Network Cloud Engine

Open Programmability
Training and Certification

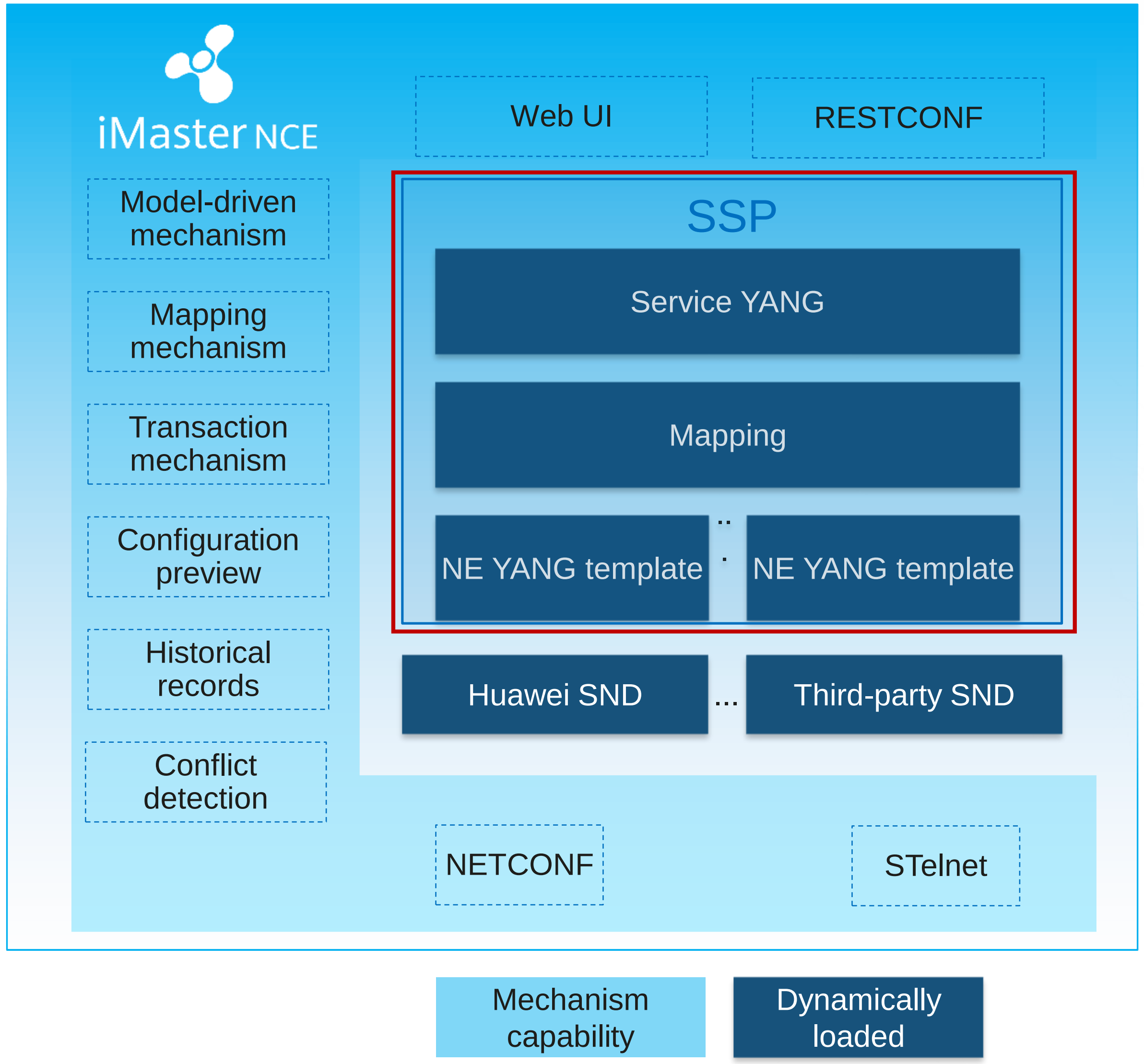
Building SSP Packages

Network Cloud Engine

01

Getting Familiar with SSP

SSP Package



Definition

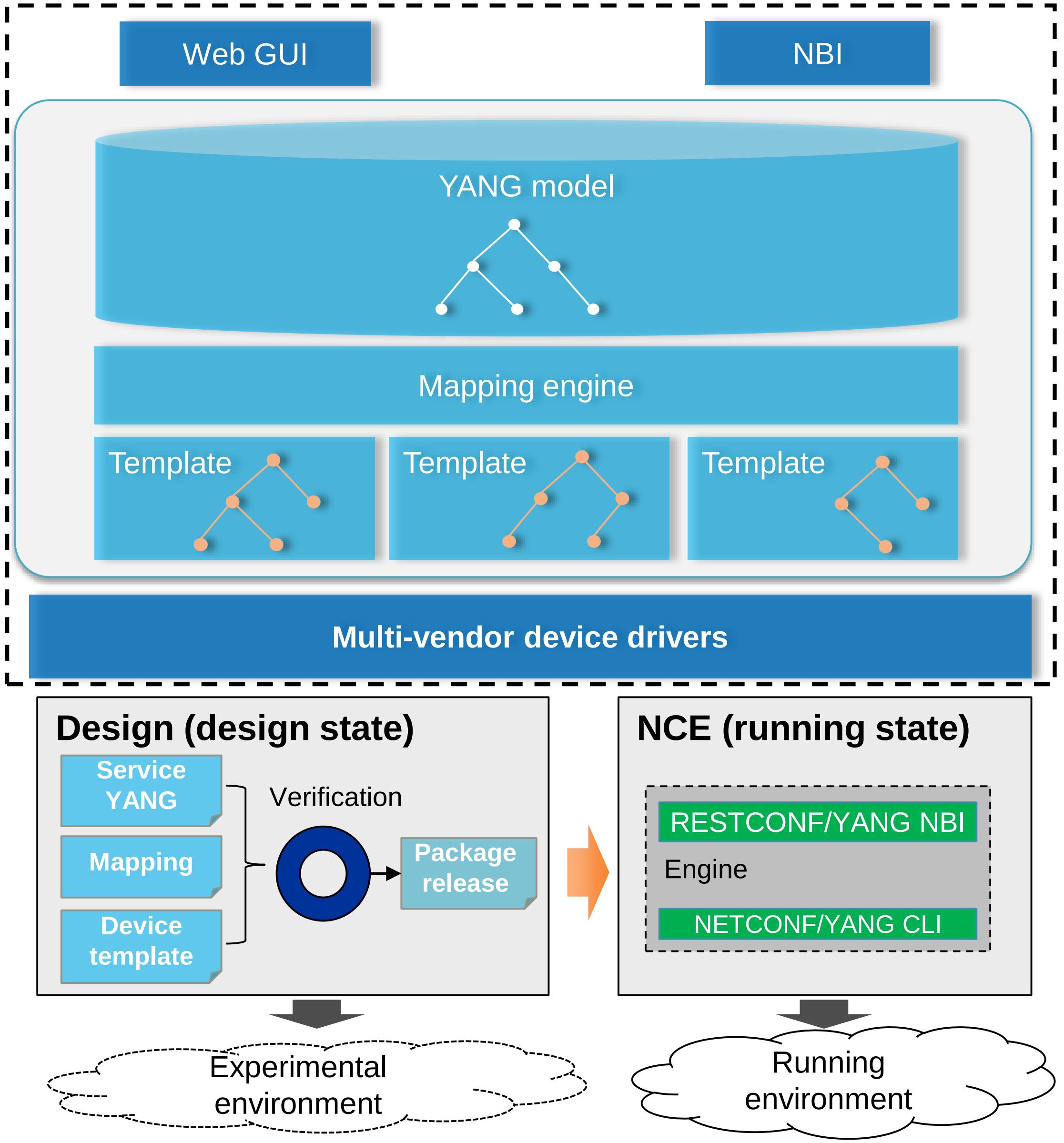
SSP: refers to Specific Service Plug-in.
Service: an application that is designed for one or more device models and uses the capabilities of one or more devices to achieve a certain function.
An SSP package defines the YANG model, Python mapping script, and Jinja2 template file for configuring network-layer services.

- The service YANG model describes service parameters and is constructed based on service input.
- The Python mapping script describes how to process submitted data and fill the data into a Jinja2 template.
- A Jinja2 template describes the data structure of an NE. In the template, variables reference service parameters or new parameters obtained through Python service processing. In addition, Jinja2 syntax is used to perform operations such as interpolation, condition-based determination, and looping.

Function

Implements agile service development and provisioning by loading an SSP package. After an SSP package is imported in the AOC, you can configure network services based on the SSP model and view NE-layer configurations that are decomposed from services.

Mapping Mechanism



Concept

- **Mapping mechanism**
  Maps a service model to an NE-layer model by verifying Python service logic and converting parameters in the SSP package.

Development

- **Building an SSP package**
  Use YANG to define northbound input parameters based on service scenarios, use Python to compile service logic and convert and map parameters, and extract parameter templates to be delivered to devices based on service scenarios.
- **Development mode**
  The design state and running state are involved. The design state is responsible for design and development. After the design is complete, data is imported to NCE for running.

SSP Package Structure

Network Cloud Engine

```
SSP_demo/
    pkg.json
    bin/
        makeFile.bat
        makeFile.sh
        pkg-tool.jar
        readme.txt
    doc/
    key/
    python/
        demo/
            demo.py
    resources/
        logger.conf
    template/
        demo/
            servicepoint.j2
    test/
        demo/
            test_demo_service.py
    yang/
        demo.yang
```

- **pkg.json**
  Package configuration file, which is used to set basic attributes and callback hooks of the current software package.

- **bin/**
  Stores executable scripts, including tools and scripts for packing.

- **python/**
  Stores Python code, including Python scripts used to implement service callback logic.

- **template/**
  Stores Jinja2 template files, including the templates used to generate NETCONF packets after mapping.

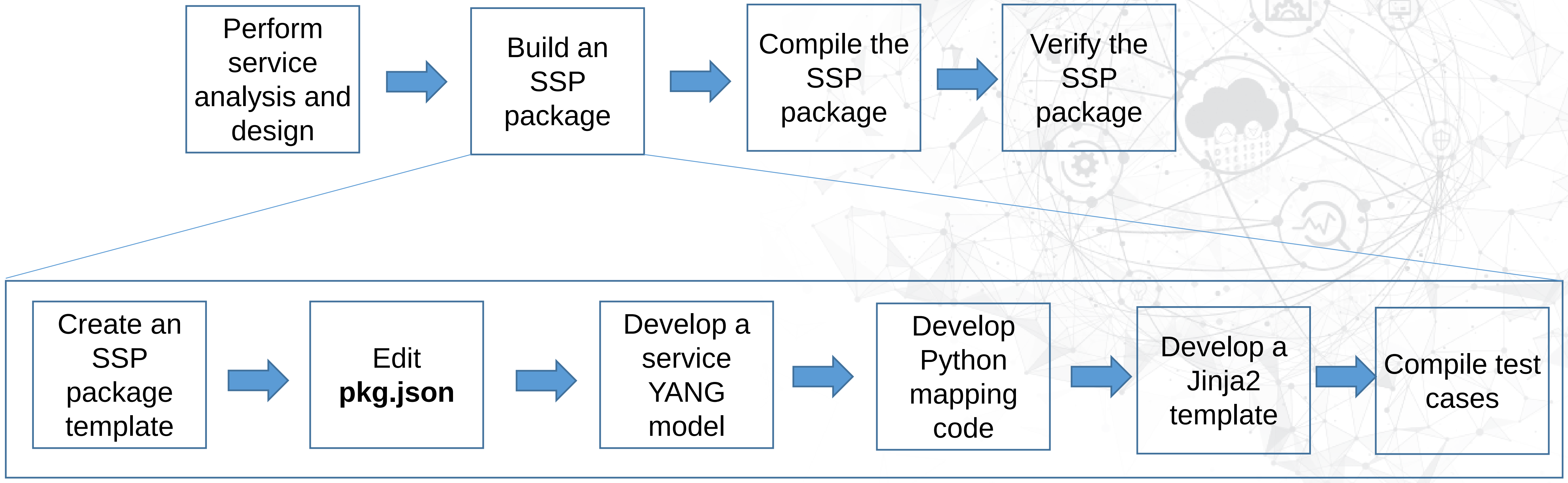
- **test/**
  Stores unit test code, including scripts used to perform local unit tests on software package functions.

- **yang/**
  Stores service YANG models.

Initial directory structure of the software package

HUAWEI

Development Process

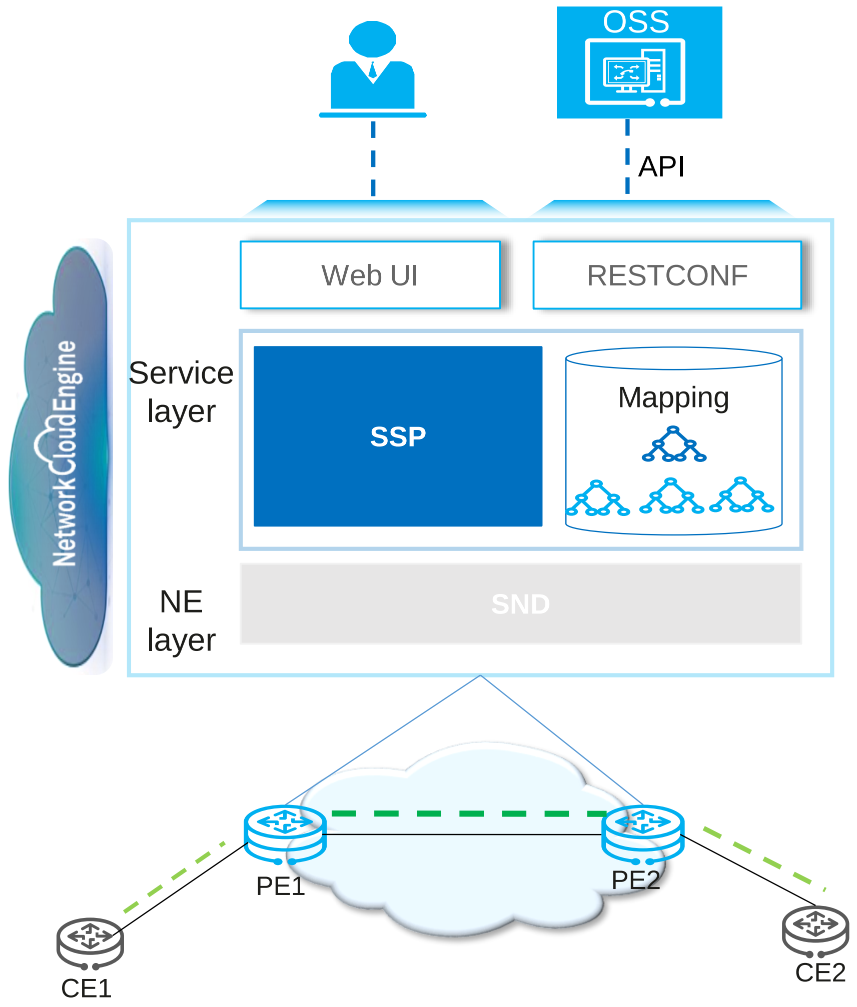

Perform service analysis and design
Build an SSP package
Compile the SSP package
Verify the SSP package
Create an SSP package template
Edit **pkg.json**
Develop a service YANG model
Develop Python mapping code
Develop a Jinja2 template
Compile test cases

Network Cloud Engine

02

Service Analysis and Design

L3VPN Service



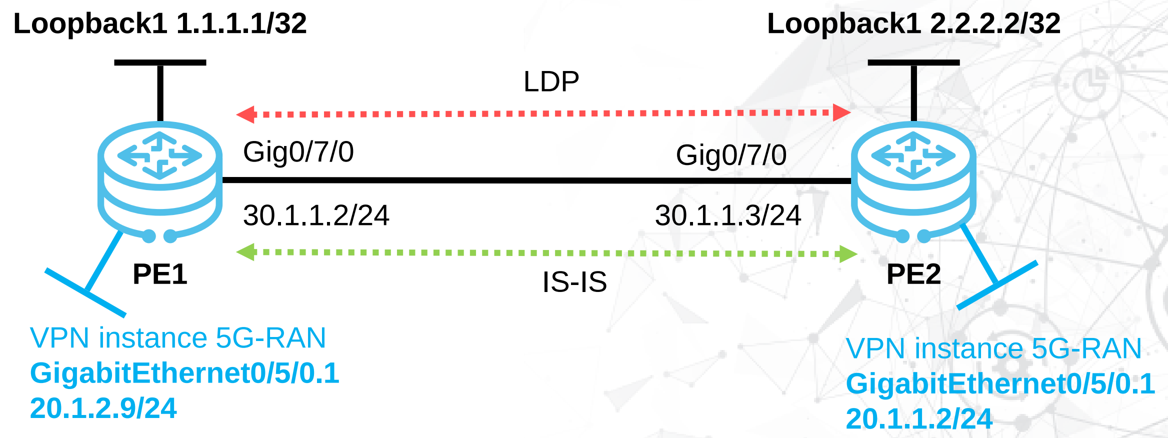
Objective

Establish a VPN between PE1 and PE2. CE1 and CE2 communicate with each other through the established VPN.

Environment

- **IDE**: local or online IDE environment
- **AOC**: local or online AOC environment
- **Devices**: two PEs (NE8000 M8 routers)

Analyzing L3VPN Service Configurations (1/3)



1.1 Basic configuration of PE1

- Create loopback1, which is used as the BGP connection interface.

```
<PE1> system-view
[PE1] interface loopback1
[PE1-LoopBack1] ip address 1.1.1.1 32
[PE1-LoopBack1] quit
```

- Bind an IP address to the physical interface to establish a connection with PE1.

```
[PE1] interface GigabitEthernet0/7/0
[PE1-GigabitEthernet0/7/0] undo shutdown
[PE1-GigabitEthernet0/7/0] ip address 30.1.1.2
255.255.255.0
[PE1-GigabitEthernet0/7/0] quit
```

- Configure IS-IS.

```
[PE1] isis 1
[PE1-isis-1] is-level level-2
[PE1-isis-1] network-entity 49.0001.0010.0000.0001.00
[PE1-isis-1] quit
[PE1] commit
[PE1] interface GigabitEthernet0/7/0
[R1-GigabitEthernet0/7/0] isis enable 1
[R1-GigabitEthernet0/7/0] isis cost 100
[R1-GigabitEthernet0/7/0] quit
```

- Configure MPLS and enable it on the connected physical interface.

```
[PE1] mpls lsr-id 1.1.1.1
[PE1] mpls
[PE1-mpls] quit
[PE1] mpls ldp
[PE1-mpls-ldp] ipv4-family
[PE1-mpls-ldp] quit
[PE1] commit
[PE1] interface GigabitEthernet0/7/0
[R1-GigabitEthernet0/7/0] mpls
[R1-GigabitEthernet0/7/0] mpls ldp
[R1-GigabitEthernet0/7/0] quit
```

- Configure BGP.

```
[PE1] bgp 100
[PE1-bgp] router-id 1.1.1.1
[PE1-bgp] peer 2.2.2.2 as-number 100
[PE1-bgp] peer 2.2.2.2 connect-
interface loopback 1
[PE1-bgp] ipv4-family vpnv4
[PE1-bgp-af-vpnv4] policy vpn-target
[PE1-bgp-af-vpnv4] peer 2.2.2.2 enable
[PE1-bgp-af-vpnv4] quit
[PE1-bgp] ipv4-family unicast
[PE1-bgp-af-unicast] peer 2.2.2.2 enable
[R1-bgp] quit
[R1] commit
```

- Configure BGP.

```
[PE1] bgp 100
[PE1-bgp] router-id 2.2.2.2
[PE1-bgp] peer 1.1.1.1 as-number 100
[PE1-bgp] peer 1.1.1.1 connect-
interface loopback 1
[PE1-bgp] ipv4-family vpnv4
[PE1-bgp-af-vpnv4] policy vpn-target
[PE1-bgp-af-vpnv4] peer 1.1.1.1 enable
[PE1-bgp-af-vpnv4] quit
[PE1-bgp] ipv4-family unicast
[PE1-bgp-af-unicast] peer 1.1.1.1 enable
[R1-bgp] quit
[R1] commit
```

1.2 Basic configuration of PE2

- Create loopback1, which is used as the BGP connection interface.

```
<PE2> system-view
[PE2] interface loopback1
[PE2-LoopBack1] ip address 2.2.2.2 32
[PE2-LoopBack1] quit
```

- Bind an IP address to the physical interface to establish a connection with PE2.

```
[PE2] interface GigabitEthernet0/7/0
[PE2-GigabitEthernet0/7/0] undo shutdown
[PE2-GigabitEthernet0/7/0] ip address 30.1.1.3
255.255.255.0
[PE2-GigabitEthernet0/7/0] quit
```

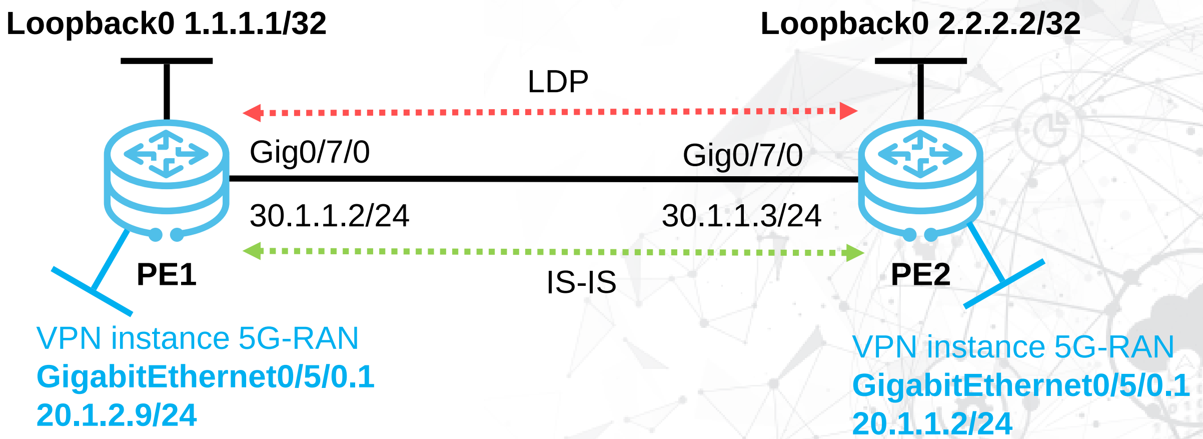
- Configure IS-IS.

```
[PE2] isis 1
[PE2-isis-1] is-level level-2
[PE2-isis-1] network-entity 49.0001.0020.0000.0003.00
[PE2-isis-1] quit
[PE2] commit
[PE2] interface GigabitEthernet0/7/0
[PE2-GigabitEthernet0/7/0] isis enable 1
[PE2-GigabitEthernet0/7/0] isis cost 100
[PE2-GigabitEthernet0/7/0] quit
```

- Configure MPLS and enable it on the connected physical interface.

```
[PE2] mpls lsr-id 1.1.1.1
[PE2] mpls
[PE2-mpls] quit
[PE2] mpls ldp
[PE2-mpls-ldp] ipv4-family
[PE2-mpls-ldp] quit
[PE2] commit
[PE2] interface GigabitEthernet0/7/0
[PE2-GigabitEthernet0/7/0] mpls
[PE2-GigabitEthernet0/7/0] mpls ldp
[PE2-GigabitEthernet0/7/0] quit
```

Analyzing L3VPN Service Configurations (2/3)



1.3 VPN configuration of PE1:

- Configure a VPN instance.

```
[PE1] ip vpn-instance 5G-RAN
[PE1-vpn-instance-A] ipv4-family
[PE1-vpn-instance-A-af-ipv4] route-distinguisher 100:1
[PE1-vpn-instance-A-af-ipv4] vpn-target 100:11 both
[PE1-vpn-instance-A-af-ipv4] quit
[PE1-vpn-instance-A] quit
[PE1] commit
```

- Bind the interface to the VPN instance.

```
[PE1] interface GigabitEthernet0/5/0.1
[PE1- GigabitEthernet0/5/0.1] vlan-type dot1q 20
[PE1- GigabitEthernet0/5/0.1] ip binding vpn-instance 5G-RAN
[PE1- GigabitEthernet0/5/0.1] ip address 20.1.2.9 255.255.255.0
[PE1- GigabitEthernet0/5/0.1] quit
```

- Import routes to the BGP VPN instance.

```
[PE1] bgp 100
[PE1-bgp] ipv4-family vpn-instance 5G-RAN
[PE1-bgp-A] peer 2.2.2.2 as-number 100
[PE1-bgp-A] import-route direct
[PE1-bgp-A] import-route static
[PE1-bgp-A] commit
```

1.4 VPN configuration of PE2:

- **Configure a VPN instance.**

```
[PE1] ip vpn-instance 5G-RAN
[PE1-vpn-instance-A] ipv4-family
[PE1-vpn-instance-A-af-ipv4] route-distinguisher 100:1
[PE1-vpn-instance-A-af-ipv4] vpn-target 100:11 both
[PE1-vpn-instance-A-af-ipv4] quit
[PE1-vpn-instance-A] quit
[PE1] commit
```

- Bind the interface to the VPN instance.

```
[PE1] interface GigabitEthernet0/5/0.1
[PE1- GigabitEthernet0/5/0.1] vlan-type dot1q 20
[PE1- GigabitEthernet0/5/0.1] ip binding vpn-instance 5G-RAN
[PE1- GigabitEthernet0/5/0.1] ip address 20.1.1.2 255.255.255.0
[PE1- GigabitEthernet0/5/0.1] quit
```

- Import routes to the BGP VPN instance.

```
[PE1] bgp 100
[PE1-bgp] ipv4-family vpn-instance 5G-RAN
[PE1-bgp-A] peer 1.1.1.1 as-number 100
[PE1-bgp-A] import-route direct
[PE1-bgp-A] import-route static
[PE1-bgp-A] commit
```

Analyzing L3VPN Service Configurations (3/3)

1
```
ip vpn-instance 5G-RAN
   ipv4-family
    route-distinguisher 100:1
    vpn-target 100:11 export-extcommunity evpn
    vpn-target 100:11 import-extcommunity evpn
```

2
```
interface GigabitEthernet 0/5/0.1
  vlan-type dot1q 20
  ip binding vpn-instance 5G-RAN
  ip address 20.1.2.9 255.255.255.0
```

3
```
bgp 100
   ipv4-family vpn-instance  5G-RAN
      import-route direct
      import-route static
      peer 2.2.2.2 as-number 100
```

4
```
bgp yang-mode enable
```

1.5 Analyze the configuration commands.

The global VPN configurations have been performed on the device. The configurations that need to be delivered for the VPN service are analyzed as follows:

1. Create VPN instance **5G-RAN** and set the RT and RD.
2. Create a sub-interface, configure the interface address and VLAN, and bind the sub-interface to the VPN instance.
3. Configure VPN routes, import static routes and direct routes, and configure BGP peers.
4. Enable the YANG interface to deliver BGP configurations.

Input Parameters for Network Configuration

Group	Parameter	Value
1	deviceName	PE1
	vrfName	5G-RAN
2	rd	100:01:00
	rt	100:11:00
3	ifName	GigabitEthernet0/5/0.1
	description	connect to pe2
	ip	20.1.2.9
	Subnet mask	255.255.255.0
4	peerAddress	2.2.2.2

2 Sort out the input parameters for network configuration.

Sort out the input parameters for network configuration based on the network service.

1. Specify the device to which a specific VPN service is to be delivered. The device name and VPN name need to be extracted, which are combined into a key to identify the VPN service.
2. Configure the RT and RD for the VPN to filter routing information. The RT and RD need to be extracted.
3. Configure the IPv4 address, bound VPN, and VLAN for an interface. The sub-interface name, sub-interface description, sub-interface IP address, and mask need to be extracted.
4. Configure BGP peers. The IP addresses of BGP peers need to be extracted.

Mapping Between Commands and YANG (1/10)

Network Cloud Engine

1

```
ip vpn-instance 5G-RAN
    ipv4-family
    route-distinguisher 100:1
    vpn-target 100:11 export-extcommunity evpn
    vpn-target 100:11 import-extcommunity evpn
```

⟷ huawei-network-instance.yang

2

```
interface GigabitEthernet 0/5/0.1
  vlan-type dot1q 20
  ip binding vpn-instance 5G-RAN
  ip address 20.1.2.9 255.255.255.0
```

⟷ huawei-ifm.yang

3

```
bgp 100
   ipv4-family vpn-instance  5G-RAN
      import-route direct
      import-route static
      peer 2.2.2.2 as-number 100
```

⟷ huawei-bgp.yang

4

```
bgp yang-mode enable
```

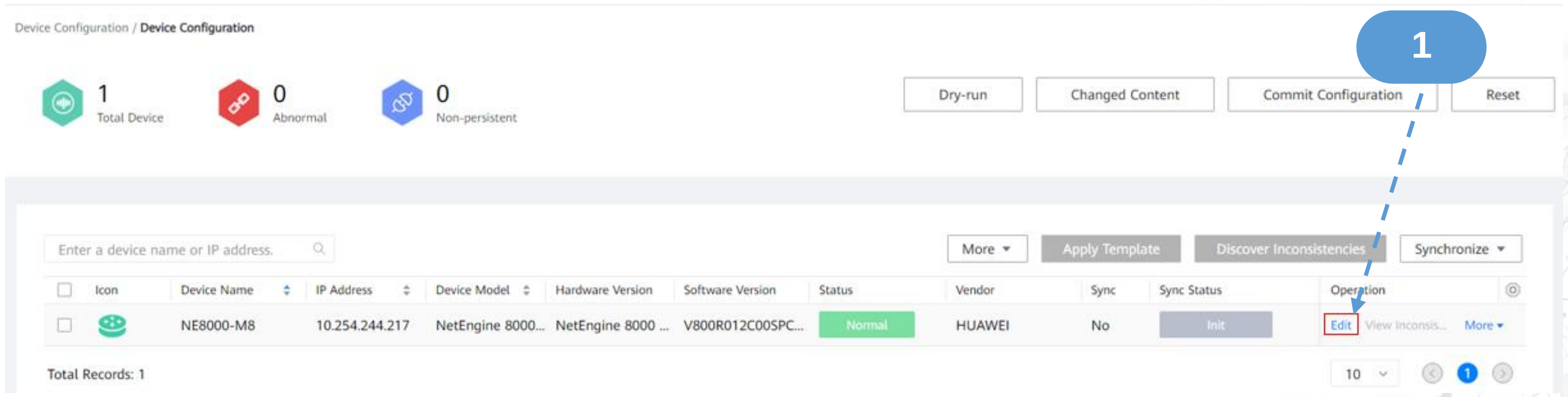
3.1 Forward method

Analyze how to deliver commands to devices through NETCONF. That is, analyze the YANG model mapped to the command and the XML data corresponding to the YANG model.

- Forward method: This method is applicable to scenarios where there is no real device, the mapping between commands and YANG is known, and the mapping content can be deduced based on the commands.

HUAWEI

Mapping Between Commands and YANG (2/10)



Device Configuration / **Device Configuration**

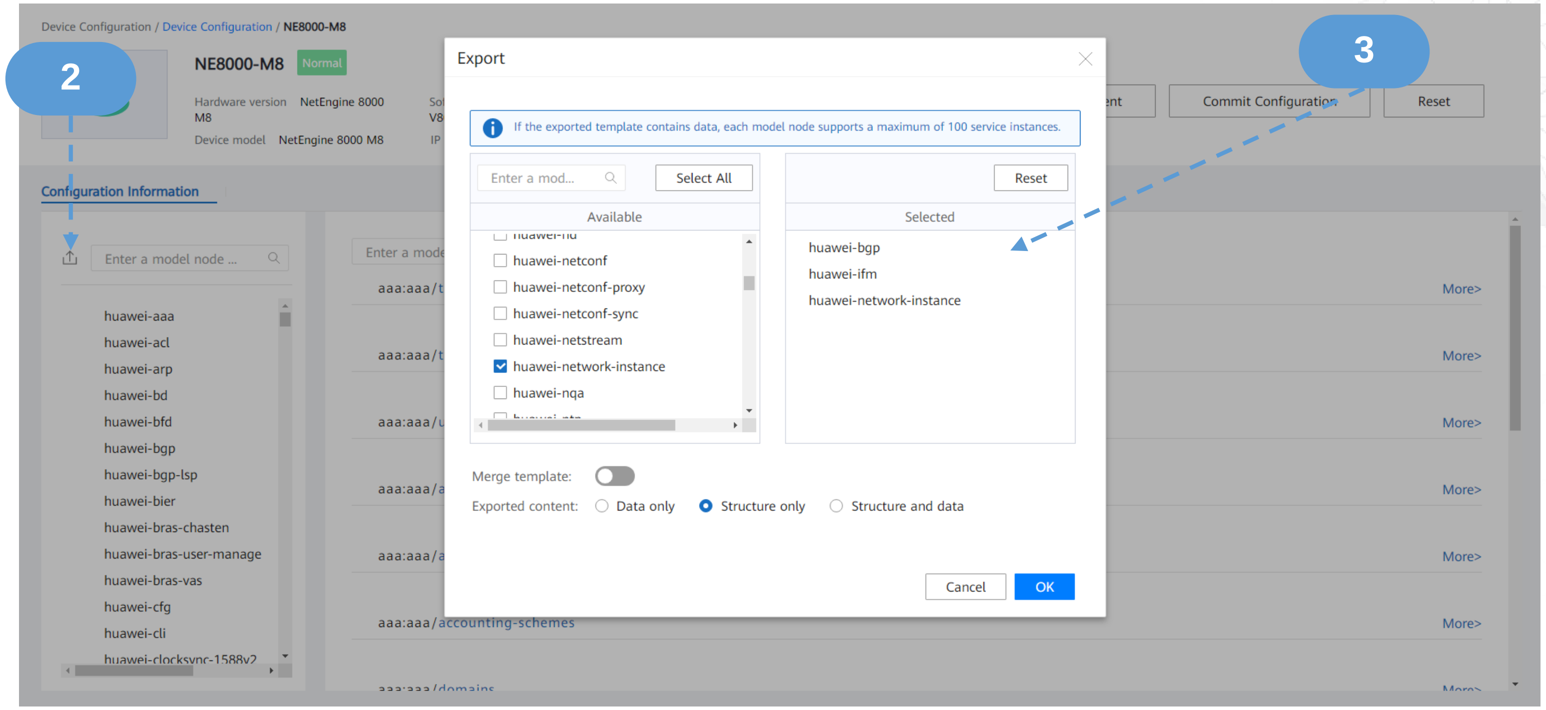
1 Total Device | 0 Abnormal | 0 Non-persistent

Dry-run | Changed Content | Commit Configuration | Reset

Enter a device name or IP address. | More | Apply Template | Discover Inconsistencies | Synchronize

Icon	Device Name	IP Address	Device Model	Hardware Version	Software Version	Status	Vendor	Sync	Sync Status	Operation
	NE8000-M8	10.254.244.217	NetEngine 8000...	NetEngine 8000 ...	V800R012C00SPC...	Normal	HUAWEI	No	Init	Edit View Inconsis... More

Total Records: 1



Device Configuration / Device Configuration / NE8000-M8

NE8000-M8 Normal

Hardware version NetEngine 8000 M8

Device model NetEngine 8000 M8

Configuration Information

Enter a model node ...

huawei-aaa
huawei-acl
huawei-arp
huawei-bd
huawei-bfd
huawei-bgp
huawei-bgp-lsp
huawei-bier
huawei-bras-chasten
huawei-bras-user-manage
huawei-bras-vas
huawei-cfg
huawei-cli
huawei-clocksync-1588v2

Export

If the exported template contains data, each model node supports a maximum of 100 service instances.

Available	Selected
huawei-netconf	huawei-bgp
huawei-netconf-proxy	huawei-ifm
huawei-netconf-sync	huawei-network-instance
huawei-netstream	
✓ huawei-network-instance	
huawei-nqa	

Merge template:

Exported content: Data only [✓ Structure only] Structure and data

Cancel | OK

aaa:aaa/accounting-schemes

aaa:aaa/domains

3.1 Forward method

1. Choose **Device Configuration** > **Device Configuration** from the main menu and click **Edit** in the **Operation** column of the NE8000 M8.
2. In this example, the YANG files **huawei-ifm**, **huawei-network-instance**, and **huawei-bgp** need to be invoked for the southbound configuration of the network service. Click .
3. Select **huawei-ifm**, **huawei-network-instance**, and **huawei-bgp**. In the **Export** dialog box that is displayed, enable **Merge template**, set the export mode to **Structure only**, and export the template.

Mapping Between Commands and YANG (3/10)

combinedTemplate_20210226044527160.zip

huawei-network-instance

```
<network-instance xmlns="urn:huawei:yang:huawei-network-instance">
 <instances>
  <instance>
   <name/>
   <afs xmlns="urn:huawei:yang:huawei-l3vpn">
    <af>
     <type/>
     <vpn-targets>
      <vpn-target>
       <value/>
       <type/>
      </vpn-target>
      <vpn-target>
       <value/>
       <type/>
      </vpn-target>
     </vpn-targets>
     ...
     <route-distinguisher/>
     <tunnel-policy/>
    </af>
   </afs>
   <bgp xmlns="urn:huawei:yang:huawei-bgp">
    <base-process>
     <afs>
      <af>
       <type/>
       <ipv4-unicast>
        <import-routes>
         <import-route>
        </import-routes>
       </ipv4-unicast>
      </af>
     </afs>
    </base-process>
   </bgp>
  </instance>
  ...
 </instances>
</network-instance>
```

huawei-ifm

```
<ifm xmlns="urn:huawei:yang:huawei-ifm">
 <interfaces>
  <interface>
   <name/>
   ...
   <ipv4 xmlns="urn:huawei:yang:huawei-ip">
    <addresses>
     <address>
      <ip/>
      <mask/>
      <type/>
     </address>
    </addresses>
   </ipv4>
   ...
   <vrf-name/>
   ...
  </interface>
</ifm>
```

3.1 Forward method

1. Decompress the exported package to the local PC.
2. Open the file in text mode. You can view the format of each feature in the file and extract the YANG model corresponding to the commands.

huawei-bgp

```
<bgp xmlns="urn:huawei:yang:huawei-bgp">
 <global>
  <yang-enable/>
 </global>
 <base-process>
  <enable/>
  <as/>
 </base-process>
</bgp>
```

Mapping Between Commands and YANG (4/10)

1
```
ip vpn-instance 5G-RAN
    ipv4-family
    route-distinguisher 100:1
    vpn-target 100:11 export-extcommunity
    vpn-target 100:11 import-extcommunity
```

2
```
interface GigabitEthernet 0/5/0.1
  vlan-type dot1q 20
  ip binding vpn-instance 5G-RAN
  ip address 20.1.2.9 255.255.255.0
```

3
```
bgp 100
   ipv4-family vpn-instance  5G-RAN
      import-route direct
      import-route static
      peer 2.2.2.2 as-number 100
```

4
```
bgp yang-mode enable
```

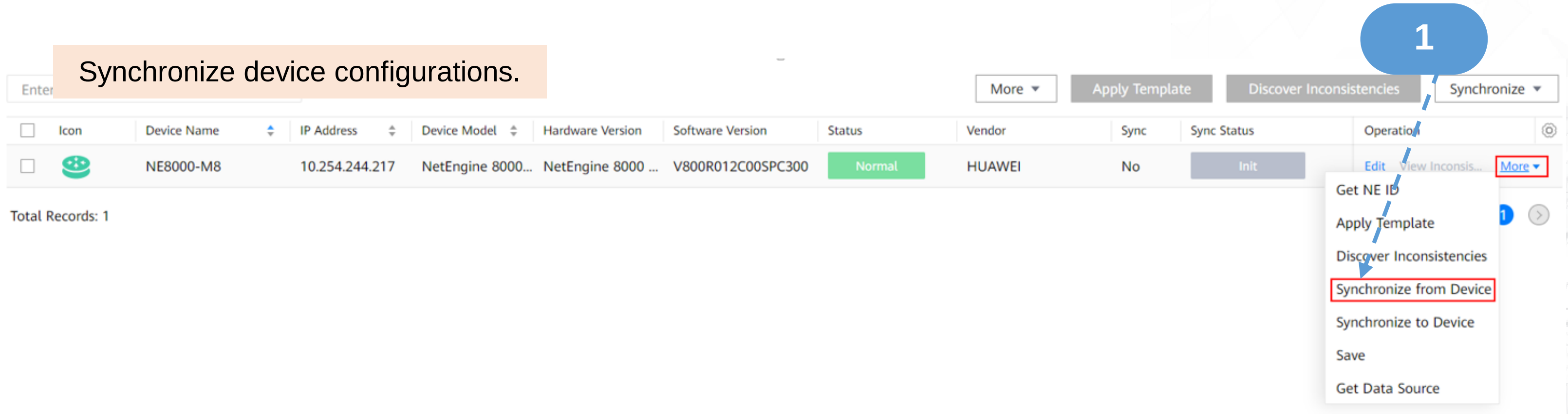
3.2 Reverse method

Analyze how to deliver commands to devices through NETCONF. That is, analyze the YANG model mapped to the command and the XML data corresponding to the YANG model.

- Reverse method: This method is applicable to scenarios where real devices are deployed. After devices are managed, device configurations are synchronized, the preceding commands are configured on the device, and the XML data template corresponding to the commands is exported based on the NE configuration difference discovery capability.

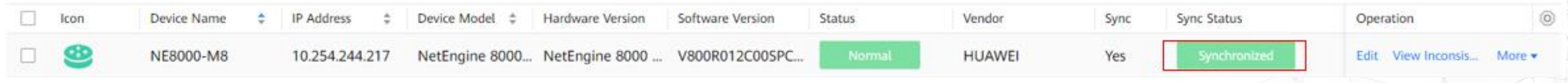
Mapping Between Commands and YANG (5/10)

Synchronize device configurations.



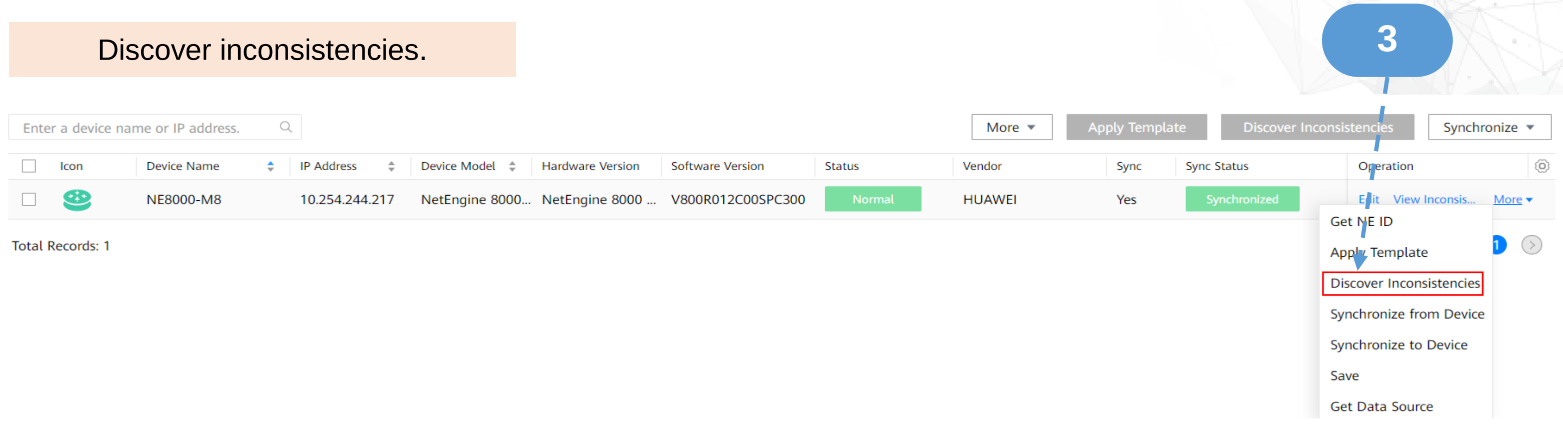
Icon	Device Name	IP Address	Device Model	Hardware Version	Software Version	Status	Vendor	Sync	Sync Status	Operation
	NE8000-M8	10.254.244.217	NetEngine 8000...	NetEngine 8000 ...	V800R012C00SPC300	Normal	HUAWEI	No	Init	Edit View Inconsis... More

Total Records: 1



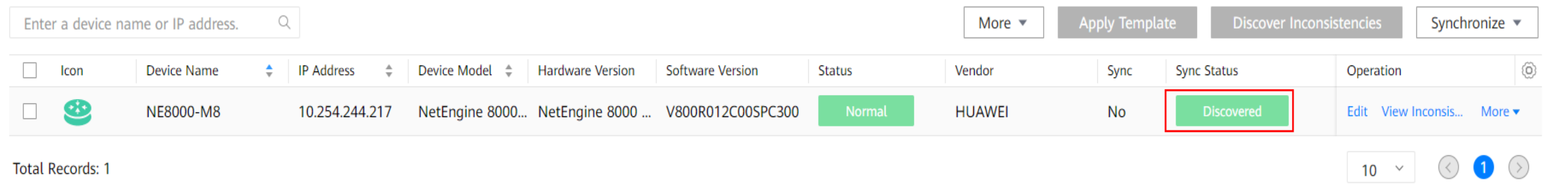
Icon	Device Name	IP Address	Device Model	Hardware Version	Software Version	Status	Vendor	Sync	Sync Status	Operation
	NE8000-M8	10.254.244.217	NetEngine 8000...	NetEngine 8000 ...	V800R00SPC...	Normal	HUAWEI	Yes	Synchronized	Edit View Inconsis... More

Discover inconsistencies.



Icon	Device Name	IP Address	Device Model	Hardware Version	Software Version	Status	Vendor	Sync	Sync Status	Operation
	NE8000-M8	10.254.244.217	NetEngine 8000...	NetEngine 8000 ...	V800R012C00SPC300	Normal	HUAWEI	Yes	Synchronized	Edit View Inconsis... More

Total Records: 1



Icon	Device Name	IP Address	Device Model	Hardware Version	Software Version	Status	Vendor	Sync	Sync Status	Operation
	NE8000-M8	10.254.244.217	NetEngine 8000...	NetEngine 8000 ...	V800R012C00SPC300	Normal	HUAWEI	No	Discovered	Edit View Inconsis... More

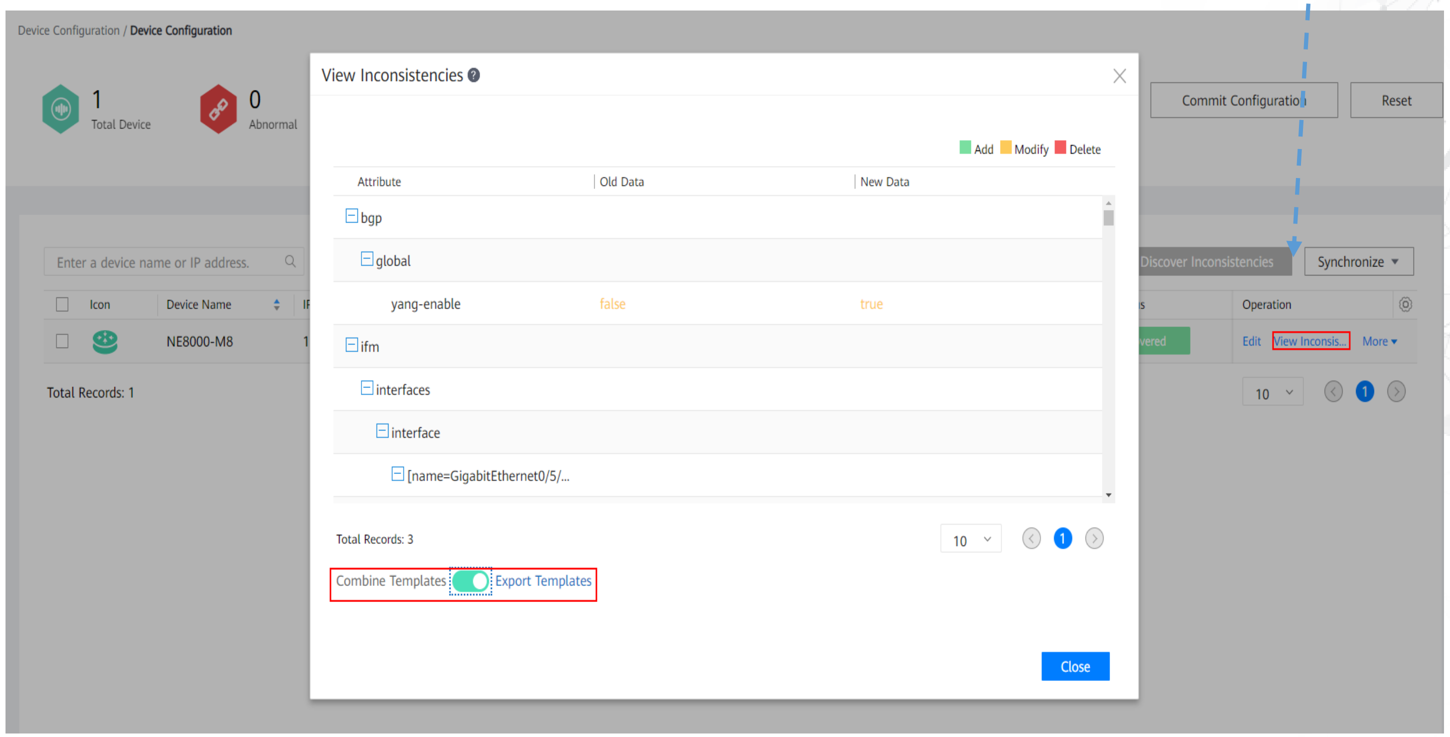
Total Records: 1

3.2 Reverse method

1. On the **Device Configuration** page, click **Synchronize** under **More** in the **Operation** column, synchronize device configurations to the controller to ensure data consistency between the controller and device.
2. Configure commands on the device.
3. Click **Discover Inconsistencies** under **More** in the **Operation** column to compare the differences after the configuration commands are executed.

Mapping Between Commands and YANG (6/10)

Export a template.



Device Configuration / Device Configuration

1 Total Device | 0 Abnormal

View Inconsistencies

🟩 Add 🟨 Modify 🟥 Delete

Attribute	Old Data	New Data
bgp		
global		
yang-enable	false	true
ifm		
interfaces		
interface		
[name=GigabitEthernet0/5/...		

Total Records: 3

Combine Templates Export Templates

1

3.2 Reverse method

1. After difference discovery, click **View Inconsistencies** in the **Operation** column to view and export the differences.

Mapping Between Commands and YANG (7/10)

View the exported template and retain only the configuration to be delivered.

```
<bgp xmlns="urn:huawei:yang:huawei-bgp" xmlns:ns0="urn:ietf:params:xml:ns:netconf:base:1.0"
ns0:operation="merge">
  <global>
    <yang-enable>true</yang-enable>
  </global>
</bgp>
<ifm xmlns="urn:huawei:yang:huawei-ifm" xmlns:ns0="urn:ietf:params:xml:ns:netconf:base:1.0"
ns0:operation="merge">
  <interfaces>
    <interface>
      <name>GigabitEthernet0/5/0.1</name>
      ...
      <ipv4 xmlns="urn:huawei:yang:huawei-ip">
        <addresses>
          <address>
            <ip>20.1.2.9</ip>
            <mask>255.255.255.0</mask>
            <type>main</type>
          </address>
        </addresses>
      </ipv4>
     ...
      <vrf-name>5G-RAN</vrf-name>
      ...
    </interface>
  </interfaces>
</ifm>
<network-instance xmlns="urn:huawei:yang:huawei-network-instance"
xmlns:ns0="urn:ietf:params:xml:ns:netconf:base:1.0" ns0:operation="merge">
  <instances>
    <instance>
      <name>5G-RAN</name>
      <bgp xmlns="urn:huawei:yang:huawei-bgp">
        <base-process>
          <peers>
            <peer>
              <address>2.2.2.2</address>
              ...
              <remote-as>100</remote-as>
              ...
              </afs>
             ...
            </peer>
          </peers>
```

3.2 Reverse method

1. Open the exported template in text mode and retain only the configuration to be delivered.

Mapping Between Commands and YANG (8/10)

View the exported template and retain only the configuration to be delivered.

```
        <afs>
          <af>
            <type>ipv4uni</type>
            ...
              <import-routes>
                <import-route>
                  <protocol>direct</protocol>
                  <process-id>0</process-id>
                </import-route>
                <import-route>
                  <protocol>static</protocol>
                  <process-id>0</process-id>
                </import-route>
              </import-routes>
             </ipv4-unicast>
          </af>
        </afs>
        ...
      </base-process>
    </bgp>
   ...
    <afs xmlns="urn:huawei:yang:huawei-l3vpn">
      <af>
        <type>ipv4-unicast</type>
        ...
        <label-mode>per-instance</label-mode>
        ...
        <vpn-targets>
          <vpn-target>
            <value>100:11</value>
            <type>export-extcommunity</type>
          </vpn-target>
          <vpn-target>
            <value>100:11</value>
            <type>import-extcommunity</type>
          </vpn-target>
        </vpn-targets>
        ...
        <route-distinguisher>100:1</route-distinguisher>
      </af>
    </afs>
  </instance>
 </instances>
</network-instance>
```

3.2 Reverse method (continued)

1. Open the exported template in text mode and retain only the configuration to be delivered.

Mapping Between Commands and YANG (9/10)

Network Cloud Engine

1

```
ip vpn-instance 5G-RAN
  ipv4-family
  route-distinguisher 100:1
  vpn-target 100:11 export-extcommunity evpn
  vpn-target 100:11 import-extcommunity evpn
```

2

```
interface GigabitEthernet 0/5/0.1
 vlan-type dot1q 20
 ip binding vpn-instance 5G-RAN
 ip address 20.1.2.9 255.255.255.0
```

3

```
bgp 100
  ipv4-family vpn-instance  5G-RAN
    import-route direct
    import-route static
    peer 2.2.2.2 as-number 100
```

4

```
bgp yang-mode enable
```

4

```
<bgp xmlns="urn:huawei:yang:huawei-bgp">
  <global>
    <yang-enable>true</yang-enable>
  </global>
</bgp>
<network-instance xmlns="urn:huawei:yang:huawei-network-instance">
  <instances>
    <instance>
```

1

```
      <name>{{l3vpn.vrfName}}</name>
      {%- if l3vpn.vrfDes %}
      <description>{{vrfDes}}</description>
      {%- endif %}
      <afs xmlns="urn:huawei:yang:huawei-l3vpn">
        <af>
          <type>ipv4-unicast</type>
          <route-distinguisher>{{ l3vpn.rd }}</route-distinguisher>
          <tunnel-policy>LDP</tunnel-policy>
          <vpn-targets>
            <vpn-target>
              <value>{{ l3vpn.rt }}</value>
              <type>export-extcommunity</type>
            </vpn-target>
            <vpn-target>
              <value>{{ l3vpn.rt }}</value>
              <type>import-extcommunity</type>
            </vpn-target>
          </vpn-targets>
        </af>
      </afs>
```

3

```
      <bgp xmlns="urn:huawei:yang:huawei-bgp">
        <base-process>
          <afs>
            <af>
              <type>ipv4uni</type>
              <ipv4-unicast>
                <import-routes>
                  <import-route>
                    <protocol>direct</protocol>
                    <process-id>0</process-id>
                  </import-route>
                  <import-route>
                    <protocol>static</protocol>
                    <process-id>0</process-id>
                  </import-route>
                </import-routes>
              </ipv4-unicast>
            </af>
          </afs>
          <peers>
            <peer>
              <address>{{ l3vpn.peerAddress }}</address>
              <remote-as>100</remote-as>
            </peer>
          </peers>
        </base-process>
      </bgp>
    </instance>
  </instances>
</network-instance>
```

3.3 Map commands to the device YANG model.

Analyze how to deliver commands to devices through NETCONF. That is, analyze the YANG model mapped to the command and the XML data corresponding to the YANG model.

- Forward method: This method is applicable to scenarios where no real device is available and the mapping content is deduced based on the commands.
- Reverse method: This method is applicable to scenarios where real devices are deployed. After devices are managed, device configurations are synchronized, the preceding commands are configured on the device, and the XML data template corresponding to the commands is exported based on the NE configuration difference discovery capability.

HUAWEI

Mapping Between Commands and YANG (10/10)

1

```
ip vpn-instance 5G-RAN
  ipv4-family
  route-distinguisher 100:1
  vpn-target 100:11 export-extcommunity evpn
  vpn-target 100:11 import-extcommunity evpn
```

2

```
interface GigabitEthernet 0/5/0.1
 vlan-type dot1q 20
 ip binding vpn-instance 5G-RAN
 ip address 20.1.2.9 255.255.255.0
```

3

```
bgp 100
  ipv4-family vpn-instance  5G-RAN
    import-route direct
    import-route static
    peer 2.2.2.2 as-number 100
```

2

```
<ifm xmlns="urn:huawei:yang:huawei-ifm">
  <interfaces>
    <interface>
      <name>{{ l3vpn.ifName }}</name>
      <type>GigabitEthernet</type>
      <description>{{ l3vpn.description }}</description>
      <vrf-name>{{ l3vpn.vrfName }}</vrf-name>
      <ipv4 xmlns="urn:huawei:yang:huawei-ip">
        <addresses>
          <address>
            <ip>{{ l3vpn.ip }}</ip>
            <type>main</type>
            <mask>{{ l3vpn.subnet }}</mask>
          </address>
        </addresses>
      </ipv4>
    </interface>
  </interfaces>
</ifm>
```

3.3 Map commands to the device YANG model. (continued)

Analyze how to deliver commands to devices through NETCONF. That is, analyze the YANG model mapped to the command and the XML data corresponding to the YANG model.

- Forward method: This method is applicable to scenarios where no real device is available and the mapping content is deduced based on the commands.
- Reverse method: This method is applicable to scenarios where real devices are deployed. After devices are managed, device configurations are synchronized, the preceding commands are configured on the device, and the XML data template corresponding to the commands is exported based on the NE configuration difference discovery capability.

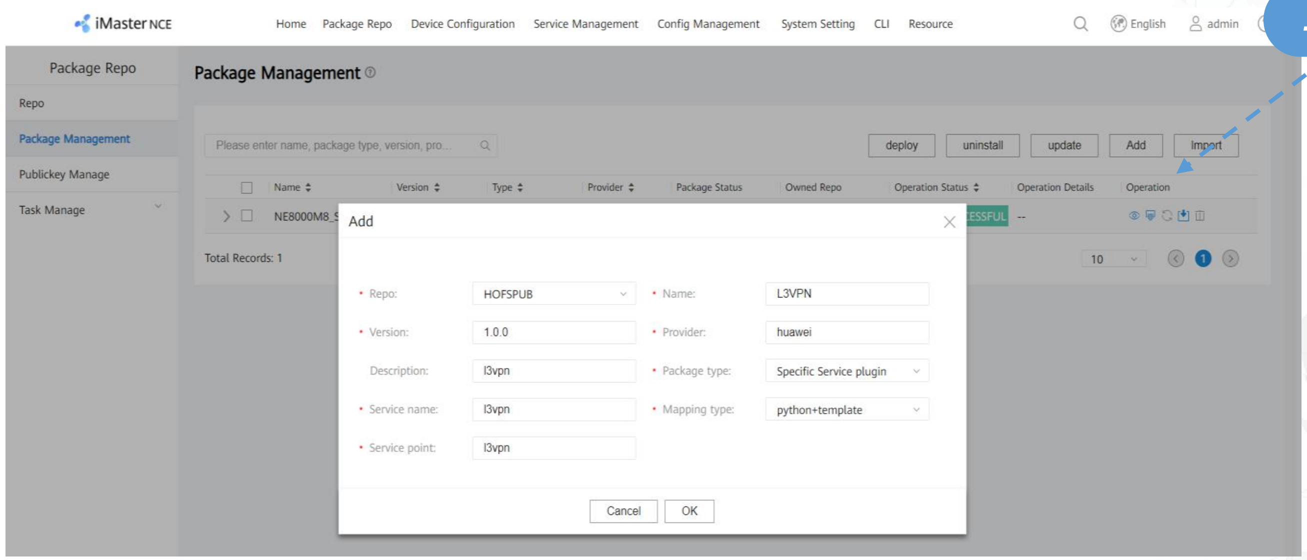
03

Building an SSP Package

Creating an SSP Package Template

NetworkCloud Engine

1.2



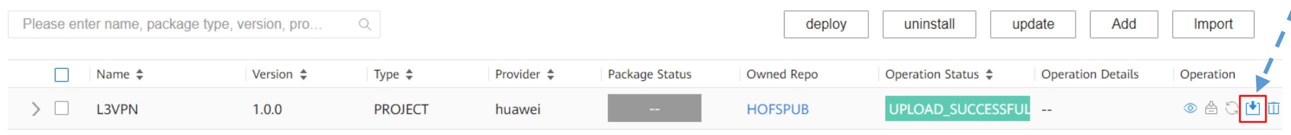
Add

• Repo:	HOFSPUB	• Name:	L3VPN
• Version:	1.0.0	• Provider:	huawei
Description:	l3vpn	• Package type:	Specific Service plugin
• Service name:	l3vpn	• Mapping type:	python+template
• Service point:	l3vpn		

Cancel OK

Parameter	Value
Name	L3VPN
Version	1.0.0
Provider	HUAWEI
Package type	Specific Service plugin
Mapping type	python+template
Service name	L3VPN
Service point	L3VPN

1.4



Name	Version	Type	Provider	Package Status	Owned Repo	Operation Status	Operation Details	Operation
L3VPN	1.0.0	PROJECT	huawei	--	HOFSPUB	UPLOAD_SUCCESSFUL	--	

1 Create an SSP package template.

1. Log in to the system, and click **Package Repo** in the **Quick entry** pane on the homepage. The **Package Repo** page is displayed.
2. In the navigation tree on the left, click **Package Management**. On the **Package Management** page, click **Add**. In the displayed dialog box, set software package parameters.
3. Click **OK**. An SSP package template is generated.
4. Click in the **Operation** column to download the template to the local PC and decompress it.

HUAWEI

Developing a Package Property File

```
{
    "name": "L3VPN",
    "version": "1.0.0",
    "description": "l3vpn",
    "package-type": "ssp",
    "producer": "huawei",
    "service-name":"l3vpn",
    "nce-min-versions":[
        "1.0.0"
    ],

    "hooks": [
      {
        "type": "mapping",
        "key": "l3vpn",
        "python-class-name": "l3vpn.l3vpn.L3VPNService"
      }
    ]
}
```

When a network-layer service is decomposed into an NE-layer service, set **type** to **mapping**.

2 Edit **pkg.json**.

1. Decompress the SSP package **L3VPN.zip** downloaded in the previous step to the IDE project.
2. Open the **pkg.json** file and modify the device information.

Developing a Service YANG Model (1/4)

Group	Parameter	Value
1	deviceName	PE1
1	vrfName	5G-RAN
2	rd	100:01:00
2	rt	100:11:00
3	ifName	GigabitEthernet0/5/0.1
3	description	connect to pe2
3	ip	20.1.2.9
3	Subnet mask	255.255.255.0
4	peerAddress	2.2.2.2

3 Sort out the input parameters for network configuration.

Sort out the input parameters for network configuration based on the network service.

1. Specify the device to which a specific VPN service is to be delivered. The device name and VPN name need to be extracted, which are combined into a key to identify the VPN service.
2. Configure the RT and RD for the VPN to filter routing information. The RT and RD need to be extracted.
3. Configure the IPv4 address, bound VPN, and VLAN for an interface. The sub-interface name, sub-interface description, sub-interface IP address, and mask need to be extracted.
4. Configure BGP peers. The IP addresses of BGP peers need to be extracted.

Developing a Service YANG Model (2/4)

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

    import huawei-ac-applications {
      prefix app;
    }
    import ietf-inet-types {
      prefix inet;
    }

    description
      "The module for L3VPN example.";

    revision 2020-11-05 {
      description "Initial revision.";
    }

    augment "/app:applications"{
      list l3vpn {
        app:application-definition "l3vpn";
  2     key "deviceName vrfName";
        leaf deviceName {
          type string {
            length "1..512";
          }
        }

        leaf vrfName {
          description "Defines a type of service component identifier.";
          type string {
            length 1..255 {
              description "VPN Routing/Forwarding instance name, support 1-255 characters.";
            }
          }
          mandatory true;
        }
```

Module name corresponding to the service name.

Corresponding to the service point.

The device name and VPN name are combined as a key.

3.1 Compile the service YANG model.

1. In the YANG model of the SSP package template, the service name corresponds to the service YANG module name, and the service point corresponds to **app:application-definition** of the service YANG model.
2. Define the service identifier in the YANG model. In this example, the combination of the device name and VPN name is used as the service identifier.
3. Define the RD and RT of VPN services in the YANG model.
4. Define the sub-interface configuration in the YANG model.
5. Define peer information in the YANG model.

Developing a Service YANG Model (3/4)

3
```
leaf rd {
  description "BGP route distinguisher";
  type string;
}

leaf rt {
  description "Route target extended community as per RFC4360";
  type string;
}
```

4
```
leaf ifName {                        <- - - - - - - - -   Sub-interface name.
  type string {
    length "0..255";
  }
}
leaf description {                   <- - - - - - - - -   Sub-interface description.
  type string {
    length "0..242";
  }
}

leaf ip {                            <- - - - - - - - -   Sub-interface IP address.
  type inet:ipv4-address;
}

leaf subnet {                        <- - - - - - - - -   Mask.
  type inet:ipv4-address;
}
```

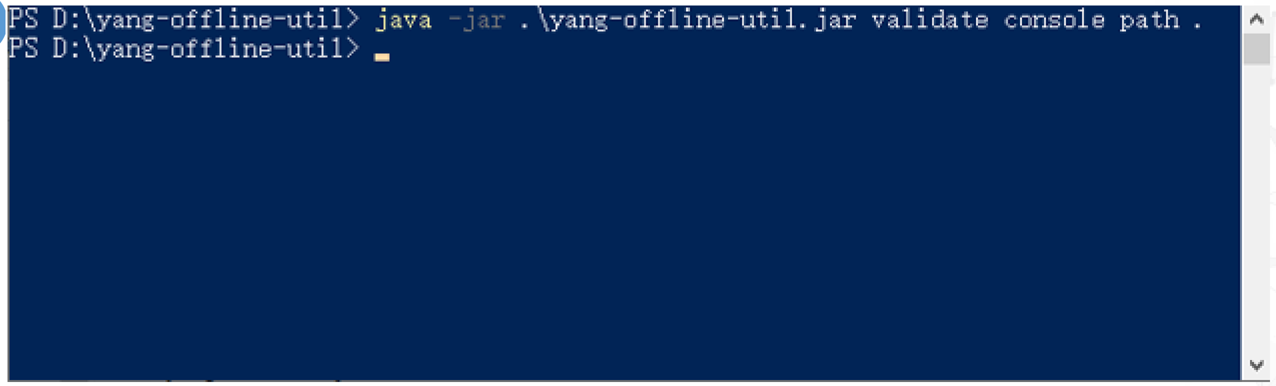
5
```
leaf peerAddress {
  type inet:ipv4-address;
}
      }
    }

  }
```

3.1 Compile the service YANG model. (continued)

1. In the YANG model of the SSP package template, the service name corresponds to the service YANG module name, and the service point corresponds to **app:application-definition** of the service YANG model.
2. Define the service identifier in the YANG model. In this example, the combination of the device name and VPN name is used as the service identifier.
3. Define the RD and RT of VPN services in the YANG model.
4. Define the sub-interface configuration in the YANG model.
5. Define peer information in the YANG model.

Developing a Service YANG Model (4/4)

1



```
PS D:\yang-offline-util> java -jar .\yang-offline-util.jar validate console path .
PS D:\yang-offline-util> _
```

Verification command:

D:\yang-offline-util> **java -jar .\yang-offline-util.jar validate console path .**

3.2 Verify the YANG file.

1. Download the YANG file verification tool from the AOC developer community.
2. Decompress **yang-offline-util.zip**.
3. Copy the service YANG model file in the **yang** directory of the software package to the directory where **yang-offline-util.zip** is decompressed.
4. Obtain the YANG files **huawei-ac-applications.yang** and **ietf-inet-types.yang** on which the service YANG model file depends from the offline tool package and place them in the same directory as the service YANG model file.
5. Open the **cmd** window, go to the directory where the verification tool is located, and run the verification command to verify the YANG model. If the command output is empty, the YANG file format is correct.

To obtain the yang-offline-util.zip package, visit: https://devzone.huawei.com/test/aoc/resDownload.html

Developing Python Mapping Code

NcsService is a base class provided by NCE and needs to be overridden to map service YANG data to Jinja2 templates. The devicemgr SDK API provides some interfaces for querying device information.

The datastore SDK API provides interfaces for reading and writing NCE database data.

AOCException is an exception class defined by NCE, which displays user-defined exceptions on the GUI.

1
```
from aoc import NcsService, devicemgr
from aoc import datastore
from aoc.exception.aocexceptions import AOCException
```

2
```
class L3VPNService(NcsService):
    def ncs_map(self, request, aoccontext=None, template=None):
        l3vpn_dict = self.xmltodict(request.xml)
        self.checkInterfaceIsExist(request, aoccontext)
        self.updateDes(l3vpn_dict)
        self.logger.info(str(l3vpn_dict))
        res = self.render('l3vpn/servicepoint.j2', l3vpn_dict)
        self.logger.info(res)
        return res
```

Fill the service data defined by the user service YANG into the Jinja2 template.

3
```
    def updateDes(self, l3vpn_dict):
        if l3vpn_dict['l3vpn'].get('description') is None:
           l3vpn_dict['l3vpn'].update({"description": 'testAOC'})
```

4
```
    def checkInterfaceIsExist(self, request, aoccontext):
        device_id = devicemgr.query_neid(request.xmldictnode.l3vpn.deviceName)
        subIfName = request.xmldictnode.l3vpn.ifName
        pointIndex = subIfName.find('.')
        parentName = subIfName[0:pointIndex]
        self.logger.info('deviceId:%s, parentName:%s' % (device_id, parentName))
        if devicemgr.is_snd(device_id, 'NE8000M8SPC300_SND') or devicemgr.is_snd(device_id, 'NE8000M8_SND'):
            path = "huawei-ac-nes:inventory-cfg/nes/ne/" + device_id + "/huawei-ifm:ifm/interfaces/"
            output = datastore.read_datastore_rdb(aoccontext, path)
            asnindex = str(output).find(parentName)
            if asnindex == -1:
                raise AOCException(parentName + ' not exist in device!')
```

Query data in the NCE configuration database.

5
```
    def add_filters(self):
        result = {"is_snd": self.is_snd}
        return result
    def is_snd(self, nename, sndid):
        neid = devicemgr.query_neid(nename)
        return devicemgr.is_snd(neid, sndid)
```

Query the device ID by device name.

Determine the SND package used by the device.

4 Develop Python mapping code.

1. Before writing Python code, import the necessary classes in the header file.
2. Override the ncs_map method to fill the service data defined by the user service YANG in the Jinja2 template.
3. Define a method for updating the VRF description.
4. Define a method for checking whether the main interface exists.
5. Add a Jinja2 template filter to map services to different devices in the Jinja2 template.

Developing a Jinja2 Template (1/3)

```
<bgp xmlns="urn:huawei:yang:huawei-bgp">
  <global>
    <yang-enable>true</yang-enable>
  </global>
</bgp>
<network-instance xmlns="urn:huawei:yang:huawei-network-instance">
  <instances>
    <instance>
      <name>{{l3vpn.vrfName}}</name>          <-- Name of a VPN.
      {%- if l3vpn.vrfDes %}
      <description>{{vrfDes}}</description>
      {%- endif %}
      <afs xmlns="urn:huawei:yang:huawei-l3vpn">
        <af>
          <type>ipv4-unicast</type>
          <route-distinguisher>{{ l3vpn.rd }}</route-distinguisher>   <-- RD of the VPN.
          <tunnel-policy>LDP</tunnel-policy>
          <vpn-targets>
            <vpn-target>
              <value>{{ l3vpn.rt }}</value>
              <type>export-extcommunity</type>     <-- RT of the VPN.
            </vpn-target>
            <vpn-target>
              <value>{{ l3vpn.rt }}</value>
              <type>import-extcommunity</type>     <-- RT of the VPN.
            </vpn-target>
          </vpn-targets>
        </af>
      </afs>
      <bgp xmlns="urn:huawei:yang:huawei-bgp">
        <base-process>
          <afs>
            <af>
              <type>ipv4uni</type>
              <ipv4-unicast>
                <import-routes>
                  <import-route>
                    <protocol>direct</protocol>
                    <process-id>0</process-id>
                  </import-route>
                  <import-route>
                    <protocol>static</protocol>
                    <process-id>0</process-id>
                  </import-route>
                </import-routes>
```

5.1 Develop a Jinja2 template.

1. Create a Jinja2 template based on **ifm**, **network-instance**, and **bgp**. The forward and reverse methods are available. Service configurations have been analyzed and the Jinja2 model of the corresponding NE has been obtained in the service analysis and design phase.
   1.1 Forward method: This method is applicable to scenarios where no real device is available and the mapping content is deduced based on the commands.
   1.2 Reverse method: This method is applicable to scenarios where real devices are deployed. After devices are managed, device configurations are synchronized, the preceding commands are configured on the device, and the XML data template corresponding to the commands is exported based on the NE configuration difference discovery capability. View the exported template and retain only the configuration to be delivered. The other configurations can be deleted.
   Finally, replace the input parameter values with the parameter names.
2. Extract the variables from the specific values and save the template to the template file **L3VPN/template/l3vpn/NE8000M8.j2**.
3. If there are multiple device models, services can be mapped to each device model.
4. Save the prepared template to the template file **L3VPN/template/l3vpn/servicepoint.j2**.

Developing a Jinja2 Template (2/3)

```
              </ipv4-unicast>
            </af>
          </afs>
          <peers>
            <peer>
              <address>{{ l3vpn.peerAddress }}</address>          ← - - - - -  Peer IP address.
              <remote-as>100</remote-as>
            </peer>
          </peers>
        </base-process>
      </bgp>
    </instance>
  </instances>
</network-instance>
<ifm xmlns="urn:huawei:yang:huawei-ifm">
  <interfaces>
    <interface>
      <name>{{ l3vpn.ifName }}</name>  ← - - - - - - - - - - - - - - - - -  Sub-interface name.
      <type>GigabitEthernet</type>
      <description>{{ l3vpn.description }}</description> ← - - - - - - - - -  Sub-interface description.
      <vrf-name>{{ l3vpn.vrfName }}</vrf-name>
      <ipv4 xmlns="urn:huawei:yang:huawei-ip">
        <addresses>
          <address>
            <ip>{{ l3vpn.ip }}</ip>      ← - - - - - - - - - - - - - - -  Sub-interface IP address.
            <type>main</type>
            <mask>{{ l3vpn.subnet }}</mask>  ← - - - - - - - - - - - -  Mask.
          </address>
        </addresses>
      </ipv4>
    </interface>
  </interfaces>
</ifm>
```

5.1 Develop a Jinja2 template. (continued)

1. Create the Jinja2 template based on **ifm**, **network-instance**, and **bgp**. The forward and reverse methods are available. Service configurations have been analyzed and the Jinja2 model of the corresponding NE has been obtained in the service analysis and design phase.
   1.1 Forward method: This method is applicable to scenarios where no real device is available and the mapping content is deduced based on the commands.
   1.2 Reverse method: This method is applicable to scenarios where real devices are deployed. After devices are managed, device configurations are synchronized, the preceding commands are configured on the device, and the XML data template corresponding to the commands is exported based on the NE configuration difference discovery capability.
   View the exported template and retain only the configuration to be delivered. The other configurations can be deleted.
   Finally, replace the input parameter values with the parameter names.
2. Extract the variables from the specific values and save the template to the template file **L3VPN/template/l3vpn/NE8000M8.j2**.
3. If there are multiple device models, services can be mapped to each device model.
4. Save the prepared template to the template file **L3VPN/template/l3vpn/servicepoint.j2**.

Developing a Jinja2 Template (3/3)

3

```
<inventory-cfg xmlns="urn:huawei:yang:huawei-ac-nes">
    <nes>
        <ne>
            <neid>{{l3vpn.deviceName| to_ne_id}}</neid>
            {%- if l3vpn.deviceName | is_snd('NE8000M8SPC300_SND')  %}
                {% include 'l3vpn/NE8000M8.j2' %}
            {% elif l3vpn.deviceName | is_snd('NE8000M8_SND') %}
                {% include 'l3vpn/NE8000M8.j2' %}
            {% endif %}
        </ne>
    </nes>
</inventory-cfg>
```

Invoke the is_snd method in the Python script to determine the SND package used by the device.

5.1 Develop a Jinja2 template. (continued)

1. Create the Jinja2 template based on **ifm**, **network-instance**, and **bgp**. The forward and reverse methods are available. Service configurations have been analyzed and the Jinja2 model of the corresponding NE has been obtained in the service analysis and design phase.
   1.1 Forward method: This method is applicable to scenarios where no real device is available and the mapping content is deduced based on the commands.
   1.2 Reverse method: This method is applicable to scenarios where real devices are deployed. After devices are managed, device configurations are synchronized, the preceding commands are configured on the device, and the XML data template corresponding to the commands is exported based on the NE configuration difference discovery capability. View the exported template and retain only the configuration to be delivered. The other configurations can be deleted.
   Finally, replace the input parameter values with the parameter names.
2. Extract the variables from the specific values and save the template to the template file **L3VPN/template/l3vpn/NE8000M8.j2**.
3. If there are multiple device models, services can be mapped to each device model.
4. Save the prepared template to the template file **L3VPN/template/l3vpn/servicepoint.j2**.

Compiling Test Cases

```
import unittest
import sys
from aoc.sys import devicemgr, datastore
from mock import Mock
sys.path.insert(0, "../../python")
from l3vpn.l3vpn import L3VPNService
class Test(unittest.TestCase):
  xml = '''
 <l3vpn xmlns="http://example.com/L3VPN">
  <deviceName>NE8000</deviceName>
  <vrfName>5G_test</vrfName>
  <rd>100:1</rd>
  <rt>100:11</rt>
  <ifName>GigabitEthernet0/5/0.1</ifName>
  <description>wireless base</description>
  <ip>20.1.2.9</ip>
  <subnet>255.255.255.0</subnet>
  <peerAddress>2.2.2.2</peerAddress>
 </l3vpn>
  '''
  def mock_get_id(self, neName):
    if neName == 'CX600':
      return 'mock_neid:CX600'
    if neName == 'NE8000':
      return 'mock_neid:NE8000'
    return 'mock_neid:NE8000'
  def mock_sys_func(self):
    devicemgr.query_neid = self.mock_get_id
    mock_read_datastore_rdb = Mock(return_value='GigabitEthernet0/5/0')
    datastore.read_datastore_rdb = mock_read_datastore_rdb
    mock_write_datastore = Mock(return_value='success')
    datastore.write_datastore = mock_write_datastore
    devicemgr.is_snd = self.mock_is_snd
  def mock_is_snd(self, nename, sndid):
    if nename == 'mock_neid:NE8000' and sndid == 'NE8000M8_SND':
      return True
    if nename == 'mock_neid:NE8000' and sndid == 'NE8000M8SPC300_SND':
      return True
    return False
  def setUp(self):
    self.mock_sys_func()
  def test_l3vpn(self):
    result = L3VPNService().ncs_map_test(self.xml)
    print(result)
if __name__ == "__main__":
  unittest.main()
```

Simulate SSP packet input.
Go to the directory where the service YANG model is located and run the **java -jar .\yang-offline-util.jar generateSubtree** command.
If no command output is displayed, the execution is successful. In this case, edit the **subtree.xml** file generated in the same directory and fill in the corresponding data.

In a local test, methods that depend on the NCE environment cannot be invoked. Instead, some methods in the code need to be mocked.

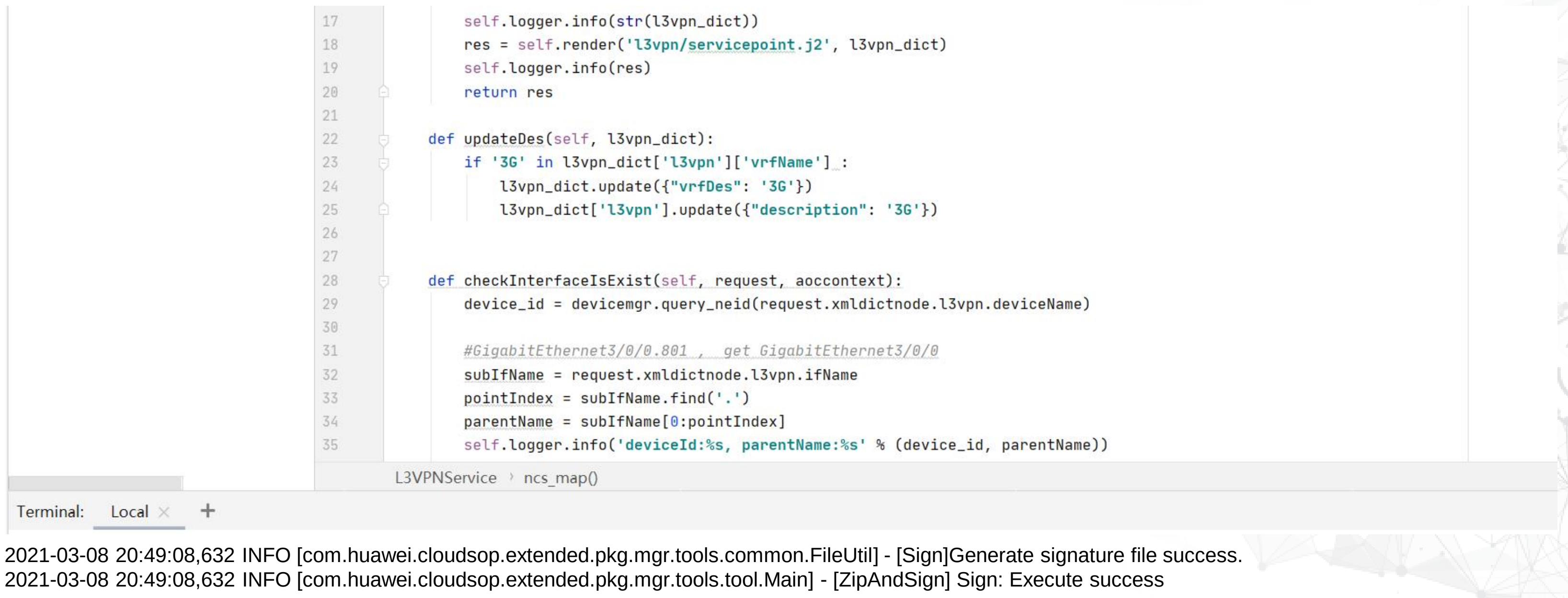
6 Compile test cases.

1. Compile the LLT test code and save it to the script file in the **test > l3vpn** directory.
2. Run the test script file. Then check whether the format and values of generated packets are correct.

04

Compiling the SSP Package

Compiling an SSP Package



```python
        self.logger.info(str(l3vpn_dict))
        res = self.render('l3vpn/servicepoint.j2', l3vpn_dict)
        self.logger.info(res)
        return res

    def updateDes(self, l3vpn_dict):
        if '3G' in l3vpn_dict['l3vpn']['vrfName'] :
            l3vpn_dict.update({"vrfDes": '3G'})
            l3vpn_dict['l3vpn'].update({"description": '3G'})


    def checkInterfaceIsExist(self, request, aoccontext):
        device_id = devicemgr.query_neid(request.xmldictnode.l3vpn.deviceName)

        #GigabitEthernet3/0/0.801 ,  get GigabitEthernet3/0/0
        subIfName = request.xmldictnode.l3vpn.ifName
        pointIndex = subIfName.find('.')
        parentName = subIfName[0:pointIndex]
        self.logger.info('deviceId:%s, parentName:%s' % (device_id, parentName))
```

```
2021-03-08 20:49:08,632 INFO [com.huawei.cloudsop.extended.pkg.mgr.tools.common.FileUtil] - [Sign]Generate signature file success.
2021-03-08 20:49:08,632 INFO [com.huawei.cloudsop.extended.pkg.mgr.tools.tool.Main] - [ZipAndSign] Sign: Execute success
```

1 Compile an SSP package.

1. Copy the private key **private.asc** in the GPG key pair generated by the Gpg4win tool to the **key** directory of the software package.
2. Open **Terminal** in PyCharm and go to the **bin** directory in the SSP package.
3. Run the **makeFile.bat** script to start packing.
4. After the packing is complete, obtain the software package and signature file from the **output** directory in the SSP package.

2

```
E:\aoc\L3VPN>cd bin
E:\aoc\L3VPN\bin>makeFile.bat
```

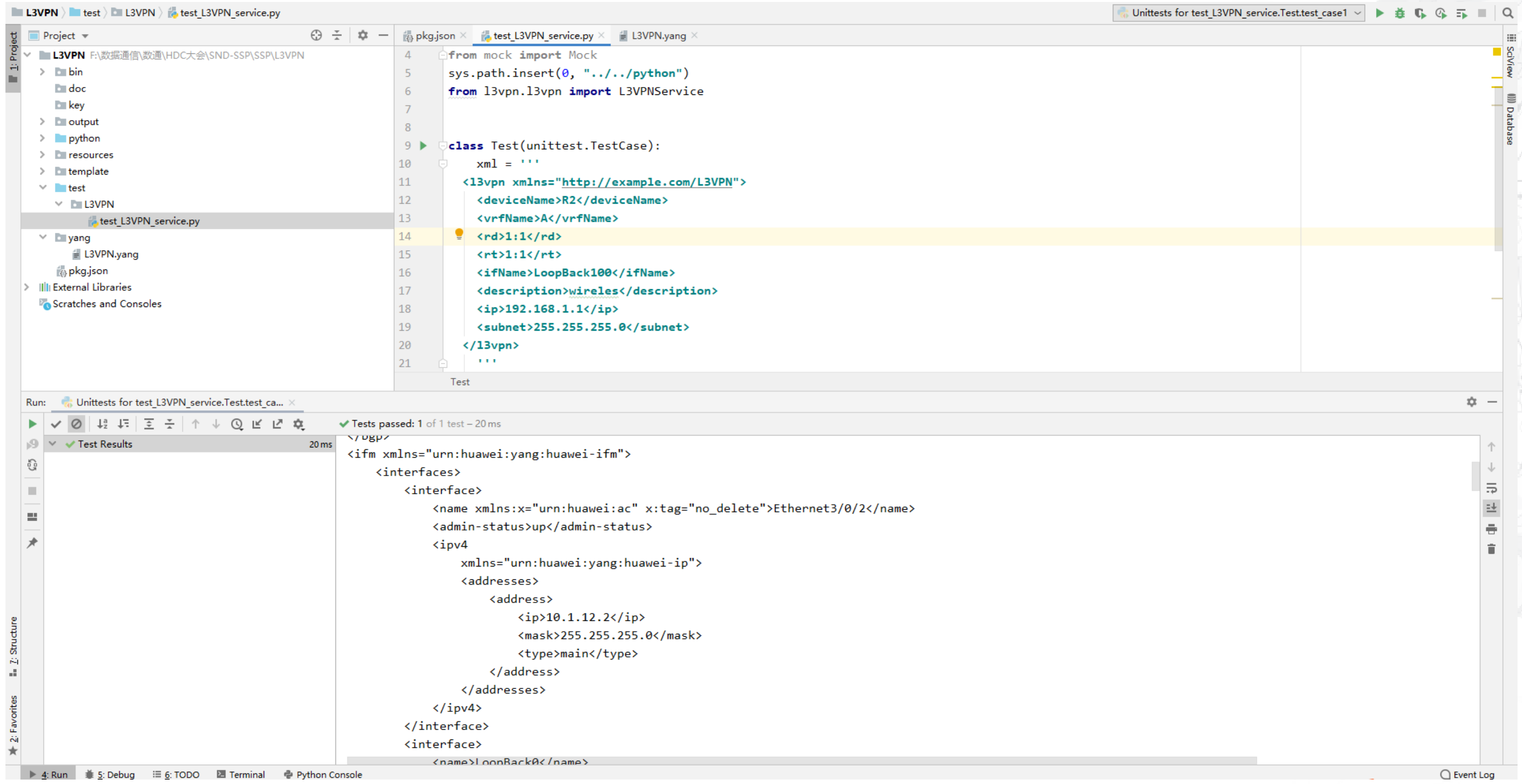
```
2021-03-08 20:49:08,632 INFO [com.huawei.cloudsop.extended.pkg.mgr.tools.common.FileUtil] - [Sign]Generate signature file success.
2021-03-08 20:49:08,632 INFO [com.huawei.cloudsop.extended.pkg.mgr.tools.tool.Main] - [ZipAndSign] Sign: Execute success
```

3

05

Verifying the SSP Package

Performing the LLT



```
from mock import Mock
sys.path.insert(0, "../../python")
from l3vpn.l3vpn import L3VPNService


class Test(unittest.TestCase):
    xml = '''
  <l3vpn xmlns="http://example.com/L3VPN">
    <deviceName>R2</deviceName>
    <vrfName>A</vrfName>
    <rd>1:1</rd>
    <rt>1:1</rt>
    <ifName>LoopBack100</ifName>
    <description>wireles</description>
    <ip>192.168.1.1</ip>
    <subnet>255.255.255.0</subnet>
  </l3vpn>
    '''
```

Test

✓ Tests passed: 1 of 1 test – 20 ms

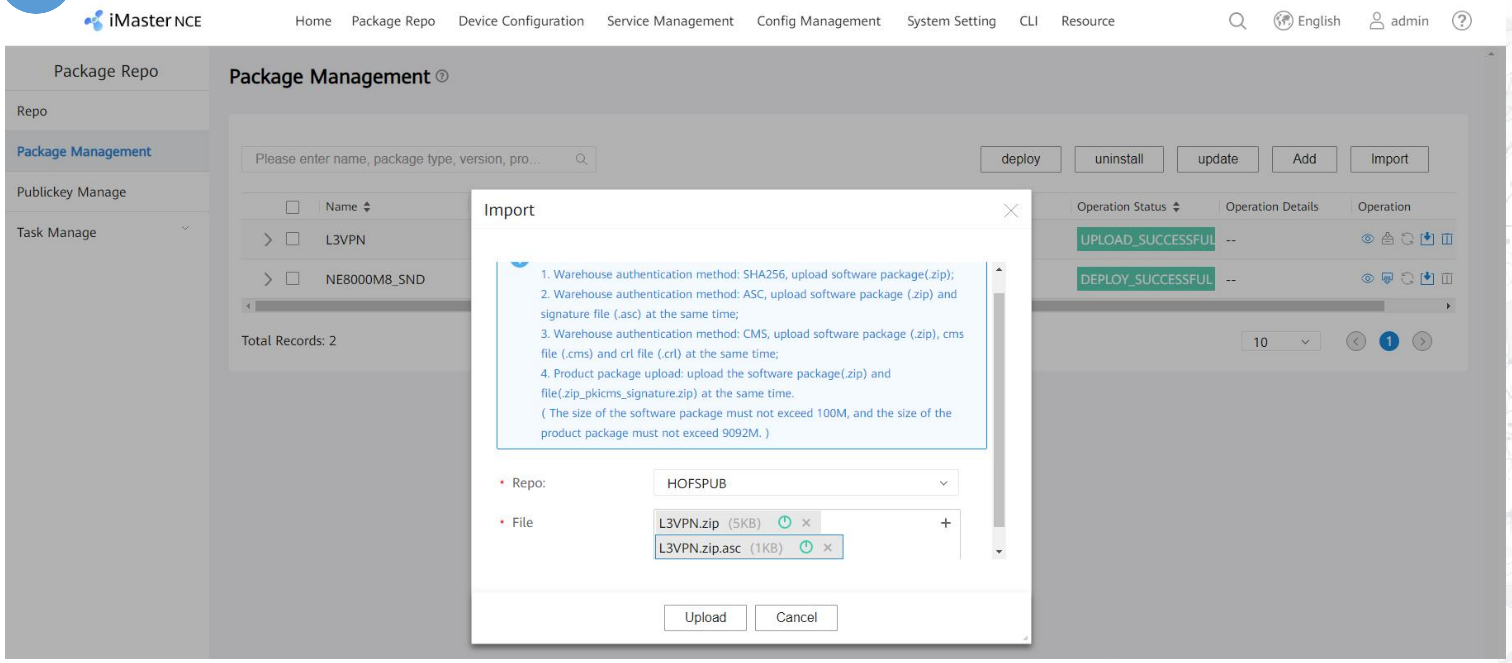
```
</bgp>
<ifm xmlns="urn:huawei:yang:huawei-ifm">
    <interfaces>
        <interface>
            <name xmlns:x="urn:huawei:ac" x:tag="no_delete">Ethernet3/0/2</name>
            <admin-status>up</admin-status>
            <ipv4
                xmlns="urn:huawei:yang:huawei-ip">
                <addresses>
                    <address>
                        <ip>10.1.12.2</ip>
                        <mask>255.255.255.0</mask>
                        <type>main</type>
                    </address>
                </addresses>
            </ipv4>
        </interface>
        <interface>
            <name>LoopBack0</name>
```

1 Perform the LLT.

1. Run the LLT test case and check whether the generated packets are correct.

Loading an SSP Package

2



iMaster NCE

Home | Package Repo | Device Configuration | Service Management | Config Management | System Setting | CLI | Resource | English | admin

Package Repo
- Repo
- Package Management
- Publickey Manage
- Task Manage

Package Management

Please enter name, package type, version, pro...

deploy | uninstall | update | Add | Import

	Name	Operation Status	Operation Details	Operation
	L3VPN	UPLOAD_SUCCESSFUL	--	
	NE8000M8_SND	DEPLOY_SUCCESSFUL	--	

Total Records: 2

10

Import

1. Warehouse authentication method: SHA256, upload software package(.zip);
2. Warehouse authentication method: ASC, upload software package (.zip) and signature file (.asc) at the same time;
3. Warehouse authentication method: CMS, upload software package (.zip), cms file (.cms) and crl file (.crl) at the same time;
4. Product package upload: upload the software package(.zip) and file(.zip_pkicms_signature.zip) at the same time.

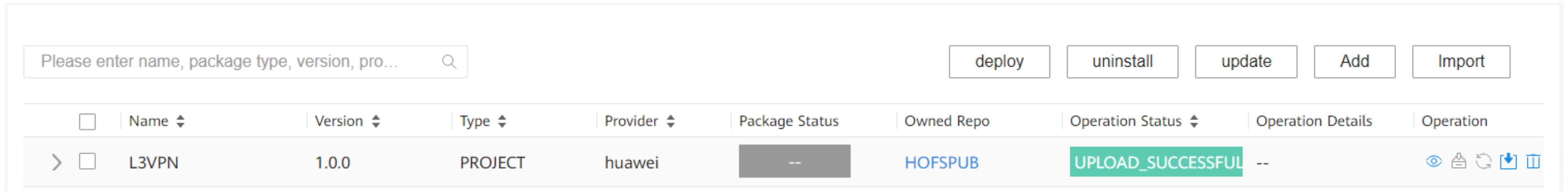
(The size of the software package must not exceed 100M, and the size of the product package must not exceed 9092M.)

* Repo: HOFSPUB

* File: L3VPN.zip (5KB) · L3VPN.zip.asc (1KB)

Upload | Cancel

3



Please enter name, package type, version, pro...

deploy | uninstall | update | Add | Import

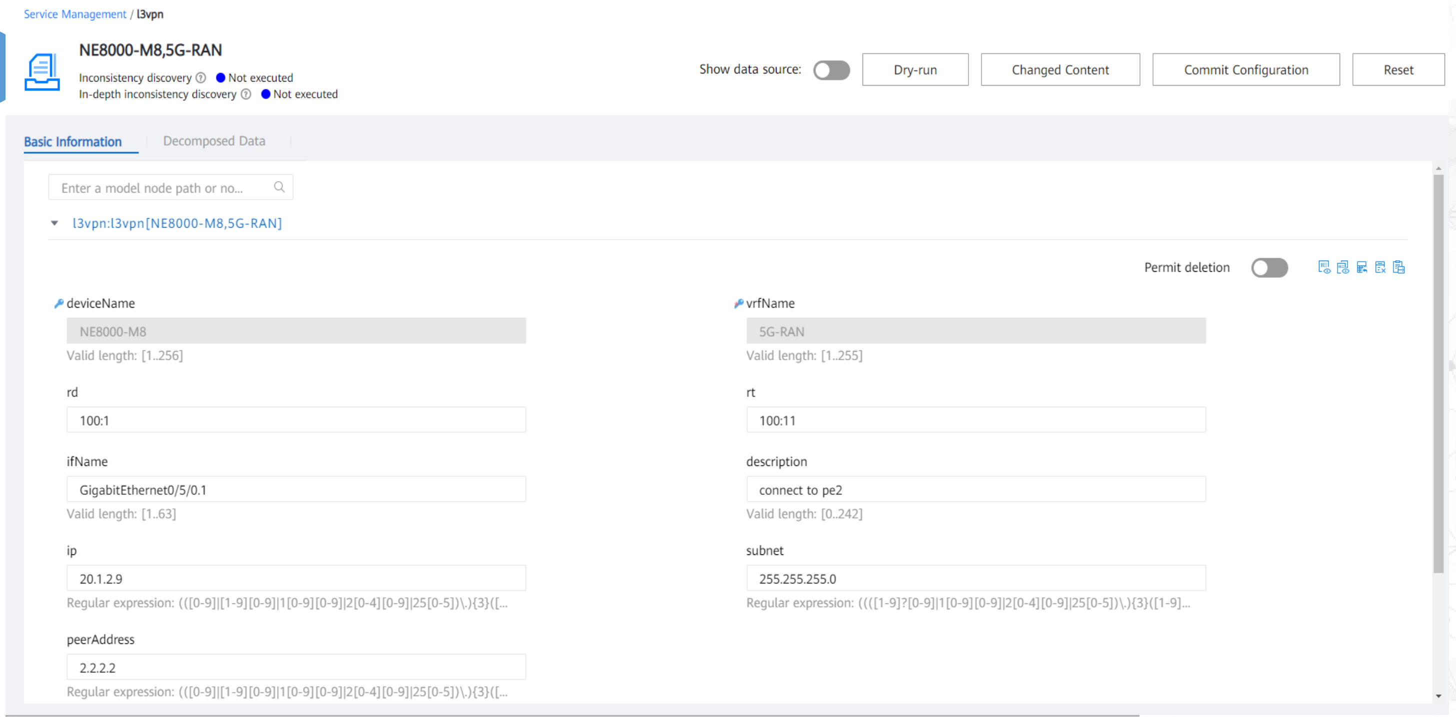
	Name	Version	Type	Provider	Package Status	Owned Repo	Operation Status	Operation Details	Operation
	L3VPN	1.0.0	PROJECT	huawei	--	HOFSPUB	UPLOAD_SUCCESSFUL	--	

2.1 Load an SSP package.

1. On the **Package Repo** page of the AOC, delete the original SSP package template.
2. Import the newly developed SSP package and signature file.
3. Click in the **Operation** column to activate the SSP package.

Delivering Network Services

3


Service Management / l3vpn

NE8000-M8,5G-RAN

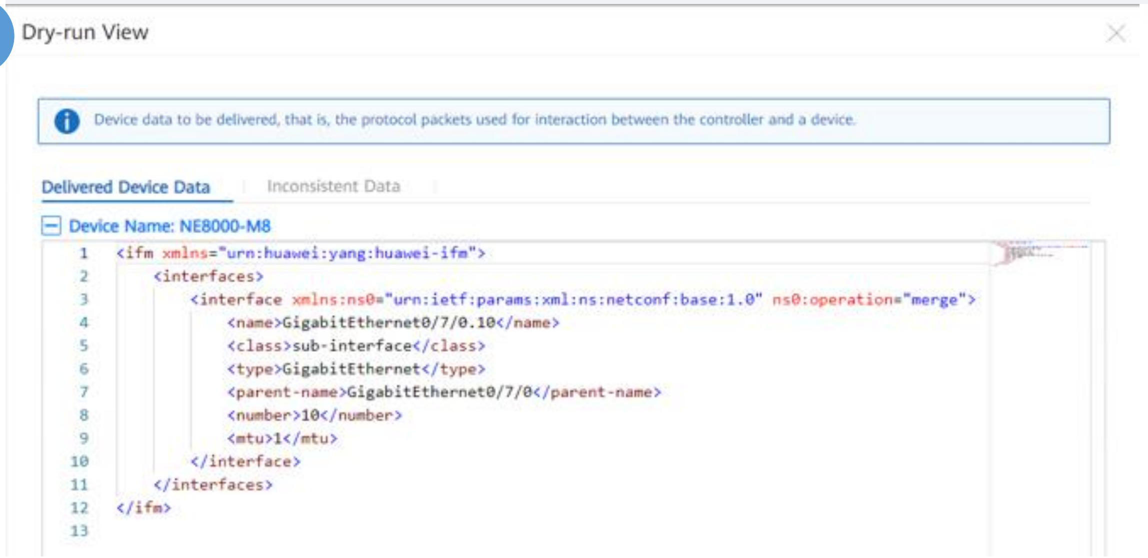
Inconsistency discovery ● Not executed
In-depth inconsistency discovery ● Not executed

l3vpn:l3vpn[NE8000-M8,5G-RAN]

Permit deletion

deviceName: NE8000-M8 (Valid length: [1..256])	vrfName: 5G-RAN (Valid length: [1..255])
rd: 100:1	rt: 100:11
ifName: GigabitEthernet0/5/0.1 (Valid length: [1..63])	description: connect to pe2 (Valid length: [0..242])
ip: 20.1.2.9 (Regular expression: (([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}([...)	subnet: 255.255.255.0 (Regular expression: ((([1-9]?[0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}([1-9]...)
peerAddress: 2.2.2.2 (Regular expression: (([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}([...)	

4


Dry-run View

Device data to be delivered, that is, the protocol packets used for interaction between the controller and a device.

Device Name: NE8000-M8

```
<ifm xmlns="urn:huawei:yang:huawei-ifm">
    <interfaces>
        <interface xmlns:ns0="urn:ietf:params:xml:ns:netconf:base:1.0" ns0:operation="merge">
            <name>GigabitEthernet0/7/0.10</name>
            <class>sub-interface</class>
            <type>GigabitEthernet</type>
            <parent-name>GigabitEthernet0/7/0</parent-name>
            <number>10</number>
            <mtu>1</mtu>
        </interface>
    </interfaces>
</ifm>

```

2.2 Deliver network services.

1. On the **Service Management** page, select the **l3vpn** service model and click **Add**.
2. On the displayed page, enter the device name and VRF name, and click **Create**.
3. Enter the sub-interface and BGP information.
4. Click **Dry-run** to view the configuration packets to be delivered.

Network Cloud Engine

Thank You!

HUAWEI