Discovery 6: Create an L3VPN Template Service

Introduction

In this activity, you will learn how to create an L3MPLS VPN service. After completing this activity, you will be able to meet these objectives:

- Use the Cisco NSO CLI to configure the service model in YANG
- Create the service template
- Compile and deploy the service

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

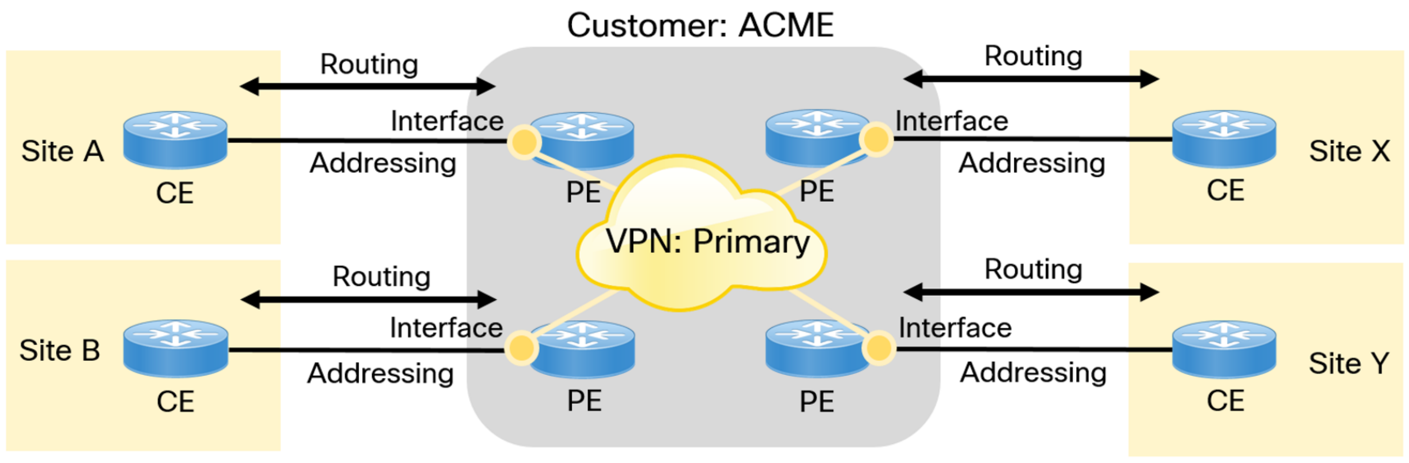
Job Aids

The following job aids are available to help you complete the lab activities:

- This lab guide
- Student guide, for general explanations

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	l3vpn	list	—	—
VPN ID	vpn-id	uint32	Range: 10001 – 19999	key for list l3vpn
VPN name	vpn-name	string	—	—
Customer	customer	leafref	—	reference to NSO customers
Link	link	list	—	—
Link ID	link-id	uint32	Range: 1 – 255	key for list link
Interface	interface	string	—	pe-device and interface combination must be unique
Routing protocol	routing-protocol	enumeration	Options: bgp static	—
PE device	pe-device	leafref	Only PE devices may be selected	pe-device and interface combination must be unique
Static route	static-route	list	Applicable only when static routing is selected	—
Prefix	prefix	inet:ipv4-address	—	key for list static-route
Mask	mask	union of inet:ipv4-address and inet:ipv6address	—	—

Job Aids for Task 2

The following data will be required for this task.

A network engineer has provided you with the actual configuration for the new service.

Cisco IOS platform (device PE11):

```
vrf definition vpn10001
```

```
  description Customer ACME VPN
  rd          1:10001
  route-target export 1:10001
  route-target import 1:10001
 !
 ip route vrf vpn10001 192.168.11.0 255.255.255.0 172.31.1.2
 interface GigabitEthernet4
  description Connection to Customer ACME
  vrf forwarding vpn10001
  ip address 172.31.1.1 255.255.255.252
 exit
 router bgp 1
  address-family ipv4 unicast vrf vpn10001
   neighbor 172.31.1.2 remote-as 65001
   neighbor 172.31.1.2 activate
   neighbor 172.31.1.2 allowas-in
   neighbor 172.31.1.2 as-override disable
   neighbor 172.31.1.2 default-originate
   redistribute connected
   redistribute static
   exit-address-family
  !
 !
```

Cisco IOS XR platform (device PE21):

```
vrf vpn10001
  description Customer ACME VPN
  address-family ipv4 unicast
   import route-target
        1:10001
   exit
   export route-target
        1:10001
   exit
  exit
 exit
 interface GigabitEthernet 0/0/0/1
  description Connection to Customer ACME
  ipv4 address 172.31.1.5 255.255.255.252
  vrf         vpn10001
 exit
 router static
  address-family ipv4 unicast
   192.168.21.0/24 GigabitEthernet0/0/0/1 172.31.1.6
  exit
 exit
 router bgp 1
  vrf vpn10001
   rd 1:10001
   address-family ipv4 unicast
        redistribute connected
        redistribute static
   exit
   neighbor 172.31.1.6
        address-family ipv4 unicast
         route-policy test in
         as-override
         default-originate
        exit
   exit
  exit
 exit
```

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`
	`        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 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 tables list 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.

Command	Comment
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:

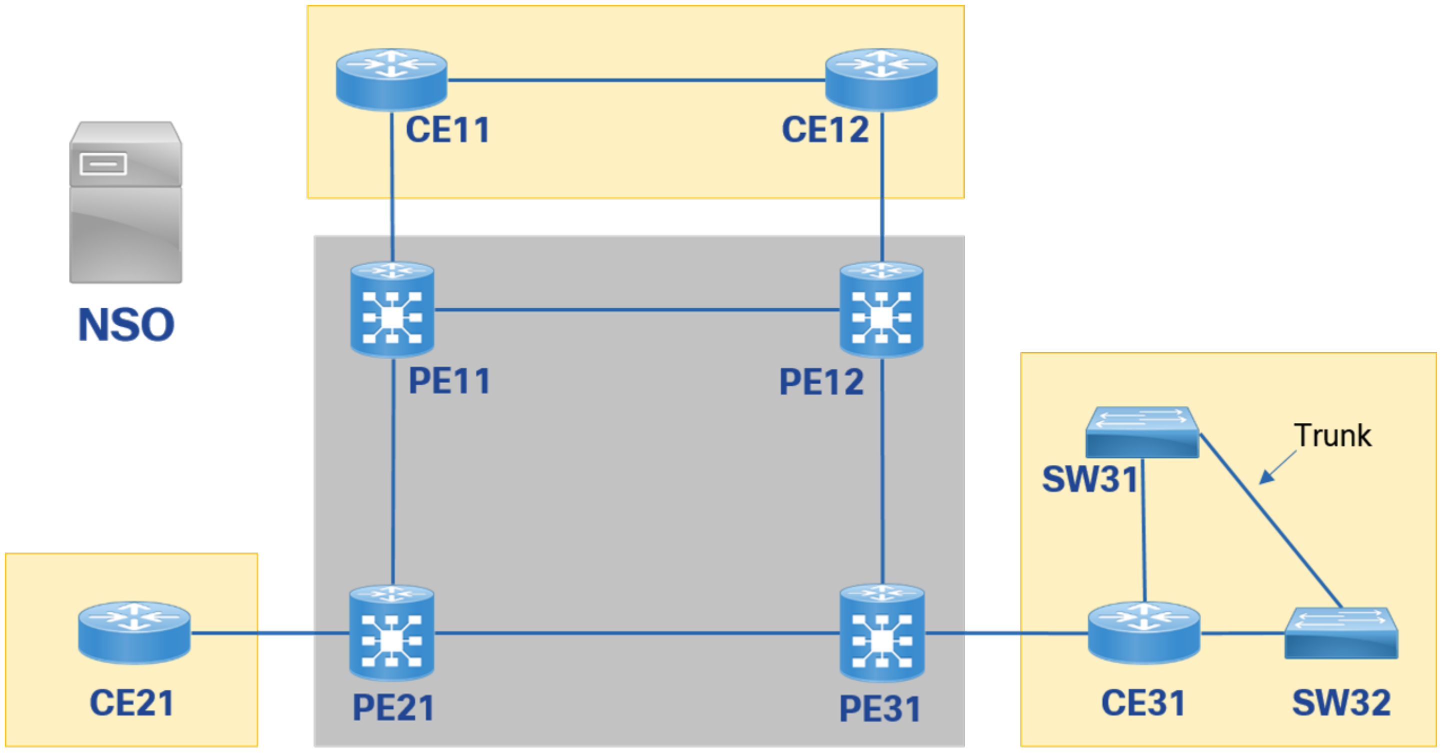
Command	Comment
switch cli	Change CLI style.
show ?	Display all command options for current mode.
configure	Enter configuration mode.
commit	Commit new configuration (configuration mode only command).
show configuration	Display new configuration that has not yet been committed (configuration mode only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

Lab Topology Information

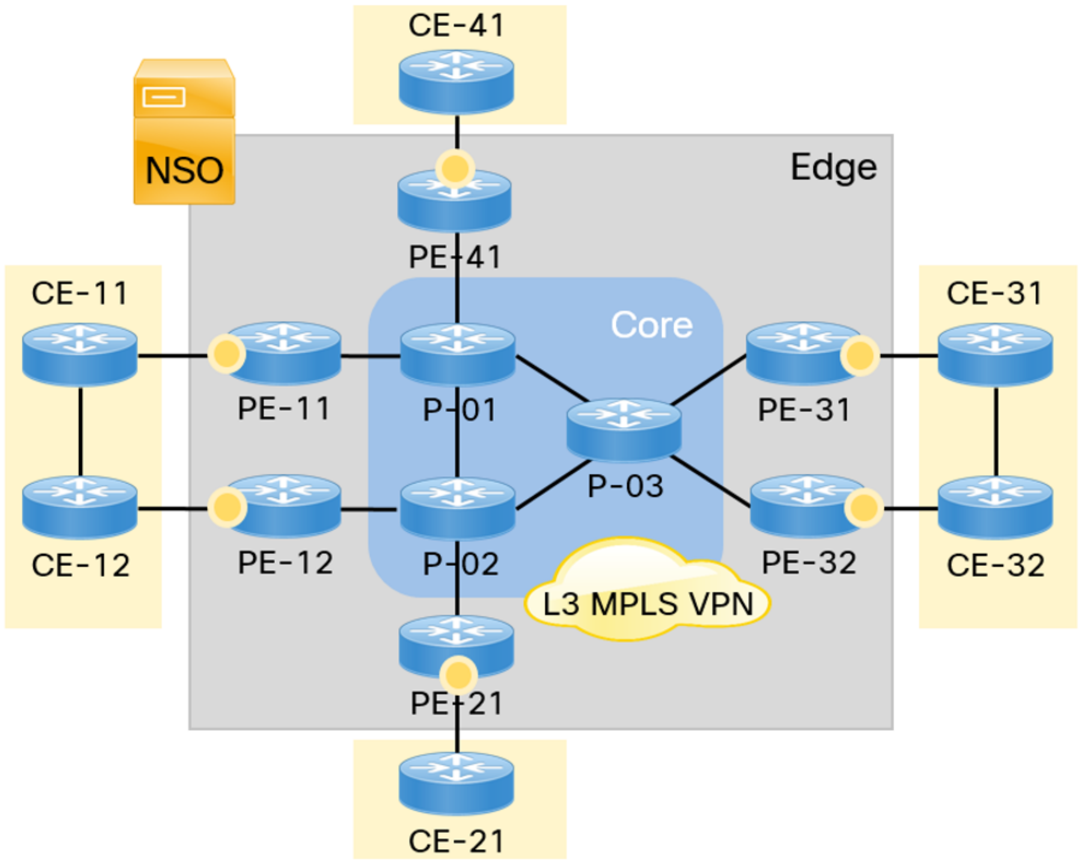
Your lab session is your own personal sandbox. Whatever you do in your session will not be reflected in anyone else s session. There are two topologies. The general one is your lab environment, and it has your workstation and a Linux server. On the Linux server within that topology is your second topology 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 preceding graphic 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 **l3vpn** 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 options.

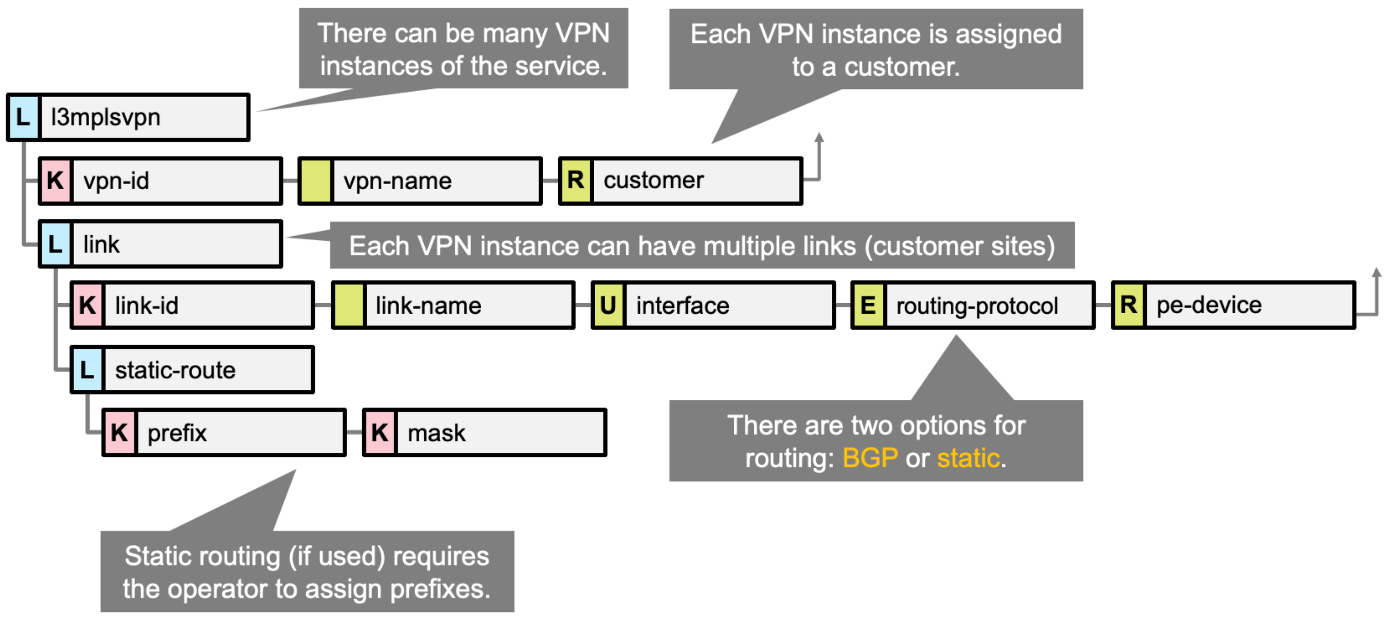
```
student@student-vm:~/nso-run/packages$ ls
cisco-ios-cli-6.85  cisco-iosxr-cli-7.41  cisco-nx-cli-5.23  loopback  vlanstudent@student-
vm:~/nso-run/packages$ ncs-make-package --service-skeleton template l3vpn
student@student-vm:~/nso-run/packages$ ls
cisco-ios-cli-6.85  cisco-iosxr-cli-7.41  cisco-nx-cli-5.23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

Step 5

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

Attribute	**Name**	**Data Type**	**Restriction**	**Other**
Service name	l3vpn	list	—	—
VPN ID	vpn-id	uint32	Range: 10001 – 19999	key for list l3vpn
VPN name	vpn-name	string	—	—
Customer	customer	leafref	—	reference to NSO customers
Link	link	list	—	—
Link ID	link-id	uint32	Range: 1 – 255	key for list link
Interface	interface	string	—	pe-device and interface combination must be unique
Routing protocol	routing-protocol	enumeration	Options: bgp static	—
PE device	pe-device	leafref	Only PE devices may be selected	pe-device and interface combination must be unique
Static route	static-route	list	Applicable only when static routing is selected	—
Prefix	prefix	inet:ipv4-address	—	key for list static-route

Attribute	Name	Data Type	Restriction	Other
Mask	mask	union of inet:ipv4-address and inet:ipv6address	—	—


There can be many VPN instances of the service.
Each VPN instance is assigned to a customer.
L l3mplsvpn
K vpn-id
vpn-name
R customer
L link
Each VPN instance can have multiple links (customer sites)
K link-id
link-name
U interface
E routing-protocol
R pe-device
L static-route
K prefix
K mask
There are two options for routing: BGP or static.
Static routing (if used) requires the operator to assign prefixes.

Step 6

Navigate to **l3vpn/src/yang** directory and open the YANG service model for editing with the **code l3vpn.yang** command.

```
student@student-vm:~/nso-run/packages$ cd l3vpn/src/yang
student@student-vm:~/nso-run/packages/l3vpn/src/yang$ code l3vpn.yang
```

You can edit the YANG service model YANG 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 7

Rename the namespace to *http://cisco.com/example/l3vpn* and remove the leaf-list device and the leaf dummy. They will not be used for the l3vpn service and will be replaced by other statements. Add an augment */ ncs:services* statement, to include the list *l3vpn* in the services branch of the CDB.

```
module l3vpn {
  namespace "http://cisco.com/example/l3vpn";
  prefix l3vpn;

  import ietf-inet-types {
    prefix inet;
  }
  import tailf-ncs {
    prefix ncs;
  }
  augment /ncs:services {
    list l3vpn {
      key name;

      uses ncs:service-data;
      ncs:servicepoint "l3vpn";

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

Step 8

Rename name leaf to *vpn-name* leaf. Define a vpn-name leaf as a key to the service list. A list is used to support multiple instances of the l3vpn service. The list records are uniquely identified, using the key attribute (vpn-name). Add another import statement for *tailf-common*, to support YANG annotations, and add node descriptions for syntax help.

```
module l3vpn {
  namespace "http://cisco.com/example/l3vpn";
  prefix l3vpn;

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

  augment /ncs:services {
    list l3vpn {
      tailf:info "L3VPN Service";
      key vpn-name;

      uses ncs:service-data;
      ncs:servicepoint "l3vpn";

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

Step 9

Assign a VPN ID to each l3vpn service instance.

```
module l3vpn {

  ...

  augment /ncs:services {
    list l3vpn {
      tailf:info "L3VPN Service";
      key vpn-name;

      uses ncs:service-data;
      ncs:servicepoint "l3vpn";

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

      leaf vpn-id {
        tailf:info "Service Instance ID";
        type uint32 {
          range "10001..19999";
        }
      }

    }
  }
}
```

Step 10

Assign a customer reference to every l3vpn service instance.

```
module l3vpn {

  ...

  augment /ncs:services {
    list l3vpn {

      ...

      leaf vpn-id {
        tailf:info "Service Instance ID";
        type uint32 {
          range "10001..19999";
        }
      }

      leaf customer {
        tailf:info "VPN Customer";
        type leafref {
          path "/ncs:customers/ncs:customer/ncs:id";
        }
      }

    }
  }
}
```

Step 11

Add a list of links. Each VPN service instance can support multiple customer links, represented with the list link. A link instance gets a unique link ID and a link name.

```


  ...

  augment /ncs:services {
    list l3vpn {

      ...

      leaf customer {
        tailf:info "VPN Customer";
        type leafref {
          path "/ncs:customers/ncs:customer/ncs:id";
        }
      }

      // Each VPN service instance can have one or more interfaces
      list link {
        tailf:info "PE-CE Attachment Point";
        key link-id;

        leaf link-id {
           tailf:info "Link ID";
           type uint32 {
              range "1..255";
           }
        }

        leaf link-name {
           tailf:info "Link Name";
           type string;
        }
      }

    }
  }
}
```

Step 12

Assign each link instance to the interface on the selected PE device. Add an additional unique restriction, which will prevent duplicate entries. The interface on a specific device can be used only for one link.

```
      ...

      // Each VPN service instance can have one or more interfaces
      list link {
        tailf:info "PE-CE Attachment Point";
        key link-id;
        unique "pe-device interface";

        leaf link-id {
           tailf:info "Link ID";
           type uint32 {
              range "1..255";
           }
        }

        leaf link-name {
           tailf:info "Link Name";
           type string;
        }

        leaf pe-device {
          tailf:info "PE Router";
          type leafref {
            path "/ncs:devices/ncs:device/ncs:name";
          }
        }
        leaf interface {
          tailf:info "Customer Facing Interface";
          type string;
        }

      }

      ...
```

Step 13

Use the XPath function starts-with() to constrain which devices the administrator can input as pe-device. To get the current node value, you must use the current() function, which returns the context node in the current expression. You can also add a relevant error message in case of an incorrect input.

Consider the following facts about the current model:

- *pe-device* can be any device currently added to NSO. This should be limited to include only "PE" devices.
- XPath supports a wide variety of functions that include strings, numbers, Booleans and nodes, such as *position(), contains(), floor(), substring(), starts-with()*¸ and more. By using these functions in combination with YANG statements such as *must*, we can constrain the valid data for a specific model. Consider which functions can be used to constrain this model.

You can find a list and an explanation of all the XPath functions here https://www.w3.org/TR/1999/REC-xpath-19991116/

```
        ...

        leaf pe-device {
          tailf:info "PE Router";
          type leafref {
            path "/ncs:devices/ncs:device/ncs:name";
          }
          must "starts-with(current(),'PE')" {
            error-message "Only PE devices can be selected.";
          }
        }

        ...
```

Step 14

Add a routing-protocol option to each instance link. If static routing is used, route prefix and route mask attributes are used. A "*when*" statement is used to make its parent data definition conditional. The "when" statement's argument is an XPath expression. For this particular step, use the wrong XPath argument – "/routing-protocol='static'" to get an XPath error when compiling the YANG file. Use **type union** for the mask because Cisco IOS demands an IP address with a subnet mask, and Cisco IOS XR demands just an IP prefix (with CIDR annotation).

```
...

        leaf interface {
          tailf:info "Customer Facing Interface";
          type string;
        }

        leaf routing-protocol {
          tailf:info "Routing option on PE-CE link";
          type enumeration {
            enum bgp;
            enum static;
          }
        }

        list static-route  {
          tailf:info "Static Route";
          key prefix;
          when "/routing-protocol='static'";

          leaf prefix {
            tailf:info "Static Route Prefix";
            type inet:ipv4-address;
          }

          leaf mask {
            tailf:info "Subnet Mask for IOS and Prefix for IOSXR";
            type union {
                type inet:ipv4-address;
                type inet:ipv4-prefix;
            }
          }
        }

        ...
```

Step 15

Verify the service model contents and save the file.

The YANG service model should resemble the following output:

```
module l3vpn {
  namespace "http://cisco.com/example/l3vpn";
  prefix l3vpn;

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

  augment /ncs:services {
    list l3vpn {
      tailf:info "L3VPN Service";
      key vpn-name;

      uses ncs:service-data;
      ncs:servicepoint "l3vpn";

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

      leaf vpn-id {
        tailf:info "Service Instance ID";
        type uint32 {
          range "10001..19999";
        }
      }

      leaf customer {
        tailf:info "VPN Customer";
        type leafref {
          path "/ncs:customers/ncs:customer/ncs:id";
        }
      }

      // Each VPN service instance can have one or more interfaces
      list link {
        tailf:info "PE-CE Attachment Point";
        key link-id;
        unique "pe-device interface";

        leaf link-id {
           tailf:info "Link ID";
           type uint32 {
              range "1..255";
           }
        }

        leaf link-name {
           tailf:info "Link Name";
           type string;
        }

        leaf pe-device {
          tailf:info "PE Router";
          type leafref {
            path "/ncs:devices/ncs:device/ncs:name";
          }
          must "starts-with(current(),'PE')" {
            error-message "Only PE devices can be selected.";
          }
        }

        leaf interface {
          tailf:info "Customer Facing Interface";
          type string;
        }

        leaf routing-protocol {
          tailf:info "Routing option on PE-CE link";
          type enumeration {
            enum bgp;
            enum static;
          }
        }

        list static-route  {
          tailf:info "Static Route";
          key prefix;
          when "/routing-protocol='static'";

          leaf prefix {
            tailf:info "Static Route Prefix";
            type inet:ipv4-address;
          }

          leaf mask {
            tailf:info "Subnet Mask for IOS and Prefix for IOSXR";
            type union {
                type inet:ipv4-address;
                type inet:ipv4-prefix;
            }
          }
        }

      }

    }
  }
}
```

Step 16

Switch to the Terminal application. You can validate your YANG file by using the **pyang** command. You can expect a warning due to the incorrect XPath.

```
student@student-vm:~/nso-run/packages/l3vpn/src/yang$ pyang l3vpn.yang
l3vpn.yang:86: warning: node "l3vpn::routing-protocol" is not found in module "l3vpn"
/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"
...
```

Step 17

To see a different and more specific warning output, 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/l3vpn/src/yang$ cd ..
student@student-vm:~/nso-run/packages/l3vpn/src$ make
mkdir -p ../load-dir
/home/student/nso-5.8/bin/ncsc `ls l3vpn-ann.yang  > /dev/null 2>&1 && echo "-a l3vpn-
ann.yang"` \
        --fail-on-warnings \
         \
        -c -o ../load-dir/l3vpn.fxs yang/l3vpn.yang
yang/l3vpn.yang:86: error: The XPath expression references an undefined node: the node
'routing-protocol' from module 'l3vpn' is not found
Makefile:26: recipe for target '../load-dir/l3vpn.fxs' failed
make: *** [../load-dir/l3vpn.fxs] Error 1
```

Step 18

You can see that the 'routing-protocol' node cannot be found because the XPath is incorrect. When you are writing "when" statements, always make sure that the desired path is correct. In this specific case, you are specifying that the 'routing-protocol' node is on the same level as the 'static-route' node. You must go outside the current (list static-route) node to address the 'routing-protocol' node. Edit the l3vpn.yang file and append ".." to the **/routing-protocol='static'** statement.

```
...

        leaf routing-protocol {
          tailf:info "Routing option on PE-CE link";
          type enumeration {
            enum bgp;
            enum static;
          }
        }

        list static-route  {
          tailf:info "Static Route";
          key prefix;
          when "../routing-protocol='static'";

          leaf prefix {
            tailf:info "Static Route Prefix";
            type inet:ipv4-address;
          }

          leaf mask {
            tailf:info "Subnet Mask for IOS and Prefix for IOSXR";
            type union {
                type inet:ipv4-address;
                type inet:ipv4-prefix;
            }
          }
        }

        ...
```

Step 19

Save the file. Validate it with the **pyang** command and compile it again. Keep making corrections to your YANG file until the **pyang** command no longer produces any output for **l3vpn.yang** file (until there are no errors).

```
student@student-vm:~/nso-run/packages/l3vpn/src$ pyang yang/l3vpn.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-devices.yang:3645: 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/l3vpn/src$ make
/home/student/nso-5.8/bin/ncsc `ls l3vpn-ann.yang  > /dev/null 2>&1 && echo "-a l3vpn-
ann.yang"` \
        --fail-on-warnings \
         \
        -c -o ../load-dir/l3vpn.fxs yang/l3vpn.yang
student@student-vm:~/nso-run/packages/l3vpn/src$
```

Step 20

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

```
student@student-vm:~/nso-run/packages/l3vpn/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 vlan
    result true
}
admin@ncs
System message at 2022-09-14 17:03:27...
    Subsystem stopped: ncs-dp-4-cisco-nx-cli-5.23:NexusDp
admin@ncs
System message at 2022-09-14 17:03:27...
    Subsystem stopped: ncs-dp-3-cisco-ios-cli-6.85:IOSDp
admin@ncs
System message at 2022-09-14 17:03:27...
    Subsystem started: ncs-dp-5-cisco-ios-cli-6.85:IOSDp
admin@ncs
System message at 2022-09-14 17:03:27...
    Subsystem started: ncs-dp-6-cisco-nx-cli-5.23:NexusDp
admin@ncs
```

Step 21

Check the availability of CLI commands to manage the service. For example, use the **services l3vpn** 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
  vlan         VLAN Service
admin@ncs services l3vpn ?
% No entries found
```

Step 22

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

```
student@student-vm:~$ cd
student@student-vm:~$ cd nso-run/packages/
student@student-vm:~/nso-run/packages$ ls -l l3vpn/
total 20
drwxrwxr-x 2 student student 4096 Sep 14 17:02 load-dir
-rw-rw-r-- 1 student student  368 Sep 14 15:05 package-meta-data.xml
drwxr-xr-x 3 student student 4096 Sep 14 15:05 src
drwxr-xr-x 2 student student 4096 Sep 14 15:05 templates
drwxr-xr-x 3 student student 4096 Jul  8 09:56 test
student@student-vm:~/nso-run/packages$ ls -l l3vpn/src/yang
total 4
-rw-rw-r-- 1 student student 2325 Sep 14 16:53 l3vpn.yang
student@student-vm:~/nso-run/packages$ ls -l l3vpn/templates
total 4
```

```
-rw-rw-r-- 1 student student 733 Sep 14 15:05 l3vpn-template.xml
```

Activity Verification

You have completed this task when you attain this result:

- You have a directory structure and file templates for a new service that matches the expected structure.

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

The purpose of this task is to configure the sample l3vpn service from the NSO CLI, to obtain the device-specific configuration in XML format. This will allow you to create a device template for the Cisco IOS and Cisco IOS XR platforms.

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

Configure the sample l3vpn service for devices **PE11** and **PE21** from NSO CLI by loading the configuration from the provided **configuration.txt** file.

```
admin@ncs config
Entering configuration mode terminal
admin@ncs(config) load merge /home/student/solutions/lab06/configuration.txt
Loading.
1.62 KiB parsed in 0.67 sec (2.41 KiB/sec)
admin@ncs(config)
```

You can find the complete l3vpn sample service configuration in Job Aids.

You can also configure the sample from the CLI in NSO by copy and pasting the whole configuration sample. To do this, enter config mode and enter the **load merge terminal** command. Next copy and paste the entire configuration. When you are finished pasting the configuration press the **Enter** key to create a new line and then press **CTRL+D** on your keyboard to parse the configuration.

Step 3

Use the **commit dry-run outformat xml** command to retrieve the XML version of the configuration for the Cisco IOS and Cisco IOS XR platforms. Copy and save these outputs as you will use them later.

Verify that the output contains all the configured parameters. If some or all 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>PE11</name>
                 <config>
                   <vrf xmlns="urn:ios">
                     <definition>
                       <name>vpn10001</name>
                       <description>L3VPN for Customer ACME</description>
                       <rd>1:10001</rd>
                       <route-target>
                         <export>
                           <asn-ip>1:10001</asn-ip>
                         </export>
                         <import>
                           <asn-ip>1:10001</asn-ip>
                         </import>
                       </route-target>
                     </definition>
                   </vrf>
                   <ip xmlns="urn:ios">
                     <route>
                       <vrf>
                         <name>vpn10001</name>
                         <ip-route-forwarding-list>
                           <prefix>192.168.11.0</prefix>
                           <mask>255.255.255.0</mask>
                           <forwarding-address>172.31.1.2</forwarding-address>
                         </ip-route-forwarding-list>
                       </vrf>
                     </route>
                   </ip>
                   <interface xmlns="urn:ios">
                     <GigabitEthernet>
                       <name>4</name>
                       <description>Connection to Customer ACME</description>
                       <vrf>
                         <forwarding>vpn10001</forwarding>
                       </vrf>
                       <ip>
                         <address>
                           <primary>
                             <address>172.31.1.1</address>
                             <mask>255.255.255.252</mask>
                           </primary>
                         </address>
                       </ip>
                     </GigabitEthernet>
                   </interface>
                   <router xmlns="urn:ios">
                     <bgp>
                       <as-no>1</as-no>
                       <address-family>
                         <with-vrf>
                           <ipv4>
                             <af>unicast</af>
                             <vrf>
                               <name>vpn10001</name>
```

```
                <redistribute>
                  <connected/>
                  <static/>
                </redistribute>
                <neighbor>
                  <id>172.31.1.2</id>
                  <remote-as annotation="first">65001</remote-as>
                  <activate/>
                  <allowas-in/>
                  <as-override>
                    <disable/>
                  </as-override>
                  <default-originate/>
                </neighbor>
              </vrf>
            </ipv4>
          </with-vrf>
        </address-family>
      </bgp>
    </router>
  </config>
</device>
<device>
  <name>PE21</name>
  <config>
    <vrf xmlns="http://tail-f.com/ned/cisco-ios-xr">
      <vrf-list>
        <name>vpn10001</name>
        <description>L3VPN for Customer ACME</description>
        <address-family>
          <ipv4>
            <unicast>
              <import>
                <route-target>
                  <address-list>
                    <name>1:10001</name>
                  </address-list>
                </route-target>
              </import>
              <export>
                <route-target>
                  <address-list>
                    <name>1:10001</name>
                  </address-list>
                </route-target>
              </export>
            </unicast>
          </ipv4>
        </address-family>
      </vrf-list>
    </vrf>
    <interface xmlns="http://tail-f.com/ned/cisco-ios-xr">
      <GigabitEthernet>
        <id>0/0/0/1</id>
        <description>Connection to Customer ACME</description>
        <vrf>vpn10001</vrf>
        <ipv4>
          <address>
            <ip>172.31.1.5</ip>
            <mask>255.255.255.252</mask>
          </address>
        </ipv4>
      </GigabitEthernet>
    </interface>
    <router xmlns="http://tail-f.com/ned/cisco-ios-xr">
      <static>
        <address-family>
```

```
                        <ipv4>
                          <unicast>
                            <routes>
                              <net>192.168.21.0/24</net>
                              <interface>GigabitEthernet0/0/0/1</interface>
                              <address>172.31.1.6</address>
                            </routes>
                          </unicast>
                        </ipv4>
                      </address-family>
                    </static>
                    <bgp>
                      <bgp-no-instance>
                        <id>1</id>
                        <vrf>
                          <name>vpn10001</name>
                          <rd>1:10001</rd>
                          <address-family>
                            <ipv4>
                              <unicast>
                                <redistribute>
                                  <connected/>
                                  <static/>
                                </redistribute>
                              </unicast>
                            </ipv4>
                          </address-family>
                          <neighbor>
                            <id>172.31.1.6</id>
                            <address-family>
                              <ipv4>
                                <unicast>
                                  <route-policy>
                                    <direction>in</direction>
                                    <name>test</name>
                                  </route-policy>
                                  <as-override/>
                                  <default-originate/>
                                </unicast>
                              </ipv4>
                            </address-family>
                          </neighbor>
                        </vrf>
                      </bgp-no-instance>
                    </bgp>
                  </router>
                </config>
              </device>
            </devices>
    }
}
admin@ncs(config) abort
admin@ncs exit
```

You can save the output of the dry-run by piping the output to the file. The following example will save the output to a file name **configuration.xml** in a directory from where you invoked the **ncs_cli** command: **commit dry-run outformat xml | save configuration.xml**

Step 4

Go to the templates subdirectory of your package skeleton (for example, $HOME/nso-run/packages/l3vpn/templates).

```
student@student-vm:~$ cd $HOME/nso-run/packages/l3vpn/templates
student@student-vm:~/nso-run/packages/l3vpn/templates$
```

Step 5

Open the template by using the **code l3vpn-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="l3vpn">
  <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>
```

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, IOS-XR).

```
<config-template xmlns="http://tail-f.com/ns/config/1.0"
                 servicepoint="l3vpn">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <!-- DEVICE -->
    <device>
      <name>{/device}</name>
      <config>

        <!-- IOS -->

        <!-- IOS-XR -->

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

Step 6

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

Because the service model supports two types of devices, the XML configuration changes segment must accommodate both. 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="l3vpn">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <!-- DEVICE -->
    <device>
      <name>{/device}</name>
      <config>

        <!-- IOS -->
        <vrf xmlns="urn:ios">
          <definition>
            <name>vpn10001</name>
            <description>L3VPN for Customer ACME</description>
            <rd>1:10001</rd>
            <route-target>
              <export>
                <asn-ip>1:10001</asn-ip>
              </export>
              <import>
                <asn-ip>1:10001</asn-ip>
              </import>
            </route-target>
          </definition>
        </vrf>
        <ip xmlns="urn:ios">
          <route>
            <vrf>
              <name>vpn10001</name>
              <ip-route-forwarding-list>
                <prefix>192.168.11.0</prefix>
                <mask>255.255.255.0</mask>
                <forwarding-address>172.31.1.2</forwarding-address>
              </ip-route-forwarding-list>
            </vrf>
          </route>
        </ip>
        <interface xmlns="urn:ios">
          <GigabitEthernet>
            <name>4</name>
            <description>Connection to Customer ACME</description>
            <vrf>
              <forwarding>vpn10001</forwarding>
            </vrf>
            <ip>
              <address>
                <primary>
                  <address>172.31.1.1</address>
                  <mask>255.255.255.252</mask>
                </primary>
              </address>
            </ip>
          </GigabitEthernet>
        </interface>
        <router xmlns="urn:ios">
          <bgp>
            <as-no>1</as-no>
            <address-family>
              <with-vrf>
                <ipv4>
                  <af>unicast</af>
                  <vrf>
                    <name>vpn10001</name>
                    <redistribute>
                      <connected/>
                      <static/>
                    </redistribute>
                    <neighbor>
                      <id>172.31.1.2</id>
                      <remote-as annotation="first">65001</remote-as>
                      <activate/>
                      <allowas-in/>
                      <as-override>
                        <disable/>
                      </as-override>
                      <default-originate/>
                    </neighbor>
                  </vrf>
                </ipv4>
              </with-vrf>
            </address-family>
          </bgp>
        </router>

        <!-- IOS-XR -->
        <vrf xmlns="http://tail-f.com/ned/cisco-ios-xr">
          <vrf-list>
            <name>vpn10001</name>
            <description>L3VPN for Customer ACME</description>
            <address-family>
              <ipv4>
                <unicast>
                  <import>
                    <route-target>
                      <address-list>
                        <name>1:10001</name>
                      </address-list>
                    </route-target>
                  </import>
                  <export>
                    <route-target>
                      <address-list>
                        <name>1:10001</name>
                      </address-list>
                    </route-target>
                  </export>
                </unicast>
              </ipv4>
            </address-family>
          </vrf-list>
        </vrf>
        <interface xmlns="http://tail-f.com/ned/cisco-ios-xr">
          <GigabitEthernet>
            <id>0/0/0/1</id>
            <description>Connection to Customer ACME</description>
            <vrf>vpn10001</vrf>
            <ipv4>
              <address>
                <ip>172.31.1.5</ip>
                <mask>255.255.255.252</mask>
              </address>
            </ipv4>
          </GigabitEthernet>
        </interface>
        <router xmlns="http://tail-f.com/ned/cisco-ios-xr">
          <static>
            <address-family>
              <ipv4>
                <unicast>
                  <routes>
                    <net>192.168.21.0/24</net>
                    <interface>GigabitEthernet0/0/0/1</interface>
                    <address>172.31.1.6</address>
                  </routes>
```

```
                </unicast>
              </ipv4>
            </address-family>
          </static>
          <bgp>
            <bgp-no-instance>
              <id>1</id>
              <vrf>
                <name>vpn10001</name>
                <rd>1:10001</rd>
                <address-family>
                  <ipv4>
                    <unicast>
                      <redistribute>
                        <connected/>
                        <static/>
                      </redistribute>
                    </unicast>
                  </ipv4>
                </address-family>
                <neighbor>
                  <id>172.31.1.6</id>
                  <address-family>
                    <ipv4>
                      <unicast>
                        <route-policy>
                          <direction>in</direction>
                          <name>test</name>
                        </route-policy>
                        <as-override/>
                        <default-originate/>
                      </unicast>
                    </ipv4>
                  </address-family>
                </neighbor>
              </vrf>
            </bgp-no-instance>
          </bgp>
        </router>

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

Step 7

Replace all the static parameters with variables, which reference service attributes according to the hierarchy of the YANG data model. Because you are implementing an l3vpn service template, the template must be able to apply multiple devices and static routes. Use XML *foreach* tags to create the logic that will handle this. Use **string()** in XPath when referencing *vpn-id* so that the context node will not be changed. This needs to be done only with references to (yang model) keys, because the template processor detects it as a loop.

```
<config-template xmlns="http://tail-f.com/ns/config/1.0" servicepoint="l3vpn">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <!-- DEVICE -->
    <?foreach {/link}?>
      <device>
        <name>{pe-device}</name>
        <config>

          <!-- IOS -->
          <vrf xmlns="urn:ios">
            <definition>
```

```
              <name>vpn{string(../vpn-id)}</name>
              <description>By NSO: L3VPN - {string(/customer)}</description>
              <rd>1:{string(../vpn-id)}</rd>
              <route-target>
                <export>
                  <asn-ip>1:{string(../vpn-id)}</asn-ip>
                </export>
                <import>
                  <asn-ip>1:{string(../vpn-id)}</asn-ip>
                </import>
              </route-target>
            </definition>
          </vrf>
          <?if {routing-protocol='static'}?>
            <ip xmlns="urn:ios">
              <route>
                <vrf>
                  <name>vpn{string(../vpn-id)}</name>
                  <?foreach {static-route}?>
                    <ip-route-forwarding-list>
                      <prefix>{prefix}</prefix>
                      <mask>{mask}</mask>
                      <forwarding-address>172.31.{../link-id}.2</forwarding-address>
                    </ip-route-forwarding-list>
                  <?end?>
                </vrf>
              </route>
            </ip>
          <?end?>
          <interface xmlns="urn:ios">
            <GigabitEthernet>
              <name>{interface}</name>
              <description>By NSO: L3VPN - {string(/customer)} - {string(link-name)}</
description>
              <vrf>
                <forwarding>vpn{string(../vpn-id)}</forwarding>
              </vrf>
              <ip>
                <address>
                  <primary>
                    <address>172.31.{link-id}.1</address>
                    <mask>255.255.255.252</mask>
                  </primary>
                </address>
              </ip>
            </GigabitEthernet>
          </interface>
          <router xmlns="urn:ios">
            <?if {routing-protocol='bgp'}?>
              <bgp>
                <as-no>1</as-no>
                <address-family>
                  <with-vrf>
                    <ipv4>
                      <af>unicast</af>
                      <vrf>
                        <name>vpn{string(../vpn-id)}</name>
                        <redistribute>
                          <connected/>
                          <static/>
                        </redistribute>
                        <neighbor>
                          <id>172.31.{link-id}.2</id>
                          <remote-as annotation="first">65001</remote-as>
                          <activate/>
                          <allowas-in/>
                          <as-override>
                            <disable/>
                          </as-override>
                          <default-originate/>
                        </neighbor>
                      </vrf>
                    </ipv4>
                  </with-vrf>
                </address-family>
              </bgp>
            <?end?>
          </router>

          <!-- IOS-XR -->
          <vrf xmlns="http://tail-f.com/ned/cisco-ios-xr">
            <vrf-list>
              <name>vpn{string(../vpn-id)}</name>
              <description>By NSO: L3VPN - {string(/customer)}</description>
              <address-family>
                <ipv4>
                  <unicast>
                    <import>
                      <route-target>
                        <address-list>
                          <name>1:{string(../vpn-id)}</name>
                        </address-list>
                      </route-target>
                    </import>
                    <export>
                      <route-target>
                        <address-list>
                          <name>1:{string(../vpn-id)}</name>
                        </address-list>
                      </route-target>
                    </export>
                  </unicast>
                </ipv4>
              </address-family>
            </vrf-list>
          </vrf>
          <interface xmlns="http://tail-f.com/ned/cisco-ios-xr">
            <GigabitEthernet>
              <id>{interface}</id>
              <description>By NSO: L3VPN - {string(/customer)} - {string(link-name)}</
description>
              <vrf>vpn{string(../vpn-id)}</vrf>
              <ipv4>
                <address>
                  <ip>172.31.{link-id}.5</ip>
                  <mask>255.255.255.252</mask>
                </address>
              </ipv4>
            </GigabitEthernet>
          </interface>
          <router xmlns="http://tail-f.com/ned/cisco-ios-xr">
            <?if {routing-protocol='static'}?>
              <static>
                <address-family>
                  <ipv4>
                    <unicast>
                      <?foreach {static-route} ?>
                        <routes>
                          <net>{prefix}</net>
                          <interface>{interface}</interface>
                          <address>172.31.{../link-id}.6</address>
                        </routes>
                      <?end?>
                    </unicast>
                  </ipv4>
                </address-family>
              </static>
            <?end?>
            <?if {routing-protocol='bgp'}?>
              <bgp>
                <bgp-no-instance>
                  <id>1</id>
                  <vrf>
                    <name>vpn{string(../vpn-id)}</name>
                    <rd>1:{string(../vpn-id)}</rd>
                    <address-family>
                      <ipv4>
                        <unicast>
                          <redistribute>
                            <connected/>
                            <static/>
                          </redistribute>
                        </unicast>
                      </ipv4>
                    </address-family>
                    <neighbor>
                      <id>172.31.{link-id}.6</id>
                      <address-family>
                        <ipv4>
                          <unicast>
                            <route-policy>
                              <direction>in</direction>
                              <name>test</name>
                            </route-policy>
                            <as-override/>
                            <default-originate/>
                          </unicast>
                        </ipv4>
                      </address-family>
                    </neighbor>
                  </vrf>
                </bgp-no-instance>
              </bgp>
            <?end?>
          </router>

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

Step 8

Save the file.

Step 9

Connect to the NSO CLI and reload the packages.

```
student@student-vm:~/nso-run/packages/l3vpn/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 vlan
    result true
}
```

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 l3vpn-template.xml, and the **packages reload** operation was successful.

Task 3: Deploy a Service Instance

The purpose of this task is to deploy a service instance based on your newly installed service.

Activity

Complete these steps:

Step 1

Connect to NSO CLI and enter the configuration mode.

```
student@student-vm:~/nso-run/packages$ ncs_cli -Cu admin
admin@ncs config
Entering configuration mode terminal
admin@ncs(config)
```

Step 2

Add a new customer **ACME** and commit the changes.

```
admin@ncs(config) customers customer ACME
admin@ncs(config-customer-ACME) top
admin@ncs(config) commit
Commit complete.
admin@ncs(config)
```

Step 3

Provision the first service instance.

```
admin@ncs(config) services l3vpn ACME vpn-id 10001
```

```
admin@ncs(config-l3vpn-ACME) customer ACME
admin@ncs(config-l3vpn-ACME) link 1 link-name Site1 pe-device PE11 interface 0/1 routing-
protocol bgp
admin@ncs(config-link-1) exit
admin@ncs(config-l3vpn-ACME) link 2 link-name Site2 pe-device PE21 interface 0/0/0/1 routing-
protocol bgp
admin@ncs(config-link-2) exit
admin@ncs(config-l3vpn-ACME) link 3 link-name Site3 pe-device PE31 interface 0/1 routing-
protocol static
admin@ncs(config-link-3) static-route 192.168.1.0 mask 255.255.255.0
admin@ncs(config-static-route-192.168.1.0) exit
admin@ncs(config-link-3) static-route 192.168.2.0 mask 255.255.255.0
admin@ncs(config-static-route-192.168.2.0) top
```

Step 4

After entering all service parameters, verify the configuration using the **show configuration** command.

The configuration should match the following output:

```
admin@ncs(config) show configuration
services l3vpn ACME
 vpn-id   10001
 customer ACME
 link 1
  link-name        Site1
  pe-device        PE11
  interface        0/1
  routing-protocol bgp
 !
 link 2
  link-name        Site2
  pe-device        PE21
  interface        0/0/0/1
  routing-protocol bgp
 !
 link 3
  link-name        Site3
  pe-device        PE31
  interface        0/1
  routing-protocol static
  static-route 192.168.1.0
   mask 255.255.255.0
  !
  static-route 192.168.2.0
   mask 255.255.255.0
  !
 !
!
```

Make sure you are at the top configuration level before running the **show configuration** command. The **show configuration** command displays only commands on the same or lower levels in the command hierarchy. You can quickly jump to the top level by issuing the **top** command.

Step 5

Preview the device changes by using the **commit dry run** command.

The configuration should match the following output:

```
admin@ncs(config) commit dry-run
```

```
cli {
    local-node {
        data  devices {
                  device PE11 {
                      config {
                          vrf {
             +                definition vpn10001 {
             +                    description "By NSO: L3VPN - ACME";
             +                    rd 1:10001;
             +                    route-target {
             +                        export 1:10001;
             +                        import 1:10001;
             +                    }
             +                }
                          }
                          interface {
             +                GigabitEthernet 0/1 {
             +                    description "By NSO: L3VPN - ACME - Site1";
             +                    vrf {
             +                        forwarding vpn10001;
             +                    }
             +                    ip {
             +                        address {
             +                            primary {
             +                                address 172.31.1.1;
             +                                mask 255.255.255.252;
             +                            }
             +                        }
             +                    }
             +                }
                          }
                          router {
             +                bgp 1 {
             +                    address-family {
             +                        with-vrf {
             +                            ipv4 unicast {
             +                                vrf vpn10001 {
             +                                    redistribute {
             +                                        connected {
             +                                        }
             +                                        static {
             +                                        }
             +                                    }
             +                                    neighbor 172.31.1.2 {
             +                                        remote-as 65001;
             +                                        activate;
             +                                        allowas-in {
             +                                        }
             +                                        as-override {
             +                                            disable;
             +                                        }
             +                                        default-originate {
             +                                        }
             +                                    }
             +                                }
             +                            }
             +                        }
             +                    }
             +                }
                          }
                      }
                  }
                  device PE21 {
                      config {
                          vrf {
             +                vrf-list vpn10001 {
             +                    description "By NSO: L3VPN - ACME";
             +                   address-family {
             +                       ipv4 {
             +                           unicast {
             +                               import {
             +                                   route-target {
             +                                       address-list 1:10001;
             +                                   }
             +                               }
             +                               export {
             +                                   route-target {
             +                                       address-list 1:10001;
             +                                   }
             +                               }
             +                           }
             +                       }
             +                   }
             +               }
                          }
                          interface {
             +               GigabitEthernet 0/0/0/1 {
             +                   description "By NSO: L3VPN - ACME - Site2";
             +                   vrf vpn10001;
             +                   ipv4 {
             +                       address {
             +                           ip 172.31.2.5;
             +                           mask 255.255.255.252;
             +                       }
             +                   }
             +               }
                          }
                          router {
                              bgp {
             +                   bgp-no-instance 1 {
             +                       vrf vpn10001 {
             +                           rd 1:10001;
             +                           address-family {
             +                               ipv4 {
             +                                   unicast {
             +                                       redistribute {
             +                                           connected {
             +                                           }
             +                                           static {
             +                                           }
             +                                       }
             +                                   }
             +                               }
             +                           }
             +                           neighbor 172.31.2.6 {
             +                               address-family {
             +                                   ipv4 {
             +                                       unicast {
             +                                           route-policy in {
             +                                               name test;
             +                                           }
             +                                           as-override {
             +                                           }
             +                                           default-originate {
             +                                           }
             +                                       }
             +                                   }
             +                               }
             +                           }
             +                       }
             +                   }
                              }
                          }
                      }
```

```
                    }
                    device PE31 {
                        config {
                            vrf {
               +                definition vpn10001 {
               +                    description "By NSO: L3VPN - ACME";
               +                    rd 1:10001;
               +                    route-target {
               +                        export 1:10001;
               +                        import 1:10001;
               +                    }
               +                }
                            }
                            interface {
               +                GigabitEthernet 0/1 {
               +                    description "By NSO: L3VPN - ACME - Site3";
               +                    vrf {
               +                        forwarding vpn10001;
               +                    }
               +                    ip {
               +                        address {
               +                            primary {
               +                                address 172.31.3.1;
               +                                mask 255.255.255.252;
               +                            }
               +                        }
               +                    }
               +                }
                            }
                        }
                    }
                }
                services {
               +    l3vpn ACME {
               +        vpn-id 10001;
               +        customer ACME;
               +        link 1 {
               +            link-name Site1;
               +            pe-device PE11;
               +            interface 0/1;
               +            routing-protocol bgp;
               +        }
               +        link 2 {
               +            link-name Site2;
               +            pe-device PE21;
               +            interface 0/0/0/1;
               +            routing-protocol bgp;
               +        }
               +        link 3 {
               +            link-name Site3;
               +            pe-device PE31;
               +            interface 0/1;
               +            routing-protocol static;
               +            static-route 192.168.1.0 {
               +                mask 255.255.255.0;
               +            }
               +            static-route 192.168.2.0 {
               +                mask 255.255.255.0;
               +            }
               +        }
               +    }
                }
    }
}
```

Step 6

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

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

Step 7

Verify that the service configuration exists and that the XPath was evaluated correctly.

```
admin@ncs show running-config services l3vpn
services l3vpn ACME
 vpn-id   10001
 customer ACME
 link 1
  link-name        Site1
  pe-device        PE11
  interface        0/1
  routing-protocol bgp
 !
 link 2
  link-name        Site2
  pe-device        PE21
  interface        0/0/0/1
  routing-protocol bgp
 !
 link 3
  link-name        Site3
  pe-device        PE31
  interface        0/1
  routing-protocol static
  static-route 192.168.1.0
   mask 255.255.255.0
  !
  static-route 192.168.2.0
   mask 255.255.255.0
  !
 !
!
admin@ncs show running-config services l3vpn | tab
VPN   VPN                LINK  LINK   PE                 ROUTING
NAME  ID     CUSTOMER   ID    NAME   DEVICE  INTERFACE  PROTOCOL  PREFIX       MASK
------------------------------------------------------------------------------------------------
ACME  10001  ACME       1     Site1  PE11    0/1        bgp
                        2     Site2  PE21    0/0/0/1    bgp
                        3     Site3  PE31    0/1        static    192.168.1.0  255.255.255.0
                                                                  192.168.2.0  255.255.255.0
admin@ncs exit
```

As you can see from the preceding output, your commit was successful, which means that the XPath was evaluated correctly.

Step 8

You can also preview the XPath entries for the l3vpn service instance ACME you created.

```
admin@ncs show running-config services l3vpn ACME | display xpath
/services/l3vpn:l3vpn[vpn-name='ACME']/vpn-id 10001
/services/l3vpn:l3vpn[vpn-name='ACME']/customer ACME
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='1']/link-name Site1
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='1']/pe-device PE11
```

```
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='1']/interface 0/1
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='1']/routing-protocol bgp
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='2']/link-name Site2
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='2']/pe-device PE21
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='2']/interface 0/0/0/1
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='2']/routing-protocol bgp
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='3']/link-name Site3
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='3']/pe-device PE31
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='3']/interface 0/1
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='3']/routing-protocol static
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='3']/static-route[prefix='192.168.1.0']/
mask 255.255.255.0
/services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='3']/static-route[prefix='192.168.2.0']/
mask 255.255.255.0
```

Step 9

Exit NSO CLI.

```
admin@ncs exit
student@student-vm:~$
```

Step 10

You can use the **ncs_cmd** utility using the **xe** command (short for XPath evaluate) to evaluate custom XPath expressions. In this example, you can try evaluating an XPath expression that references an existing vpn-id (vpn-name ACME and link-id 1) and a non-existing vpn-id (ABME).

```
student@student-vm:~$ ncs_cmd -c "xe /services/l3vpn:l3vpn[vpn-name='ACME']/link[link-id='1']"
1Site1PE110/1bgp
student@student-vm:~$ ncs_cmd -c "xe /services/l3vpn:l3vpn[vpn-name='ABME']"

student@student-vm:~$ ncs_cmd -c "xe boolean(/services/l3vpn:l3vpn[vpn-name='ABME'])"
false
student@student-vm:~$
```

If the XML node does not exist, a response without the content is returned. If you want to test whether a node exists or not, you can test that by using the XPath function **boolean**. An empty node-set is always false, and a non-empty is always true.

Activity Verification

You have completed this task when you attain this result:

- The service has been successfully deployed.