Discovery 9: Create an SVI Python-Template Service

Introduction

In this activity, you will learn how to create a Python template service. After completing this activity, you will be able to meet these objectives:

- Define the service model in YANG
- Create feature templates
- Write Python mapping code
- Compile and deploy the service
- Troubleshoot a Python script

Required Resources

The following resources and equipment are required for completing the activities in this lab guide:

- PC or laptop with a web browser
- Access to the Internet

Command Syntax Reference

This lab guide uses the following conventions for command syntax:

Formatting	Description and Examples
show running config	Commands in steps use this formatting.

Formatting	Description and Examples
Example	Type **show running config**
Example	Use the **name** command.
`show running config`	Commands in CLI outputs and configurations use this formatting.
highlight	CLI output that is important is highlighted.
Example	`student@student-vm:~$ ncs -version` `5.8.2.1`
Example	Save your current configuration as the default **startup config**. `Router Name# copy running startup`
brackets ([ ])	Indicates the optional element. You can choose one of the options.
Example:	`(config-if)# frame-relay lmi-type {ansi|cisco|q933a}`
italics font	Arguments for which you supply values.
Example	Open file **ip tcp window-size** *bytes*
angle brackets (<>)	In contexts that do not allow italics, arguments for which you supply values are enclosed in angle brackets [<>]. Do not type the brackets when entering the command.
Example	If the command syntax is **ping** *<ip_address>*, you enter ping *192.32.10.12*
string	A nonquoted set of characters. Type the characters as-is.
Example	(config)# **hostname MyRouter**
vertical line (|)	Indicates that you enter one of the choices. The vertical line separates choices. Do not type the vertical line when entering the command.
Example	If the command syntax is **show ip route|arp**, you enter either **show ip route** or **show ip arp**, but not both.

Command List

The following are the most common commands that you will need:

Linux Shell:

Command	Comment
source /home/student/ nso-5.8/ncsrc	Source NSO environmental variables.
ls|ll	Display contents of the current directory.
cd	Move directly to user home directory.
cd ..	Exit the current directory.
cd test	Move into folder "test," which is a subfolder of the current directory.
cd /home/student/test	Move into folder "test" by specifying direct path to it, starting from the root of directory system.
ncs_cli -Cu admin	Log in to NSO CLI directly from local server.

NSO CLI:

Command	Comment
switch cli	Change CLI style.
show ?	Display all command options for current mode.
configure	Enter configuration mode.
commit	Commit new configuration (configuration mode only command).
show configuration	Display new configuration that has not yet been committed (configuration mode only command).

Job Aids

The following job aid is available to help you complete the lab activities:

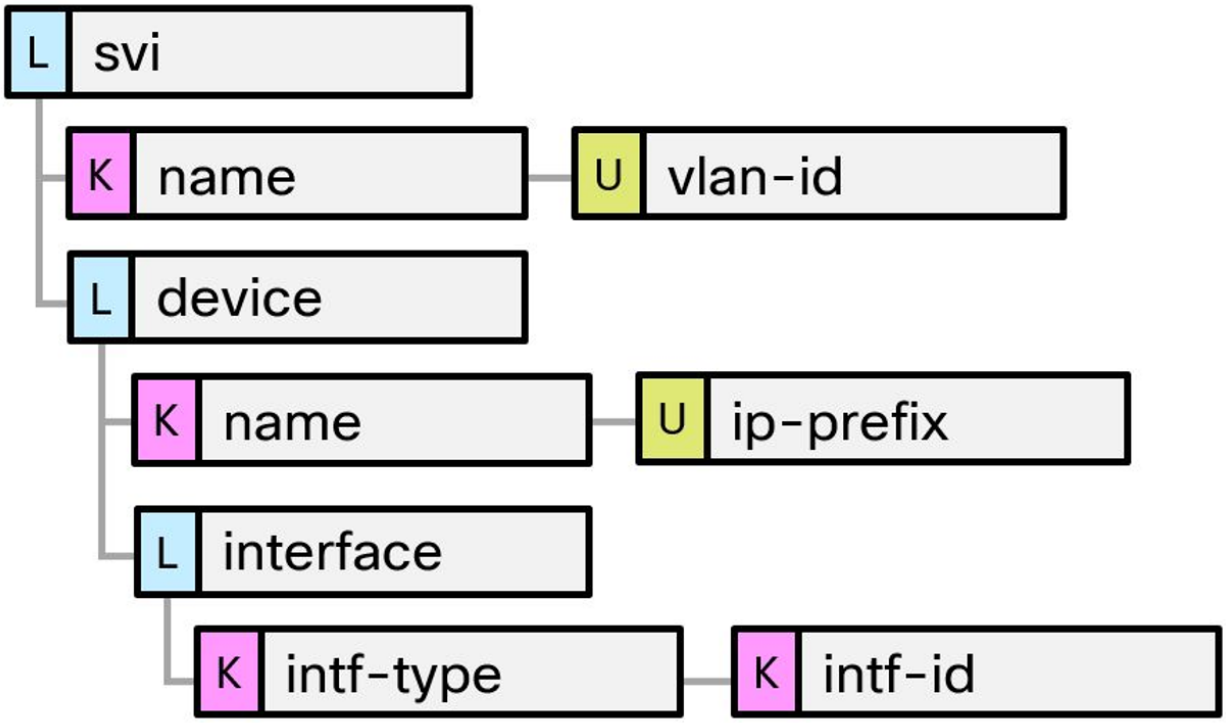
- This lab guide
- Student guide, for general explanations

The following table contains passwords that you might need.

Device	User Name	Password
Student-VM	student	1234QWer
NSO application	admin	admin

Job Aids for Task 1

The following data will be required for this task.



The following table describes the service characteristics and requirements. Use this information as a starting point for creating a service model.

Attribute	Name	Data Type	Restriction	Other
Service name	*svi*	list	—	—
Service instance name	*name*	string	—	key for list svi
VLAN ID	*vlan-id*	uint32	Range: 1 – 4096	unique value
Device	*device*	list	—	—
Device name	*name*	leafref	—	key for list device
IP prefix	*ip-prefix*	inet:ip-prefix	—	—
Interface type	*intf-type*	enumeration	Options: FastEthernet GigabitEthernet Ethernet	intf-type and intf-id combination must be unique
Interface ID	*intf-id*	string	—	intf-type and intf-id combination must be unique

Job Aids for Task 2

A network engineer has provided you with the actual configuration for the new service, one for each device type.

Cisco IOS platform (device SW31):

```
interface Vlan 88
```

```
  ip address 10.172.19.88 255.255.255.0
!
```

Cisco NX-OS platform (device SW32):

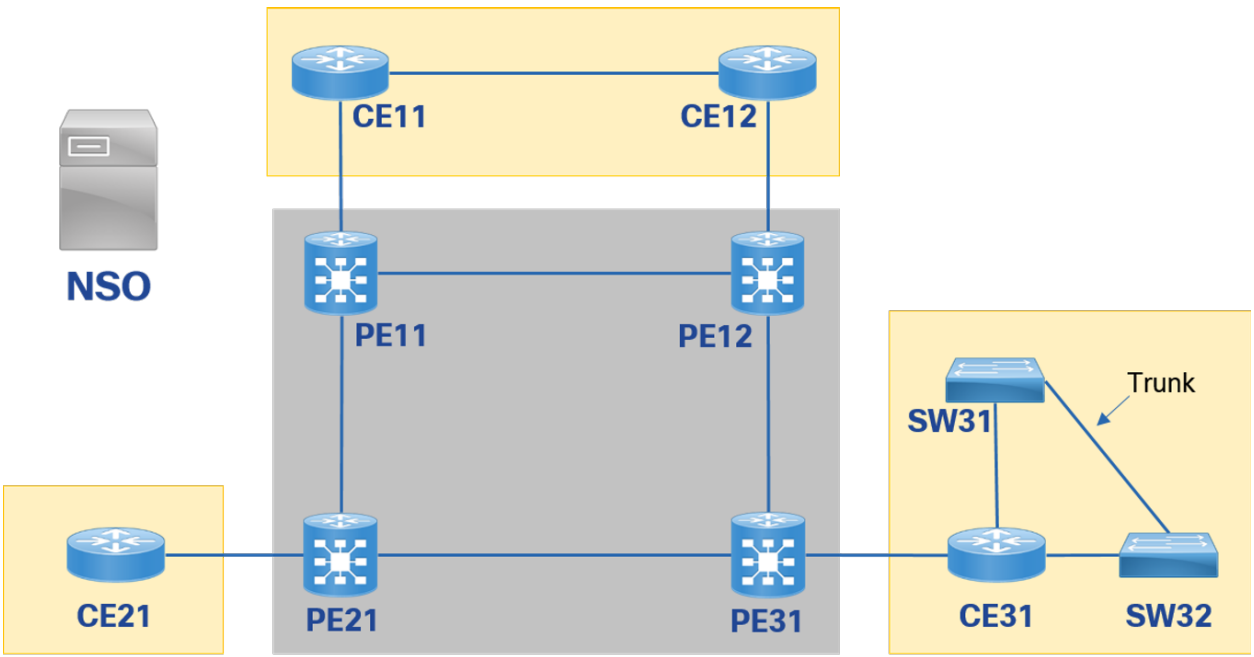
```
interface Vethernet 168
  ip address 10.168.23.66/24
!
```

Lab Topology Information

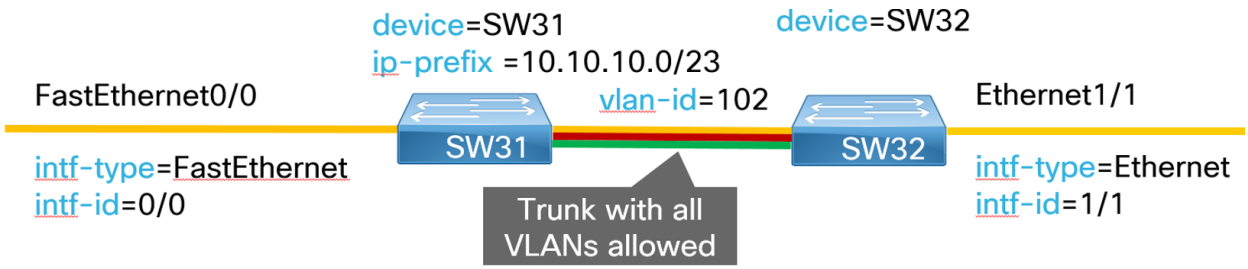
Your lab session is your own personal sandbox. Whatever you do in your session will not be reflected in anyone else's session. There are two topologies. The general one is your lab environment, and it has your workstation and a Linux server. On the Linux server within that topology is your second topology with your NSO installation, together with numerous netsim routers and switches that are logically grouped into a network topology. This network will be the one that you will orchestrate with your NSO.

- The network topology is designed to cover both the service provider and enterprise use cases. It is a simulated netsim network; devices have no control or data plane. Devices will, however, accept or reject a configuration sent by the NSO, just as real devices would.

Topology



Visual Objective



The graphic above provides a visual aid for this activity.

Task 1: Design a Service Model

The first task requires you to design a service model based on the provided high-level service requirements.

> The final solutions for all labs, including this one, are located in the ~/solutions directory. You can use them for copy-pasting longer pieces of code and as a reference point for troubleshooting your packages.

Activity

Complete these steps:

Step 1

Connect to the NSO server by clicking the NSO icon.

Step 2

Open the terminal window using the Terminal icon on the bottom bar.

```
student@student-vm:~$
```

Step 3

Change your current location to the directory **$HOME/nso-run/packages**.

```
student@student-vm:~$ cd $HOME/nso-run/packages
```

Step 4

Create a template-based service skeleton by using the **ncs-make-package** command. Use **svi** as the name of the new service.

> You can use **ncs-make-package --help** or **man ncs-make-package** to learn about all available command options.

```
student@student-vm:~/nso-run/packages$ ls
cisco-ios-cli-6.85  cisco-iosxr-cli-7.41  cisco-nx-cli-5.23  l3vpn
loopback  vlan
student@student-vm:~/nso-run/packages$ ncs-make-package --service-
skeleton python-and-template svi
student@student-vm:~/nso-run/packages$ ls
cisco-ios-cli-6.85  cisco-iosxr-cli-7.41  cisco-nx-cli-5.23  l3vpn
loopback  svi  vlan
student@student-vm:~/nso-run/packages$
```

Step 5

Navigate to the *svi/src/yang* directory and open the *svi.yang* file to display the YANG service model of the svi service.

This is how the file should look when you open it for the first time:

```
student@student-vm:~/nso-run/packages$ cd svi/src/yang/
student@student-vm:~/nso-run/packages/svi/src/yang$ cat svi.yang
module svi {

  namespace "http://example.com/svi";
  prefix svi;

  import ietf-inet-types {
    prefix inet;
  }
  import tailf-common {
    prefix tailf;
  }
  import tailf-ncs {
    prefix ncs;
  }

  description
    "Bla bla...";

  revision 2016-01-01 {
    description
      "Initial revision.";
  }

  list svi {
    description "This is an RFS skeleton service";

    key name;
    leaf name {
      tailf:info "Unique service id";
      tailf:cli-allow-range;
      type string;
    }

    uses ncs:service-data;
    ncs:servicepoint svi-servicepoint;
```

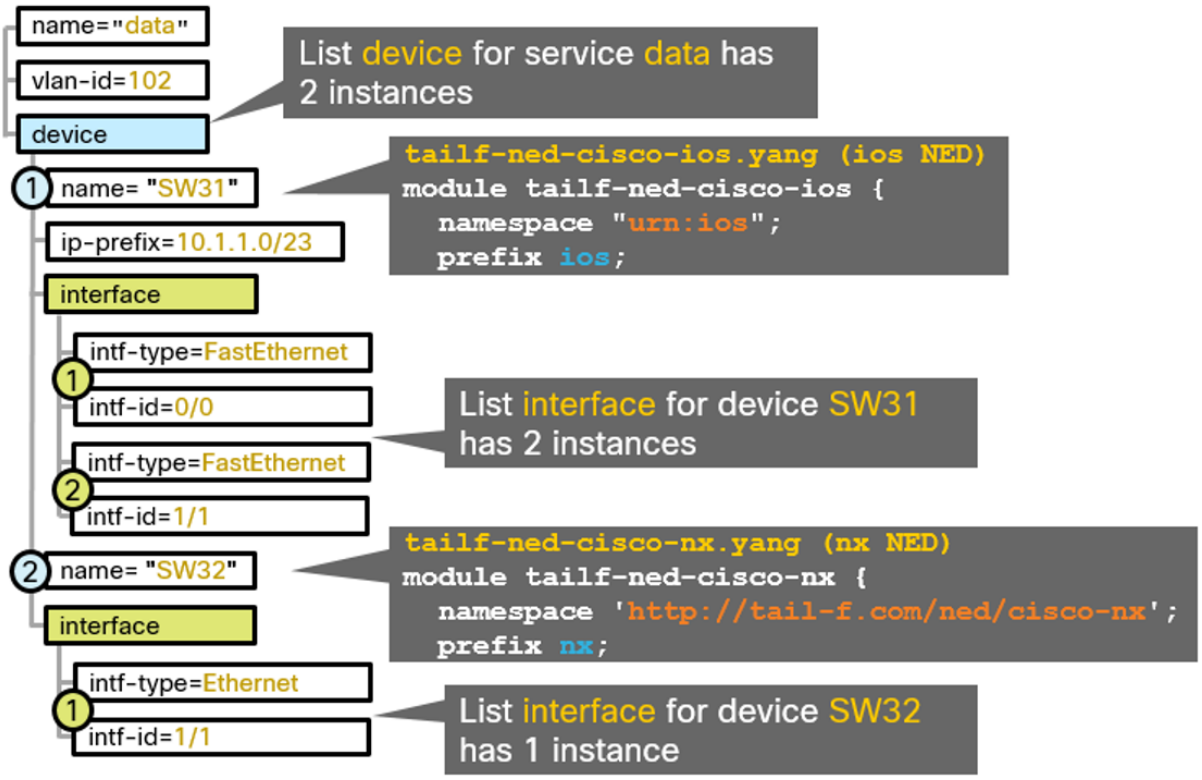
```
    // may replace this with other ways of refering to the devices.
    leaf-list device {
      type leafref {
        path "/ncs:devices/ncs:device/ncs:name";
      }
    }

    // replace with your own stuff here
    leaf dummy {
      type inet:ipv4-address;
    }
  }
}
```

Step 6

Use the following table to fine-tune the service attributes that you will use in your YANG service model.

Attribute	**Name**	**Data Type**	**Restriction**	**Other**
Service name	*svi*	list	—	—
Service instance name	*name*	string	—	key for list svi
VLAN ID	*vlan-id*	uint32	Range: 1 – 4096	unique value
Device	*device*	list	—	—
Device name	*name*	leafref	—	key for list device
IP prefix	*ip-prefix*	inet:ip-prefix	—	—
Interface type	*intf-type*	enumeration	Options: FastEthernet GigabitEthernet Ethernet	intf-type and intf-id combination must be unique
Interface ID	*intf-id*	string	—	intf-type and intf-id combination must be unique



Step 7

Open the YANG service model for editing with the **code svi.yang** command.

```
student@student-vm:~/nso-run/packages/svi/src/yang$ code svi.yang
```


You can edit the YANG service model file by using Visual Studio Code or any editor of your choice. Visual Studio Code text editor on the workstation will provide you with syntax highlighting for YANG.

Step 8

Rename the namespace to *http://cisco.com/examples/svi*, remove the leaf-list device and the leaf dummy. Those will not be used for the SVI service and will be replaced by other statements. Add an **augment /ncs:services** statement, to include the list SVI in the services branch of the CDB. You can also add a custom description and revision information.

```
module svi {
  namespace "http://cisco.com/examples/svi";
  prefix svi;

  import ietf-inet-types {
    prefix inet;
  }
  import tailf-common {
    prefix tailf;
```

```
  }
  import tailf-ncs {
    prefix ncs;
  }

  description "This is SVI Python service package.";

  revision 2022-09-01 {
    description
      "Initial revision.";
  }

  augment /ncs:services {
    list svi {
      tailf:info "SVI Python Service";

      key name;
      leaf name {
        tailf:info "Unique service id";
        tailf:cli-allow-range;
        type string;
      }

      uses ncs:service-data;
      ncs:servicepoint svi-servicepoint;

    }
  }
}
```


YANG allows a module to insert additional nodes into existing data models. The "augment" statement defines the location in the data model hierarchy where new nodes are inserted. In your case, the location is under the **services** branch of the CDB.

Step 9

Assign a VLAN ID to each SVI. Use the standard VLAN range from 1 to 4096.

```
module svi {

  namespace "http://cisco.com/examples/svi";
  prefix svi;

  import ietf-inet-types {
    prefix inet;
  }
  import tailf-common {
    prefix tailf;
  }
  import tailf-ncs {
    prefix ncs;
  }
```

```
  description "This is SVI Python service package.";

  revision 2022-09-01 {
    description
      "Initial revision.";
  }

  augment /ncs:services {
    list svi {
      tailf:info "SVI Python Service";

      key name;
      leaf name {
        tailf:info "Unique service id";
        tailf:cli-allow-range;
        type string;
      }

      uses ncs:service-data;
      ncs:servicepoint svi-servicepoint;

      leaf vlan-id {
        tailf:info "Unique VLAN ID";
        type uint32 {
          range "1..4096";
        }
      }

    }

  }
}
```

Step 10

Assign a list of devices to each SVI instance.

The list device is used, and the leaf name is a leafref that points to the existing device in the devices list of the CDB. Add an interface list, that represents interfaces of the selected VLAN ID. Each interface can be one of the three available types, and it must have a unique identifier (intf-id).

You can reuse the model structure from the **vlan** service model. You can open the **vlan** service model file by using the **code $HOME/nso-run/packages/vlan/src/yang/vlan.yang** command and copy the *list device* section of the model.

```
module svi {
  namespace "http://cisco.com/examples/svi";
  prefix svi;

  import ietf-inet-types {
    prefix inet;
```

```
  }
  import tailf-common {
    prefix tailf;
  }
  import tailf-ncs {
    prefix ncs;
  }

  description "This is SVI Python service package.";

  revision 2022-09-01 {
    description
      "Initial revision.";
  }

  augment /ncs:services {
    list svi {
      tailf:info "SVI Python Service";

      key name;
      leaf name {
        tailf:info "Unique service id";
        tailf:cli-allow-range;
        type string;
      }

      uses ncs:service-data;
      ncs:servicepoint svi-servicepoint;

      leaf vlan-id {
        tailf:info "Unique VLAN ID";
        type uint32 {
          range "1..4096";
        }
      }

      list device {
        tailf:info "L3 Switch";
        key name;

        leaf name {
          tailf:info "Device Name";
          type leafref {
            path "/ncs:devices/ncs:device/ncs:name";
          }
        }

        list interface {
          tailf:info "Ethernet Interface";
          key "intf-type intf-id";

          leaf intf-type {
            tailf:info "Ethernet Interface Type";
            type enumeration {
              enum Ethernet;
              enum FastEthernet;
              enum GigabitEthernet;
            }
```

```
        }

        leaf intf-id {
          tailf:info "Ethernet Interface ID";
          type string;
        }
      }

    }

  }
}
```

Step 11

Add the ip-prefix leaf inside the device list structure. The ip-prefix leaf represents an IP network that will be used on the SVI.

```
module svi {

  ...

  augment /ncs:services {
    list svi {

      ...

      list device {
        tailf:info "L3 Switch";
        key name;

        leaf name {
          tailf:info "Device Name";
          type leafref {
            path "/ncs:devices/ncs:device/ncs:name";
          }
        }

        leaf ip-prefix {
          tailf:info "Unique IPv4 Prefix for VLAN";
          type inet:ip-prefix;
        }

        list interface {
          tailf:info "Ethernet Interface";
          key "intf-type intf-id";

          leaf intf-type {
            tailf:info "Ethernet Interface Type";
            type enumeration {
              enum Ethernet;
              enum FastEthernet;
              enum GigabitEthernet;
            }
```

```
            }

            leaf intf-id {
              tailf:info "Ethernet Interface ID";
              type string;
            }
          }

        }

      }

    }
}
```

Step 12

Save the file.

Step 13

You can validate your YANG file by using the **pyang** command. Keep making corrections to your YANG file until pyang no longer produces any output for svi.yang (that is until there are no errors).

```
tudent@student-vm:~/nso-run/packages/svi/src/yang$ pyang svi.yang
/home/student/nso-5.8/src/ncs/yang/tailf-ncs-cluster.yang:224: error:
unexpected keyword "default"
/home/student/nso-5.8/src/ncs/yang/tailf-ncs-cluster.yang:225: error:
unexpected keyword "default"
/home/student/nso-5.8/src/ncs/yang/tailf-ncs-cluster.yang:226: error:
unexpected keyword "default"
...
/home/student/nso-5.8/src/ncs/yang/tailf-ncs-packages.yang:1006:
warning: node "tailf-ncs::high-availability" is not found in module
"tailf-ncs-packages"
student@student-vm:~/nso-run/packages/svi/src/yang$
```


Pyang can show errors and warnings in its output. In the lab, only errors pertaining to **svi.yang** must to be fixed.

Step 14

Change the directory to the parent directory where the Makefile is located. Use the **make** command to compile the package.

```
student@student-vm:~/nso-run/packages/svi/src/yang$ cd ..
student@student-vm:~/nso-run/packages/svi/src$ make
mkdir -p ../load-dir
mkdir -p java/src//
/home/student/nso-5.8/bin/ncsc  `ls svi-ann.yang  > /dev/null 2>&1 &&
echo "-a svi-ann.yang"` \
```

```
              -c -o ../load-dir/svi.fxs yang/svi.yang
student@student-vm:~/nso-run/packages/svi/src$
```

Step 15

After the package is compiled, log in to NSO CLI and issue the **packages reload** command.

```
student@student-vm:~/nso-run/packages/svi/src$ ncs_cli -Cu admin
admin@ncs# packages reload

>>> 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-ios-cli-6.85
    result true
}
reload-result {
    package cisco-iosxr-cli-7.41
    result true
}
reload-result {
    package cisco-nx-cli-5.23
    result true
}
reload-result {
    package l3vpn
    result true
}
reload-result {
    package loopback
    result true
}
reload-result {
    package svi
    result true
}
reload-result {
    package vlan
    result true
}
admin@ncs#
System message at 2022-09-18 12:40:53...
    Subsystem stopped: ncs-dp-6-cisco-nx-cli-5.23:NexusDp
admin@ncs#
System message at 2022-09-18 12:40:53...
    Subsystem stopped: ncs-dp-5-cisco-ios-cli-6.85:IOSDp
admin@ncs#
System message at 2022-09-18 12:40:53...
    Subsystem started: ncs-dp-7-cisco-ios-cli-6.85:IOSDp
admin@ncs#
System message at 2022-09-18 12:40:53...
    Subsystem started: ncs-dp-8-cisco-nx-cli-5.23:NexusDp
```

```
admin@ncs#
```

Step 16

Check the availability of CLI commands to manage the service. For example, use the **services svi** command, followed by a question mark, to investigate the new CLI options that are available for the provisioning of service instances.

```
admin@ncs# services ?
Possible completions:
  check-sync   Check if device configuration is according to the
services
  l3vpn        L3VPN Service
  loopback     Loopback Service
  svi          SVI Python Service
  vlan         VLAN Service
admin@ncs# services svi ?
% No entries found
admin@ncs#
```

Step 17

Exit NSO CLI. Verify the directory structure and file templates for a new service.

```
admin@ncs# exit
student@student-vm:~/nso-run/packages/svi/src$ cd $HOME/nso-run/
packages
student@student-vm:~/nso-run/packages$ ls
cisco-ios-cli-6.85  cisco-iosxr-cli-7.41  cisco-nx-cli-5.23  l3vpn
loopback  svi  vlan
student@student-vm:~/nso-run/packages$ ls -l svi/
total 28
drwxrwxr-x 2 student student 4096 Sep 18 11:35 load-dir
-rw-rw-r-- 1 student student  374 Sep 18 10:44 package-meta-data.xml
drwxr-xr-x 3 student student 4096 Sep 18 10:44 python
-rw-rw-r-- 1 student student  742 Sep 18 10:44 README
drwxr-xr-x 4 student student 4096 Sep 18 11:35 src
drwxr-xr-x 2 student student 4096 Sep 18 10:44 templates
drwxr-xr-x 3 student student 4096 Jul  8 09:56 test
student@student-vm:~/nso-run/packages$ ls -l svi/src/yang/
total 4
-rw-rw-r-- 1 student student 1601 Sep 18 11:29 svi.yang
student@student-vm:~/nso-run/packages$ ls -l svi/templates/
total 4
-rw-r--r-- 1 student student 699 Jul  8 09:56 svi-template.xml
student@student-vm:~/nso-run/packages$ ls -l svi/python/
svi
student@student-vm:~/nso-run/packages$
```

Activity Verification

You have completed this task when you attain this result:

- You have a directory structure and file templates that matches the one displayed in the previous step.

You are still unable to provision services, because the service template is only a placeholder at this point. You will develop it properly in the next task.

Task 2: Create Feature Templates

In this task, you will configure the sample SVI service from NSO CLI to obtain the device-specific configuration in XML format. This will allow you to create feature templates for the Cisco IOS and Cisco NX-OS platforms. You will create two different templates - one to configure an interface and the other one to configure a VLAN. Using a template for each feature is considered a good practice, since it makes developing and modifying the service easier.

Activity

Complete these steps:

Step 1

Connect to NSO CLI.

```
student@student-vm:~/nso-run/packages$ ncs_cli -Cu admin
admin@ncs#
```

Step 2

A network engineer has provided you with the actual configuration for the new service, one for each device type.

Cisco IOS platform (device SW31):

```
interface Vlan 88
  ip address 10.172.19.88 255.255.255.0
!
```

Cisco NX-OS platform (device SW32):

```
interface Vethernet 168
  ip address 10.168.23.66/24
!
```


You can find the configuration for the new service in the Job Aids.

Step 3

Configure the preceding examples from NSO CLI.

Use the **devices device SW31** configuration command as a starting point to configure the Cisco IOS sample.

Use the **devices device SW32** configuration command as a starting point to configure the Cisco NX-OS sample.

```
admin@nso# config
Entering configuration mode terminal
admin@nso(config)# devices device SW31 config interface Vlan 88
admin@nso(config-if)# ip address 10.172.19.88 255.255.255.0
admin@nso(config-if)# top
admin@nso(config)# devices device SW32 config interface Vethernet 168
admin@nso(config-if)# ip address 10.168.23.66/24
admin@nso(config-if)# top
```

Step 4

Use the **commit dry-run outformat xml** command to retrieve the XML version of the configuration.

Verify that the output contains all the configured parameters. If some or all of the changes were committed earlier, the output will be missing those parts.

```
admin@ncs(config)# commit dry-run outformat xml
result-xml {
    local-node {
        data <devices xmlns="http://tail-f.com/ns/ncs">
               <device>
                 <name>SW31</name>
                 <config>
                   <interface xmlns="urn:ios">
                     <Vlan>
                       <name>88</name>
                       <ip>
                         <address>
                           <primary>
                             <address>10.172.19.88</address>
                             <mask>255.255.255.0</mask>
                           </primary>
                         </address>
                       </ip>
                     </Vlan>
                   </interface>
                 </config>
               </device>
               <device>
                 <name>SW32</name>
                 <config>
                   <interface xmlns="http://tail-f.com/ned/cisco-nx">
                     <Vethernet>
```

```
                        <name>168</name>
                        <ip>
                          <address>
                            <ipaddr>10.168.23.66/24</ipaddr>
                          </address>
                        </ip>
                      </Vethernet>
                    </interface>
                  </config>
                </device>
              </devices>
    }
}
admin@ncs(config)#
```

Step 5

Abort the configuration mode, exit NSO CLI and go to the templates subdirectory of your package skeleton (for example, $HOME/nso-run/packages/svi/templates).

```
admin@ncs(config)# abort
admin@ncs# exit
student@student-vm:~/nso-run/packages$ cd svi/templates
student@student-vm:~/nso-run/packages/svi/templates$
```

Step 6

Rename the template file.

```
student@student-vm:~/nso-run/packages/svi/templates$ mv svi-
template.xml svi-intf-template.xml
```

Step 7

Open the template by using the **code svi-intf-template.xml** command.

This is how the template should look when you open it for the first time:

```
<config-template xmlns="http://tail-f.com/ns/config/1.0">
  <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, java code sets a variable DUMMY, use
```

```
it
            to set something on the device e.g.:
            <ip-address-on-device>{$DUMMY}</ip-address-on-device>
        -->
      </config>
    </device>
  </devices>
</config-template>
```

Step 8

Because the service model supports two types of devices, the XML configuration changes segment must accommodate both. Remove the provided comments under the <device> and <config> tags and add your own (DEVICE, IOS, NX-OS).

```
<config-template xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <!-- DEVICE -->
    <device>
      <name>{/device}</name>
      <config>

        <!-- IOS -->

        <!-- NX-OS -->

      </config>
    </device>
  </devices>
</config-template>
```

Step 9

Insert the XML configuration created with **commit dry-run outformat xml** in the modified XML skeleton. Edit the segment with XML configuration code. Add both the Cisco IOS and NX-OS parts of the configuration. Remove provided comments under the **<device>** and **<config>** tags.

```
<config-template xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <!-- DEVICE -->
    <device>
      <name>{/device}</name>
      <config>

        <!-- IOS -->
        <interface xmlns="urn:ios">
          <Vlan>
            <name>88</name>
            <ip>
              <address>
                <primary>
                  <address>10.172.19.88</address>
```

```
                <mask>255.255.255.0</mask>
              </primary>
            </address>
          </ip>
        </Vlan>
      </interface>

      <!-- NX-OS -->
      <interface xmlns="http://tail-f.com/ned/cisco-nx">
        <Vethernet>
          <name>168</name>
          <ip>
            <address>
              <ipaddr>10.168.23.66/24</ipaddr>
            </address>
          </ip>
        </Vethernet>
      </interface>

    </config>
  </device>
 </devices>
</config-template>
```

Step 10

Replace all static parameters with variables, which reference service attributes according to the hierarchy of the YANG data model. Add the device name condition, which will apply this configuration only to one of the devices in the list.

```
<config-template xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <!-- DEVICE -->
    <device>
      <name>{/device/name}</name>
      <config>

        <!-- IOS -->
        <interface xmlns="urn:ios">
          <?if {string(name)=$SVI-DEVICE}?>
            <Vlan>
              <name>{$VLAN-ID}</name>
              <ip>
                <address>
                  <primary>
                    <address>{$IP-ADDR}</address>
                    <mask>{$NETMASK}</mask>
                  </primary>
                </address>
              </ip>
            </Vlan>
          <?end?>
        </interface>

        <!-- NX-OS -->
        <interface xmlns="http://tail-f.com/ned/cisco-nx">
```

```
          <?if {string(name)=$SVI-DEVICE}?>
            <Vethernet>
              <name>{$VLAN-ID}</name>
              <ip>
                <address>
                  <ipaddr>{$IP-PREFIX}</ipaddr>
                </address>
              </ip>
            </Vethernet>
          <?end?>
        </interface>

      </config>
    </device>
  </devices>
</config-template>
```


Template variables are defined and assigned via the Python or Java API by a user. The variables $DEVICE and $TEMPLATE_NAME are set internally by NSO. $DEVICE is set to the name of the current device. The variable $TEMPLATE_NAME is set to the name of the current template. None of these variables can be set by a user, it can however be used in a template as any other variable.

Step 11

Save the file.

Step 12

Copy the second template from the existing VLAN service. The template will be reused for the VLAN configuration.

```
student@student-vm:~/nso-run/packages/svi/templates$ cp $HOME/nso-run/
packages/vlan/templates/vlan-template.xml svi-vlan-template.xml
student@student-vm:~/nso-run/packages/svi/templates$ ls
svi-intf-template.xml  svi-vlan-template.xml
```

Step 13

Open the template for editing by using the **code svi-vlan-template.xml** command.

Step 14

Remove the **servicepoint** attribute with the value **vlan** from the **<config-template>** tag. Modify the variables, which will be referenced inside the Python code.

```
<config-template xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <!-- DEVICE -->
    <device>
      <name>{/device/name}</name>
```

```
<config>

  <!-- IOS -->
  <vlan xmlns="urn:ios">
    <vlan-list>
      <id>{$VLAN-ID}</id>
    </vlan-list>
  </vlan>
  <?foreach {interface}?>
    <interface xmlns="urn:ios">
      <?if {intf-type='FastEthernet'}?>
        <FastEthernet>
          <name>{intf-id}</name>
          <switchport>
            <mode>
              <access/>
            </mode>
            <access>
              <vlan>{$VLAN-ID}</vlan>
            </access>
          </switchport>
        </FastEthernet>
      <?end?>
      <?if {intf-type='GigabitEthernet'}?>
        <GigabitEthernet>
          <name>{intf-id}</name>
          <switchport>
            <mode>
              <access/>
            </mode>
            <access>
              <vlan>{$VLAN-ID}</vlan>
            </access>
          </switchport>
        </GigabitEthernet>
      <?end?>

    </interface>
  <?end?>

  <!-- NX-OS -->
  <vlan xmlns="http://tail-f.com/ned/cisco-nx">
    <vlan-list>
      <id>{$VLAN-ID}</id>
    </vlan-list>
  </vlan>
  <?foreach {interface}?>
    <interface xmlns="http://tail-f.com/ned/cisco-nx">
      <Ethernet>
        <name>{intf-id}</name>
        <switchport>
          <mode>access</mode>
          <access>
            <vlan>{$VLAN-ID}</vlan>
          </access>
        </switchport>
      </Ethernet>
    </interface>
```

```
        <?end?>

      </config>
    </device>
  </devices>
</config-template>
```

Step 15

Save the file.

Step 16

Reload the packages.

```
student@student-vm:~/nso-run/packages/svi/templates$ ncs_cli -Cu admin
admin@ncs# packages reload
reload-result {
    package cisco-ios-cli-6.85
    result true
}
reload-result {
    package cisco-iosxr-cli-7.41
    result true
}
reload-result {
    package cisco-nx-cli-5.23
    result true
}
reload-result {
    package l3vpn
    result true
}
reload-result {
    package loopback
    result true
}
reload-result {
    package svi
    result true
}
reload-result {
    package vlan
    result true
}
admin@ncs#
System message at 2022-09-18 13:54:22...
    Subsystem stopped: ncs-dp-8-cisco-nx-cli-5.23:NexusDp
admin@ncs#
System message at 2022-09-18 13:54:22...
    Subsystem stopped: ncs-dp-7-cisco-ios-cli-6.85:IOSDp
admin@ncs#
System message at 2022-09-18 13:54:22...
    Subsystem started: ncs-dp-9-cisco-ios-cli-6.85:IOSDp
admin@ncs#
System message at 2022-09-18 13:54:22...
```

```
    Subsystem started: ncs-dp-10-cisco-nx-cli-5.23:NexusDp
admin@ncs#
```

Troubleshooting Hints

The variables must reference data according to the hierarchy of the YANG data model.

Activity Verification

You have completed this task when you attain this result:

- You have successfully developed the svi-intf-template.xml and svi-vlan-template.xml, and the packages reload operation was successful.

Task 3: Write Python Mapping Code

In this task, you will write Python code for service attribute mapping.

Activity

Complete these steps:

Step 1

Open the Terminal application.

```
student@student-vm:~$
```

Step 2

Go to the *python/svi* subfolder inside the svi package.

```
student@student-vm:~$ cd $HOME/nso-run/packages/svi/python/svi
student@student-vm:~/nso-run/packages/svi/python/svi$ ls
__init__.py  main.py
```

Step 3

Open the Python code for editing by the **code main.py** command.

```
student@student-vm:~/nso-run/packages/svi/python/svi$ code main.py
```

This is how the main.py Python code should look when you open it for the first time:

```
# -*- mode: python; python-indent: 4 -*-
import ncs
from ncs.application import Service

# ------------------------
```

```
# SERVICE CALLBACK EXAMPLE
# ------------------------
class ServiceCallbacks(Service):

    # The create() callback is invoked inside NCS FASTMAP and
    # must always exist.
    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        self.log.info('Service create(service=', service._path, ')')

        vars = ncs.template.Variables()
        vars.add('DUMMY', '127.0.0.1')
        template = ncs.template.Template(service)
        template.apply('svi-template', vars)

    # The pre_modification() and post_modification() callbacks are
optional,
    # and are invoked outside FASTMAP. pre_modification() is invoked
before
    # create, update, or delete of the service, as indicated by the
enum
    # ncs_service_operation op parameter. Conversely
    # post_modification() is invoked after create, update, or delete
    # of the service. These functions can be useful e.g. for
    # allocations that should be stored and existing also when the
    # service instance is removed.

    # @Service.pre_lock_create
    # def cb_pre_lock_create(self, tctx, root, service, proplist):
    #     self.log.info('Service plcreate(service=', service._path,
')')

    # @Service.pre_modification
    # def cb_pre_modification(self, tctx, op, kp, root, proplist):
    #     self.log.info('Service premod(service=', kp, ')')

    # @Service.post_modification
    # def cb_post_modification(self, tctx, op, kp, root, proplist):
    #     self.log.info('Service postmod(service=', kp, ')')

# ---------------------------------------------
# COMPONENT THREAD THAT WILL BE STARTED BY NCS.
# ---------------------------------------------
class Main(ncs.application.Application):
    def setup(self):
        # The application class sets up logging for us. It is
accessible
        # through 'self.log' and is a ncs.log.Log instance.
        self.log.info('Main RUNNING')

        # Service callbacks require a registration for a 'service
point',
        # as specified in the corresponding data model.
        #
        self.register_service('svi-servicepoint', ServiceCallbacks)

        # If we registered any callback(s) above, the Application class
        # took care of creating a daemon (related to the service/action
```

```
point).

        # When this setup method is finished, all registrations are
        # considered done and the application is 'started'.

    def teardown(self):
        # When the application is finished (which would happen if NCS
went
        # down, packages were reloaded or some error occurred) this
teardown
        # method will be called.

        self.log.info('Main FINISHED')
```


Indentation is important in Python. Indentation refers to the spaces at the beginning of a code line. Python uses indentation to indicate a block of code. The number of spaces is up to you as a programmer, but it has to be at least one. You have to use the same number of spaces in the same block of code, otherwise, Python will give you an error. The number of spaces is usually indicated at the beginning of the Python file in a form of a comment: # -*- mode: python; python-indent: 4 -*- ./

Step 4

Remove the dummy code. Create a Python dictionary called svi, which will store service attributes. Read the first attribute VLAN ID from the service object and store it.

```
# -*- mode: python; python-indent: 4 -*-
import ncs
from ncs.application import Service

# ------------------------
# SERVICE CALLBACK EXAMPLE
# ------------------------
class ServiceCallbacks(Service):

    # The create() callback is invoked inside NCS FASTMAP and
    # must always exist.
    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        self.log.info('Service create(service=', service._path, ')')
        svi = {'vlan-id': "",
               'svi-device': "",
               'ip-prefix': "",
               'ip-addr': "",
               'netmask': ""}

        svi['vlan-id'] = service.vlan_id


    # The pre_modification() and post_modification() callbacks are
optional,
    # and are invoked outside FASTMAP. pre_modification() is invoked
```

```
before
    ...
```


You can enable whitespace rendering in Visual Studio Code to have better control over the indentation. Spaces will appear as dots. This feature can be enabled in the **View, Render Whitespaces** menu.

Step 5

Create a Python for loop that will loop over the device list. For each device, check whether ip-prefix is set. If that is true, read all interface-specific attributes and store them in the svi dictionary. Add another import statement for the Python netaddr module, which is used to build a network and network mask from the IP prefix notation.

```
# -*- mode: python; python-indent: 4 -*-
import ncs
from ncs.application import Service
import netaddr

# ------------------------
# SERVICE CALLBACK EXAMPLE
# ------------------------
class ServiceCallbacks(Service):

    # The create() callback is invoked inside NCS FASTMAP and
    # must always exist.
    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        self.log.info('Service create(service=', service._path, ')')

        svi = {'vlan-id': "",
               'svi-device': "",
               'ip-prefix': "",
               'ip-addr': "",
               'netmask': ""}

        svi['vlan-id'] = service.vlan_id

        for device in service.device:
            self.log.info('Entering /device list = ', device.name)
            if device.ip_prefix:
                self.log.info('SVI device = ', device.name)

                ip_net = netaddr.IPNetwork(device.ip_prefix)
                svi['svi-device'] = device.name
                svi['ip-prefix'] =
str(ip_net[2])+"/"+str(ip_net.prefixlen)
                svi['ip-addr'] = str(ip_net[1])
                svi['netmask'] = str(ip_net.netmask)


    # The pre_modification() and post_modification() callbacks are
optional,
```

```
    # and are invoked outside FASTMAP. pre_modification() is invoked
before
    ...
```

Step 6

Create a template variables object called **svi_tvars**. Assign values for all interface-specific variables used within the **svi-intf-template**. Apply the template, using the variables defined.

```
# -*- mode: python; python-indent: 4 -*-
import ncs
from ncs.application import Service
import netaddr

# ------------------------
# SERVICE CALLBACK EXAMPLE
# ------------------------
class ServiceCallbacks(Service):

    # The create() callback is invoked inside NCS FASTMAP and
    # must always exist.
    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        self.log.info('Service create(service=', service._path, ')')
        svi = {'vlan-id': "",
               'svi-device': "",
               'ip-prefix': "",
               'ip-addr': "",
               'netmask': ""}

        svi['vlan-id'] = service.vlan_id

        for device in service.device:
            self.log.info('Entering /device list = ', device.name)
            if device.ip_prefix:
                self.log.info('SVI device = ', device.name)

                ip_net = netaddr.IPNetwork(device.ip_prefix)
                svi['svi-device'] = device.name
                svi['ip-prefix'] =
str(ip_net[2])+"/"+str(ip_net.prefixlen)
                svi['ip-addr'] = str(ip_net[1])
                svi['netmask'] = str(ip_net.netmask)

                svi_tvars = ncs.template.Variables()
                svi_tvars.add('VLAN-ID', svi['vlan-id'])
                svi_tvars.add('SVI-DEVICE', svi['svi-device'])
                svi_tvars.add('IP-PREFIX', svi['ip-prefix'])
                svi_tvars.add('IP-ADDR', svi['ip-addr'])
                svi_tvars.add('NETMASK', svi['netmask'])
                svi_template = ncs.template.Template(service)
                svi_template.apply('svi-intf-template', svi_tvars)


    # The pre_modification() and post_modification() callbacks are
```

```
optional,
    # and are invoked outside FASTMAP. pre_modification() is invoked
before
    ...
```

Step 7

Create a template variables object called **vlan_tvars**. Assign a value for a VLAN ID variable used within the **svi-vlan-template**. Apply the template by using the variable defined.

```
# -*- mode: python; python-indent: 4 -*-
import ncs
from ncs.application import Service
import netaddr

# ------------------------
# SERVICE CALLBACK EXAMPLE
# ------------------------
class ServiceCallbacks(Service):

    # The create() callback is invoked inside NCS FASTMAP and
    # must always exist.
    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        self.log.info('Service create(service=', service._path, ')')

        svi = {'vlan-id': "",
               'svi-device': "",
               'ip-prefix': "",
               'ip-addr': "",
               'netmask': ""}

        svi['vlan-id'] = service.vlan_id

        for device in service.device:
            self.log.info('Entering /device list = ', device.name)
            if device.ip_prefix:
                self.log.info('SVI device = ', device.name)

                ip_net = netaddr.IPNetwork(device.ip_prefix)
                svi['svi-device'] = device.name
                svi['ip-prefix'] =
str(ip_net[2])+"/"+str(ip_net.prefixlen)
                svi['ip-addr'] = str(ip_net[1])
                svi['netmask'] = str(ip_net.netmask)

                svi_tvars = ncs.template.Variables()
                svi_tvars.add('VLAN-ID', svi['vlan-id'])
                svi_tvars.add('SVI-DEVICE', svi['svi-device'])
                svi_tvars.add('IP-PREFIX', svi['ip-prefix'])
                svi_tvars.add('IP-ADDR', svi['ip-addr'])
                svi_tvars.add('NETMASK', svi['netmask'])
                svi_template = ncs.template.Template(service)
                svi_template.apply('svi-intf-template', svi_tvars)
```

```
        vlan_tvars = ncs.template.Variables()
        vlan_tvars.add('VLAN-ID', svi['vlan-id'])
        vlan_template = ncs.template.Template(service)
        vlan_template.apply('svi-vlan-template', vlan_tvars)


    # The pre_modification() and post_modification() callbacks are
optional,
    # and are invoked outside FASTMAP. pre_modification() is invoked
before
    ...
```

Step 8

Save the file.

Step 9

Navigate to the *svi/src* directory and open the Makefile by using the **code Makefile** command.

```
student@student-vm:~/nso-run/packages/svi/python/svi$ cd ../../src
student@student-vm:~/nso-run/packages/svi/src$ code Makefile
```

Step 10

Add some other instruction to the Makefile that will allow you to check the syntax and style of your Python code. You can do this by running the pylint tool on all Python source files. The tool already has been installed in the lab environment.

```
all: fxs pylint
.PHONY: all

# Include standard NCS examples build definitions and rules
include $(NCS_DIR)/src/ncs/build/include.ncs.mk

SRC = $(wildcard yang/*.yang)
DIRS = ../load-dir java/src/$(JDIR)/$(NS)
FXS = $(SRC:yang/%.yang=../load-dir/%.fxs)

## Uncomment and patch the line below if you have a dependency to a NED
## or to other YANG files
# YANGPATH += ../../<ned-name>/src/ncsc-out/modules/yang \
#       ../../<pkt-name>/src/yang

NCSCPATH   = $(YANGPATH:%=--yangpath %)
YANGERPATH = $(YANGPATH:%=--path %)

PYLINT = pylint
PYLINTFLAGS = --disable=R,C --reports=n
PYDIR = ../python/svi
PYTHONFILES = $(wildcard $(PYDIR)/*.py)

pylint:$(patsubst %.py,%.pylint,$(PYTHONFILES))
```

```
%.pylint:
        $(PYLINT) $(PYLINTFLAGS) $*.py || (test $$? -ge 4)

fxs: $(DIRS) $(FXS)

$(DIRS):
        mkdir -p $@

../load-dir/%.fxs: yang/%.yang
        $(NCSC)  `ls $*-ann.yang  > /dev/null 2>&1 && echo "-a $*-
ann.yang"` \
                 --fail-on-warnings \
                 $(NCSCPATH) \
                 -c -o $@ $<

clean:
        rm -rf $(DIRS)
.PHONY: clean
```


Make sure to use the TAB spaces when indenting commands in Makefiles.

Step 11

Save the file.

Step 12

Use the **make** command to compile the package.

```
student@student-vm:~/nso-run/packages/svi/src$ make
pylint --disable=R,C --reports=n ../python/svi/__init__.py || (test $?
-ge 4)
pylint --disable=R,C --reports=n ../python/svi/main.py || (test $? -ge
4)
************* Module svi.main
/home/student/nso-run/packages/svi/python/svi/main.py:16:49: W0212:
Access to a protected member _path of a client class (protected-access)

------------------------------------------------------------------
Your code has been rated at 9.71/10
```

Step 13

Connect to the NSO CLI and reload the packages.

```
student@student-vm:~/nso-run/packages/svi/src$ ncs_cli -Cu admin
admin@ncs# packages reload
reload-result {
    package cisco-ios-cli-6.85
    result true
}
reload-result {
```

```
    package cisco-iosxr-cli-7.41
    result true
}
reload-result {
    package cisco-nx-cli-5.23
    result true
}
reload-result {
    package l3vpn
    result true
}
reload-result {
    package loopback
    result true
}
reload-result {
    package svi
    result true
}
reload-result {
    package vlan
    result true
}
admin@ncs#
System message at 2022-09-18 14:27:47...
    Subsystem stopped: ncs-dp-10-cisco-nx-cli-5.23:NexusDp
admin@ncs#
System message at 2022-09-18 14:27:47...
    Subsystem stopped: ncs-dp-9-cisco-ios-cli-6.85:IOSDp
admin@ncs#
System message at 2022-09-18 14:27:47...
    Subsystem started: ncs-dp-11-cisco-ios-cli-6.85:IOSDp
admin@ncs#
System message at 2022-09-18 14:27:47...
    Subsystem started: ncs-dp-12-cisco-nx-cli-5.23:NexusDp
admin@ncs#
```

Activity Verification

You have completed this task when you attain these results:

- You have successfully developed the main.py, and the packages reload operation was successful.

Task 4: Deploy a Service Instance

In this task, you will deploy a service instance based on your newly installed service.

Activity

Complete these steps:

Step 1

Connect to NSO CLI.

```
student@student-vm:~/nso-run/packages$ ncs_cli -Cu admin
admin@ncs#
```

Step 2

Provision the first service instance.

```
admin@ncs# config
Entering configuration mode terminal
admin@ncs(config)# services svi 102 vlan-id 102
admin@ncs(config-svi-102)# device SW31 interface FastEthernet 0/0
admin@ncs(config-interface-FastEthernet/0/0)# exit
admin@ncs(config-device-SW31)# ip-prefix 10.10.10.0/23
admin@ncs(config-device-SW31)# exit
admin@ncs(config-svi-102)# device SW32 interface Ethernet 1/1
admin@ncs(config-interface-Ethernet/1/1)# exit
admin@ncs(config-device-SW32)# ip-prefix 10.10.10.0/23
admin@ncs(config-device-SW32)# top
admin@ncs(config)#
```

Step 3

After all service parameters are entered, preview the device changes by using the **commit dry-run** command.

```
admin@ncs(config)# commit dry-run
cli {
    local-node {
        data  devices {
                  device SW31 {
                      config {
                          vlan {
            +                 vlan-list 102 {
            +                 }
                          }
                          interface {
                              FastEthernet 0/0 {
            +                     switchport {
            +                         mode {
            +                             access {
            +                             }
            +                         }
            +                         access {
            +                             vlan 102;
            +                         }
            +                     }
                              }
            +                 Vlan 102 {
            +                     ip {
            +                         address {
            +                             primary {
            +                                 address 10.10.10.1;
            +                                 mask 255.255.254.0;
            +                             }
```

```
+                           }
+                       }
+                   }
                }
            }
        }
        device SW32 {
            config {
                vlan {
+                   vlan-list 102 {
+                   }
                }
                interface {
+                   Vethernet 102 {
+                       ip {
+                           address {
+                               ipaddr 10.10.10.2/23;
+                           }
+                       }
+                   }
                    Ethernet 1/1 {
                        switchport {
-                           mode trunk;
+                           mode access;
                            access {
+                               vlan 102;
                            }
                        }
                    }
                }
            }
        }
     }
     services {
+    svi 102 {
+        vlan-id 102;
+        device SW31 {
+            ip-prefix 10.10.10.0/23;
+            interface FastEthernet 0/0;
+        }
+        device SW32 {
+            ip-prefix 10.10.10.0/23;
+            interface Ethernet 1/1;
+        }
+    }
     }
  }
```

Step 4

Commit the service instance creation by using the **commit** command, and exit the configuration mode.

```
admin@ncs(config)# commit
Commit complete.
```

```
admin@ncs(config)# exit
```

Step 5

Use the **show running-config services svi** command to verify that the service has been successfully deployed.

```
admin@ncs# show running-config services svi
services svi 102
 vlan-id 102
 device SW31
  ip-prefix 10.10.10.0/23
  interface FastEthernet 0/0
  !
 !
 device SW32
  ip-prefix 10.10.10.0/23
  interface Ethernet 1/1
  !
 !
!
admin@ncs# show running-config services svi | tab
      VLAN                                        INTF
NAME  ID    NAME  IP PREFIX      INTF TYPE      ID
-----------------------------------------------------
102   102   SW31   10.10.10.0/23  FastEthernet  0/0
            SW32   10.10.10.0/23  Ethernet      1/1
```

Activity Verification

You have completed this task when you attain this result:

- The service has been successfully deployed.

Task 5: Troubleshoot a Python service

In this task, you will intentionally cause some commonly occurring errors in the Python code to learn how to troubleshoot them.

Activity

Complete these steps:

Step 1

Open the Terminal application.

```
student@student-vm:~$
```

Step 2

Go to the *python/svi* subfolder inside the svi package.

```
student@student-vm:~$ cd $ nso-run/packages/svi/python/svi
student@student-vm:~/nso-run/packages/svi/python/svi$ ls
__init__.py  main.py
```

Step 3

Open the Python file for editing with the **code main.py** command.

```
student@student-vm:~/nso-run/packages/svi/python/svi$ code main.py
```

Step 4

Change the name of the SVI interface template and remove indentation from one line.

```
# -*- mode: python; python-indent: 4 -*-
import ncs
from ncs.application import Service
import netaddr

# ------------------------
# SERVICE CALLBACK EXAMPLE
# ------------------------
class ServiceCallbacks(Service):

    # The create() callback is invoked inside NCS FASTMAP and
    # must always exist.
    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        self.log.info('Service create(service=', service._path, ')')

        svi = {'vlan-id': "",
               'svi-device': "",
               'ip-prefix': "",
               'ip-addr': "",
               'netmask': ""}

        svi['vlan-id'] = service.vlan_id

        for device in service.device:
            self.log.info('Entering /device list = ', device.name)
            if device.ip_prefix:
                self.log.info('SVI device = ', device.name)

                ip_net = netaddr.IPNetwork(device.ip_prefix)
                svi['svi-device'] = device.name
                svi['ip-prefix'] =
str(ip_net[2])+"/"+str(ip_net.prefixlen)
                svi['ip-addr'] = str(ip_net[1])
                svi['netmask'] = str(ip_net.netmask)

                svi_tvars = ncs.template.Variables()
                svi_tvars.add('VLAN-ID', svi['vlan-id'])
```

```
                svi_tvars.add('SVI-DEVICE', svi['svi-device'])
                svi_tvars.add('IP-PREFIX', svi['ip-prefix'])
                svi_tvars.add('IP-ADDR', svi['ip-addr'])
                svi_tvars.add('NETMASK', svi['netmask'])
                svi_template = ncs.template.Template(service)
                svi_template.apply('svi-interface-template', svi_tvars)

        vlan_tvars = ncs.template.Variables()
vlan_tvars.add('VLAN-ID', svi['vlan-id'])
        vlan_template = ncs.template.Template(service)
        vlan_template.apply('svi-vlan-template', vlan_tvars)


    # The pre_modification() and post_modification() callbacks are
optional,
    # and are invoked outside FASTMAP. pre_modification() is invoked
before
    ...
```

Step 5

Save the file and switch back to the Terminal application.

Step 6

Connect to NSO CLI.

```
student@student-vm:~/nso-run/packages/svi/python/svi$ ncs_cli -Cu admin
admin@ncs#
```

Step 7

Redeploy the *svi* package. If the changes to the package only affect the code (Python, Java) or the templates, you can perform a redeploy of packages instead of a full reload, which is a much faster operation. The redeploy of the package should fail.

```
admin@ncs# packages package svi redeploy
result false
admin@ncs# *** ALARM package-load-failure: IndentationError: unexpected
indent
```

Step 8

You can check the Python log for more information about what went wrong. Each service has its own Python log where errors and the file line in which they occur are logged. You can see that the indentation must be fixed in the 47th line.

```
admin@ncs# exit
student@student-vm:~/nso-run/packages/svi/src$ tail -n 20 ~/nso-run/
logs/ncs-python-vm-svi.log
<INFO> 18-Sep-2022::14:55:52.853 svi MainThread: - Exiting...
<INFO> 18-Sep-2022::14:55:52.942 svi MainThread: - joining
```

```
ComponentThread svi.main.Main...
<INFO> 18-Sep-2022::14:55:52.944 svi ComponentThread:main: - Main
FINISHED
<INFO> 18-Sep-2022::14:55:52.944 svi MainThread: - joined
ComponentThread svi.main.Main
<INFO> 18-Sep-2022::14:55:52.945 svi MainThread: - Exited
<INFO> 18-Sep-2022::14:55:52.945 svi MainThread: - Goodbye cruel
world...
<INFO> 18-Sep-2022::14:55:53.189 svi MainThread: - Python 3.8.0
(default, Dec  9 2021, 17:53:27) [GCC 8.4.0]
<INFO> 18-Sep-2022::14:55:53.189 svi MainThread: - Using callpoint-
model: threading
<INFO> 18-Sep-2022::14:55:53.190 svi MainThread: - Starting...
<INFO> 18-Sep-2022::14:55:53.204 svi MainThread: - Started
<ERROR> 18-Sep-2022::14:55:53.214 svi MainThread: - Traceback (most
recent call last):
  File "/home/student/nso-5.8/src/ncs/pyapi/ncs_pyvm/startup.py", line
238, in start_components
    ct = ComponentThread(
  File "/home/student/nso-5.8/src/ncs/pyapi/ncs_pyvm/ncsthreads.py",
line 153, in __init__
    mod = __import__(modname)
  File "/home/student/nso-run/state/packages-in-use/1/svi/python/svi/
main.py", line 47
    vlan_template = ncs.template.Template(service)
    ^
IndentationError: unexpected indent
```

Step 9

Switch back to Python file, fix the indentation error and save the file.

```
# -*- mode: python; python-indent: 4 -*-
import ncs
from ncs.application import Service
import netaddr

# ------------------------
# SERVICE CALLBACK EXAMPLE
# ------------------------
class ServiceCallbacks(Service):

    # The create() callback is invoked inside NCS FASTMAP and
    # must always exist.
    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        self.log.info('Service create(service=', service._path, ')')

        svi = {'vlan-id': "",
               'svi-device': "",
               'ip-prefix': "",
               'ip-addr': "",
               'netmask': ""}

        svi['vlan-id'] = service.vlan_id
```

```
        for device in service.device:
            self.log.info('Entering /device list = ', device.name)
            if device.ip_prefix:
                self.log.info('SVI device = ', device.name)

                ip_net = netaddr.IPNetwork(device.ip_prefix)
                svi['svi-device'] = device.name
                svi['ip-prefix'] =
str(ip_net[2])+"/"+str(ip_net.prefixlen)
                svi['ip-addr'] = str(ip_net[1])
                svi['netmask'] = str(ip_net.netmask)

                svi_tvars = ncs.template.Variables()
                svi_tvars.add('VLAN-ID', svi['vlan-id'])
                svi_tvars.add('SVI-DEVICE', svi['svi-device'])
                svi_tvars.add('IP-PREFIX', svi['ip-prefix'])
                svi_tvars.add('IP-ADDR', svi['ip-addr'])
                svi_tvars.add('NETMASK', svi['netmask'])
                svi_template = ncs.template.Template(service)
                svi_template.apply('svi-interface-template', svi_tvars)

        vlan_tvars = ncs.template.Variables()
        vlan_tvars.add('VLAN-ID', svi['vlan-id'])
        vlan_template = ncs.template.Template(service)
        vlan_template.apply('svi-vlan-template', vlan_tvars)


    # The pre_modification() and post_modification() callbacks are
optional,
    # and are invoked outside FASTMAP. pre_modification() is invoked
before
    ...
```

Step 10

Switch back to the Terminal application. Redeploy the package again. Note that even though the Python code is still incorrect, the package is successfully redeployed. This is because NSO has no insight into whether the Python code is working correctly or not, only if the syntax is correct.

```
student@student-vm:~/nso-run/packages/svi/src$ ncs_cli -Cu admin
admin@ncs# packages package svi redeploy
result true
admin@ncs#
```

Step 11

Deploy a new SVI service instance.

```
admin@ncs# config
Entering configuration mode terminal
admin@ncs(config)# services svi 103 vlan-id 103
admin@ncs(config-svi-103)# device SW31 interface FastEthernet 2/0
admin@ncs(config-interface-FastEthernet/2/0)# exit
```

```
admin@ncs(config-device-SW31)# ip-prefix 10.10.20.0/23
admin@ncs(config-device-SW31)# top
admin@ncs(config)#
```

Step 12

Try committing the service instance. An error occurs in the callback create function because the template file **svi-interface-template** does not exist.

```
admin@ncs(config)# commit
Aborted: Python cb_create error. item does not exist (1): The template
file svi-interface-template has not been loaded.
admin@ncs(config)#
```

Step 13

To avoid configuring the service instance once again, you will commit the changes without running the service logic. Service parameters will be stored in CDB for later deployment.

```
admin@ncs(config)# commit no-deploy
Commit complete.
admin@ncs(config)# exit
admin@ncs#
```

Step 14

Switch back Visual Studio Code and fix the template name in Python file. Save the file.

```
        ...

        for device in service.device:
            self.log.info('Entering /device list = ', device.name)
            if device.ip_prefix:
                self.log.info('SVI device = ', device.name)

                ip_net = netaddr.IPNetwork(device.ip_prefix)
                svi['svi-device'] = device.name
                svi['ip-prefix'] =
str(ip_net[2])+"/"+str(ip_net.prefixlen)
                svi['ip-addr'] = str(ip_net[1])
                svi['netmask'] = str(ip_net.netmask)

                svi_tvars = ncs.template.Variables()
                svi_tvars.add('VLAN-ID', svi['vlan-id'])
                svi_tvars.add('SVI-DEVICE', svi['svi-device'])
                svi_tvars.add('IP-PREFIX', svi['ip-prefix'])
                svi_tvars.add('IP-ADDR', svi['ip-addr'])
                svi_tvars.add('NETMASK', svi['netmask'])
                svi_template = ncs.template.Template(service)
```

```
                svi_template.apply('svi-intf-template', svi_tvars)

        vlan_tvars = ncs.template.Variables()
        vlan_tvars.add('VLAN-ID', svi['vlan-id'])
        vlan_template = ncs.template.Template(service)
        vlan_template.apply('svi-vlan-template', vlan_tvars)


    # The pre_modification() and post_modification() callbacks are
optional,
    # and are invoked outside FASTMAP. pre_modification() is invoked
before
    ....
```

Step 15

Switch back to NSO CLI and redeploy the package again.

```
admin@ncs# packages package svi redeploy
result true
admin@ncs#
```

Step 16

Verify whether the service logic of your created service instance you have previously committed but not deployed is now working properly.

```
admin@ncs# services svi 103 re-deploy dry-run
cli {
    local-node {
        data  devices {
                    device SW31 {
                        config {
                            vlan {
              +                 vlan-list 103 {
              +                 }
                            }
                            interface {
              +                 FastEthernet 2/0 {
              +                     switchport {
              +                         mode {
              +                             access {
              +                             }
              +                         }
              +                         access {
              +                             vlan 103;
              +                         }
              +                     }
              +                 }
              +                 Vlan 103 {
              +                     ip {
              +                         address {
              +                             primary {
              +                                 address 10.10.20.1;
```

```
                +                                       mask 255.255.254.0;
                +                                   }
                +                               }
                +                           }
                +                       }
                                    }
                                }
                            }
                        }

    }
}
```

Step 17

Redeploy your service instance.

```
admin@ncs# services svi 103 re-deploy
admin@ncs#
System message at 2022-09-18 15:20:41...
Commit performed by admin via ssh using cli.
admin@ncs#
```

Step 18

Verify that the service configuration matches with the following configuration.

```
admin@ncs# show running-config services svi 103
services svi 103
 vlan-id 103
 device SW31
  ip-prefix 10.10.20.0/23
  interface FastEthernet 2/0
  !
 !
!
```

Activity Verification

You have completed this task when you attain this result:

- The service instance has been successfully deployed.