# Discovery 4: Create a Loopback Template Service

## Introduction

In this activity, you will create a Loopback template service, which you will then apply to devices. After completing this activity, you will be able to meet these objectives:

- Create a package skeleton and define the service model in YANG
- Create the service template
- Compile and deploy the service

## Job Aids

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

- This lab guide
- Student guide, for general explanations

### Job Aids for Task 1

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 | *loopback* | list | — | — |
| Service instance name | *name* | string | — | key for list loopback |
| Interface number | *loopback-intf* | uint32 | Range: 1100 – 1199 | — |
| IP address | *ip-address* | inet:ipv4-address | Range: 10.100.x.x – 10.199.x.x | — |
| Device | *device* | leafref | — | Reference to device list in NSO |

### 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 PE31):

```
interface Loopback172
 ip address 10.172.19.88 255.255.255.255
!
```

Cisco IOS XR platform (device PE21):

```
interface Loopback168
 ipv4 address 10.168.23.66/32
!
```

## 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. |
| *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`<br>`5.8.2.1` |
| *Example* | Save your current configuration as the default **startup config**.<br>`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 non-quoted 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. |

| Command | Comment |
|---|---|
| **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 a 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:

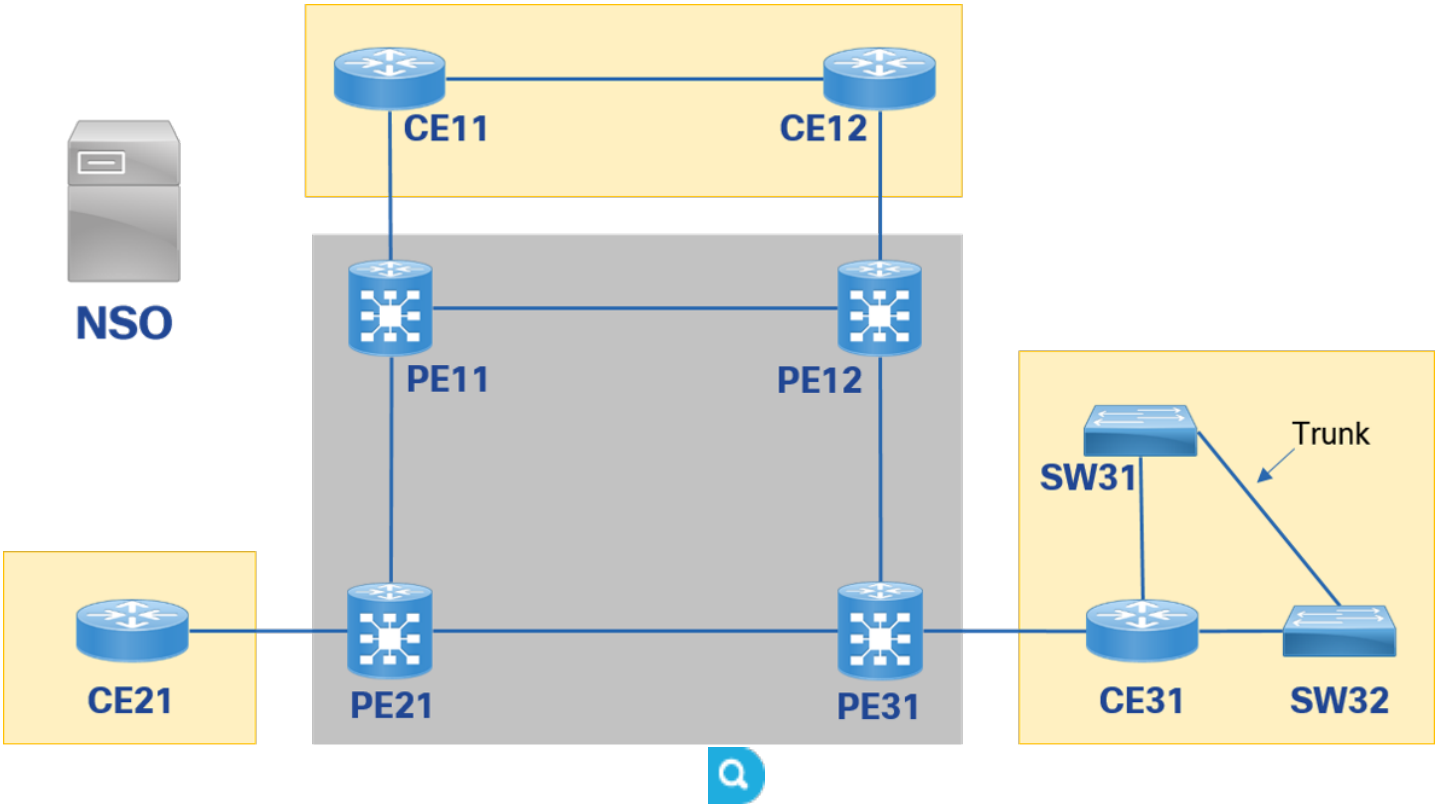
| Command | Comment |
|---|---|
| **switch cli** | Change CLI style. |
| **show ?** | Display all command options for current mode. |
| **configure** | Enter configuration mode. |
| **commit** | Commit new configuration (configuration mode only command). |
| **show configuration** | Display new configuration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

## 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 is 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



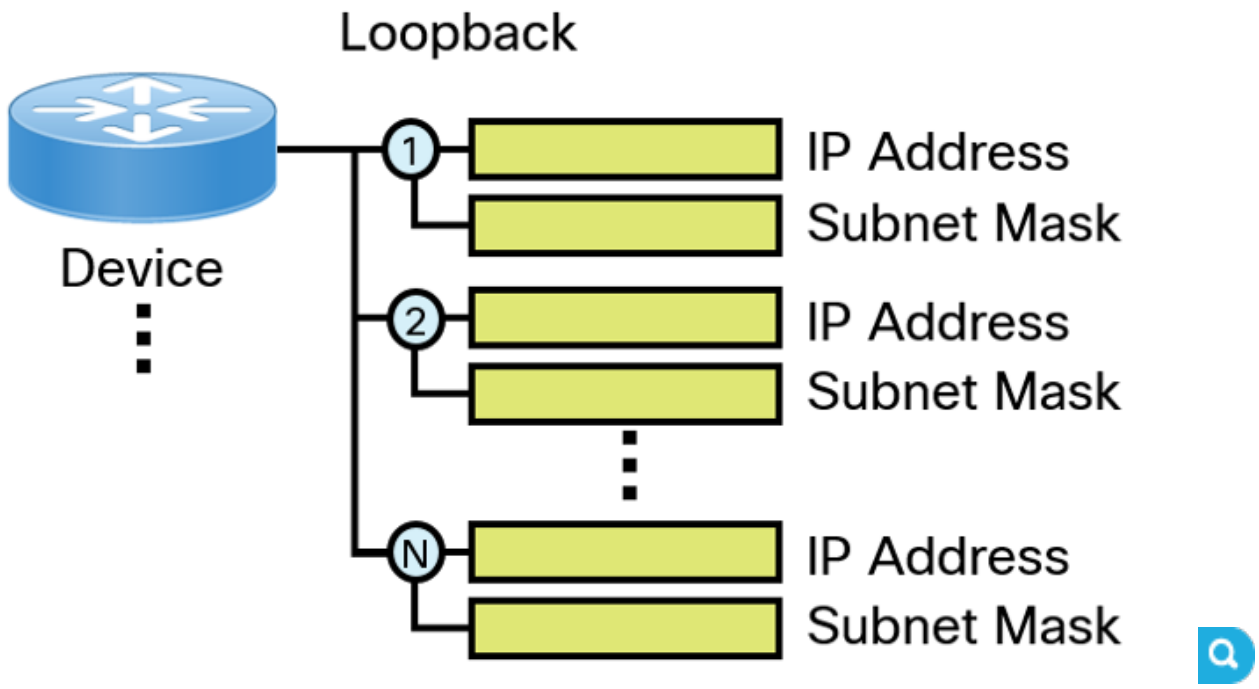

## Task 1: Design a Service Model

The first task requires you to design a service model that is 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

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 | Description |
|---|---|
| Service name | Unique identifier describing an instance of a deployed service |
| Interface number | Unique number for the loopback interface |
| IP address | Unique IP address for the loopback interface |
| Device | Device on which the service will be deployed |

Complete these steps:

***Step 1***

Connect to the NSO server by clicking the **NSO** icon.

***Step 2***

On the NSO server, launch the Terminal application and go to a directory where service packages are located.

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

***Step 3***

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

You can use **ncs-make-package --help** or man **ncs-make-package** to learn about all the 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
student@student-vm:~/nso-run/packages$ ncs-make-package --service-skeleton template loopback
student@student-vm:~/nso-run/packages$ ls
cisco-ios-cli-6.54  cisco-iosxr-cli-7.26  cisco-nx-cli-5.15  loopback
```

***Step 4***

Display the YANG service model of the **loopback** service.

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

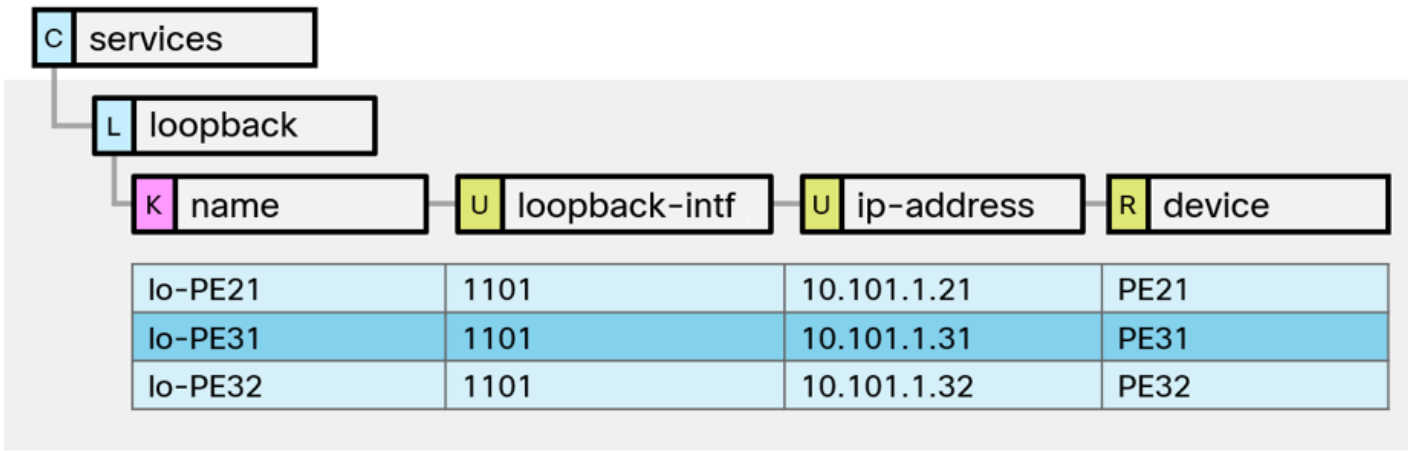
```
student@student-vm:~/nso-run/packages$ cd loopback/src/yang
student@student-vm:~/nso-run/packages/loopback/src/yang$ cat loopback.yang
module loopback {
  namespace "http://com/example/loopback";
  prefix loopback;

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

  list loopback {
    key name;
```

```
    uses ncs:service-data;
    ncs:servicepoint "loopback";

    leaf name {
      type string;
    }
    // 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 5***

Refer to the following table for fine-tuning the service attributes that you will use in your YANG service model.

| Attribute | Name | Data Type | Restriction | Other |
|---|---|---|---|---|
| Service name | *loopback* | list | — | — |
| Service instance name | *name* | string | — | key for list loopback |
| Interface number | *loopback-intf* | uint32 | Range: 1100 – 1199 | — |
| IP address | *ip-address* | inet:ipv4-address | Range: 10.100.x.x – 10.199.x.x | — |
| Device | *device* | leafref | — | Reference to device list in NSO |




You can find the data table in the Job Aids.

***Step 6***

Open and edit the YANG service model with the Visual Studio Code, by entering the **code loopback.yang** command.

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

You can edit the service model YANG file by using your favorite text editor. It is recommended that you use the Visual Studio Code text editor on the workstation, which will provide you with syntax highlighting for YANG.

***Step 7***

Change the namespace of the module to a unique distinctive value, for example “http://cisco.com/examples/loopback”, that can be used to easily identify the service package.

```
module loopback {
  namespace "http://cisco.com/example/loopback";
  prefix loopback;

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

***Step 8***

Remove the l**eaf-list device** and the **leaf dummy**. They will not be used for the loopback service and will be replaced by other statements.

After removing the nodes, the YANG service module should look exactly like this:

```
module loopback {
  namespace "http://cisco.com/example/loopback";
  prefix loopback;

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

  list loopback {
    key name;

    uses ncs:service-data;
    ncs:servicepoint "loopback";

    leaf name {
      type string;
    }
  }
}
```

***Step 9***

Add an **augment /ncs:services** statement, to include the list loopback in the services branch of the CDB.

Make sure that you correctly embed the augment statement contents, by using the opening and closing curly braces like in the following output:

```
module loopback {
  namespace "http://cisco.com/example/loopback";
  prefix loopback;

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

  augment /ncs:services {
    list loopback {
      key name;

      uses ncs:service-data;
      ncs:servicepoint "loopback";

      leaf name {
        type string;
      }
    }
  }
}
```

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 10***

Add a reference to the **tailf-common** module.

Add a reference to the **tailf-common** module by importing that module. The **tailf-common** module contains NSO specific YANG extensions and statements. You will use the module's **info** statement for the NSO CLI syntax help that you will implement.

```
module loopback {
  namespace "http://cisco.com/example/loopback";
  prefix loopback;

  import ietf-inet-types {
    prefix inet;
  }
  import tailf-ncs {
    prefix ncs;
  }
  import tailf-common {
     prefix tailf;
  }
   augment /ncs:services {
    ...
  }
}
```

### *Step 11*

Add node descriptions to your service model.

Add node descriptions to your service model. You will use the **tailf:info** statement from the previously imported **tailf-common module** to implement the descriptions and NSO CLI syntax help.

```
module loopback {
  namespace "http://cisco.com/example/loopback";
  prefix loopback;

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

  augment /ncs:services {
    list loopback {
      tailf:info "Loopback Service";
      key name;

      uses ncs:service-data;
      ncs:servicepoint "loopback";

      leaf name {
        tailf:info "Service Instance Name";
        type string;
      }
    }
  }
}
```

### *Step 12*

Add a leaf for a device. Each loopback is assigned to a device on which it will be configured. The type leafref is used to point to the existing device in the devices list of the CDB.

```
module loopback {
  namespace "http://cisco.com/example/loopback";
  prefix loopback;

  ...

  augment /ncs:services {
    list loopback {
      tailf:info "Loopback Service";
      key name;

      uses ncs:service-data;
      ncs:servicepoint "loopback";

      leaf name {
        tailf:info "Service Instance Name";
        type string;
      }

      leaf device {
        tailf:info "Router name";
```

```
        mandatory true;
        type leafref {
          path "/ncs:devices/ncs:device/ncs:name";
        }
      }

    }
  }
}
```

***Step 13***

Add an interface number (loopback-intf) for each loopback interface. Based on the service requirements, loopback-intf is limited to the range 1100 to 1199.

```
module loopback {
  namespace "http://cisco.com/example/loopback";
  prefix loopback;

  ...

  augment /ncs:services {
    list loopback {

      ...

      leaf name {
        tailf:info "Service Instance Name";
        type string;
      }

      leaf device {
        tailf:info "Router name";
        mandatory true;
        type leafref {
          path "/ncs:devices/ncs:device/ncs:name";
        }
      }

      leaf loopback-intf {
        tailf:info "Loopback Interface Number from 1100 to 1199";
        mandatory true;
        type uint32 {
          range "1100..1199";
        }
      }

    }
  }
}
```

***Step 14***

The last service attribute to add is ip-address, which is configured on the selected loopback. The valid IP address range is defined by means of a regular expression inside the pattern statement. Because the loopback interface number and IP address must be unique, the loopback instance gets a unique loopback-intf and ip-address.

```
module loopback {
  ...

  augment /ncs:services {
    list loopback {
```

```
      ...

      leaf loopback-intf {
        tailf:info "Loopback Interface Number from 1100 to 1199";
        mandatory true;
        type uint32 {
          range "1100..1199";
        }
      }

      leaf ip-address {
        tailf:info "Valid IP range from 10.100.x.x to 10.199.x.x.";
        mandatory true;
        type inet:ipv4-address {
          pattern "10.1[0-9][0-9].[0-9]+.[0-9]+";
        }
      }

    }
  }
}
```

***Step 15***

Save the file.

***Step 16***

You can validate your YANG file by using **pyang**. Keep making corrections to your YANG file until **pyang** no longer produces any errors for your YANG module.

Observe the two valid outputs the pyang tool can produce:

```
student@student-vm:~/nso-run/packages/loopback/src/yang$ pyang loopback.yang
student@student-vm:~/nso-run/packages/loopback/src/yang$
```

or:

```
student@student-vm:~/nso-run/packages/loopback/src/yang$ pyang loopback.yang
/home/student/nso-5.8/src/ncs/yang/ietf-yang-schema-mount.yang:2: error: bad value "1.1"
(should be version)
/home/student/nso-5.8/src/ncs/yang/ietf-yang-schema-mount.yang:9: error: unexpected keyword
"reference"
/home/student/nso-5.8/src/ncs/yang/ietf-yang-schema-mount.yang:15: error: unexpected keyword
"reference"
/home/student/nso-5.8/src/ncs/yang/tailf-ncs-cluster.yang:2: error: bad value "1.1" (should be
version)
/home/student/nso-5.8/src/ncs/yang/tailf-ncs-cluster.yang:222: error: "leaf-list" node "tailf-
ncs::public-key" cannot be refined with "default"
/home/student/nso-5.8/src/ncs/yang/tailf-ncs-cluster.yang:224: error: unexpected keyword
"default"
...
```

In the second case, no errors for the loopback module are present.

Pyang can show errors and warnings in its output. In the lab, only errors for the **loopback** module must be fixed. It is possible that errors from other modules will be displayed as shown in the above example. You can ignore them.

### *Step 17*

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

### *Step 18*

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

```
student@student-vm:~/nso-run/packages/loopback/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 loopback
    result true
}
admin@ncs
System message at 2022-09-06 14:02:24...
    Subsystem stopped: ncs-dp-2-cisco-nx-cli-5.23:NexusDp
admin@ncs
System message at 2022-09-06 14:02:24...
    Subsystem stopped: ncs-dp-1-cisco-ios-cli-6.85:IOSDp
admin@ncs
System message at 2022-09-06 14:02:24...
    Subsystem started: ncs-dp-3-cisco-ios-cli-6.85:IOSDp
admin@ncs
System message at 2022-09-06 14:02:24...
    Subsystem started: ncs-dp-4-cisco-nx-cli-5.23:NexusDp
admin@ncs
```

### *Step 19*

Check the availability of CLI commands for managing the service. For example, use the **services** command, followed by a question mark, to investigate the new CLI options available for the provisioning of service instances. These options are defined in the *loopback.yang* file you edited in this task.

```
admin@ncs config
Entering configuration mode terminal
admin@ncs(config) services ?
Possible completions:
  check-sync                    Check if device configuration is according to the services
  commit-queue-notifications    Configuration to send service-commit-queue-event notifications.
  customer-service              Service that can be linked to customer
  global-settings
  logging                       Configure service logging
  loopback                      Loopback Service
  plan-notifications            Configuration to send plan-state-change notifications for plan
state transitions.
  service                       List of resource facing services
admin@ncs(config) services loopback ?
% No entries found
Possible completions:
  Service Instance Name
admin@ncs(config) services loopback
```

***Step 20***

Exit the NSO CLI. Verify that the directory structure and file templates for the new service package match the expected structure.

```
student@student-vm:~/nso-run/packages$ 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  loopback
student@student-vm:~/nso-run/packages$ ls -l loopback/
total 20
drwxrwxr-x 2 student student 4096 Sep  6 13:48 load-dir
-rw-rw-r-- 1 student student  374 Sep  6 12:25 package-meta-data.xml
drwxr-xr-x 3 student student 4096 Sep  6 12:25 src
drwxr-xr-x 2 student student 4096 Sep  6 12:25 templates
drwxr-xr-x 3 student student 4096 Jul  8 09:56 test
student@student-vm:~/nso-run/packages$ ls -l loopback/src/yang/
total 4
-rw-rw-r-- 1 student student 1089 Sep  6 13:39 loopback.yang
student@student-vm:~/nso-run/packages$ ls -l loopback/templates/
total 4
-rw-rw-r-- 1 student student 736 Sep  6 12:25 loopback-template.xml
```

**Activity Verification**

You have completed this task when you attain this result:

- You have a directory structure and file templates that match the expected structure in the previous output.

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 Configuration Templates

In this task, you will configure the sample loopback service from the NSO CLI in order to obtain the device-specific configuration in XML format. Then you can create a device template for Cisco IOS and Cisco IOS XR devices.

## Activity

Complete these steps:

***Step 1***

Connect to the 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 PE31):

```
interface Loopback172
 ip address 10.172.19.88 255.255.255.255
!
```

Cisco IOS XR platform (device PE21):

```
interface Loopback168
 ipv4 address 10.168.23.66/32
!
```

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

***Step 3***

Configure the preceding examples by using the NSO CLI. Use the **devices device PE31** configuration command as a starting point to configure the Cisco IOS sample. Use the **devices device PE21** configuration command as a starting point for configuring the Cisco IOS XR sample.

```
admin@ncs config
Entering configuration mode terminal
admin@nso(config) devices device PE31 config interface Loopback 172
admin@nso(config-if) ip address 10.172.19.88 255.255.255.255
admin@nso(config-if) top
admin@nso(config)
admin@nso(config) devices device PE21 config interface Loopback 168
admin@nso(config-if) ipv4 address 10.168.23.66 255.255.255.255
admin@nso(config-if) top
admin@nso(config)
```

***Step 4***

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

You will use this output later in this task.

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>PE21</name>
                 <config>
                   <interface xmlns="http://tail-f.com/ned/cisco-ios-xr">
                     <Loopback>
                       <id>169</id>
                       <ipv4>
                         <address>
                           <ip>10.168.23.66</ip>
                           <mask>255.255.255.255</mask>
                         </address>
                       </ipv4>
                     </Loopback>
                   </interface>
                 </config>
               </device>
               <device>
                 <name>PE31</name>
                 <config>
                   <interface xmlns="urn:ios">
                     <Loopback>
                       <name>172</name>
                       <ip>
                         <address>
                           <primary>
                             <address>10.172.19.88</address>
                             <mask>255.255.255.255</mask>
                           </primary>
                         </address>
                       </ip>
                     </Loopback>
                   </interface>
                 </config>
               </device>
             </devices>
    }
}
admin@ncs(config) abort
admin@ncs exit
```

### *Step 5*

Go to the templates subdirectory of your loopback package skeleton ($HOME/nso-run/packages/loopback/templates).

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

### *Step 6*

Open the loopback-template by using the **code loopback-template.xml** command.

The following output shows how the template should look when you open it for the first time:

```
<config-template xmlns="http://tail-f.com/ns/config/1.0"
                 servicepoint="loopback">
  <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>
        -->
      </config>
    </device>
  </devices>
</config-template>
```

***Step 7***

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

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

        <!-- IOS -->

        <!-- IOS-XR -->

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

***Step 8***

Insert the XML configuration created with **commit dry-run outformat xml** in the modified XML skeleton sections.

Because the service model supports two types of devices, the XML configuration changes segment must accommodate both of them. Edit the segment with XML configuration code. Add both the Cisco IOS and Cisco IOS XR parts of the configuration.

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

        <!-- IOS -->
        <interface xmlns="urn:ios">
          <Loopback>
            <name>172</name>
            <ip>
              <address>
                <primary>
                  <address>10.172.19.88</address>
```

```
                <mask>255.255.255.255</mask>
              </primary>
            </address>
          </ip>
        </Loopback>
      </interface>

      <!-- IOS-XR -->
      <interface xmlns="http://tail-f.com/ned/cisco-ios-xr">
        <Loopback>
          <id>168</id>
          <ipv4>
            <address>
              <ip>10.168.23.66</ip>
              <mask>255.255.255.255</mask>
            </address>
          </ipv4>
        </Loopback>
      </interface>

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

***Step 9***

Replace all static parameters with variables, which reference service attributes according to the hierarchy of the YANG data model.

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

        <!-- IOS -->
        <interface xmlns="urn:ios">
          <Loopback>
            <name>{/loopback-intf}</name>
            <ip>
              <address>
                <primary>
                  <address>{/ip-address}</address>
                  <mask>255.255.255.255</mask>
                </primary>
              </address>
            </ip>
          </Loopback>
        </interface>

        <!-- IOS-XR -->
        <interface xmlns="http://tail-f.com/ned/cisco-ios-xr">
          <Loopback>
            <id>{/loopback-intf}</id>
            <ipv4>
              <address>
                <ip>{/ip-address}</ip>
                <mask>255.255.255.255</mask>
              </address>
            </ipv4>
          </Loopback>
```

```
        </interface>

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

***Step 10***

Save the file.

***Step 11***

Reload the packages and make sure the reload-result of the loopback package is set to true.

```
student@student-vm:~/nso-run/packages/loopback/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 loopback
    result true
}
admin@ncs
System message at 2022-09-06 14:26:55...
    Subsystem stopped: ncs-dp-4-cisco-nx-cli-5.23:NexusDp
admin@ncs
System message at 2022-09-06 14:26:55...
    Subsystem stopped: ncs-dp-3-cisco-ios-cli-6.85:IOSDp
admin@ncs
System message at 2022-09-06 14:26:55...
    Subsystem started: ncs-dp-5-cisco-ios-cli-6.85:IOSDp
admin@ncs
System message at 2022-09-06 14:26:55...
    Subsystem started: ncs-dp-6-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 the following result:

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

## Task 3: 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 the 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 loopback OSPF device PE11 loopback-intf 1100 ip-address 10.100.0.11
```

***Step 3***

After all service parameters are entered, preview the device changes by using the **commit dry-run** command. To better understand what is happening when you are applying templates, use the CLI pipe **debug template** command.

```
admin@ncs(config-loopback-OSPF) commit dry-run | debug template
Evaluating "/device" (from file "loopback-template.xml", line 6)
Context node: /services/loopback:loopback[name='OSPF']
Result:
For /services/loopback:loopback[name='OSPF'], it evaluates to "PE11"
Operation 'merge' on existing node: /devices/device[name='PE11'] (from file "loopback-
template.xml", line 6)
Evaluating "/loopback-intf" (from file "loopback-template.xml", line 12)
Context node: /services/loopback:loopback[name='OSPF']
Result:
For /services/loopback:loopback[name='OSPF'], it evaluates to "1100"
Operation 'merge' on non-existing node: /devices/device[name='PE11']/config/ios:interface/
Loopback[name='1100'] (from file "loopback-template.xml", line 12)
Operation 'merge' on non-existing node: /devices/device[name='PE11']/config/ios:interface/
Loopback[name='1100']/ip/address/primary/address (from file "loopback-template.xml", line 16)
Evaluating "/ip-address" (from file "loopback-template.xml", line 16)
Context node: /services/loopback:loopback[name='OSPF']
Result:
For /services/loopback:loopback[name='OSPF'], it evaluates to "10.100.0.11"
Setting /devices/device[name='PE11']/config/ios:interface/Loopback[name='1100']/ip/address/
primary/address to "10.100.0.11"
Operation 'merge' on non-existing node: /devices/device[name='PE11']/config/ios:interface/
Loopback[name='1100']/ip/address/primary/mask (from file "loopback-template.xml", line 17)
Fetching literal "255.255.255.255" (from file "loopback-template.xml", line 17)
Setting /devices/device[name='PE11']/config/ios:interface/Loopback[name='1100']/ip/address/
primary/mask to "255.255.255.255"
The device /devices/device[name='PE11'] does not support namespace 'http://tail-f.com/ned/
cisco-ios-xr' for node "interface" (from file "loopback-template.xml", line 25)
Skipping...
cli {
    local-node {
        data  devices {
                  device PE11 {
                      config {
                          interface {
            +                 Loopback 1100 {
            +                     ip {
            +                         address {
```

```
               +                               primary {
               +                                   address 10.100.0.11;
               +                                   mask 255.255.255.255;
               +                               }
               +                           }
               +                       }
               +                   }
                               }
                           }
                       }
                }
                services {
               +    loopback OSPF {
               +        device PE11;
               +        loopback-intf 1100;
               +        ip-address 10.100.0.11;
               +    }
                }
    }
}
```

### *Step 4*

Commit the service instance creation, using the **commit** command.

```
admin@ncs(config-loopback-OSPF) top
admin@ncs(config) commit
Commit complete.
admin@ncs(config) exit
```

### *Step 5*

Verify the service configuration.

```
admin@ncs show running-config services loopback
services loopback OSPF
 device        PE11
 loopback-intf 1100
 ip-address    10.100.0.11
!
admin@ncs show running-config services loopback | tab
             LOOPBACK
NAME  DEVICE  INTF      IP ADDRESS
--------------------------------------
OSPF  PE11    1100      10.100.0.11
```

### Activity Verification

You have completed this task when you attain this result:

- The service has been successfully deployed.