know what will NSO actually SEND to your device. You can run the NSO **dry-run** option before committing, and the **native** format shows you exactly what will be sent away.

```
admin@ncs(config)# commit dry-run outformat native
native {
    device{
        name CSR1
        data interface Loopback0
            ip redirects
            ip address 1.1.1.1 255.255.255.0
            no shutdown
            exit
    }
}
admin@ncs(config)# commit
Commit complete.
```



# Rollback

```
admin@ncs(config)# show configuration rollback changes 1000
```

Possible completions:

- 10001 2020-11-30 21:45:11 by system via system
- 10002 2020-11-30 21:50:21 by admin via cli
- 10003 2020-11-30 21:58:07 by admin via cli
- 10004 2020-11-30 21:59:55 by admin via cli
- 10005 2020-11-30 22:01:40 by admin via cli
- 10006 2020-11-30 22:01:45 by admin via cli
- 10007 2020-11-30 22:04:54 by admin via cli
- 10008 2020-11-30 22:05:57 by admin via cli
- 10009 2020-11-30 22:10:54 by admin via cli

<cr> latest

```
admin@ncs(config)# show configuration rollback changes 10009
```

devices device CSR1

config

no interface Loopback0



# Rollback

```
admin@ncs(config)# rollback configuration 10009
admin@ncs(config)# show configuration
devices device CSR1
config
  no interface Loopback0
!
!
admin@ncs(config)# commit
Commit complete.
```



# Device template

using Device Templates to configure something on many devices.

```
admin@ncs(config)# devices template set_dns_server //创建模版
admin@ncs(config-template-set_dns_server)# ned-id cisco-ios-cli-6.42
admin@ncs(config-ned-id-cisco-ios-cli-6.42)# config //配置
admin@ncs(config-config)# ip name-server name-server-list 114.114.114.114 //配置DNS server
admin@ncs(config-name-server-list-114.114.114.114)# top
```



# Device template

```
admin@ncs(config)# show configuration  
devices template set_dns_server  
ned-id cisco-ios-cli-6.42  
config  
    ip name-server name-server-list 114.114.114.114  
!  
!  
!  
!  
admin@ncs(config)# commit //应用配置  
Commit complete.
```



# Device template

```
admin@ncs# show devices list
```

NAME	ADDRESS	DESCRIPTION	NED ID	ADMIN STATE
<hr/>				
CSR1	10.1.1.253	-	cisco-ios-cli-6.42	unlocked
CSR17	10.1.1.252	-	cisco-ios-cli-6.42	unlocked

# Device Group

```
admin@ncs(config)# devices device-group csr_group
admin@ncs(config-device-group-csr_group)# device-name CSR1
admin@ncs(config-device-group-csr_group)# device-name CSR17
admin@ncs(config-device-group-csr_group)# top
admin@ncs(config)# show configuration
devices device-group csr_group
  device-name [ CSR1 CSR17 ]
!
admin@ncs(config)# commit
Commit complete.
```



# Apply the template

```
admin@ncs(config)# devices device-group csr_group apply-template template-name
set_dns_server
apply-template-result {
    device CSR1
    result ok
}
apply-template-result {
    device CSR17
    result ok
}
admin@ncs(config)# commit
Commit complete.
```



# Device template with variable

```
admin@ncs(config)# devices template qyt_tmp_with_vars
admin@ncs(config-template-qyt_tmp_with_vars)# ned-id cisco-ios-cli-6.42
admin@ncs(config-ned-id-cisco-ios-cli-6.42)# config
admin@ncs(config-config)# ip access-list standard std-named-acl {$qyt-acl-name}
admin@ncs(config-std-named-acl-{$qyt-acl-name})# std-access-list-rule {$qyt-rule-1}
admin@ncs(config-std-access-list-rule-{$qyt-rule-1})# exit
admin@ncs(config-std-named-acl-{$qyt-acl-name})# std-access-list-rule {$qyt-rule-2}
admin@ncs(config-std-access-list-rule-{$qyt-rule-2})# exit
admin@ncs(config-std-named-acl-{$qyt-acl-name})# commit dry-run outformat xml
admin@ncs(config-std-named-acl-{$qyt-acl-name})# top
admin@ncs(config)# commit
Commit complete.
```



# Apply Device template with variable

```
admin@ncs(config)# devices device-group csr_group apply-template template-name qyt_tmpl_with_vars variable { name qyt-acl-name value 'qyt-acl' } variable { name qyt-rule-1 value 'permit\\ 1.1.1.1' } variable { name qyt-rule-2 value 'permit\\ 2.2.2.2' }
apply-template-result {
    device CSR1
    result ok
}
apply-template-result {
    device CSR17
    result ok
}
admin@ncs(config)# commit
Commit complete.
```

# 执行Show命令

```
admin@ncs(config)# devices device CSR1 live-status exec show clock  
result  
*01:35:50.014 UTC Tue Dec 1 2020  
CSR1#  
admin@ncs(config)# devices device CSR1 live-status exec any "show clock"  
result  
*01:36:00.936 UTC Tue Dec 1 2020  
CSR1#
```



# 创建Service

```
[root@localhost ~]# source /nso/ncsrc
[root@localhost ~]# cd /nso-instance/packages
[root@localhost packages]# pwd
/nso-instance/packages
[root@localhost packages]# ncs-make-package --service-skeleton template qyt_radius --augment
/ncs:services
[root@localhost packages]# ls
cisco-asa-cli-6.7 cisco-ios-cli-6.42 cisco-iosxr-cli-7.18 cisco-nx-cli-5.13 qyt_radius
```

# 配置Yang

```
[root@localhost ~]# cd /nso-instance/packages/qyt_radius/src/yang/  
[root@localhost yang]# mv qyt_radius.yang qyt_radius_bak.yang  
[root@localhost yang]# vi qyt_radius.yang
```

```
module qyt_radius {  
    namespace "http://com/example/qyt_radius";  
    prefix qyt_radius;  
  
    import ietf-inet-types {  
        prefix inet;  
    }  
    import tailf-ncs {  
        prefix ncs;  
    }  
  
    augment /ncs:services {  
        list qyt_radius {  
            key radius_server_name;  
  
            uses ncs:service-data;  
            ncs:servicepoint "qyt_radius";  
  
            leaf radius_server_name {  
                type string;  
            }  
  
            leaf radius_server_ip {  
                type inet:ipv4-address;  
            }  
  
            leaf-list device {  
                type leafref {  
                    path "/ncs:devices/ncs:device/ncs:name";  
                }  
            }  
  
            leaf dummy {  
                type inet:ipv4-address;  
            }  
        } // augment /ncs:services {  
    }
```

# Make

```
[root@localhost -]# yum install gcc automake autoconf libtool make  
  
[root@localhost -]# cd /nso-instance/packages/qyt_radius/src  
[root@localhost src]# ls  
Makefile yang  
[root@localhost src]# make  
mkdir -p ..../load-dir  
/nso/bin/ncsc `ls qyt_radius-ann.yang > /dev/null 2>&1 && echo "-a qyt_radius-ann.yang"\`  
-c -o ..../load-dir/qyt_radius.fxs yang/qyt_radius.yang
```



# 修改Template

```
[root@localhost ~]# cd /nso-instance/packages/qyt_radius/templates/  
[root@localhost templates]# cp qyt_radius-template.xml qyt_radius-  
template_bak.xml  
[root@localhost templates]# vi qyt_radius-template.xml
```

```
<config-template xmlns="http://tail-f.com/ns/config/1.0" servicepoint="qyt_radius">  
  <devices xmlns="http://tail-f.com/ns/ncs">  
    <device>  
      <!--  
        Select the devices from some data structure in the service  
        model. In this skeleton the devices are specified in a leaf-list.  
        Select all devices in that leaf-list:  
      -->  
      <name>/device</name>  
      <config>  
        <!--  
          Add device-specific parameters here.  
          In this skeleton the service has a leaf "dummy"; use that  
          to set something on the device e.g.:  
          <ip-address-on-device>/dummy</ip-address-on-device>  
        -->  
        <radius xmlns="urn:ios">  
          <server>  
            <id>/radius_server_name</id>  
            <address>  
              <ipv4>  
                <host>/radius_server_ip</host>  
                <auth-port>1812</auth-port>  
                <acct-port>1813</acct-port>  
              </ipv4>  
            </address>  
          </server>  
        </radius>  
      </config>  
    </device>  
  </devices>  
</config-template>
```



# Reload package

```
[root@k8s-master01 templates]# ncs_cli -C -u admin  
  
admin connected from 10.1.1.100 using ssh on k8s-master01  
admin@ncs# packages reload force  
  
>>> System upgrade is starting.  
>>> Sessions in configure mode must exit to operational mode.  
>>> No configuration changes can be performed until upgrade has completed.  
>>> System upgrade has completed successfully.  
reload-result{  
    package cisco-asa-cli-6.7  
    result true  
}  
reload-result{  
    package cisco-ios-cli-6.42  
    result true  
}  
reload-result{  
    package cisco-iosxr-cli-7.18  
    result true  
}  
reload-result{  
    package cisco-nx-cli-5.13  
    result true  
}  
reload-result {  
    package qyt_radius  
    result true
```

# 使用service

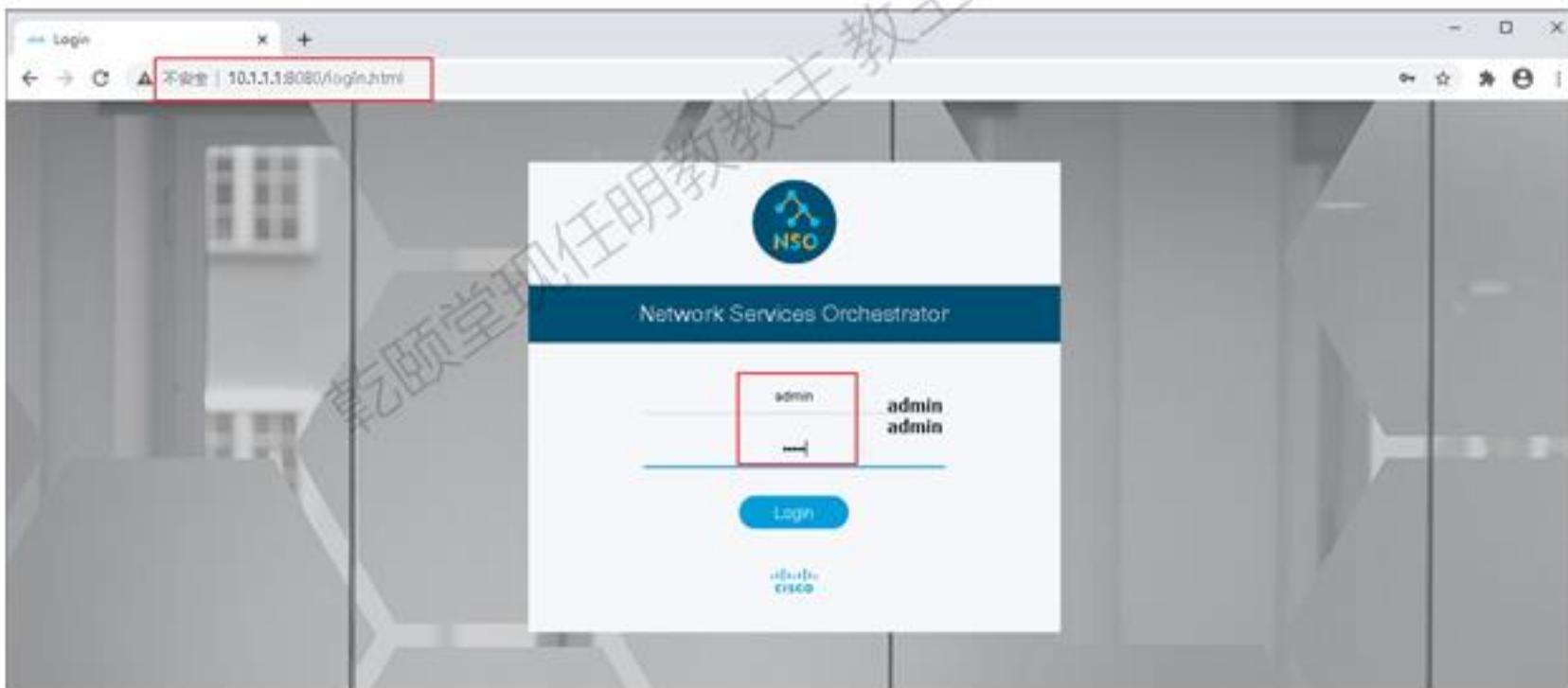
```
admin@ncs(config)# services qyt_radius qyt_radius_server
admin@ncs(config-qyt_radius-qyt_radius_server)# radius_server_ip 10.1.1.123
admin@ncs(config-qyt_radius-qyt_radius_server)# device CSR1
admin@ncs(config-qyt_radius-qyt_radius_server)# commit dry-run
cli {
    local-node{
        data devices{
            device CSR1{
                config{
                    radius{
                        +     server qyt_radius_server {
                        +         address{
                        +             ipv4{
                        +                 host 10.1.1.123;
                        +                 auth-port 1812;
                        +                 acct-port 1813;
                        +             }
                        +         }
                        +     }
                    }
                }
            }
        }
    }
}
services{
+  qyt_radius qyt_radius_server {
+    radius_server_ip 10.1.1.123;
+    device [CSR1];
+  }
}
}
admin@ncs(config-qyt_radius-qyt_radius_server)# commit
Commit complete.
```



# NSO Web Interface

You can use the NSO WebUI as well as the CLI or one of the northbound interfaces to interact with NSO.

<http://10.1.1.90:8080> 默认用户名: admin 密码: admin



# NSO Web Interface

The screenshot displays the Cisco NSO Application hub interface. At the top left is the Cisco logo and the text "Application hub NSO VERSION 5.3". At the top right is the user "admin". Below the header, there are two main sections: "CONFIGURATION" and "MONITORING".

**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**

- Dashboard**: Monitor number of devices, user sessions and service instances. Overview service progress monitoring.

At the bottom, there is a legend with icons and labels:

- C Commit manager**
- E Configuration editor**
- B Dashboard**
- D Device manager**
- S Service manager**



# Adding devices to Cisco NSO - 1

The screenshot shows the Cisco NSO Application hub interface. At the top left is the Cisco logo and "Application hub" text. In the top right corner, there is a "admin" user icon. Below the header, there are two main sections: "CONFIGURATION" and "MONITORING".

**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. This section is highlighted with a red box and has a red circle with the number 1 above it.
- Service manager**: Find and synchronize your services. See deployment status and access the service configuration data.

**MONITORING**

- Dashboard**: Monitor number of devices, user sessions and service instances. Overview service progress monitoring.

At the bottom of the screen, there is a navigation bar with icons and labels:

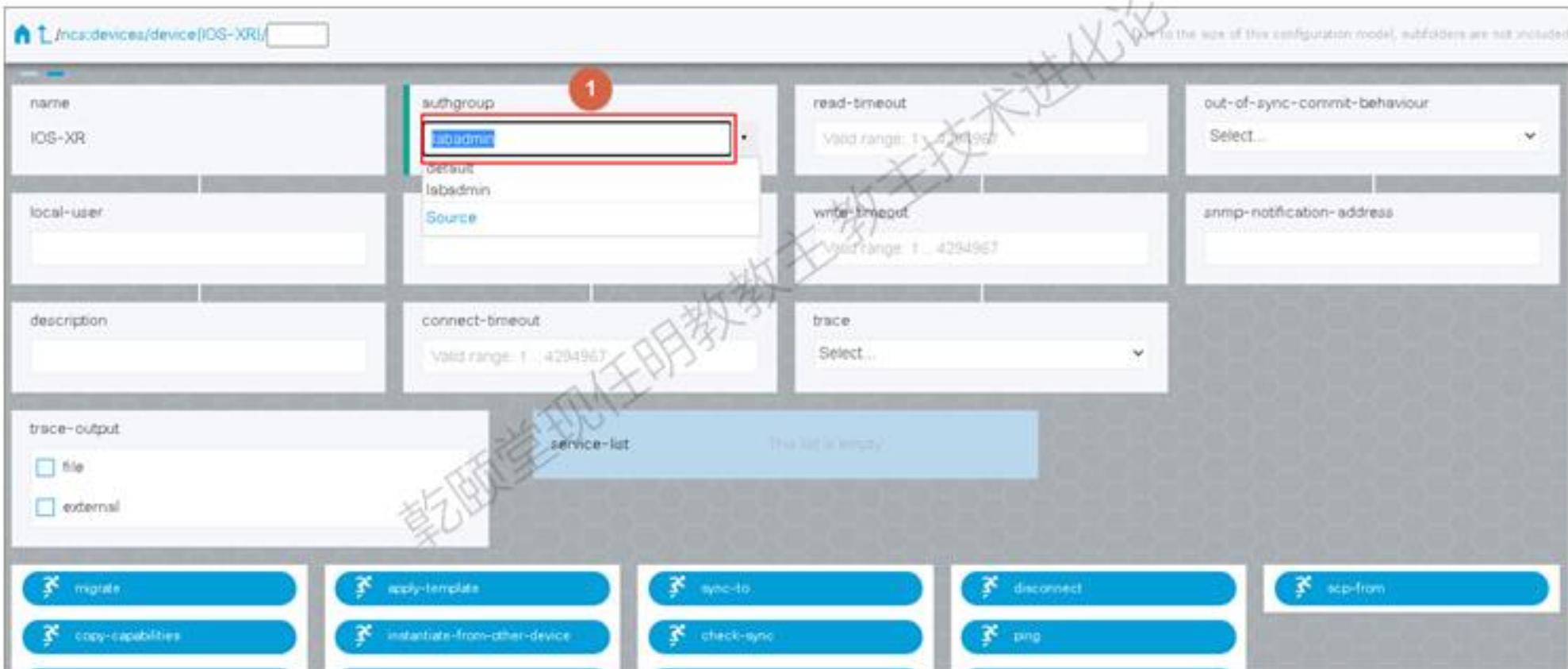
- C Commit manager
- E Configuration editor
- B Dashboard
- D Device manager
- S Service manager



# Adding devices to Cisco NSO-2

The screenshot shows the Cisco NSO Device manager interface. A modal dialog box titled "Add device" is open in the center. The dialog contains a "name" input field with the value "IOS-XR" and two buttons at the bottom: "cancel" and "confirm". The "confirm" button is highlighted with a red circle and labeled with the number 3. In the background, the main interface shows a table of devices with one entry: "IOSXR-1000V 10.1.1.100 cisco-ios-cl-6.42:cisco-ios-cl-6.42 0". The "name" column has a red box around its header and a red circle with the number 1. The "address" column has a red box around its header.

# Adding devices to Cisco NSO-3



# Adding devices to Cisco NSO-4

The screenshot shows the Cisco Network Services Orchestrator (NSO) interface for adding devices. The URL in the address bar is `/nca/devices/device[IOS-XR]/`. The main content area is titled "address-choice". A red box highlights the "address" input field, which contains the value "10.1.1.103". A red circle with the number "1" is positioned next to this field. Below the address field are other input fields: "port" (with a placeholder "Valid range: 0 ... 65535") and "remote-node" (an empty input field). At the top of the page, there are four blue buttons: "add-capability", "sync-from", "connect", and "scp-to". A note at the top right states: "Due to the size of this configuration model, subfolders are not included".

# Adding devices to Cisco NSO-5

The screenshot shows the Cisco NSO Configuration Editor interface. At the top, it displays "Configuration editor" and "NSO VERSION 5.3". On the right, there are "View options" and "admin" buttons. The main area has a URL bar with a red box around it, containing the path `/ncs:devices/device(IOS-XR)/ssh/`. A red circle with the number 1 is placed over this URL bar. Below the URL bar, there is a dropdown menu labeled "host-key-verification" with the option "none - Accept any host key" selected. A red box surrounds this dropdown, and a red circle with the number 2 is placed over it. Below the dropdown is a blue button labeled "fetch-host-keys". At the bottom of the screen, there is a section titled "host-key" with the message "This list is empty" and a "Add list item" button with a plus sign.



# Adding devices to Cisco NSO-6

Configuration editor  
NSO VERSION 5.3

View options • admin •

1 /ncs/devices/device{IOS-XR}/device-type/cli/

2 ned-id\*  
cisco-ios-cli-6.42:cisco-ios-cli-6.42

3 protocol  
telnet

Due to the size of this configuration model, subfolders are not included

The screenshot shows the Cisco NSO Configuration Editor interface. At the top left is the Cisco logo and the text 'Configuration editor' and 'NSO VERSION 5.3'. On the right are 'View options' and 'admin' buttons. A red circle labeled '1' highlights the URL bar containing the path '/ncs/devices/device{IOS-XR}/device-type/cli/'. Below the URL bar is a red box around a dropdown menu labeled 'ned-id\*' with the value 'cisco-ios-cli-6.42:cisco-ios-cli-6.42', which is also highlighted by a red circle labeled '2'. Further down is another red box around a dropdown menu labeled 'protocol' with the value 'telnet', also highlighted by a red circle labeled '3'. A note at the bottom right states 'Due to the size of this configuration model, subfolders are not included'. A large watermark reading '乾颐堂现任明教教主技术进化论' is diagonally across the page.

# Adding devices to Cisco NSO-7

The screenshot shows the Cisco NSO Configuration editor interface. The top navigation bar includes the Cisco logo, 'Configuration editor', 'NSO VERSION 5.3', and user options like 'View options' and 'admin'. A red box labeled '1' highlights the URL bar containing the path '/ncs:devices/device(IOS-XR)/state/'. A red box labeled '2' highlights the 'admin-state' field, which is set to 'unlocked - Device is open for config and sout'. The interface displays various configuration parameters such as 'oper-state', 'last-transaction-id', 'open-state-error-tag', 'transaction-mode', and 'last-modules-state/'. A note at the top right states: 'Due to the size of this configuration model, subfolders are not included'.



# Adding devices to Cisco NSO-8

Commit manager  
NSO VERSION 8.3

Current transaction is VALID    Revert    Load/Save    admin +

changes	errors	warnings	config	native config	commit queue
/ncs:devices/device/OS-XR/device-type/cli/led-id					Operation: value_set
/ncs:devices/device/OS-XR/device-type/cli/protocol					Old value: telnet
/ncs:devices/device/OS-XR/address					New value: 10.1.1.103
/ncs:devices/device/OS-XR/state/admin-state					Old value: unlocked
/ncs:devices/device/OS-XR/authgroup					New value: labadmin
/ncs:devices/device/OS-XR					Operation: created

Commit changes to NSO?  
No commit options set

cancel    Yes, commit    1

1    C\* Commit manager    E Configuration editor    B Dashboard    D Device manager    S Service manager

# Adding devices to Cisco NSO - 9

Device manager  
NSO VERSION: 5.3

+ - +Add filter admin

1 / 2

name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	co
<input checked="" type="checkbox"/> IOS-XR	10.1.1.103		cisco-ios-cli-6.42:cisco-ios-cli-6.42	0	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">ping</span>	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">connect</span>	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">check-sync</span>	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">sync-from</span>	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">sync-to</span>	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">compare-config</span>		
<p>Action: connect      Performed: 2020-11-19 20:31:52      Status: completed</p> <p>Result: device was connected</p> <p>true (admin) Connected to IOS-XR - 10.1.1.103:23</p>												
<input type="checkbox"/> IOSXR-1000V	10.1.1.102		cisco-ios-cli-6.42:cisco-ios-cli-6.42	0	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">ping</span>	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">connect</span>	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">check-sync</span>	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">sync-from</span>	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">sync-to</span>	<span style="background-color: #0070C0; color: white; border-radius: 50%; padding: 2px 5px;">compare-config</span>		

# Config Device

Device manager  
NSO VERSION 5.3

0 / 2 + - grid list Add filter

address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
<input type="checkbox"/> 10.1.1.103		cisco-ios-cli-6.42:cisco-ios-cli-6.42	0	ping	connect	check-sync	sync-from	sync-to	compare-config	1	configuration
<input type="checkbox"/> 0V	10.1.1.102	cisco-ios-cli-6.42:cisco-ios-cli-6.42	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration

Action: compare-config      Performed: 2020-11-19 22:05:15      Status: completed

Result: CDB and device config matches

# Config Device interface

The screenshot shows the Cisco Configuration Editor interface. At the top left is the Cisco logo and the text "Configuration editor NSO VERSION 5.3". On the right are "View options" and "admin" dropdown menus. The URL bar shows the path: "/ncs:devices/device[IOS-XR]/config/ios/interface/Loopback[2]/ip/address/primary/". A note on the right states: "Due to the size of this configuration model, subfolders are not included". The main area contains two input fields: "address" with value "111.1.1.1" and "mask" with value "255.255.255.255".

# Config router

The screenshot shows the Cisco Configuration editor interface. The top bar displays the Cisco logo, "Configuration editor", "NSO VERSION 5.3", and user status. The URL bar shows the path: `/ncs:devices/device[IOS-XR]/config/ios:router/ospf[1]/network[1.1.1.1 0.0.0.0]/`. A note on the right states: "Due to the size of this configuration model, subfolders are not included". On the left, there is a sidebar with configuration parameters:

- ip: 1.1.1.1
- mask: 0.0.0.0
- area: 0

# Config Device

**CISCO Commit manager NSO VERSION 5.3**

Current transaction is VALID    Revert    Load/Save    **Commit** (2)

changes	errors	warnings	config	native config	commit queue
Path					
/ncs:devices/device[IOS-XR]/config/ios:router/ospf[1]/network[1.1.1.1 0.0.0.0]/areas					value_set 0
/ncs:devices/device[IOS-XR]/config/ios:router/ospf[1]/network[1.1.1.1 0.0.0.0]					created
/ncs:devices/device[IOS-XR]/config/ios:router/ospf[1]/router-id					value_set 1.1.1.1
/ncs:devices/device[IOS-XR]/config/ios:router/ospf[1]					creased
/ncs:devices/device[IOS-XR]/config/ios:router/eigrp[90]/network-ip/network[10.1.1.1 10.1.1.103]					creased
/ncs:devices/device[IOS-XR]/config/ios:router/eigrp[90]					modified
/ncs:devices/device[IOS-XR]/config/ios:interface/Loopback[2]/ip/address/primary/address					value_set 1.1.1.1
/ncs:devices/device[IOS-XR]/config/ios:interface/Loopback[2]/ip/address/primary/mask					value_set 255.255.255.255
/ncs:devices/device[IOS-XR]/config/ios:interface/Loopback[2]					creased
/ncs:devices/device[IOS-XR]/config/ios:ios					created
/ncs:devices/device[IOS-XR]					modified

**Commit manager**    Configuration editor    Dashboard    Device manager    Service manager