TECOPS-2201

From Zero to Hero:

Cisco Network Services Orchestrator (NSO)

Intermediate

Sofia Athanasiou - Customer Success Specialist

Hector Oses – Customer Delivery Software Architect

Table of Contents

[Overview 3](#_Toc157866926)

[Learning Objectives 3](#_Toc157866927)

[Disclaimer 4](#_Toc157866928)

[NSO Overview 4](#_Toc157866929)

[Scenario 5](#_Toc157866930)

[Network Diagram 5](#_Toc157866931)

[Lab environment 6](#_Toc157866932)

[Lab Introduction and Verification 6](#_Toc157866933)

[Task 1: Configure build-in High Availability 8](#_Toc157866934)

[Step 1: Verifications before configuring High Availability 9](#_Toc157866935)

[Step 2: Enable High Availability 9](#_Toc157866936)

[Step 3: Verify High-Availability 9](#_Toc157866937)

[Task 2: Layered Services Architecture 10](#_Toc157866938)

[Task 3: Kickers and Subscribers 11](#_Toc157866939)

[Task 4: Nano-services 11](#_Toc157866940)

[Summary 11](#_Toc157866941)

[Appendix A 12](#_Toc157866942)

[Use Makefile to configure netsim devices 12](#_Toc157866943)

[Appendix B 12](#_Toc157866944)

[Start NSO and verify status 12](#_Toc157866945)

[Related Sessions at Ciscolive 14](#_Toc157866946)

Overview

Network operators and service providers today are struggling to control the difference between the growth of their operating costs and their revenue. Introduction and deployment of new services is much slower compared to service demand and availability on the market. It is because of inadequate provisioning processes where services are either configured manually or hard coded inside the Operations Support Systems (OSS). Cisco Network Service Orchestrator (NSO) is the answer to the above challenge. NSO architecture decouples network services from specific components, while automatically configuring the network according to the service specifications. NSO enabled by NETCONF and YANG models, enables operators to dynamically adopt the service configuration solution according to changes in the offered service portfolio.

This session is intended to familiarize the novice NSO user with the architecture and capabilities of the platform, touching standards utilized by NSO, such as NETCONF and YANG. The session will further discuss NSO components, service and device abstraction, integration with northbound systems via Application Programming Interfaces (APIs), communication procedure with southbound devices via Network Element Drivers (NEDs), configuration compliance, and configuration data collection.

Learning Objectives

Upon completion of this lab, you will be able to:

* Configure NSO build-in High Availability,
* Scale your services by using Layered Services Architecture,
* Configure kickers and use them inside your service,
* Use nano-services to create l3mplsvpn service.

The lab has 4 main excercises:

* **Exercise 1:** Configure build-in High Availability. It will help you get familiar with the concept of High-Availability and its benefits.
* **Exercise 2:** Create a service using Layered Service Architecture.
* **Exercise 3:** Configure a service taking advantage of FASTMAP using kickers.
* **Exercise 4:** L3MPLSVPN service creation usingnano-services.

Disclaimer

This training document is to familiarize with Cisco NSO intermediate topics for Automating your network. Although the lab design and configuration examples could be used as a reference, it’s not a real design, thus not all recommended features are used, or enabled optimally. For the design related questions please contact your representative at Cisco, or a Cisco partner.

NSO Overview

Cisco® Network Services Orchestrator (NSO) enabled by Tail-f® is an industry-leading orchestration platform for hybrid networks. It provides comprehensive lifecycle service automation to enable you to design and deliver high-quality services faster and more easily.

Figure 1: NSO high-level architecture



OSS

Service Order

Minimal Device Reconfigurations

**NSO**

The network is a foundation for revenue generation. Therefore, service providers must implement network orchestration to simplify the entire lifecycle management for services. For today’s virtualized networks, this means transparent orchestration that spans multiple domains in your network and includes network functions virtualization (NFV) and software-defined networking (SDN) as well as your traditional physical network and all its components.

NSO is a model driven (YANG) platform for automating your network orchestration. It supports multi-vendor networks through a rich variety of Network Element Drivers (NEDs).

We support the process of validating, implementing and abstracting your network config and network services, providing support for the entire transformation into intent-based networking.

Scenario

There are two servers with NSO version 6.1.5 installed that are part of the same subnet. NSO is running 4 simulated devices taking the roles of Provider Edge in the network. Two of them run Cisco IOS, and the other two Cisco IOS XR.

Network Diagram

A diagram of a cloud computing system

Description automatically generated

Figure 2: Network Diagram

**Table 1:** Lab first aid

|  |  |  |  |
| --- | --- | --- | --- |
| Host name | IP address | Username | Password |
| NSO-01 server | 198.18.134.28 | cisco | C1sco12345 |
| NSO-02 server | 198.18.134.29 | cisco | C1sco12345 |
| PE\_00 | 127.0.0.1 | admin | admin |
| PE\_01 | 127.0.0.1 | admin | admin |
| PE\_11 | 127.0.0.1 | admin | admin |
| PE\_10 | 127.0.0.1 | admin | admin |

Lab environment

The lab runs inside dCloud in a Windows machine. NSO is installed in a Linux host and can be reached through SSH, GUI and some APIs (RESTCONF will be used) from the windows machine. Ways of development possible for this lab.

1. (preferred) Visual Studio Code. You can find a shortcut in the desktop to this application. When you start it, you will be connected to NSO server and you will be able to view and edit files from your local Windows. A Terminal is available as well. See Appendix A for more information.
2. Connect to NSO host through putty and edit the files directly there by ‘vim’. Desktop shortcut available. A computer screen with a lightning bolt

   Description automatically generated A computer screen with a lightning bolt

   Description automatically generated

Lab Introduction and Verification

The NSO version 6.1.5 is already installed and the required Network Element Drivers (NEDS) are loaded.

Desktop shortcut ‘NSO 1 Host’ and ‘NSO 2 Host’ allows you to connect to the Linux host where NSO is running as user ‘cisco’.

1. Change user to both NSOs be using nsoadmin user:

cisco@nso1:~$su nsoadmin

Password: nsoadmin

nsoadmin@nso1:/home/cisco$

1. Verify that the 4 netsim devices are up and running:

nsoadmin@nso1:/home/cisco$ cd /var/opt/ncs

nsoadmin@nso1:/var/opt/ncs$ ncs-netsim is-alive

DEVICE PE\_00 OK

DEVICE PE\_01 OK

DEVICE PE\_10 OK

DEVICE PE\_11 OK

If you are facing issues, follow Appendix A on how to bring back up the netsim devices. From same directory (/var/opt/ncs) use Makefile:

nsoadmin@nso1:/var/opt/ncs$ make rebuild-netsim

1. Access to NSO CLI. There are different ways to connect to NSO-01 CLI and to NSO-02 CLI.
   1. Double click on the icon shown here to access: A computer screen with a lightning bolt

      Description automatically generated
   2. Run from the previous terminal ‘ncs\_cli -u nsoadmin’ (default password for user admin is admin)

nsoadmin@nso1:/var/opt/ncs$ ncs\_cli –u nsoadmin

nsoadmin connected from 127.0.0.1 using console on ubuntu

nsoadmin@ncs>

* 1. When enabled in ncs.conf file, NSO allows direct access to CLI through SSH connection. Example:

nsoadmin@nso1:/var/opt/ncs$ ssh -l nsoadmin -p 2024 localhost

The authenticity of host '[localhost]:2024 ([127.0.0.1]:2024)' can't be established.

RSA key fingerprint is SHA256:nzmXDxz2gP7F8r5OYnz2d6OI20uwoHTRw+sstvftHI8.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[localhost]:2024' (RSA) to the list of known hosts.

nsoadmin@localhost's password:

nsoadmin connected from 127.0.0.1 using ssh on ubuntu

nsoadmin@nso1>

1. Cisco NSO allows 2 types of CLI to interact with it. During this workbook we will use Cisco based CLI. To use Cisco based CLI you can do it in 2 ways:
   1. You can run the command above and then ‘switch cli’.

nsoadmin@nso1:/var/opt/ncs$ **ncs\_cli –u nsoadmin**

admin connected from 127.0.0.1 using console on ubuntu

nsoadmin@nso1> **switch cli**

nsoadmin@nso1#

* 1. Run the above command with the additional option -C for cisco

nsoadmin@nso1:/var/opt/ncs $ **ncs\_cli –Cu nsoadmin**

nsoadmin connected from 127.0.0.1 using console on ubuntu

nsoadmin@ncs#

1. Let’s verify the 4 netsim devices are loaded into NSO

nsoadmin@nso1# **show devices list**

NAME ADDRESS DESCRIPTION NED ID ADMIN STATE

--------------------------------------------------------

PE\_00 127.0.0.1 - cisco-ios unlocked

PE\_01 127.0.0.1 - cisco-ios unlocked

PE\_10 127.0.0.1 - cisco-ios-xr unlocked

PE\_11 127.0.0.1 - cisco-ios-xr unlocked

1. Verify the required packages are loaded

nsoadmin@nso1# **show packages package package-version**

PACKAGE

NAME VERSION

------------------------------

cisco-ios-cli-6.100 6.100.9

cisco-iosxr-cli-7.53 7.53

juniper-junos-nc-4.14 4.14.3

l3mplsvpn 1.0

loopbackbasic 1.0

nsoadmin@nso1# **show packages package oper-status**

PACKAGE

PROGRAM META FILE

CODE JAVA BAD NCS PACKAGE PACKAGE CIRCULAR DATA LOAD ERROR

NAME UP ERROR UNINITIALIZED VERSION NAME VERSION DEPENDENCY ERROR ERROR INFO

---------------------------------------------------------------------------------------------------------

cisco-ios-cli-6.100 **X** - - - - - - - - -

cisco-iosxr-cli-7.53 **X** - - - - - - - - -

- -

Juniper-junos-nc- 4.14 X - - - - - - - - -

l3mplsvpn X - - - - - - - - -

loopbackbasic X - - - - - - - - -

1. Visual Studio code will be connected already with NSO-01, the window will be open and corresponding to the nso running directory. That way you do not need to change or create files in the terminal of the NSO-01 though through Visual Studio Code.
2. So that you obtain the latest code from lab, you need to pull the repository that is under /home/cisco directory.

nsoadmin@nso1:$ cd /home/cisco/CLEMEAR20\_TECNMS-4175

nsoadmin@nso1:/home/cisco/CLEMEAR20\_TECNMS-4175$ git pull

Task 1: Configure build-in High Availability

There are three different ways that High-Availability can be used:

1. Build-in HA
2. Using Tailf-hcc package and BGP capabilities
3. HA-RAFT

In this lab, you will configure the build-in HA. This type of High-Availability is used when NSO-Primary and NSO-Secondary lay in the same subnet, in other words, when they belong to the same network.

On the contrary, using High-Availability with Tailf-hcc package is used when you want to succeed High-Availability with NSOs that lay either on different networks in the same area either on different networks in different areas (Geo HA). In order to use it, you need to enable and configure BGP and a Virtual IP (VIP) that will be the bash of communication between the two NSO instances.

Lastly, HA-RAFT was introduced in 6.x.x version of NSO and is supporting further with auto-recovery and fully resolving the issues of the Split Brain. In order to configure it, you need more that 2 NSO instances, so far the recommended Cisco approach refers to using 3 NSO instances.

During High-Availability, a NSO node can be in three different modes:

1. Primary
2. Secondary
3. None

None is supporting so that there are no additional commits permitted while Primary is Down, that way we are not facing any longer problems of Split Brain.

Another important note is that while configuring the high-availability token, in local install the token between Primary and Secondary might be different. In system install they should be the **same**, otherwise Primary and Secondary will not be able to communicate and form a cluster.

More information regarding configuration of High-Availability of NSO 6.1.5: <https://developer.cisco.com/docs/nso-guides-6.1/#!high-availability> .

Step 1: Verifications before configuring High Availability

Let’s start exploring how to configure build-in high-availability in NSO 6.1.5. Before starting with high-availability configuration. There are some verifications needed:

1. Connect to NSO-01 CLI and verify that high-availability is not enable:
2. Verify the same for NSO-01
3. Verify that both NSOs are having same packages:
4. Verify netsim devices are up and running in NSO-01
5. Verify there are no services configured

Step 2: Enable High Availability

1. Connect to NSO Host 1 and navigate to /etc/ncs directory. There you will find the configuration files of NSO.

cd /etc/ncs

ls

1. Use vi to navigate in ncs.conf file and find ha field:

vi ncs.conf

/<ha>

1. Enable High-Availability by changing value in true
2. Replicate the same for NSO-02
3. Restart NSO-01 and NSO-02
4. Configure High-Availability to NSO-01, that NSO will act as Primary
5. Configure High-Availability to NSO-02, that NSO will act as Secondary
6. Enable High-Availability in NSO-01
7. Enable High-Availability in NSO-02
8. Verify High-Availability is up and running and Primary with Secondary has formed a cluster.

Step 3: Verify High-Availability

1. Enter configuration mode in NSO-02

You should not be able to enter since right now NSO-02 is working as Secondary and has read only abilities.

1. Verify in NSO-02 that there is no loopback interface configured in PE\_00
2. Go back to NSO-01 and configure device PE\_00 with a loopback interface:
3. Verify that loopback interface is configured for PE\_00 in both NSOs:
4. Now configure service loopback through NSO-01 and verify the same will be visible in NSO-02:
5. It is time to bring down NSO-01 and see NSO-02 taking over, stop NSO-01
6. Verify that NSO-02 has taken over
7. Bring back NSO-01 and revert roles :
8. Verify High-Availability is working between the two nodes.
9. To conclude disable HA again from both by changing ncs.conf file and restart both NSOs:

Task 2: Layered Services Architecture

Stacked services design conceptually consists of several services that are divided into customer-facing and resource-facing services. The approach to service development is almost the same as with traditional service development. Now, however, you need to consider multiple services being stacked and integrated together, to present themselves as one large service.

There are two types of services for working with stacked services design. CFSs and resource-facing services (RFSs) are identified as two types of services that are implemented together to achieve service stacking. The customer-facing service is assigned to a customer and can consist of several RFSs, as shown in the following figure.

A diagram of a diagram of a company

Description automatically generated

Figure 3 : Layered Service Architecture – CFS & RFS

Following the figure above, NSO-01 with be responsible for the CFS to RFS information mapping and later on RFS will retrieve that information and will be responsible for the RFS to device configuration mapping.

More information regarding Layered Service Architecture : <https://developer.cisco.com/docs/nso-guides-6.1/#!lsa-overview>

The aim of the section is to develop a stacked service called l2vpn-qos.

In this task, the goal is from network service perspective:

* Configure a QoS policy with one class – default – and defined average bit rate shaping. Apply the QoS policy to a GigabitEthernet interface.
* Create an L2VPN VPWS tunnel, for this we will reuse the previously created l2vpn package.

A screenshot of a computer

Description automatically generated

Figure 4: Task 2 Service Architecture

Step 1: Create QoS Template Service Package

1. Get to packages directory in NSO-01:

nsoadmin@nso1:$ cd /var/opt/ncs/packages

nsoadmin@nso1:/var/opt/ncs/packages$

1. Generate a package with skeleton template-based only:

nsoadmin@nso1:/var/opt/ncs/packages$ ncs-make-package –-service-skeleton template qos

1. Start creating the YANG model by using the following instructions bellow, the YANG model will be in:

nsoadmin@nso1:$cd /var/opt/ncs/packages/qos/src

nsoadmin@nso1: /var/opt/ncs/packages/qos/src$ ls

Makefile yang

nrsoadmin@nso1:/var/opt/ncs/packages/qos/src$ cd yang

**Table 1:** Lab first aid

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | Parameter type | Restrictions | mandatory |
| qos | List | No | - |
| Service-id | String | (key to qos list) | - |
| Ce-devices | List | 2 devices always | - |
| Lan-ge-interface | String | - | - |
| Policy-name | String | - | True |
| Average-bit-rate | Uint32 | Multiple of 8000 | True |

module qos {

namespace "http://com/example/qos";

prefix qos;

import tailf-common {

prefix tailf;

}

import ietf-inet-types {

prefix inet;

}

import tailf-ncs {

prefix ncs;

}

description

"TECOPS-2201: QoS service";

revision 2024-02-05 {

description

"Initial revision.";

}

list qos {

uses ncs:service-data;

// the keyword ncs:servicepoint is used to link a yang model to a configuration template or service code

// the service is template-based only. Thus, you will find inside qos-template.xml a link to servicepoint "qos"

ncs:servicepoint "qos";

// service-id identifies the qos service instance

key service-id;

leaf service-id {

// thanks to tailf:info, "Unique service id" description will be printed in the NSO CLI

tailf:info "Unique service id";

tailf:cli-allow-range;

type string;

}

// Exactly 2 devices can be configured by one service instance.

// The minimum number of devices to configure is defined by "min-elements".

// The maximum number of devices to configure is defined by "max-elements".

list ce-devices {

min-elements 2;

max-elements 2;

key ce-device;

tailf:info "device name";

leaf ce-device {

type leafref {

// reference to the list of registered devices in NSO

// in the path the usage of prefix "ncs" is mandatory because the elements "ncs:devices", "ncs:device" and "ncs:name" are not part of the local yang module

path "/ncs:devices/ncs:device/ncs:name";

}

}

// The LAN facing interface that the policy-map will be applied to

leaf lan-ge-interface {

type string;

}

}

// the policy-name to be configured on the devices in the leaf-list

leaf policy-name {

tailf:info "Policy name";

// "mandatory true" means that this leaf must be filled. Otherwise NSO won't accept the service instantiation

mandatory true;

type string;

}

leaf average-bit-rate {

tailf:info "Target Bit Rate (bits per second)";

mandatory true;

type uint32 {

// the predefined type unint32 is locally refined

range "8000..max";

}

}

}

}

1. Define the qos configuration template. Template will be located:

nsoadmin@nso1:$ cd /var/opt/ncs/packages/qos/templates

nsoadmin@nso1: var/opt/ncs/packages/qos/templates$ ls

qos-template.xml

<!-- Here we find the same servicepoint like in the qos yang file-->

<config-template xmlns="http://tail-f.com/ns/config/1.0" servicepoint="qos">

<devices xmlns="http://tail-f.com/ns/ncs">

<?foreach {/ce-devices}?>

<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>{ce-device}</name>

<config>

<!-- With namespace "urn:ios" we explicitly define which YANG model NED to be used to configure the equipment -->

<policy-map xmlns="urn:ios">

<!-- Here we specify a variable which will be filled directly from the YANG model -->

<name>{/policy-name}</name>

<class-default>

<class>

<!-- A static parameter -->

<name>class-default</name>

<shape>

<average>

<bit-rate>{/average-bit-rate}</bit-rate>

</average>

</shape>

</class>

</class-default>

</policy-map>

<interface xmlns="urn:ios">

<GigabitEthernet>

<name>{lan-ge-interface}</name>

<service-policy>

<output>{/policy-name}</output>

</service-policy>

</GigabitEthernet>

</interface>

</config>

</device>

<?end?>

</devices>

</config-template>

1. Compile the package:

nsoadmin@nso1:$ cd /var/opt/ncs/packages/qos/src

nsoadmin@nso1: /var/opt/ncs/packages/qos/src$ make clean all

rm -rf ../load-dir  
mkdir -p ../load-dir  
/opt/ncs/ncs-5.7.11/bin/ncsc `ls qos-ann.yang > /dev/null 2>&1 && echo "-a qos-ann.yang"` \

--fail-on-warnings \ \

-c -o ../load-dir/qos.fxs yang/qos.yang

1. Perform packages reload to onboard package to NSO:

nsoadmin@nso1: /var/opt/ncs/packages/qos/src$ ncs\_cli -Cu nsoadmin

nsoadmin@nso1# packages reload

>>> System upgrade is starting.

>>> SessdiColnosud:iTnhecCoinscfoiDguemreo mode must exit to operational mode.

>>> No configuration changes can be performed until upgrade has completed.

>>> System upgrade has completed successfully.

1. Configure a qos service:

admin@ncs#

admin@ncs# **config**

Entering configuration mode terminal

admin@ncs(config)# qos test average-bit-rate 500000 policy-name test-policy ce-devices PE\_00 lan-ge-interface 0/3

admin@ncs(config-ce-devices-CE-A\_Site\_1)# exit admin@ncs(config-qos-CUST-PMAP-S-L2ETH)# ce-devices PE\_10 lan-ge-interface 0/3

nsoadmin@nso1(config-ce-devices-CE-A\_Site\_2)# top  
nsoadmin@nso1 (config)# show configuration  
qos test

ce-devices PE\_00 lan-ge-interface 0/3

!  
ce-devices PE\_10

lan-ge-interface 0/3 !

policy-name test-policy

average-bit-rate 500000

!

nsoadmin@nso1 (config)# **commit**

Commit complete.

1. Delete the created QoS service instance

nsoadmin@nso1(config)# no qos test

nsoadmin@nso1(config)# commit dry-run outformat native native {

device {  
name PE\_00

data interface GigabitEthernet0/3  
no service-policy output test-policy

exit  
no policy-map test-policy

}  
device {

name PE\_10  
data interface GigabitEthernet0/3

no service-policy output test

exit

no policy-map test-policy

nsoadmin@nso1(config)# commit

Step 2: Create l2vpn Template Service Package

Step 3: Create l2vpn-qos Python and Template Service Package

Task 3: Kickers and Subscribers

Task 4: Nano-services

**Table:** Lorem ipsum dolor sit amet

|  |  |  |  |
| --- | --- | --- | --- |
| Lorem ipsum | Lorem ipsum | Lorem ipsum | Lorem ipsum |
| Lorem ipsum | 1 | Lorem ipsum | Lorem ipsum |
| Lorem ipsum | 2 | Lorem ipsum | Lorem ipsum |
| Lorem ipsum | 3 | Lorem ipsum | Lorem ipsum |
| Lorem ipsum | 4 | Lorem ipsum | Lorem ipsum |

Summary

Deliver high-quality services faster and more easily through network automation. Cisco Network Services Orchestrator (NSO) is industry-leading software for automating services across traditional and virtualized networks. Use NSO to add, change, and delete services without disrupting overall service, and help ensure that services are delivered in real time.

NSO is now [free to download](https://developer.cisco.com/site/nso/?utm_campaign=nso19&utm_source=website&utm_medium=nso-try) for non-production use! Download NSO to evaluate and learn how to automate your network and orchestrate your services using NETCONF and YANG today. Also do not miss the opportunity to practice more through : <https://developer.cisco.com/site/nso/> , there you can find lab guides, sandbox labs and learning tracks to support you in your automation journey.

[Graphical user interface, text, application, chat or text message

Description automatically generated](https://www.cisco.com/c/en/us/solutions/service-provider/solutions-cloud-providers/network-services-orchestrator-solutions/business-benefits-of-automation-and-orchestration.html)

Ref.: <https://www.cisco.com/c/en/us/solutions/service-provider/solutions-cloud-providers/network-services-orchestrator-solutions.html>

Appendix A

Use Makefile to configure netsim devices

If netsim devices are not up and running, there is a Makefile create that will support you with netsim creation, netsim loading and netsim deletion.

cd /var/opt/ncs

ls

cat Makefile

make rebuild-netsim

Appendix B

Start NSO and verify status

1. Start NSO in system install:

nsoadmin@nso1:$/etc/init.d/ncs start

1. Verify NSO status:

nsoadmin@nso1:~/var/opt/ncs$ **ncs --status**

vsn: 6.1.5

SMP support: yes, using 4 threads

Using epoll: yes

available modules: backplane,netconf,cdb,cli,snmp,webui

running modules: backplane,netconf,cdb,cli,snmp,webui

status: started

…

Or

/etc/init.d/ncs status

1. Check the ncs.conf file for default NSO configuration parameters:

nsoadmin@nso1:$ cd /etc/ncs

nsoadmin@nso1:$ **cat ncs.conf**

<!-- -\*- nxml -\*- -->

<!-- Example configuration file for ncs. -->

<ncs-config xmlns="http://tail-f.com/yang/tailf-ncs-config">

<!-- NCS can be configured to restrict access for incoming connections -->

<!-- to the IPC listener sockets. The access check requires that -->

<!-- connecting clients prove possession of a shared secret. -->

<ncs-ipc-access-check>

<enabled>false</enabled>

<filename>${NCS\_DIR}/etc/ncs/ipc\_access</filename>

</ncs-ipc-access-check>

<!-- Where to look for .fxs and snmp .bin files to load -->

…

1. Login to NSO CLI Juniper mode (default password for user admin is admin):

nsoadmin@nso1:~/var/opt/ncs$ **ncs\_cli -u nsoadmin**

admin connected from 192.168.234.3 using ssh on nso-host

nsoadmin@nso1>

OR

nsoadmin@nso1:/var/opt/ncs$ **ssh -l nsoadmin -p 2024 localhost**

The authenticity of host '[localhost]:2024 ([127.0.0.1]:2024)' can't be established.

RSA key fingerprint is SHA256:ZLWvfBSWDj4yqS1a68ZpTT4nTVsrrCC8CVTBlDPJuO0.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[localhost]:2024' (RSA) to the list of known hosts.

Nsoadmin@nso1 password:

nsoadmin connected from 127.0.0.1 using ssh on nso-host

nsoadmin@nso1>

1. Switch to Cisco style CLI:

nsoadmin@nso1> **switch cli**

nsoadmin@nso1# << Cisco style CLI>>

nsoadmin@nso1#

nsoadmin@nso1#

nsoadmin@nso1# **switch cli**

[ok][2019-11-27 05:38:02]

nsoadmin@nso1> << Juniper Style>>

nsoadmin@nso1>

nsoadmin@nso1>

OR

## Cisco Style CLI

nsoadmin@cisco:/var/opt/ncs$ ncs\_cli -u nsoadmin -C

nsoadmin connected from 192.168.234.3 using ssh on nso-host

nsoadmin@ncs#

## Juniper Style CLI

nsoadmin@cisco:/var/opt/ncs$ ncs\_cli -u nsoadmin -J

nsoadmin connected from 192.168.234.3 using ssh on nso-host

nsoadmin@ncs>

1. Explore different show commands from NSO CLI

# show packages packages package-version

# show devices list

# show running-config

1. Observe NSO startup process through different logs: ncs-java-vm.log

nsoadmin@cisco:/var/opt/ncs$

nsoadmin@cisco:/var/opt/ncs$ cd /var/log/ncs

nsoadmin@cisco:/var/log/ncs$ ls -lrt

total 240

-rw-rw-r-- 1 developer developer 0 Nov 27 05:30 netconf.log

-rw-rw-r-- 1 developer developer 0 Nov 27 05:30 snmp.log

-rw-rw-r-- 1 developer developer 13 Nov 27 05:30 ncserr.log.siz

-rw-rw-r-- 1 developer developer 18 Nov 27 05:30 ncserr.log.idx

-rw-rw-r-- 1 developer developer 8 Nov 27 05:30 ncserr.log.1

-rw-rw-r-- 1 developer developer 0 Nov 27 05:31 **ncs-python-vm.log**

-rw-rw-r-- 1 developer developer 826 Nov 27 05:31 rollback10001

-rw-rw-r-- 1 developer developer 0 Nov 27 05:31 localhost:8080.access

-rw-rw-r-- 1 developer developer 7274 Nov 27 05:31 **ncs-java-vm.log**

-rw-rw-r-- 1 developer developer 21401 Nov 27 05:32 ncs.log

-rw-rw-r-- 1 developer developer 53451 Nov 27 05:40 xpath.trace

-rw-rw-r-- 1 developer developer 127736 Nov 27 05:40 devel.log

-rw-rw-r-- 1 developer developer 6097 Nov 27 05:40 audit.log

developer@nso-host:~/nso521/ncs-run-521/logs$

For example if you want to observe the steps of High-Availability and any issues in the establishment of it, you can use devel.log , it will provide with information regarding communication between Primary and Secondary.

Related Sessions at Ciscolive

You can search CiscoLive Amsterdam content catalog with specific keyword and recommend sessions that are relevant to your lab.

[Content Catalog Link](https://www.ciscolive.com/emea/learn/session-catalog.html?search=nso#/)

1. Embracing DevOps for my NSO Use Cases lifecycle – [DEVNET-2224], Alfonso Sandoval Rosas, Software Consulting Engineer, Cisco.
2. Automating Services with NSO – [LABOPS-1507], Spyridon Spyriadis, Software Consulting Engineer, Sofia Athanasiou, Customer Success Specialist, Cisco.
3. Advanced YANG Data Modeling for Cisco NSO – [DEVNET-3014], Bartosz Luraniec, Customer Delivery Software Architect, Cisco.
4. Automatic Services with Cisco NSO and Model-Driven Telemetry – [DEVWKS-3230] – Bartosz Luraniec, Customer Delivery Software Architect, Cisco.
5. Real-time Services Automation with NSO and Model-Driven Telemetry – [LABOPS-1305], Spyridon Spyriadis, Software Consulting Engineer, Sofia Athanasiou, Customer Success Specialist, Cisco.