Network Cloud Engine

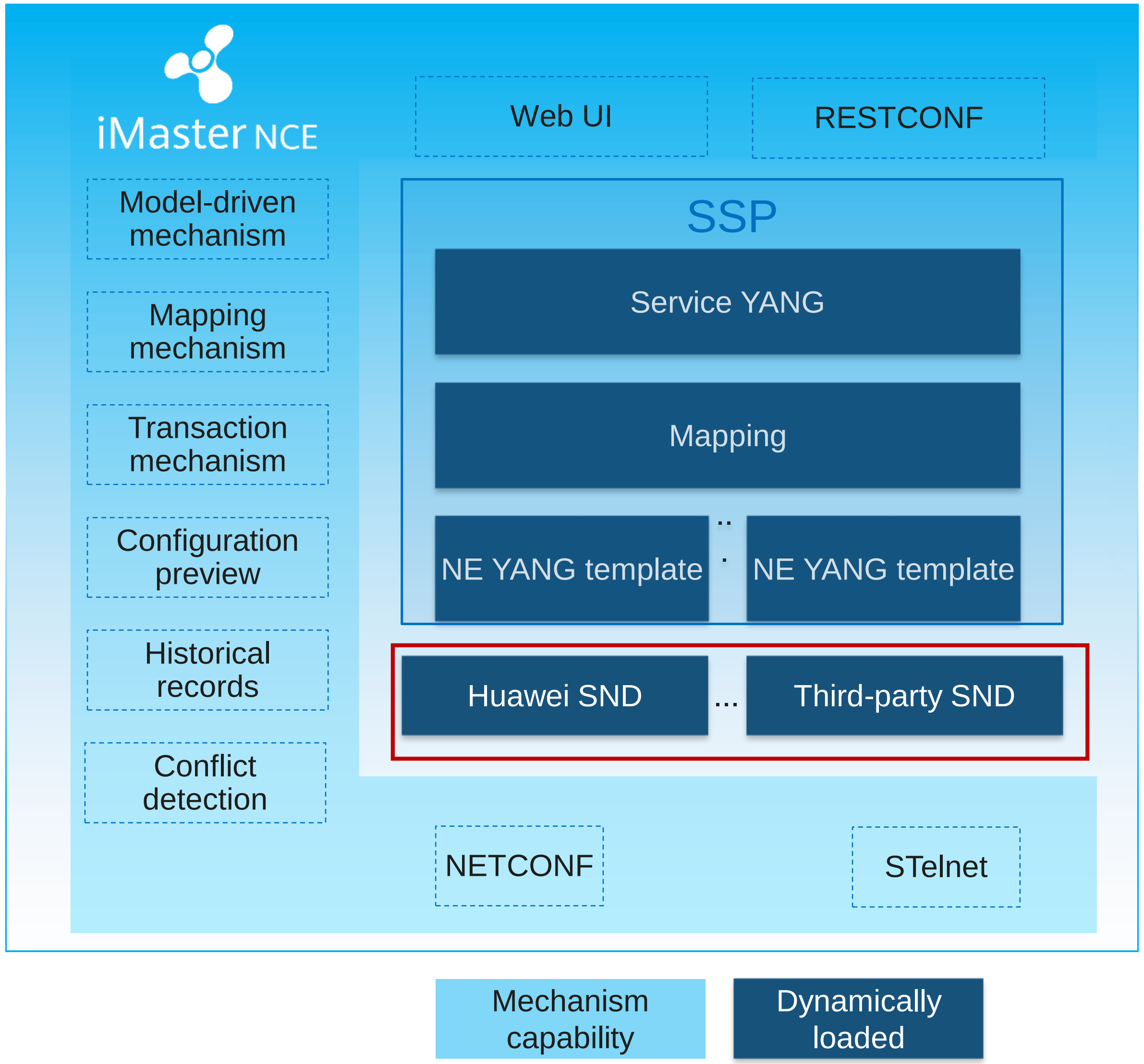
Open Programmability
Training and Certification

Building NETCONF SND Packages

01

Getting Familiar with SND

SND Package



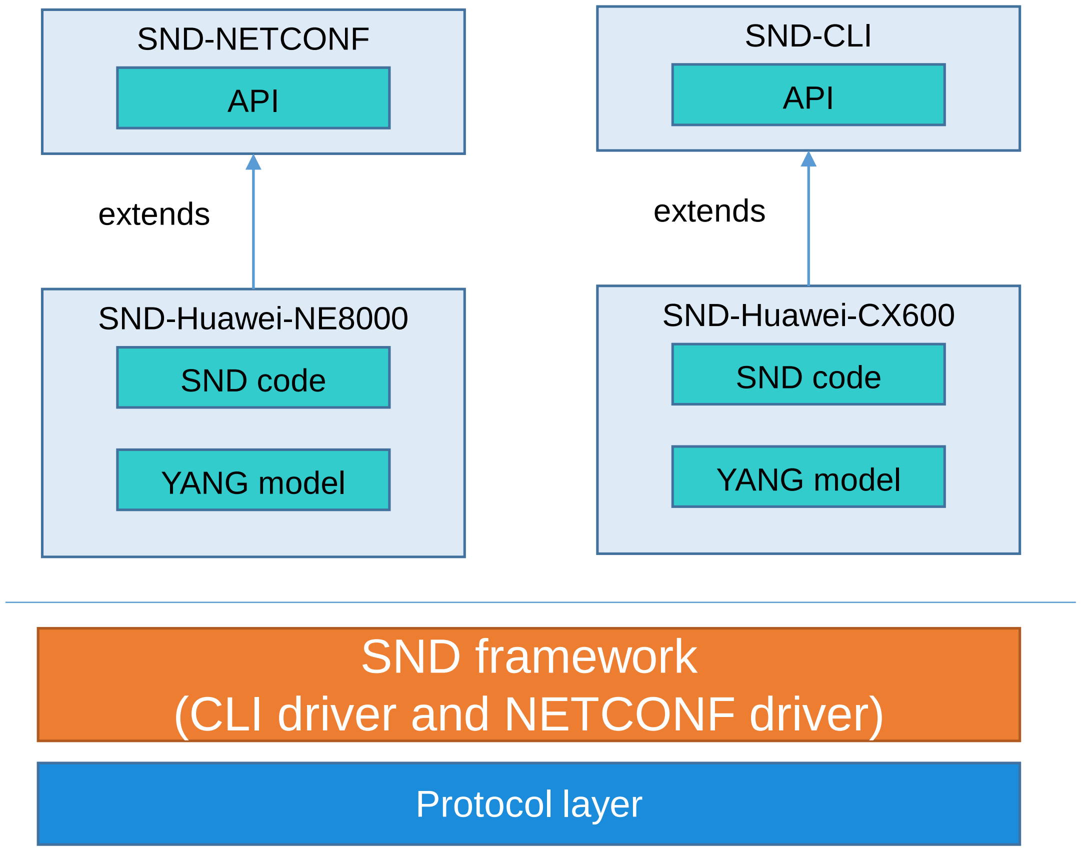
Definition

SND: refers to specific NE driver. An SND package defines the device YANG model to shield differences caused because of different protocols used on devices. The SND framework provides a general conversion mechanism to quickly customize and deliver the conversion from the YANG model to protocols.

Function

By loading the developed SND package, the framework system can **recognize devices and understand their capabilities**. After importing the SND package of a device into the Agile Open Container (AOC), you can create a device, establish a connection with the device, collect device data, obtain configuration differences from the device, and deliver the configuration through the device management GUI and northbound API automatically generated based on the device YANG model.

SND Package Types

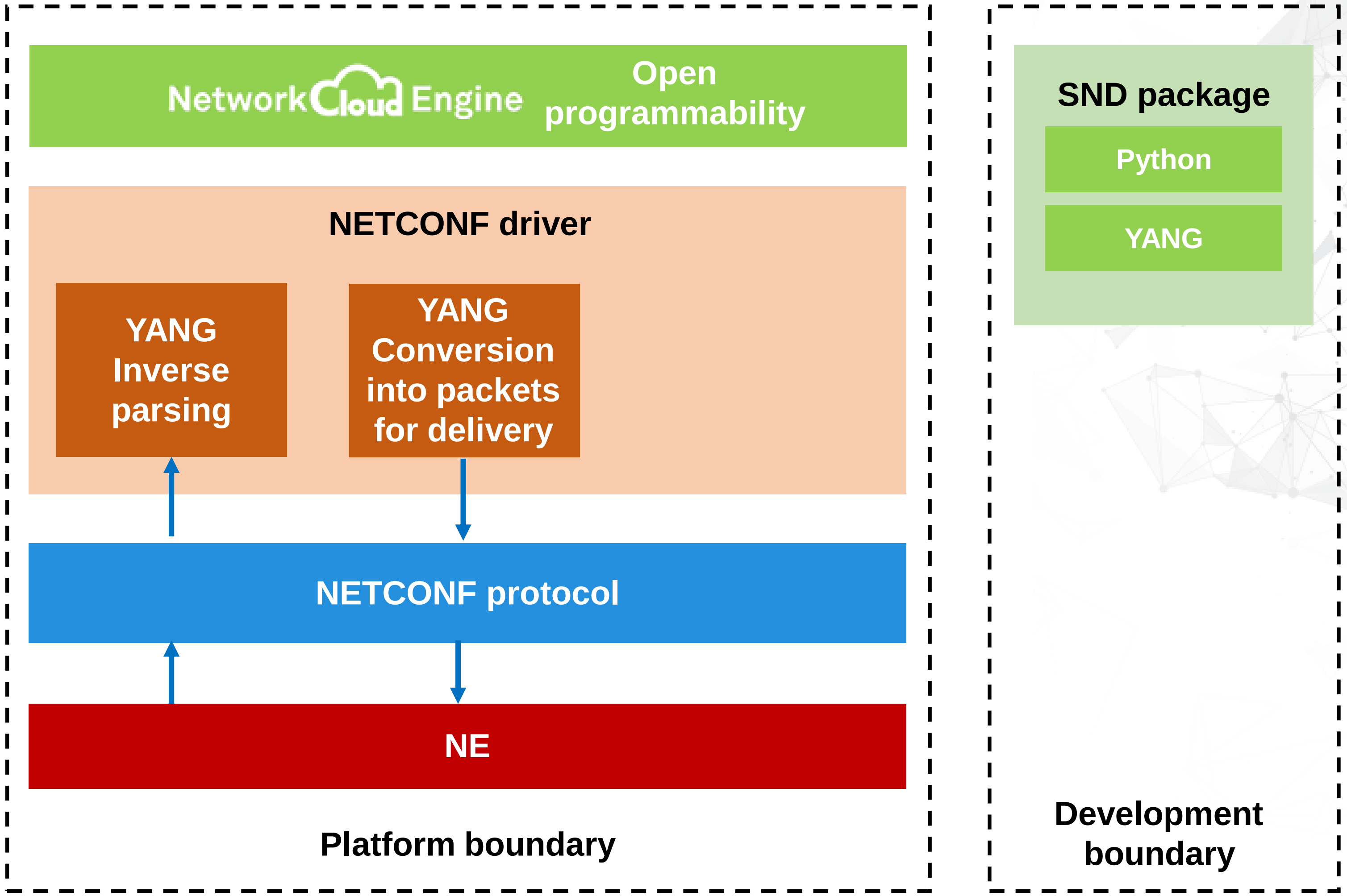


Type

In the AOC, SND packages are classified into the following types by protocol:

- **NETCONF SND**
  The general-purpose driver for converting the YANG model into NETCONF packets is provided for NETCONF-capable devices. You can quickly customize the device driver based on the NETCONF packet format.
- **CLI SND**
  The bidirectional rendering driver for converting the YANG model into the CLI configuration is provided for devices interconnected through the CLI, for example, NETCONF-incapable old devices. You can quickly customize device drivers based on the NE CLI configuration.

NETCONF Driver



Concept

- **Device YANG model**
  Abstracts device capabilities, and includes multiple device YANG modules. Each module corresponds to a function on the device. Based on the device YANG model, the mechanism can automatically generate southbound protocol packets, automatically generate data tables, support CRUD operations, and automatically generate northbound APIs and GUIs.
- **NETCONF driver framework**
  Provides a general-purpose conversion mechanism based on the device YANG model. For configurations to be delivered, the mechanism automatically converts the configurations into packets to deliver based on the YANG model. For configurations to be synchronized from the device to the controller, the mechanism automatically converts packets into the corresponding YANG model structure.
- **NETCONF protocol**
  Provides a channel for connecting to the device to issue CLI packets for delivering or querying configurations to the device.

Development

- **Building third-party packages**
  Saves the YANG file **supported by the device** to the **yang** directory and develops the handshake parameters for establishing a connection with the device.

NETCONF SND Package Structure

```
SND_demo_Python/
    pkg.json
    python/
        com/
            __init__.py
            huawei/
                controller/
                    devicetype/
                        snd.py
    resources/
        logger.conf
    yang/
        huawei-bfd.yang
        huawei-bgp.yang
        huawei-ccc.yang
        huawei-ifm.yang
        huawei-ip.yang
        huawei-monitor-group.yang
        huawei-vlan.yang
```

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

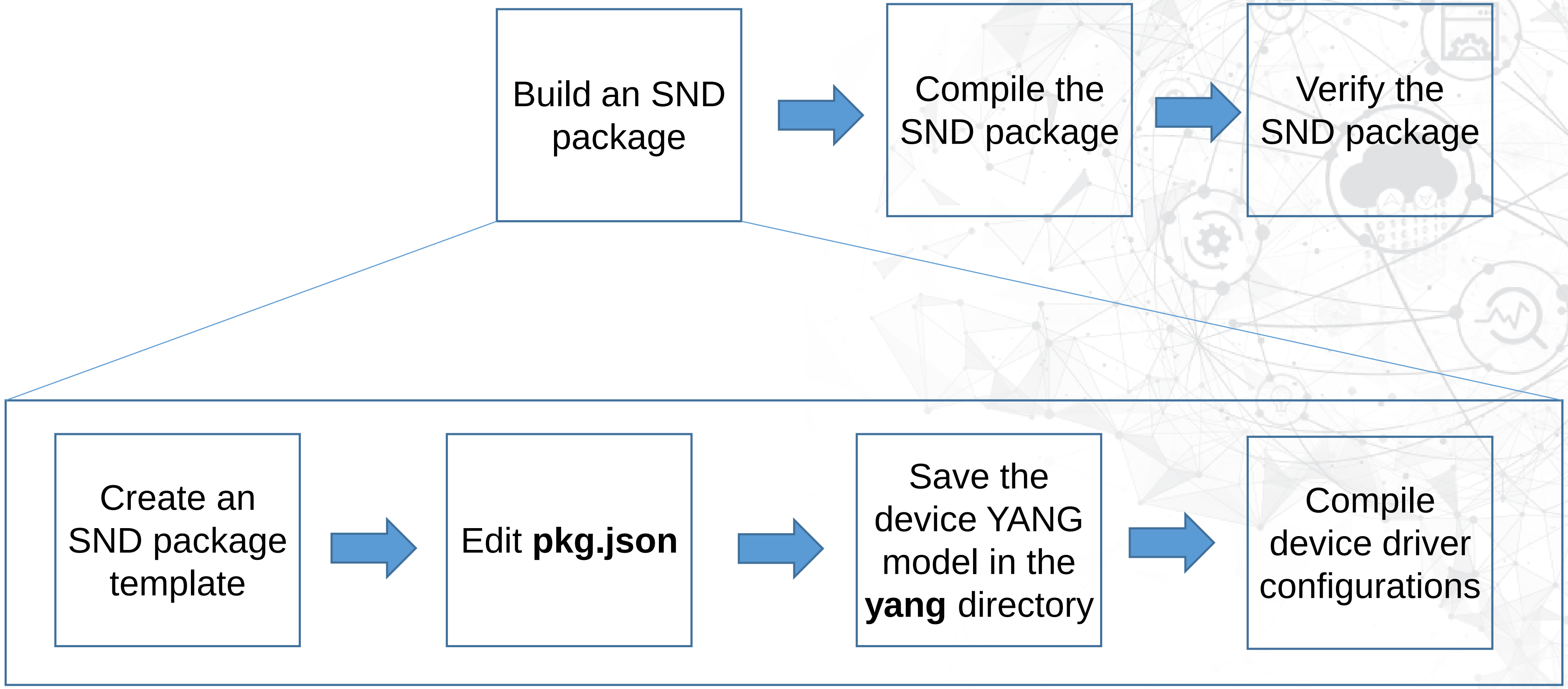
- **python**
  Information required for device interconnection, such as device information, connection information, driver information, protocol parameter information, and differentiated device customization.

- **Resource file**
  Stores log configuration files or customized files used in Python.

- **yang/***
  YANG module of the device. Each module corresponds to a function on the device. Together, they form the device YANG model.

Directory structure

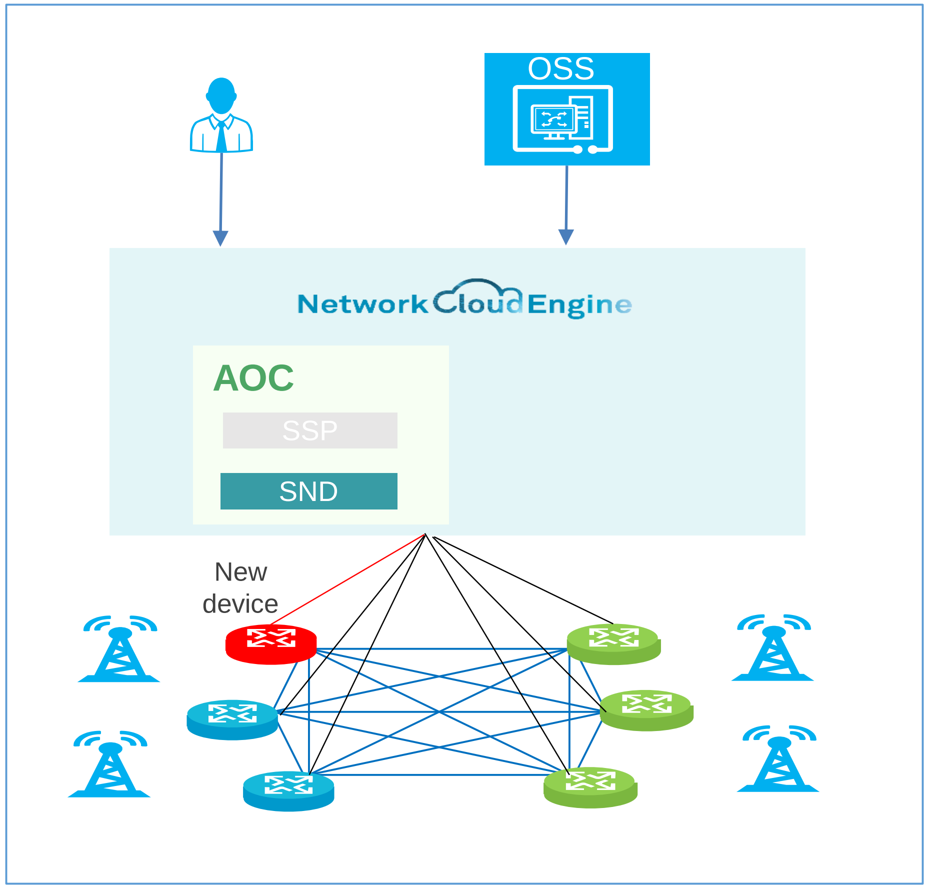
Development Process


Build an SND package
Compile the SND package
Verify the SND package
Create an SND package template
Edit **pkg.json**
Save the device YANG model in the **yang** directory
Compile device driver configurations

02

Building an SND Package

Building a NETCONF SND Package for the NE8000 M8



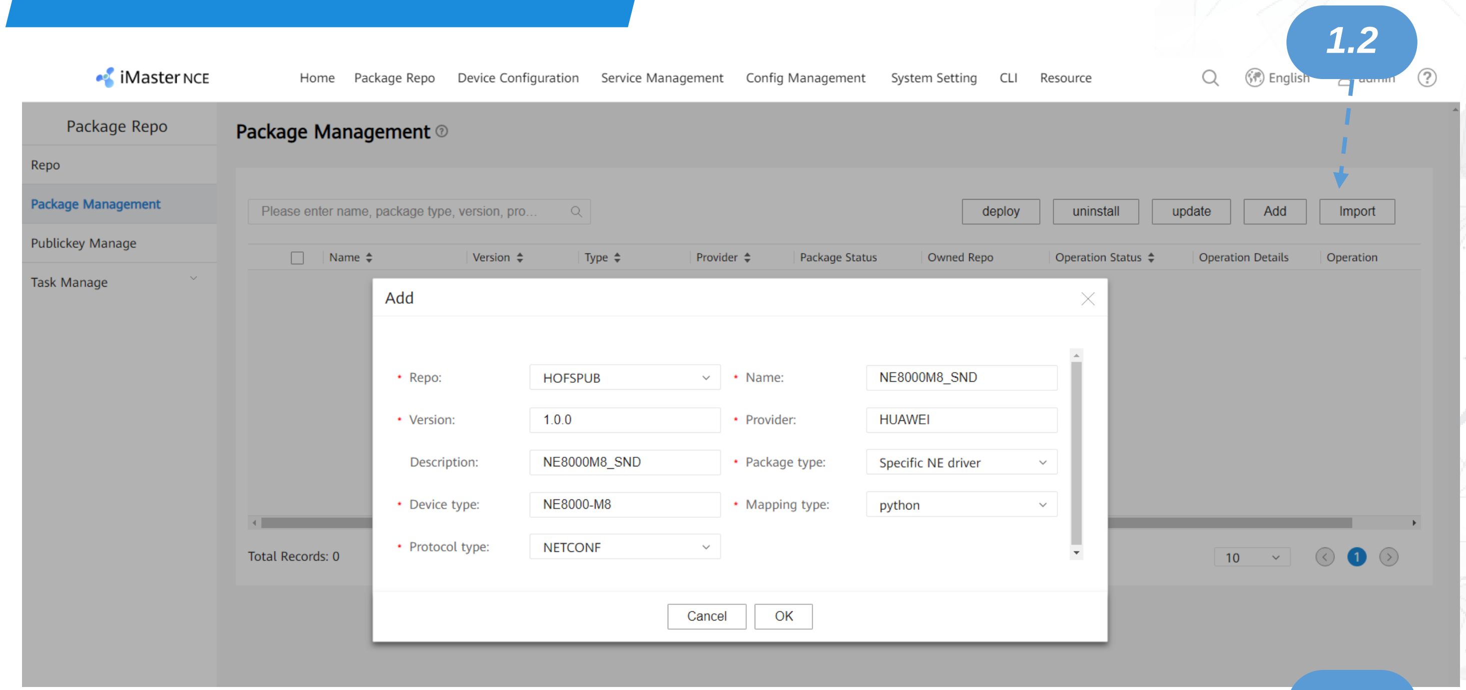
Objective

Describe how to develop a NETCONF SND package for Huawei NE8000 M8.

Environment

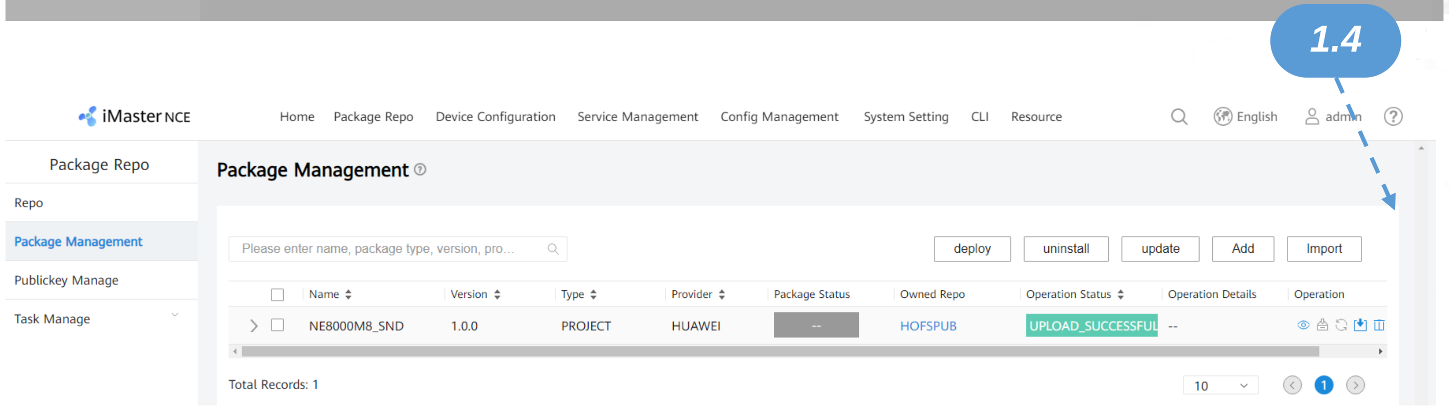
- **IDE**: local or online IDE environment
- **AOC**: local or online AOC environment

Creating an SND Package Template



Add

• Repo:	HOFSPUB	• Name:	NE8000M8_SND
• Version:	1.0.0	• Provider:	HUAWEI
Description:	NE8000M8_SND	• Package type:	Specific NE driver
• Device type:	NE8000-M8	• Mapping type:	python
• Protocol type:	NETCONF		



Name	Version	Type	Provider	Package Status	Owned Repo	Operation Status	Operation Details	Operation
NE8000M8_SND	1.0.0	PROJECT	HUAWEI	--	HOFSPUB	UPLOAD_SUCCESSFUL	--	

1 Create an SND 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 SND package template is generated.
4. Click ⬇ in the **Operation** column to download the template to the local PC and decompress it.

Developing the Package Property File

```
{
  "name": "NE8000M8_SND",
  "version": "1.0.0",
  "description": "NE8000M8_SND",
  "package-type": "snd",
  "producer": "HUAWEI",
  "nce-min-versions": [
    "1.0.0"
  ],
  "snd-id": "NE8000M8_SND",
  "devices": [
   {
    "vendor": "HUAWEI",
    "device-type": "NetEngine 8000 M8",
    "device-version": "V800R012C00SPC300"
   }
  ],
  "hooks": [
   {
    "type": "snd",
    "key": "ecs-driver",
    "python-class-name": "com.huawei.controller.devicetype.ne8000m8snd.NE8000M8NETCONF"
   }
  ]
}
```

Basic information about the SND package.

Triplet, which is used to map the device to the SND package.

Device vendor.

Device type.

Device version.

Callback mapping information, which is used to import device capabilities into the SND package.

Callback mapping type, which is set to driver parameter information.

Specific driver parameter information.

2 Edit **pkg.json**.

1. Decompress the SND package **NE8000M8_SND.zip** downloaded in the previous step to the PyCharm project.
2. Open the **pkg.json** file and modify the device information.

Saving the Device YANG Model in the yang Directory

```
NE8000M8_SND/
    pkg.json
    python/
    resources/
    yang/
        huawei-aaa@2020-01-09.yang
        huawei-aaa-deviations-NE8000-M8@2019-04-23.yang
        huawei-aaa-lam@2020-01-09.yang
        huawei-aaa-type@2020-01-09.yang
        huawei-acl@2020-02-20.yang
        huawei-acl-deviations-NE8000-M8@2020-02-20.yang
        huawei-acl-ucl@2020-02-20.yang
        huawei-acl-ucl-deviations-NE8000-M8@2020-02-20.yang
        huawei-arp@2020-02-18.yang
        huawei-bd@2019-04-29.yang
        huawei-bd-ext@2020-03-06.yang
        huawei-bfd@2020-02-20.yang
```

3.1 Obtain the device YANG model and save it in the **yang** directory.

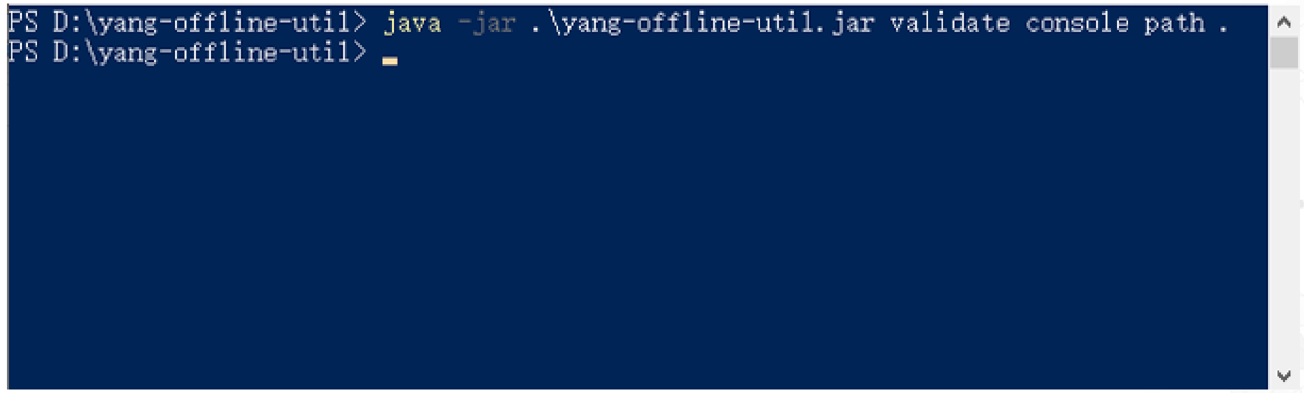
1. Obtain the YANG model file of the NE8000 M8 from the vendor.
2. Delete the default YANG model file from the SND package template.
3. Decompress the obtained YANG model to the **yang** folder in the SND package template.

- **yang/***
  YANG module of the device. Each module corresponds to a function on the device. Together, they form the device YANG model.

Directory structure

Saving the Device YANG Model in the yang Directory

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 YANG model file in the **yang** directory of the software package to the directory where **yang-offline-util.zip** is decompressed.
4. 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

Compiling Device Driver Configurations

4.1 Import necessary classes.

1. Before writing Python code, import necessary classes.

```
from aoc.snd.netconfsnd import NetconfSND
from aoc.snd.snd_model_pb2.sysoidinfo_pb2 import SysoidInfo
from aoc.snd.snd_model_pb2.connectinfo_pb2 import ConnectInfos, ProtocolEntity, DEFAULT_CONNECT, PRIMARY_CONNECTION,  HelloEntity
from aoc.snd.snd_model_pb2.channelInfo_pb2 import SINGLE_CHANNEL, PROTECTED_MODE
from aoc.snd.snd_model_pb2.ecsdriver_pb2 import CommonDriverInfo
from aoc.snd.snd_model_pb2.ecsdriver_pb2 import NetconfDriverInfo
```

NetconfSND is the parent class of the SND package. The default configuration has been implemented in the parent class. For customized configurations, inherit NetconfSND in the SND package and override methods in the parent class.

Import the SysoidInfo class to configure the device type, vendor, and model corresponding to the SND. This class is used in the getSysoidInfo method.

Import the ConnectInfos class to configure device connection information, the ProtocolEntity class to configure the device connection protocol, and the HelloEntity class to configure Hello packet information. Additionally, import the DEFAULT_CONNECT and PRIMARY_CONNECTION constants. This class is used in the getConnectInfo method.

Import the SINGLE_CHANNEL or PROTECTED_MODE constant, which needs to be performed when the device connection capability is configured. This constant is used in the getConnectInfo method to configure a single channel or dual channels that work in active/standby mode for protection.

Import the following classes:
CommonDriverInfo: configures the general-purpose device driver. This class is used in the getCommonDriverInfo method.
NETCONFDriverInfo: sets the NETCONF driver of the device. This class is used in the getNetconfDriverInfo method.

Import the **devicemgr.py** file, which is used to query certain data from the device in real time.

Compiling Device Driver Configurations

4.2 Configure device management parameters.

1. Register device information.
2. Configure the device connection capability.

1

```
def getSysoidInfo(self, aoccontext, request=None):
    sysoidInfo = SysoidInfo()
    sysoidEntity = sysoidInfo.sysoidEntity.add()
    sysoidEntity.sysoid = "1.3.6.1.4.1.2011.2.360.1.10"
    sysoidEntity.deviceType = "ROUTER"
    sysoidEntity.deviceModel = "NetEngine 8000 M8"
    sysoidEntity.deviceVendor = "HUAWEI"
    return sysoidInfo
```

Sysoid of the device to be registered.

Type of the device to be registered.

Model of the device to be registered.

Vendor of the device to be registered.

2

```
# Use the NETCONF protocol to establish a shared single read and write channel.
def getConnectInfo(self, aoccontext, request=None):
    self.logger.info('getConnectInfo start.')
    connectInfos = ConnectInfos()
    primaryConnectInfo = connectInfos.connectInfo.add()
    primaryConnectInfo.protocolEntity.protocolType = ProtocolEntity.netconf
    primaryConnectInfo.connectPolicy = DEFAULT_CONNECT
    primaryConnectInfo.channelInfo.readChannel = SINGLE_CHANNEL
    primaryConnectInfo.channelInfo.is_read_share_write = True
    primaryConnectInfo.protocolEntity.helloEntity.helloType = HelloEntity.extendType
    return primaryConnectInfo
```

If the read and write channels are not shared, the read channel is a single channel, and the write channels work in active/standby mode for protection, set **is_read_share_write** to **False**, **readChannel** to **SINGLE_CHANNEL**, and **writeChannel** to **PROTECTED_MODE**.

Compiling Device Driver Configurations

4.3 Compile configuration management parameters.

1. Customize a general-purpose device driver.
2. Customize the device NETCONF driver.

1

```
def getCommonDriverInfo(self, aoccontext, request=None):
    common_driver = CommonDriverInfo()
    common_driver.unsupportedOperations = "create,delete"
    syncToDel = common_driver.para.add()
    syncToDel.key = "sync-to-del-enable"
    syncToDel.value = "true"
    return common_driver
```

Operations that devices do not support, which will be automatically converted after being configured. The value **create** indicates that the create operation is not supported, which needs to be converted into the merge operation. The value **delete** indicates that the delete operation is not supported, which needs to be converted into the remove operation. The value **create,delete** indicates that the create and delete operations are not supported, which need to be converted into merge and remove, respectively.

Whether the configuration instance of southbound devices can be deleted during data consistency verification. For **sync-to-del-enable**, the value **true** indicates that the configuration instance can be deleted; the value **false** indicates that the configuration instance cannot be deleted.

2

```
def getNetconfDriverInfo(self, aoccontext, request=None):
    netconf_driver = NetconfDriverInfo()
    netconf_driver.phase = "two"
    netconf_driver.classification = "huawei-v5"
    netconf_driver.testOption = "set"
    return netconf_driver
```

Number of phases in which device configurations are delivered. The value **one** indicates that configurations are delivered in one phase; the value **two** indicates that configurations are delivered in two phases.

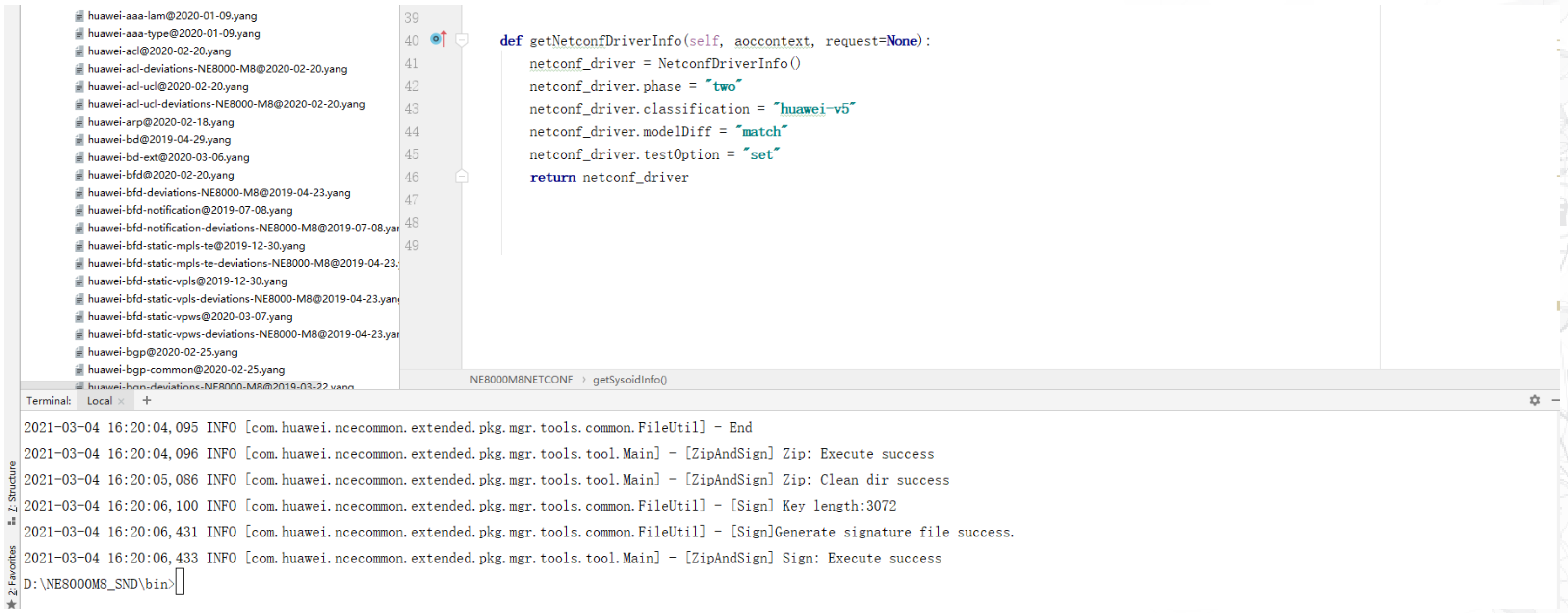
Device prototype. The value is **huawei-v5** for most Huawei device models and is **huawei-v8** for some old Huawei device models.

Whether packets are delivered in sequential order. The value **set** indicates that packets are delivered in non-sequential order.

03

Compiling the SND Package

Compiling an SND Package



```
huawei-aaa-lam@2020-01-09.yang
huawei-aaa-type@2020-01-09.yang
huawei-acl@2020-02-20.yang
huawei-acl-deviations-NE8000-M8@2020-02-20.yang
huawei-acl-ucl@2020-02-20.yang
huawei-acl-ucl-deviations-NE8000-M8@2020-02-20.yang
huawei-arp@2020-02-18.yang
huawei-bd@2019-04-29.yang
huawei-bd-ext@2020-03-06.yang
huawei-bfd@2020-02-20.yang
huawei-bfd-deviations-NE8000-M8@2019-04-23.yang
huawei-bfd-notification@2019-07-08.yang
huawei-bfd-notification-deviations-NE8000-M8@2019-07-08.yar
huawei-bfd-static-mpls-te@2019-12-30.yang
huawei-bfd-static-mpls-te-deviations-NE8000-M8@2019-04-23.
huawei-bfd-static-vpls@2019-12-30.yang
huawei-bfd-static-vpls-deviations-NE8000-M8@2019-04-23.yan
huawei-bfd-static-vpws@2020-03-07.yang
huawei-bfd-static-vpws-deviations-NE8000-M8@2019-04-23.yar
huawei-bgp@2020-02-25.yang
huawei-bgp-common@2020-02-25.yang
huawei-bgp-deviations-NE8000-M8@2019-03-22.yang
```

```python

def getNetconfDriverInfo(self, aoccontext, request=None):
    netconf_driver = NetconfDriverInfo()
    netconf_driver.phase = "two"
    netconf_driver.classification = "huawei-v5"
    netconf_driver.modelDiff = "match"
    netconf_driver.testOption = "set"
    return netconf_driver
```

NE8000M8NETCONF › getSysoidInfo()

Terminal: Local

```
2021-03-04 16:20:04,095 INFO [com.huawei.ncecommon.extended.pkg.mgr.tools.common.FileUtil] - End
2021-03-04 16:20:04,096 INFO [com.huawei.ncecommon.extended.pkg.mgr.tools.tool.Main] - [ZipAndSign] Zip: Execute success
2021-03-04 16:20:05,086 INFO [com.huawei.ncecommon.extended.pkg.mgr.tools.tool.Main] - [ZipAndSign] Zip: Clean dir success
2021-03-04 16:20:06,100 INFO [com.huawei.ncecommon.extended.pkg.mgr.tools.common.FileUtil] - [Sign] Key length:3072
2021-03-04 16:20:06,431 INFO [com.huawei.ncecommon.extended.pkg.mgr.tools.common.FileUtil] - [Sign]Generate signature file success.
2021-03-04 16:20:06,433 INFO [com.huawei.ncecommon.extended.pkg.mgr.tools.tool.Main] - [ZipAndSign] Sign: Execute success
D:\NE8000M8_SND\bin>
```

1 Compile the SND 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 SND 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 SND package.

1

```
(dem) D:\ NE8000M8_SND >copy path\to\privkey.asc .\key\privkey.asc
1 file(s) copied.
```

2

```
(dem) D:\ NE8000M8_SND >cd bin
(dem) D:\ NE8000M8_SND \bin>makeFile.bat
```

3

```
2021-03-04 16:20:04,096 INFO [com.huawei.ncecommon.extended.pkg.mgr.tools.tool.Main] - [ZipAndSign] Zip: Execute success
2021-03-04 16:20:05,086 INFO [com.huawei.ncecommon.extended.pkg.mgr.tools.tool.Main] - [ZipAndSign] Zip: Clean dir success
2021-03-04 16:20:06,100 INFO [com.huawei.ncecommon.extended.pkg.mgr.tools.common.FileUtil] - [Sign] Key length:3072
2021-03-04 16:20:06,431 INFO [com.huawei.ncecommon.extended.pkg.mgr.tools.common.FileUtil] - [Sign]Generate signature file success.
2021-03-04 16:20:06,433 INFO [com.huawei.ncecommon.extended.pkg.mgr.tools.tool.Main] - [ZipAndSign] Sign: Execute success
```

04

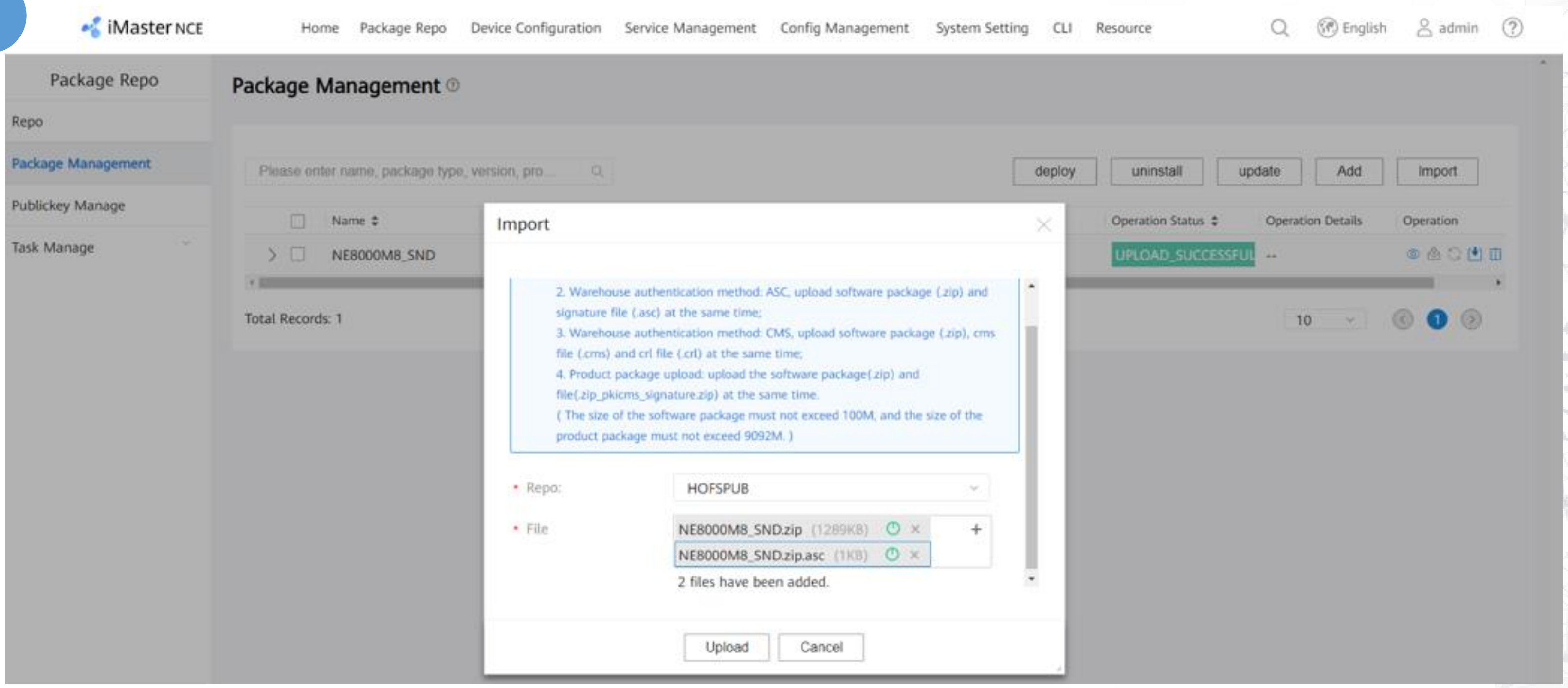
Verifying the SND Package

Loading an SND Package

1 Load an SND package.

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

2



Package Management

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

Name	Operation Status	Operation Details	Operation
NE8000M8_SND	UPLOAD_SUCCESSFUL	--	

Total Records: 1

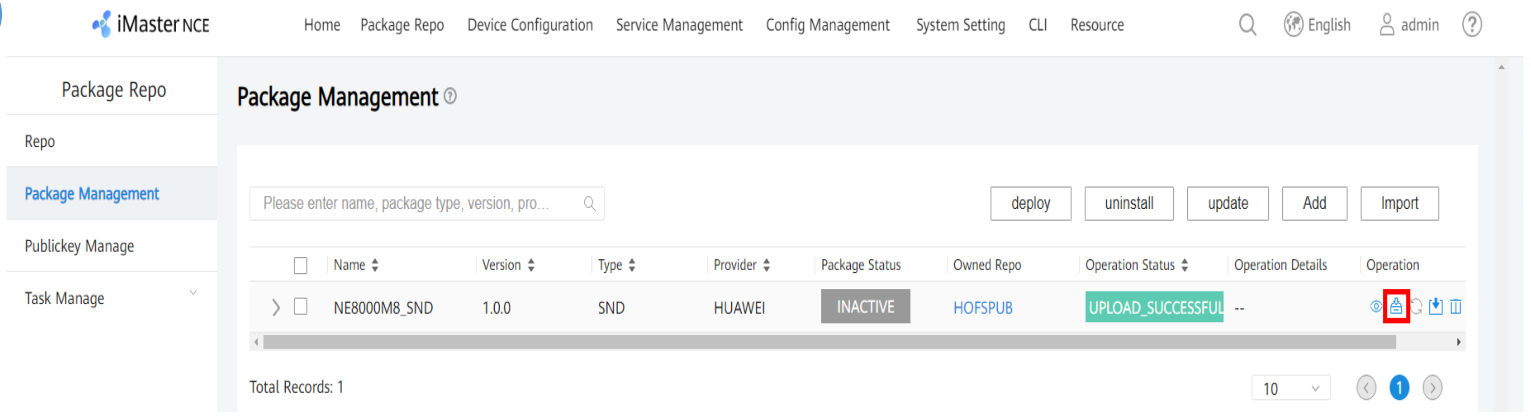
Import

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: NE8000M8_SND.zip (1289KB)
NE8000M8_SND.zip.asc (1KB)

2 files have been added.

3



Package Management

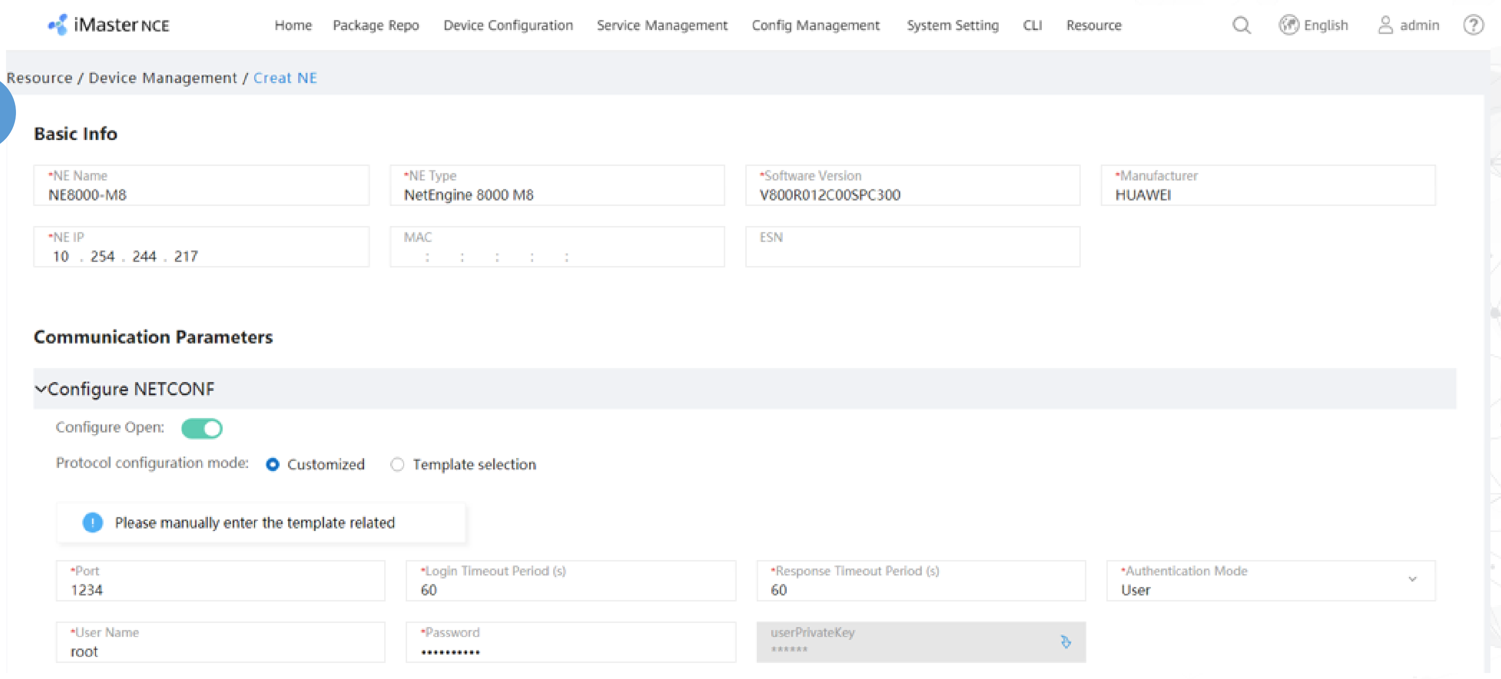
Please enter name, package type, version, pro...

Name	Version	Type	Provider	Package Status	Owned Repo	Operation Status	Operation Details	Operation
NE8000M8_SND	1.0.0	SND	HUAWEI	INACTIVE	HOFSPUB	UPLOAD_SUCCESSFUL	--	

Total Records: 1

Managing Devices

1



Resource / Device Management / Creat NE

Basic Info

*NE Name: NE8000-M8	*NE Type: NetEngine 8000 M8	*Software Version: V800R012C00SPC300	*Manufacturer: HUAWEI
*NE IP: 10 . 254 . 244 . 217	MAC:	ESN:	

Communication Parameters

Configure NETCONF

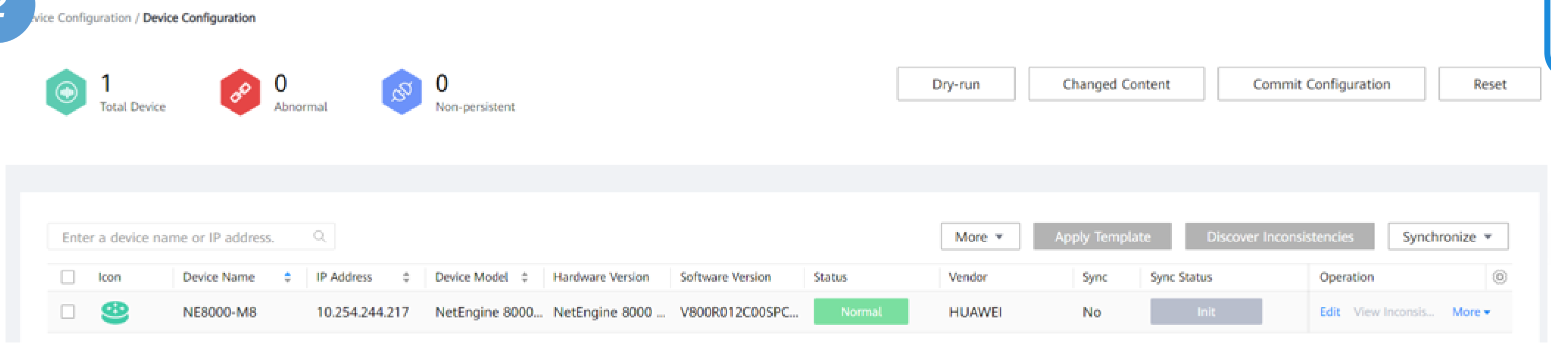
Configure Open: ●

Protocol configuration mode: [✓ Customized] Template selection

Please manually enter the template related

*Port: 1234	*Login Timeout Period (s): 60	*Response Timeout Period (s): 60	*Authentication Mode: User
*User Name: root	*Password: ••••••••••	userPrivateKey: ******	

2



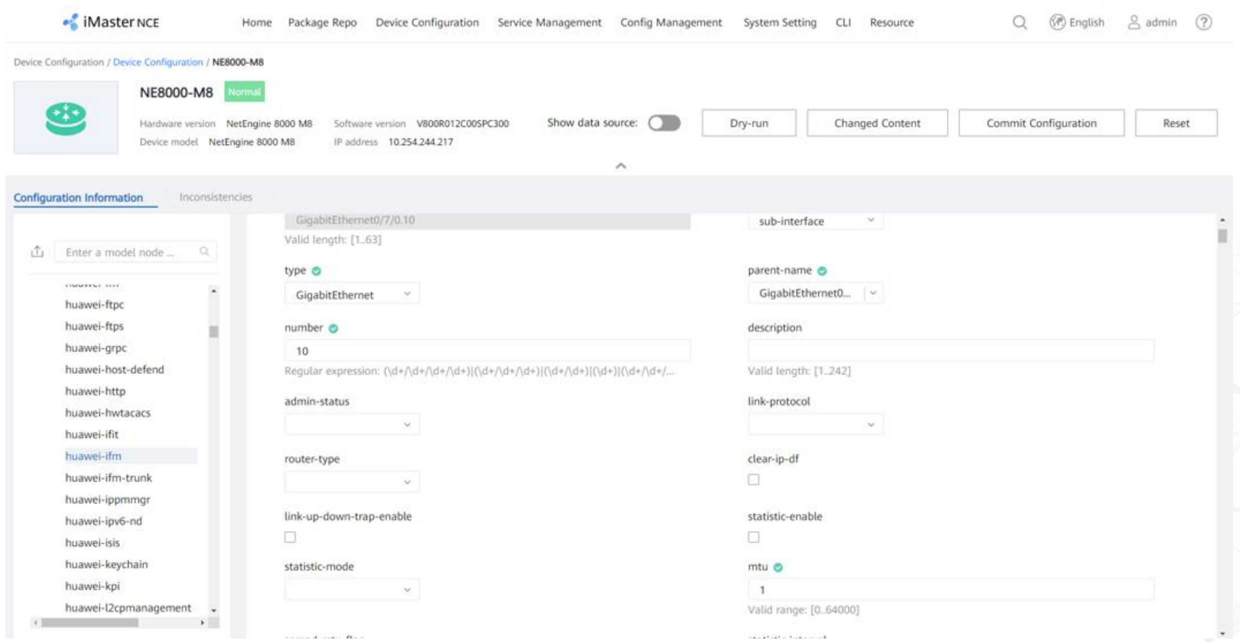
Icon	Device Name	IP Address	Device Model	Hardware Version	Software Version	Status	Vendor	Sync	Sync Status	Operation
	NE8000-M8	10.254.244.217	NetEngine 8000...	NetEngine 8000 ...	V800R012C00SPC...	Normal	HUAWEI	No	Init	Edit View Inconsis... More

2 Manage devices.

1. On the AOC, choose **Resource** > **Device Management**.
2. Click **Create**. On the **Create NE** page, enter basic information. The NE type, software version, and vendor must be the same as those in **devices** in the SND package. In addition, enable NETCONF.
3. After the device is added, you can view the new device on the **Device Management** page.

Delivering the Configuration

1



Device Configuration / Device Configuration / NE8000-M8

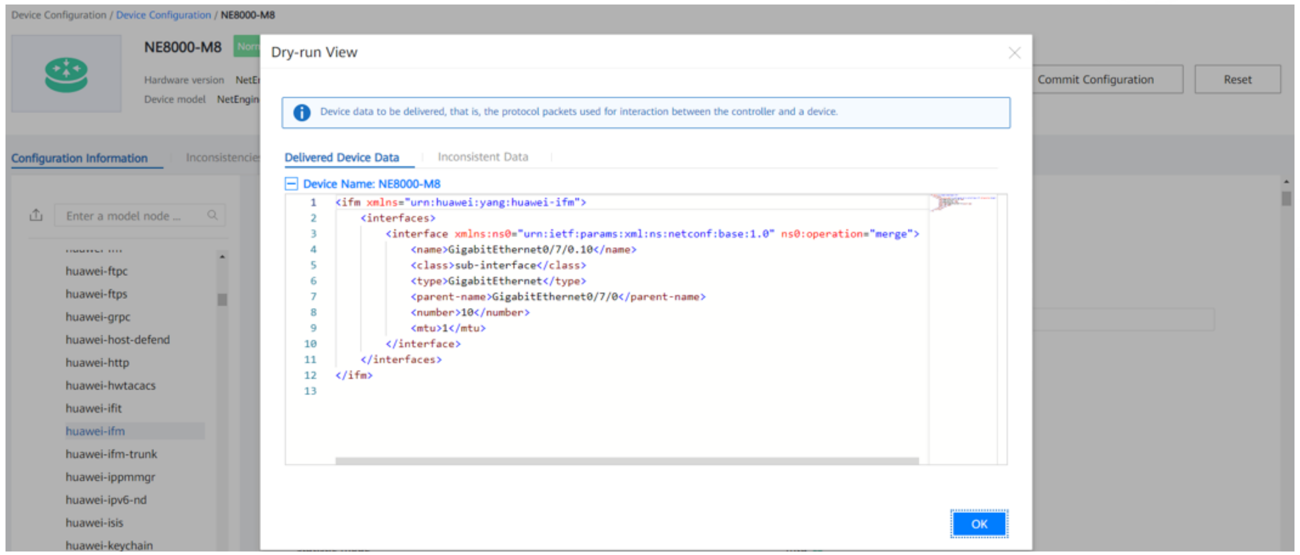
NE8000-M8 Normal

Hardware version NetEngine 8000 M8 Software version V800R012C00SPC300
Device model NetEngine 8000 M8 IP address 10.254.244.217

huawei-ftpc
huawei-ftps
huawei-grpc
huawei-host-defend
huawei-http
huawei-hwtacacs
huawei-ifit
huawei-ifm
huawei-ifm-trunk
huawei-ippmmgr
huawei-ipv6-nd
huawei-isis
huawei-keychain
huawei-kpi
huawei-l2cpmanagement

GigabitEthernet0/7/0.10
Valid length: [1..63]
class: sub-interface
type: GigabitEthernet
parent-name: GigabitEthernet0...
number: 10
Regular expression: (\d+/\d+/\d+/\d+)|(\d+/\d+/\d+)|(\d+/\d+)|(\d+)|(\d+/\d+/...
description
Valid length: [1..242]
admin-status
link-protocol
router-type
clear-ip-df
link-up-down-trap-enable
statistic-enable
statistic-mode
mtu: 1
Valid range: [0..64000]

2



Device Configuration / Device Configuration / NE8000-M8

NE8000-M8

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>
```

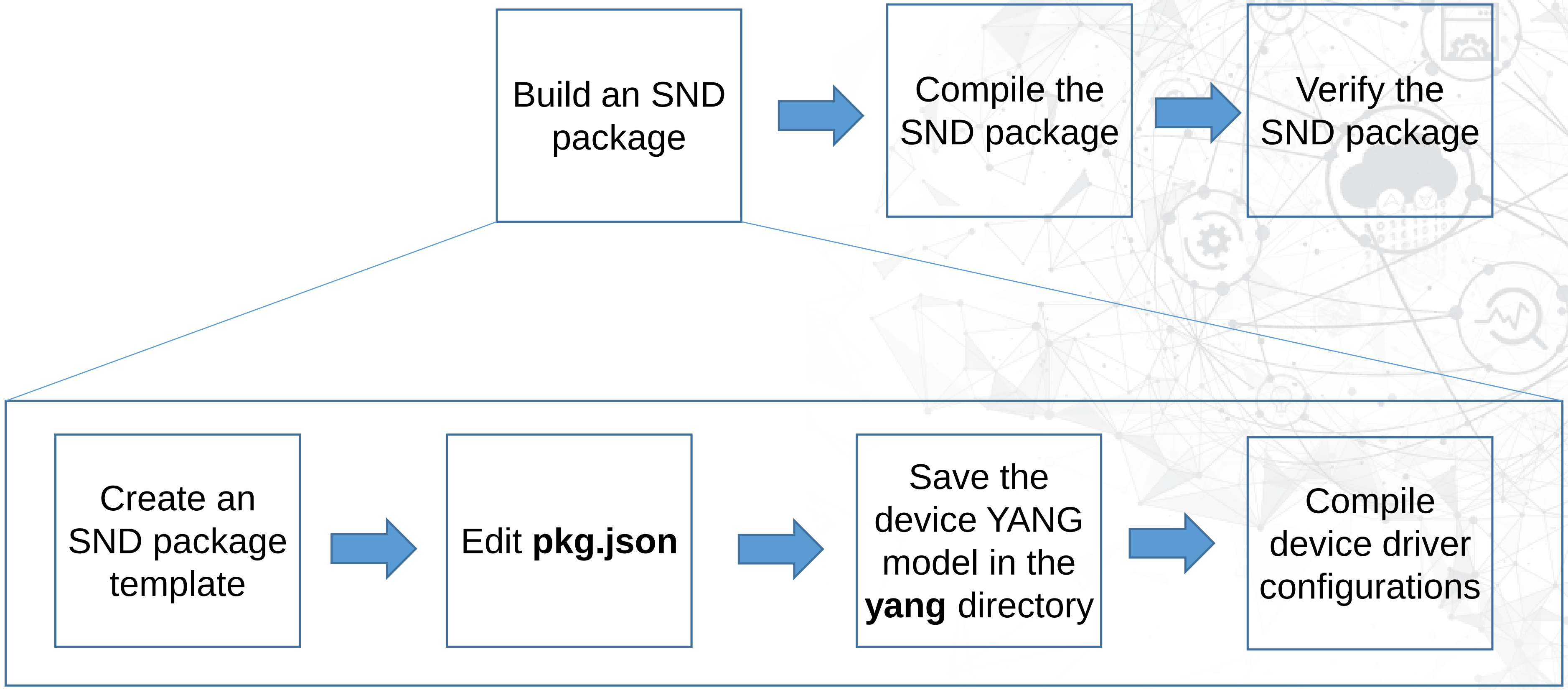
huawei-ftpc
huawei-ftps
huawei-grpc
huawei-host-defend
huawei-http
huawei-hwtacacs
huawei-ifit
huawei-ifm
huawei-ifm-trunk
huawei-ippmmgr
huawei-ipv6-nd
huawei-isis
huawei-keychain

3 Deliver the configuration.

1. Choose **Device Configuration** > **Device Configuration** from the main menu and click **Edit** in the **Operation** column.
2. In the displayed dialog box, select a model, edit the model data, and click **Dry-run** to check the protocol packets delivered to the device.

```
<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>
```

Development Process


Build an SND package
Compile the SND package
Verify the SND package
Create an SND package template
Edit **pkg.json**
Save the device YANG model in the **yang** directory
Compile device driver configurations

Thank You!