ATOM Platform Guide - Service Modeling

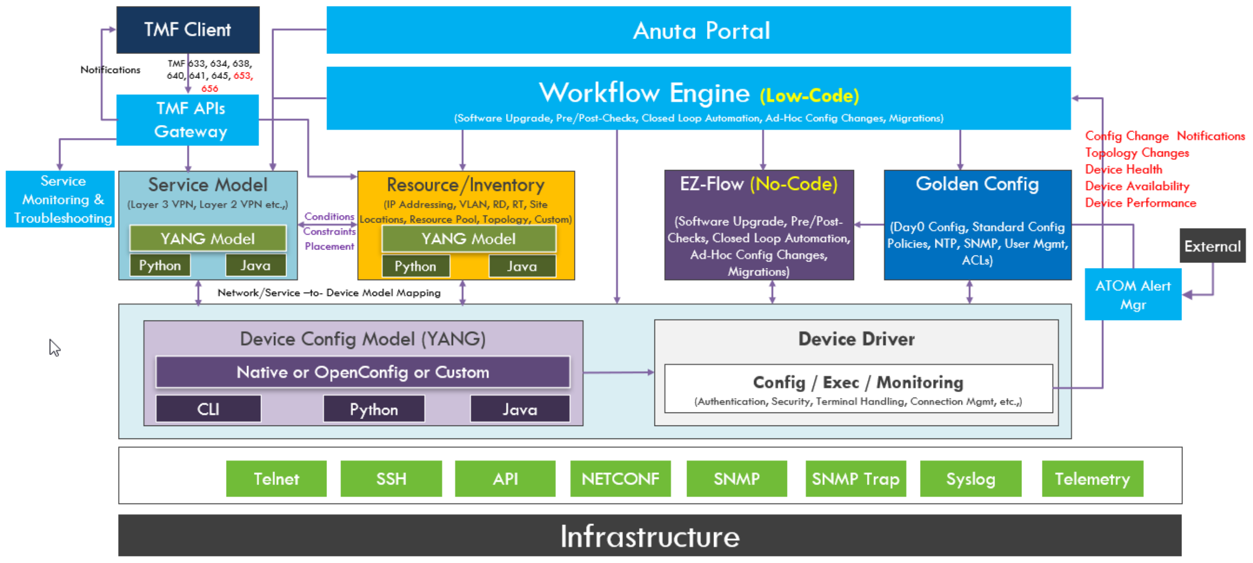
version 11.9

Table of Contents

Outline of the document

ATOM Platform provides users to develop various extensions to out-of-the box capabilities.

1) Device Drivers - Device Drivers allow ATOM to work with devices to Collect configuration, Provision Configuration, Collect Performance & Other Operational Data, Execute Show and Diagnostic Commands.
    a) Configuration Discovery & Provisioning
    b) Performance & Inventory Collection (SNMP, SNMP Trap, Syslog, Telemetry)
2) Network Automation
    a) Stateful Services like Application Delivery, L3 VPN, L2 VPN, Day0, Cloud Interconnect etc.,
    b) MOP Automation like Software Upgrade, Password Rotation etc.,


TMF Client
Notifications
TMF 633, 634, 638, 640, 641, 645, 653, 656
TMF APIs Gateway
Anuta Portal
Workflow Engine (Low-Code)
(Software Upgrade, Pre/Post-Checks, Closed Loop Automation, Ad-Hoc Config Changes, Migrations)
Service Monitoring & Troubleshooting
Service Model
(Layer 3 VPN, Layer 2 VPN etc.,)
YANG Model
Python
Java
Conditions
Constraints
Placement
Resource/Inventory
(IP Addressing, VLAN, RD, RT, Site Locations, Resource Pool, Topology, Custom)
YANG Model
Python
Java
EZ-Flow (No-Code)
(Software Upgrade, Pre/Post-Checks, Closed Loop Automation, Ad-Hoc Config Changes, Migrations)
Golden Config
(Day0 Config, Standard Config Policies, NTP, SNMP, User Mgmt, ACLs)
Config Change Notifications
Topology Changes
Device Health
Device Availability
Device Performance
External
ATOM Alert Mgr
Network/Service –to- Device Model Mapping
Device Config Model (YANG)
Native or OpenConfig or Custom
CLI
Python
Java
Device Driver
Config / Exec / Monitoring
(Authentication, Security, Terminal Handling, Connection Mgmt, etc.,)
Telnet
SSH
API
NETCONF
SNMP
SNMP Trap
Syslog
Telemetry
Infrastructure

This document covers Network Automation Stateful Services Development flows that have the following life cycle:

a. Create - Create a green field service
b. Update - Update Service. This may be repeated multiple times
c. Delete - Retire the service

Following is a high level breakdown of the content:

1. Working with Development tooling
2. Services Development against ATOM platform
3. Deploying, Upgrading & Operating Services in ATOM

In the Appendix, additional examples, extensions, library utils and FAQs are available in detail.

Intended Audience

This document is meant for the reader who is interested in developing network services or applications against ATOM.

ATOM relies heavily on YANG modeling language and RESTCONF. Hence, a good knowledge of

YANG and working knowledge of RESTCONF are required.

Service logic is implemented in python hence good working knowledge of python is required.

ATOM SDK is built on top of the Gradle build system. This document explains the build commands in detail.

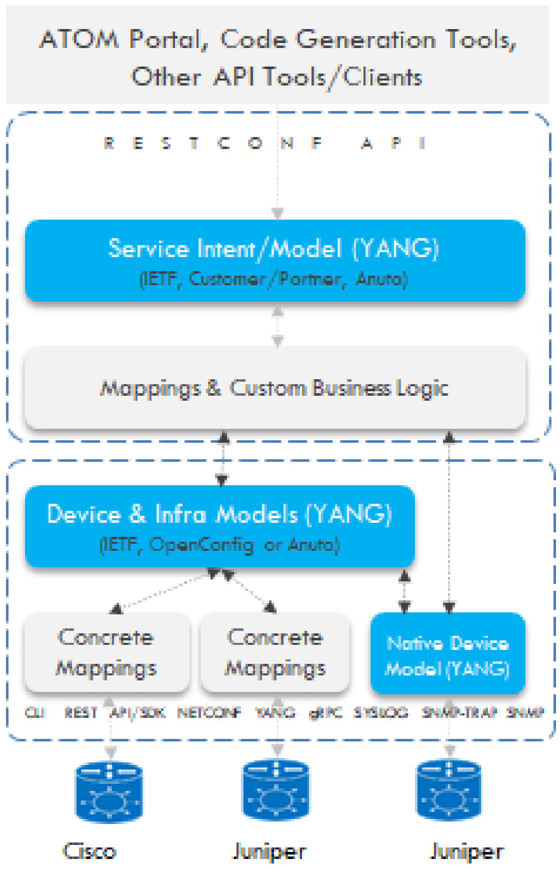
YANG : https://tools.ietf.org/html/rfc7950

RESTCONF : https://tools.ietf.org/html/rfc8040

Gradle : https://gradle.org/

Overview of Modelling in ATOM

Anuta ATOM platform with it's layered, YANG model driven approach helps in delivering vendor neutral, extensible and maintainable network services for multiple domains such as branch/CPE, Data Center, Cloud, and Carrier Core networks.

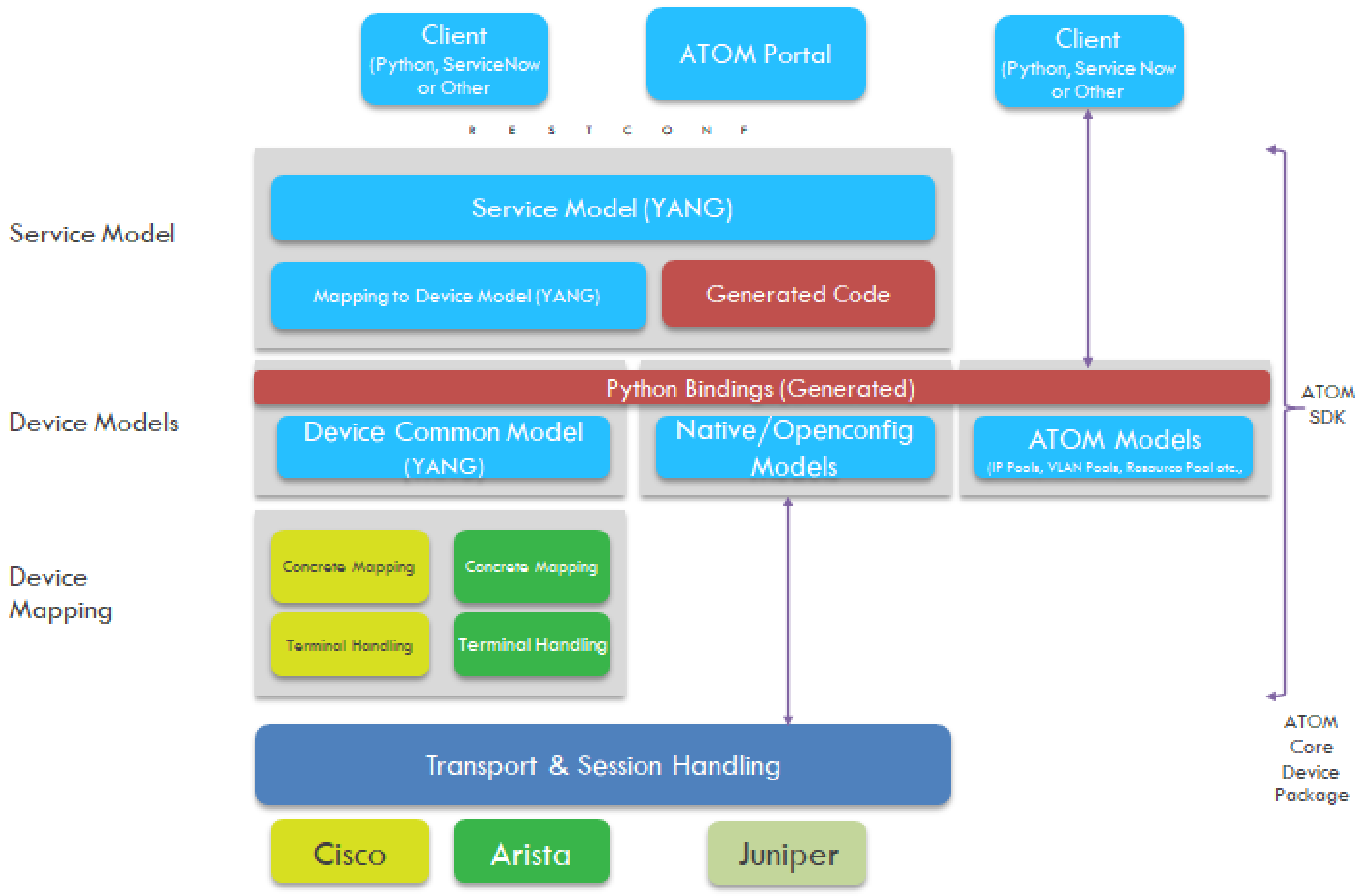

ATOM Portal, Code Generation Tools, Other API Tools/Clients
RESTCONF API
Service Intent/Model (YANG)
(IETF, Customer/Partner, Anuta)
Mappings & Custom Business Logic
Device & Infra Models (YANG)
(IETF, OpenConfig or Anuta)
Concrete Mappings
Concrete Mappings
Native Device Model (YANG)
CLI REST API/SDK NETCONF YANG gRPC SYSLOG SNMP-TRAP SNMP
Cisco
Juniper
Juniper

Network Service or application development involves the following major components:

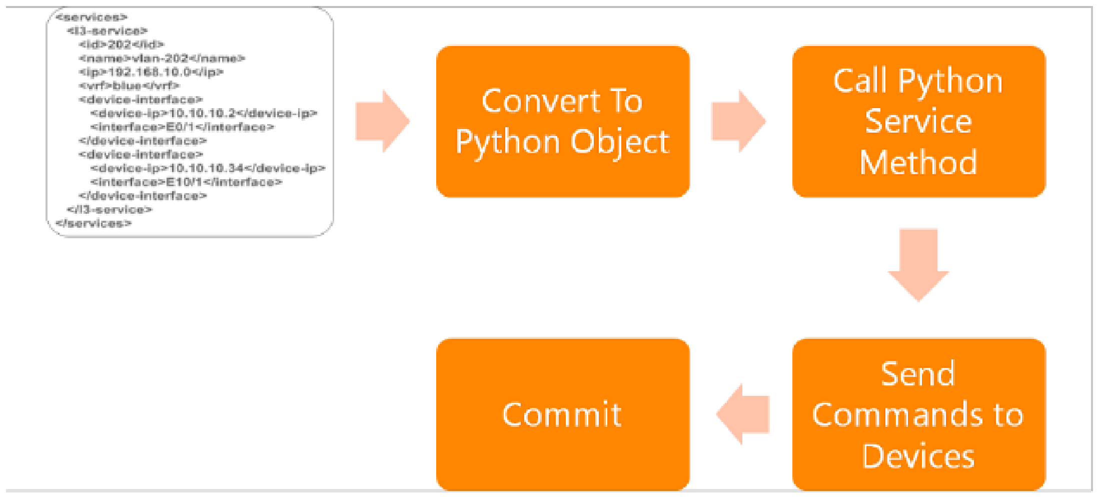
1. Device Model (Not required for Native/Openconfig Models)
    a. Abstracted object model defined using IETF YANG
    b. Defined once for each device feature or function
2. Vendor Plugin (Not required for Native/Openconfig Models)
    a. Common model mapped to vendor specific CLI or API
    b. Defined once for each vendor or OS type of a platform
3. Service Model
    a. Service description in YANG
    b. Mapping to Device models
    c. Glue logic to augment service model & mappings with business logic (Optional)

ATOM SDK and Platform provides a full life cycle for network services & apps in - Design, Develop, Package, Deploy, Test, Upgrade.

Each of the individual blocks covered under ATOM SDK development are illustrated below:


Client (Python, ServiceNow or Other
ATOM Portal
Client (Python, Service Now or Other
RESTCONF
Service Model
Service Model (YANG)
Mapping to Device Model (YANG)
Generated Code
Device Models
Python Bindings (Generated)
Device Common Model (YANG)
Native/Openconfig Models
ATOM Models (IP Pools, VLAN Pools, Resource Pool etc.,
ATOM SDK
Device Mapping
Concrete Mapping
Concrete Mapping
Terminal Handling
Terminal Handling
ATOM Core Device Package
Transport & Session Handling
Cisco
Arista
Juniper

How Service Modeling Works


```
<services>
  <l3-service>
    <id>202</id>
    <name>vlan-202</name>
    <ip>192.168.10.0</ip>
    <vrf>blue</vrf>
    <device-interface>
      <device-ip>10.10.10.2</device-ip>
      <interface>E0/1</interface>
    </device-interface>
    <device-interface>
      <device-ip>10.10.10.34</device-ip>
      <interface>E10/1</interface>
    </device-interface>
  </l3-service>
</services>
```

Service Package Development

Create the Service package

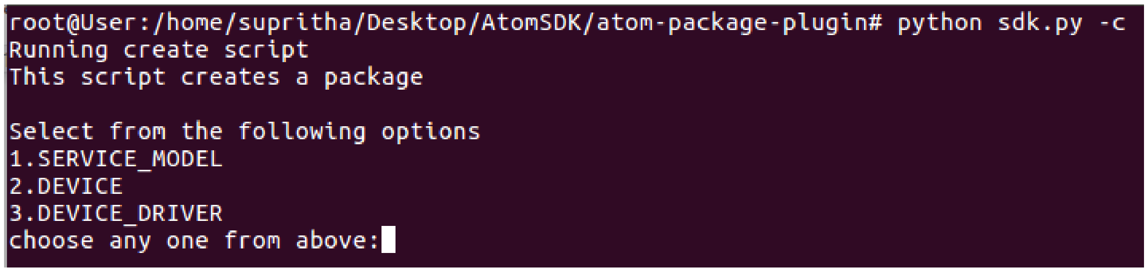
After the successful one time setup of the ATOM SDK environment (Refer Appendix section

ATOM SDK), you can create packages of your choice.

1. Run the command to create the requisite package:

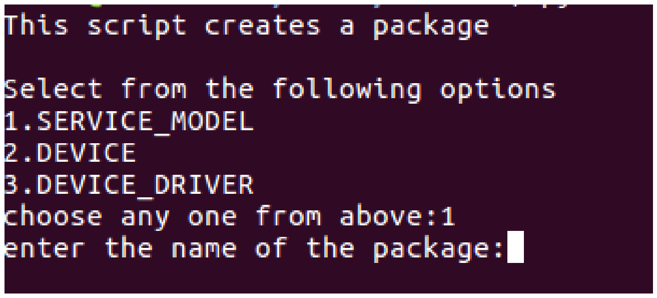
```
python sdk.py -c
```

create.py: This script helps you create different types of package; service package, device package, or device driver package.



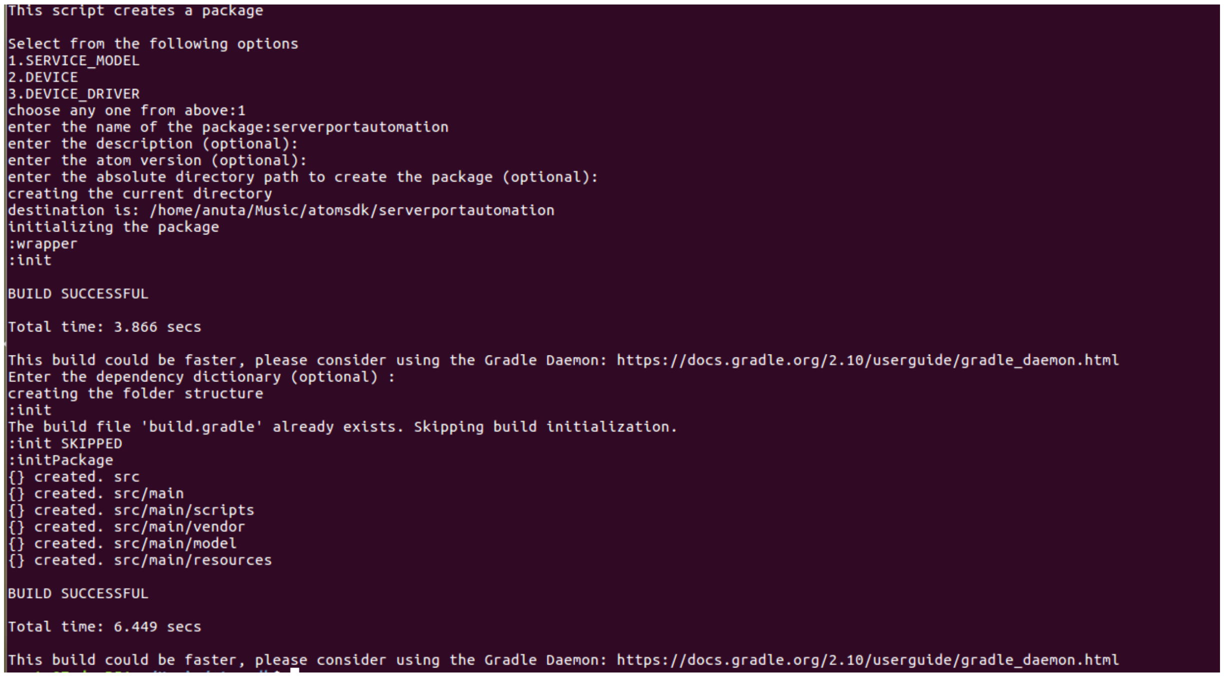
```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin# python sdk.py -c
Running create script
This script creates a package

Select from the following options
1.SERVICE_MODEL
2.DEVICE
3.DEVICE_DRIVER
choose any one from above:
```

2. Select the type SERVICE_MODEL package type as shown below:



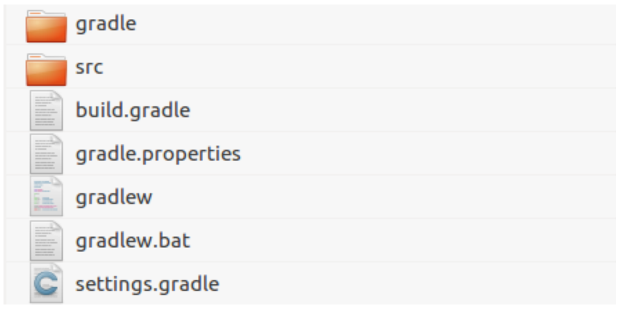
```
This script creates a package

Select from the following options
1.SERVICE_MODEL
2.DEVICE
3.DEVICE_DRIVER
choose any one from above:1
enter the name of the package:
```

3. Enter the name of the package and other inputs as shown below



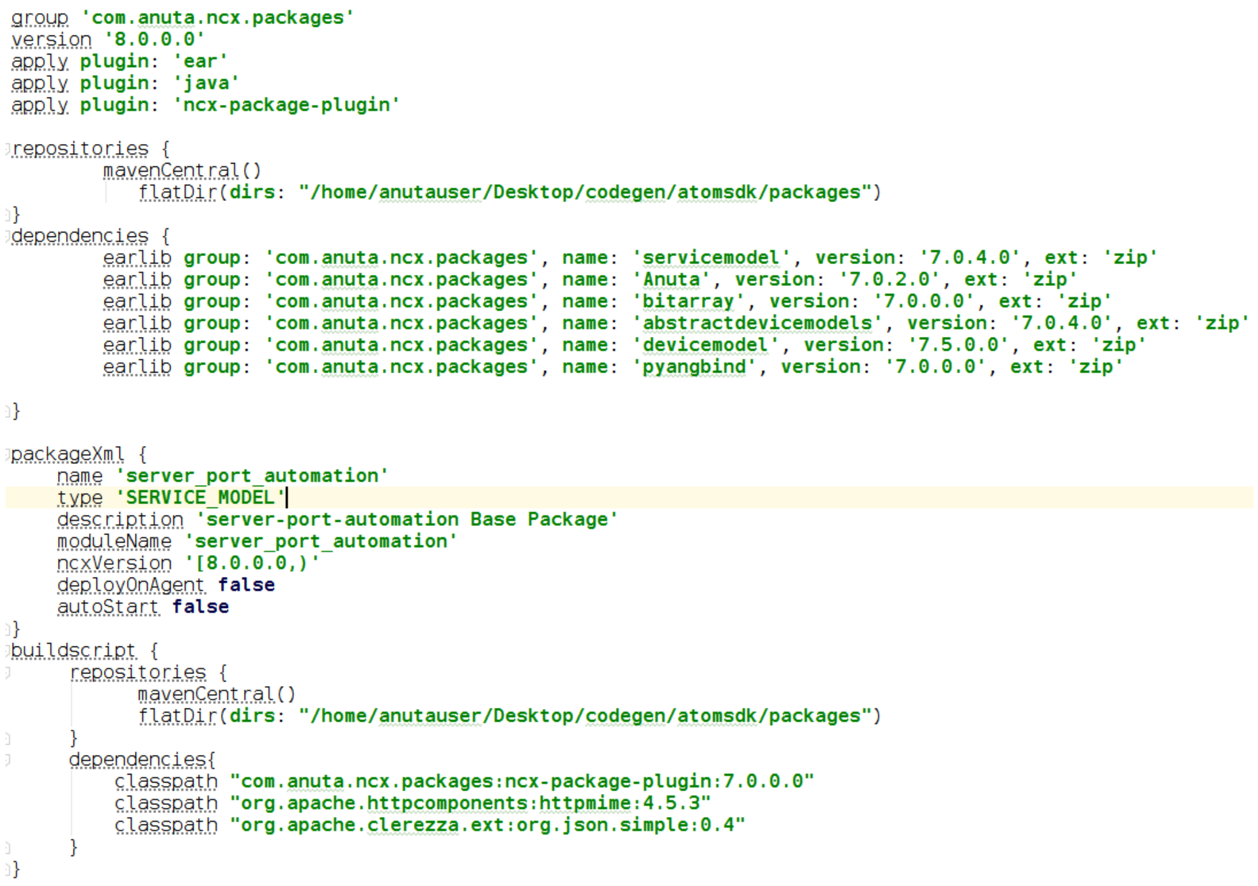
```
This script creates a package

Select from the following options
1.SERVICE_MODEL
2.DEVICE
3.DEVICE_DRIVER
choose any one from above:1
enter the name of the package:serverportautomation
enter the description (optional):
enter the atom version (optional):
enter the absolute directory path to create the package (optional):
creating the current directory
destination is: /home/anuta/Music/atomsdk/serverportautomation
initializing the package
:wrapper
:init

BUILD SUCCESSFUL

Total time: 3.866 secs

This build could be faster, please consider using the Gradle Daemon: https://docs.gradle.org/2.10/userguide/gradle_daemon.html
Enter the dependency dictionary (optional) :
creating the folder structure
:init
The build file 'build.gradle' already exists. Skipping build initialization.
:init SKIPPED
:initPackage
{} created. src
{} created. src/main
{} created. src/main/scripts
{} created. src/main/vendor
{} created. src/main/model
{} created. src/main/resources

BUILD SUCCESSFUL

Total time: 6.449 secs

This build could be faster, please consider using the Gradle Daemon: https://docs.gradle.org/2.10/userguide/gradle_daemon.html
```

After the successful run of the above build, the service package folder structure for service development purpose is created. The **service package** folder contains the following artifacts


gradle
src
build.gradle
gradle.properties
gradlew
gradlew.bat
settings.gradle

Update the Dependencies & Version in build.gradle

After a successful creation of a package, there could be some additional package(s) required as 'dependencies'. Accordingly modify the default dependencies listed in the build.gradle file, which is located in the root level of the created package.


```
group 'com.anuta.ncx.packages'
version '8.0.0.0'
apply plugin: 'ear'
apply plugin: 'java'
apply plugin: 'ncx-package-plugin'

repositories {
        mavenCentral()
           flatDir(dirs: "/home/anutauser/Desktop/codegen/atomsdk/packages")
}
dependencies {
        earlib group: 'com.anuta.ncx.packages', name: 'servicemodel', version: '7.0.4.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'Anuta', version: '7.0.2.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'bitarray', version: '7.0.0.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'abstractdevicemodels', version: '7.0.4.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'devicemodel', version: '7.5.0.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'pyangbind', version: '7.0.0.0', ext: 'zip'

}

packageXml {
    name 'server_port_automation'
    type 'SERVICE_MODEL'
    description 'server-port-automation Base Package'
    moduleName 'server_port_automation'
    ncxVersion '[8.0.0.0,)'
    deployOnAgent false
    autoStart false
}
buildscript {
     repositories {
           mavenCentral()
           flatDir(dirs: "/home/anutauser/Desktop/codegen/atomsdk/packages")
     }
     dependencies{
         classpath "com.anuta.ncx.packages:ncx-package-plugin:7.0.0.0"
         classpath "org.apache.httpcomponents:httpmime:4.5.3"
         classpath "org.apache.clerezza.ext:org.json.simple:0.4"
     }
}
```

In scenarios where service performs API invocations against Device or other Models, make sure dependency of that respective model package is there along with the servicemodel package which the library utils.

Let's consider a Service which performs invocations against Juniper Device Models, then make sure dependencies of *servicemodel-7.0.4.0*, *juniper-8.0.0.1, juniper_cli-8.0.0.1* are mentioned as below. (The package names are a combination of the name of the package and the version number separated by a hyphen)

```
dependencies {
        earlib group: 'com.anuta.ncx.packages', name: 'juniper', version: '8.0.0.1', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'juniper_cli', version: '8.0.0.1', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'servicemodel', version: '7.0.4.0', ext: 'zip'
```

Resolve the dependencies

Run the command : **gradle build --refresh-dependencies**

Successful execution of this command ensures that the dependencies mentioned in the *build.gradle* file are mapped fine.

Update the version

In the "*build.gradle*" file, metadata about the package is present in the version & packageXml object. Update the version based on the revision of the service package you are working on.

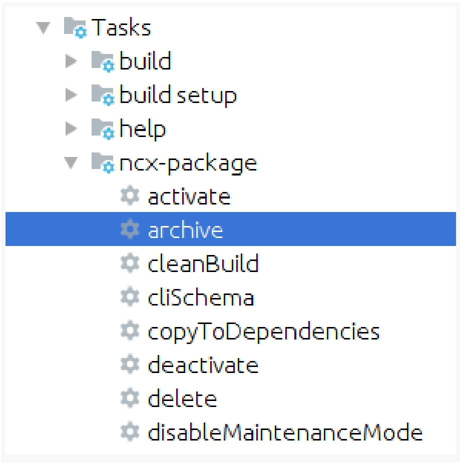
```
group 'com.anuta.ncx.packages'
version '8.0.0.0'
apply plugin: 'ear'
apply plugin: 'java'
apply plugin: 'ncx-package-plugin'
```

Yang Modelling and Service Logic

Network Service modelling and defining of Service logic in python can be done within the model and scripts folder respectively of the created service package structure. Refer to Service Modelling section for detailed procedure of it.

Archive the Service Package

Once the modelling and service logic are defined in the respective package structure, use the gradle task ***"gradle archive"*** for creating the uploadable zip with its dependencies.

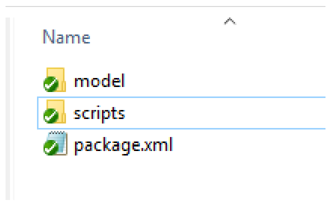


The zip will be stored in the build folder and is ready for Upload to ATOM.

Service Yang Modeling and Logic

Contents of a Service package

The service package which can be uploaded into ATOM as a zip format typically contains the following entities:


Name
model
scripts
package.xml

1. The model folder contains the following:
    a. *<service_name>.yang* file - Contains the schema of the service defined in YANG
2. The scripts folder contains python based service logic consiqueting following files typically:
    a. *<service_name>.py* - Contains the logic binding of the service to the device

       Generates the device model operations with reference to the service defined in the yang model file and also registers service in ATOM

    b. *plugin.py* - Code for adding the service as a new plug-in to ATOM
    c. *_init_py* - Required to make ATOM treat the directories as containing packages
3. The package.xml file contains the metadata about the service.
    a. Information such as package name, version, module name, type and description should be provided.

This section outlines the procedure for creating, deploying, and testing a service package built by using the ATOM SDK.

Procedure for Modelling & Defining Logic

Follow these steps for creating a service package:

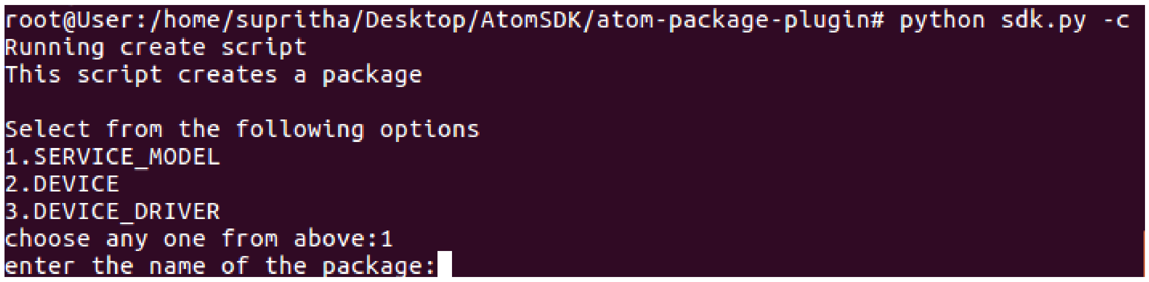
1. Define the service model file - write the service.yang
    a. Add Yang validations & constraints for building intelligence in the model
    b. Define ATOM extensions useful to map service node to device node
    c. Above extensions will help for codegen in step2
2. AutoGenerate service logic using SDK & Define any custom logic in python

Let us take an example of building a service package (**day0service package**) from *service.yang* file (*day0config.yang*) using Plugin Tasks for package development. This day0service is intended to deploy some of the day0 features on the device after it was plugged into the lab with management reachability.

Create a Service model .yang file

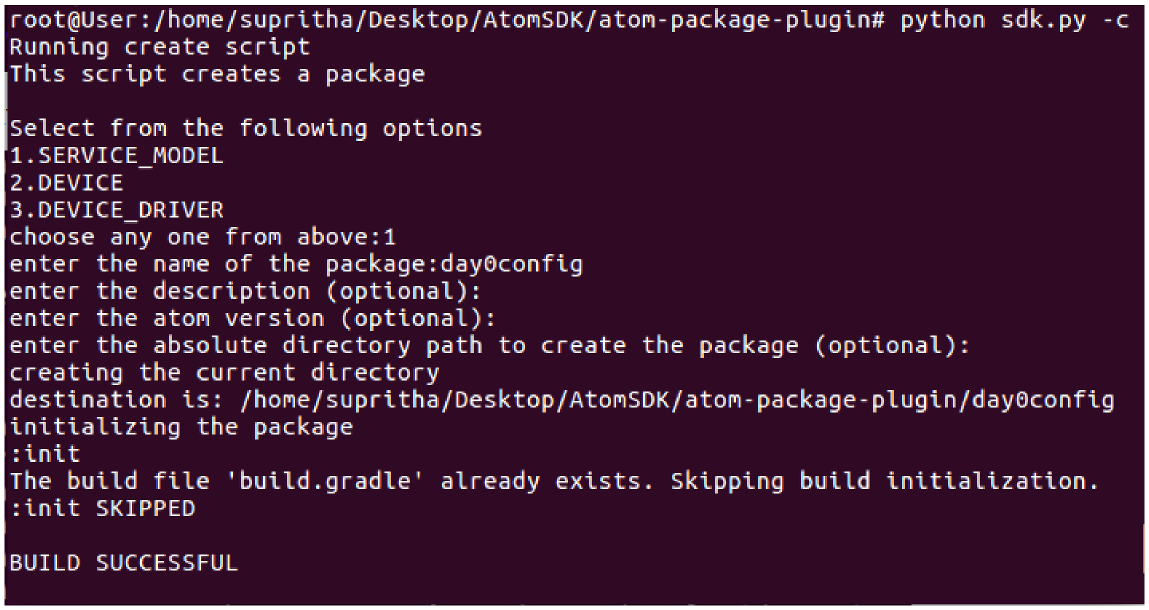
1. Extract the provided ATOM SDK .zip and Setup the environment as per steps mentioned in Setting up the environment for package development.

2. Select the required package type as SERVICE_MODEL.



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin# python sdk.py -c
Running create script
This script creates a package

Select from the following options
1.SERVICE_MODEL
2.DEVICE
3.DEVICE_DRIVER
choose any one from above:1
enter the name of the package:
```

3. Give the name of the service model: day0config



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin# python sdk.py -c
Running create script
This script creates a package

Select from the following options
1.SERVICE_MODEL
2.DEVICE
3.DEVICE_DRIVER
choose any one from above:1
enter the name of the package:day0config
enter the description (optional):
enter the atom version (optional):
enter the absolute directory path to create the package (optional):
creating the current directory
destination is: /home/supritha/Desktop/AtomSDK/atom-package-plugin/day0config
initializing the package
:init
The build file 'build.gradle' already exists. Skipping build initialization.
:init SKIPPED

BUILD SUCCESSFUL
```

After the successful run of the build, the service package, **day0config** gets created.

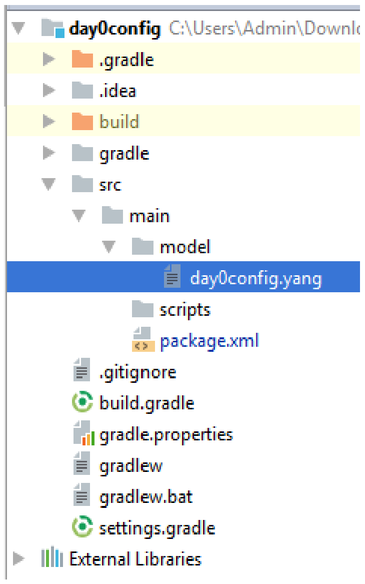
Note:: In the service package development process, make sure the package folder name, yang file and module names are the same.

4. Take reference of the **.yang** present in the examples folder of ATOMSDK to develop **day0config.yang** service model and place it day0config package path **day0config/src/main/model**. This model is for creating multiple user login a/c's on the device as part of day0 configurations to be done on the device.

```
module day0config {
  namespace "http://anutanetworks.com/day0config";
  prefix day0config;


  import controller {
    prefix ac;
  }
```

```
import user {
  prefix cu;
}
import ncx-extensions {
  prefix n-ext;
}

organization
  "Anuta Networks";
description
  "This module contains a collection of YANG definitions for
   day0 configuration ";

revision 2016-06-16 {
  description
    "Initial revision";
}

augment "/ac:services" {
  container day0services {
    list day0service {
      key "name";
      leaf name {
        type string;
        description
          "string";
      }
      leaf device-ip {
        type leafref {
          path "/ac:devices/ac:device/ac:id";
        }
        mandatory true;
        description
          "device-ip";
      }
      container users {
        list user {
          key "name";
```

```
            leaf name {
              type string;
              description
                "string";
              n-ext:maps-to "/ac:devices/ac:device[ac:id=current()/../../device-ip]/cu:users/user/name";
            }
            leaf password {
              type string;
              description
                "string";
              n-ext:maps-to
      "/ac:devices/ac:device[ac:id=current()/../../device-ip]/cu:users/user/password";
            }
            leaf password-level {
                    type enumeration {
                     enum "0";
                                enum "7";
                 }
                    description
                       "password encryption level indicator";
                   n-ext:maps-to "/ac:devices/ac:device[ac:id=current()/../../device-
      ip]/cu:users/user/password-level";
             }
          }
        }
      }
    }
   }
}
```



5. In the above yang model we can see various Yang Constraints and ATOM extensions added to bring intelligence into the model and enhance the capabilities of ATOM UI/platform and SDK.

AutoGenerate & Define Service logic in Python

For any given service.yang ATOM SDK has the capability to generate service package structure and the required files for service logic by using python libraries available at **atomsdk/packages/servicemodel/scripts/**. For detailed reading on python libraries refer to the Appendix section Library Utils for Service Modelling.

SDK can auto generate python based business logic completely where custom logic addition is not required at all. To utilize the functionality of the full code generation, use the extension, **maps-to,** for all leaf nodes in a service.yang file.

Example: n-ext:maps-to <device model x-path/rc-path>

Few more ATOM yang extensions relevant to UI and code-generation refer to the Appendix section ATOM Extensions to yang.

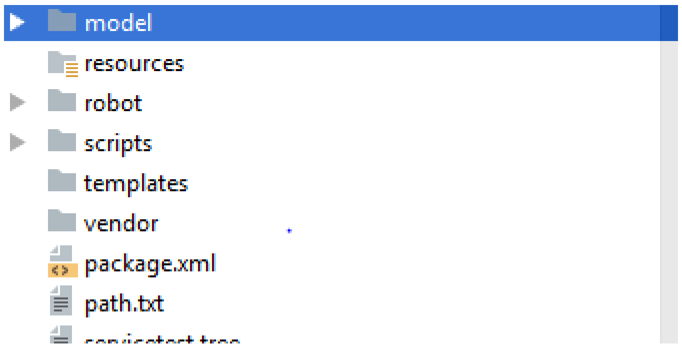
To generate a service package, make sure the service yang is inside the directory **'src/main/model'**. The user has the flexibility to generate the output in '**build/generated'** or in **'src/main'** itself. For details refer plugin task.

Change to current directory i.e. the service package directory and run the commands

gradle tasks --all (this will display all the tasks)

gradle generateServicePackage(Runs the ServicePackage generate task)

Verify the package is created in the destination folder with below folders and file:



In the **Scripts** folder, multiple python files and folders are generated as explained below:

❖ A sample of the *plugin.py* file that is required for registering and unregistering of package to ATOM is shown below:

```
def get_plugin_info():
  return Plugin('day0config', '1.0.0')

from com.anuta.service.python.plugin import PythonPlugin
from com.anuta.service.python.plugin import PythonPluginType

from servicemodel import util
import day0config
from day0config_lib import log

class Plugin(PythonPlugin):
  """Class to register day0config plugin to ATOM
  """
  def __init__(self, name, version):
    self.setName(name)
    self.setVersion(version)
            self.setPluginType(PythonPluginType.SERVICE_MODEL)
            self.setDescription('DAY0CONFIGService Plugin')
  def init(self):
    log('registering day0config')
    day0config.DAY0CONFIGService.getInstance().register()

  def shutdown(self):
    log('unregistering day0config')
    day0config.DAY0CONFIGService.getInstance().unregister()
```

❖ A sample of the *“day0config.py”* file is as below and explained:

Import Block:

Below block represents importing library servicemodel.util, servicemodel.yang and servicemodel.devicemgr

```
from servicemodel import util
from servicemodel import yang
from servicemodel import devicemgr
from day0config_lib import getCurrentObjectConfig
import day0services.day0service.day0service
import day0services.day0service.users.user.user
```

Handler-Maps

Every use-case contains one main file, here day0config.py with its class DAY0CONFIGService() having handler maps as below.

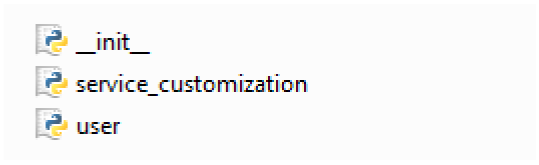
Significance of Handler-Maps: When client enters data from UI or from RESTCONF Client of any entity (container or list) in usecase, respective handler map is called from python glue logic. Each handler-map is associated with its entity class definition.

```
class DAY0CONFIGService(yang.AbstractYangServiceHandler):
  """Class for handling day0config service creation request.
  """
  _instance = None


  def create(self, id, sdata):
    config = getCurrentObjectConfig(id, sdata, None)


  def __init__(self):
    yang.AbstractYangServiceHandler.__init__(self)
    self.handler_map = {
      'day0config:day0services/day0service':
day0services.day0service.day0service.Day0Service.getInstance(),
      'day0config:day0services/day0service/users/user':
day0services.day0service.users.user.user.User.getInstance(),
    }
  @staticmethod
  def getInstance():
    if(DAY0CONFIGService._instance is None):
      DAY0CONFIGService._instance = DAY0CONFIGService()
    return DAY0CONFIGService._instance
```

❖ The supporting methods specific to this service package are available in the form of *"day0config_lib.py"* file. In addition to these, you can add any other **lib python modules,** if required for this service package.

When ATOM SDK is used for code generation of a service.yang file, it will generate a folder hierarchy within the scripts folder of the generated package as per yang tree structure.

For each folder there will be a service_customization.py file to add modifications required in auto generated python service logic. For example, a **user** model which is of a yang type list will have the following file structure.



Below we can see the structure of code generated within these files.

Class Structure Hierarchy

The structure of the User class is shown below:

```
class User(yang.AbstractYangServiceHandler):
  _instance = None

  def __init__(self):
    self.delete_pre_processor = service_customization.DeletePreProcessor()
    self.create_pre_processor = service_customization.CreatePreProcessor()
    self.opaque_args = {}

  def create(self, id, sdata):
    #Create block


  @staticmethod
  def getInstance():
    if(User._instance == None):
      User._instance = User()
    return User._instance
```

Create (Codegen)

When a client creates the service from UI or via RESTCONF, the payload is sent as an input to the **create()** block of service code as **sdata.** ATOM Platform keeps track of all the references for device entries/entities it created during a service create operation.

'create' code contains below basic structure of generated code

Fetching input values

The input values sent by the client to the python glue logic is captured in **inputdict** as shown in the following snippet of the code:

```
def create(self, id, sdata):
  sdata.getSession().addYangSessionPreReserveProcessor(self.create_pre_processor)

  #Fetch Local Config Object
  config = getCurrentObjectConfig(id, sdata, 'user')

  #Fetch Service Model Context Object
  smodelctx = ServiceModelContext(id, sdata)

  #Fetch Parent Object
  parentobj = getParentObject(sdata)

  dev = []
  inputkeydict = {}
  devbindobjs={}
  inputdict = {}
  opaque_args = self.opaque_args

  # START OF FETCHING THE LEAF PARAMETERS
   inputdict = getDictFromConfig(config)
  # END OF FETCHING THE LEAF PARAMETERS
```

Create logic using input values

The input values given for service yang leafs are assigned to respective leaf variables of device models based on the maps-to statement defined for that service yang leaf. This mapping is maintained in a dictionary called mapping_dict.

Create method requires this mapping_dict as one input along with sdata, dev(which is device object), and addReference.

```
import servicemodel.device_abs_lib.users.user

if inputdict.get('name') is not None:
        servicemodel.device_abs_lib.users.user.user().create(sdata, dev, fill_map_devices_device_users_user(inputdict,
sdata=sdata), addref=True)
```

```
def fill_map_devices_device_users_user(inputdict, sdata=None, pinputdict={}, delete=False, update=False):
  mapping_dict_devices_device_users_user = {}
  mapping_dict_devices_device_users_user['password'] = inputdict.get('password') if not delete else '' if
inputdict.get('password') is not None else inputdict.get('password')
  mapping_dict_devices_device_users_user['password_level'] = inputdict.get('password_level') if not delete else '' if
inputdict.get('password_level') is not None else inputdict.get('password_level')
  mapping_dict_devices_device_users_user['name'] = inputdict.get('name') if not update else inputdict.get('name') if
inputdict.get('name') is not None else pinputdict.get('name')
  return mapping_dict_devices_device_users_user
```

In the above we see the create method present in library servicemodel.device_abs_lib.users.user is invoked which is based on the maps-to statements defined in the service yang. This create method translates to device entity creation in ATOM with reference held in the platform based on addReference being True/False.

In case the developer wants to code additional custom logic for service Create, he can do it on top of generated code within the **create()** block.

Update (AutoUpdate)

When a client tries to edit an existing service entry, then the payload is sent as an input to the **create()** block itself as **sdata.** The code within the **create** itself will be executed with latest updated service parameter values.

ATOM platform handles the update on device entries/entities automatically using the references it had previously. Hence ATOM as a platform gives **AutoUpdate** functionality.

In case the developer wants to code additional custom logic for service Update without affecting Create logic, he can still do it using a flag **sdata.autoupdate** which gets set as True when client did an update of service entry. Hence the update specific code customization can be added under **if (sdata.autoupdate)** within the create block itself.

Delete (AutoDelete)

With the service entity deleted, ATOM Platform handles the delete of device entries/entities automatically using the references it had previously. Hence the ATOM as a platform gives **AutoDelete** functionality.

In case the developer wants to code additional custom logic for service Delete, he can still do it by having delete() definition defined.

```
def delete(self, id, sdata):
   #Delete custom logic
```

Extending Service Model & Custom Logic

Let us take an example of adding an extra day0 feature (dns-name-server) to the existing day0service. The service logic should be modified to accommodate the changes in the service thereby extending the service model.

1. Open the *day0config.yang* file present in **src/main/model** and add the dns-name-server list entry as shown below.
2. Extend the *day0config.yang* as shown below:

```
container dns {
    list name-server {
      key "name-server";
      leaf name-server {
                description
                  "Valid IPv4 Address (A.B.C.D for e.x: 172.16.1.1)";
                type inet:ipv4-address;
      }
      leaf vrf-name {
                type string;
                description
                  "string, Ex:management";
      }
    }
}
```

3. For the service.yang generate service package using ATOM SDK
   i. Change to current directory i.e /ServicePackage
   ii. Run the commands

```
gradle tasks --all (this will display all the tasks )
gradle generateServicePackage (Runs the ServicePackage generate task)
```

4. Verify the package is updated with new service logic and new folders.

   i. Open the "*name_server.py"* file (in the **<pkg>/scripts/day0services/day0service/dns/name_server** folder) and look for the existing python logic.

   As ATOM Extensions of the 'maps-to' attribute has not been added for the name-server, the python logic is not fully complete. The logic does not contain the device bindings and the corresponding payload.

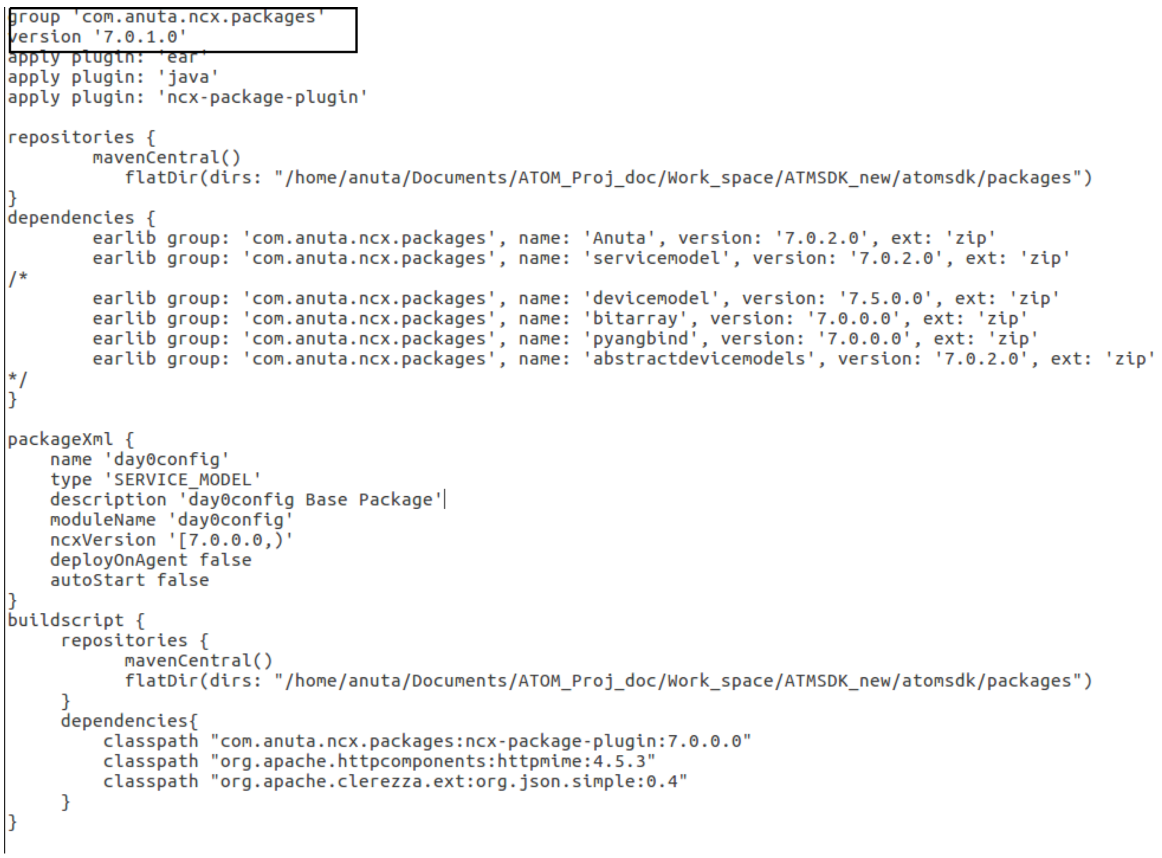
5. Add the required custom logic in the "*service_customization.py"* file (in the **<pkg>/scripts/day0services/day0service/dns/name_server** folder).

Add the missing name-server device bindings and then the create operation with name-server payload.

An excerpt of the code from the **def process_service_device_bindings** of the *"service_customization.py"* file is shown below:

```
if modify:
  config = kwargs['config']
  inputdict = kwargs['inputdict']
  inputkeydict = kwargs['inputkeydict']
  devbindobjs = kwargs['devbindobjs']
  id = kwargs['id']
  opaque_args = kwargs['hopaque']

  if dev is None or (isinstance(dev,list) and len(dev)==0):
    return

  import servicemodel.device_abs_lib.dns_server.name_server

  if inputdict.get('name_server') is not None:
          servicemodel.device_abs_lib.dns_server.name_server.name_server().create(sdata, dev,
      fill_map_devices_device_dns_server_name_server(inputdict, sdata=sdata), addref=True)
```

6. Open the **build.gradle** and upgrade service package by modifying value of version field from 7.0.0.0 to 7.0.1.0



```
group 'com.anuta.ncx.packages'
version '7.0.1.0'
apply plugin: 'ear'
apply plugin: 'java'
apply plugin: 'ncx-package-plugin'

repositories {
        mavenCentral()
           flatDir(dirs: "/home/anuta/Documents/ATOM_Proj_doc/Work_space/ATMSDK_new/atomsdk/packages")
}
dependencies {
        earlib group: 'com.anuta.ncx.packages', name: 'Anuta', version: '7.0.2.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'servicemodel', version: '7.0.2.0', ext: 'zip'
/*
        earlib group: 'com.anuta.ncx.packages', name: 'devicemodel', version: '7.5.0.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'bitarray', version: '7.0.0.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'pyangbind', version: '7.0.0.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'abstractdevicemodels', version: '7.0.2.0', ext: 'zip'
*/
}

packageXml {
    name 'day0config'
    type 'SERVICE_MODEL'
    description 'day0config Base Package'
    moduleName 'day0config'
    ncxVersion '[7.0.0.0,)'
    deployOnAgent false
    autoStart false
}
buildscript {
     repositories {
           mavenCentral()
           flatDir(dirs: "/home/anuta/Documents/ATOM_Proj_doc/Work_space/ATMSDK_new/atomsdk/packages")
     }
     dependencies{
         classpath "com.anuta.ncx.packages:ncx-package-plugin:7.0.0.0"
         classpath "org.apache.httpcomponents:httpmime:4.5.3"
         classpath "org.apache.clerezza.ext:org.json.simple:0.4"
     }
}
```

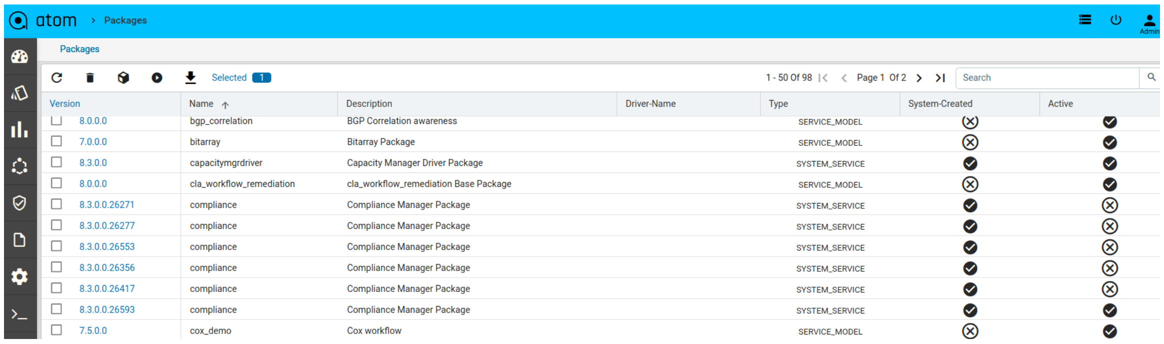
7. This package can be archived and uploaded to ATOM for use. For archiving manually zip the model, scripts folders, and the *package.xml* and upload to ATOM.

   For archiving using SDK, change to the directory level of package and run command:
   gradle archive

Deploying & Operating Service Packages

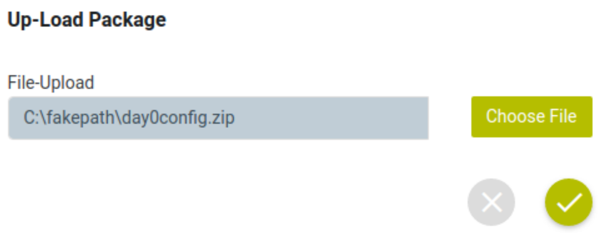
Deploying a Service Package

1. Considering the previously created day0config service, Upload the **day0config.zip** into ATOM.

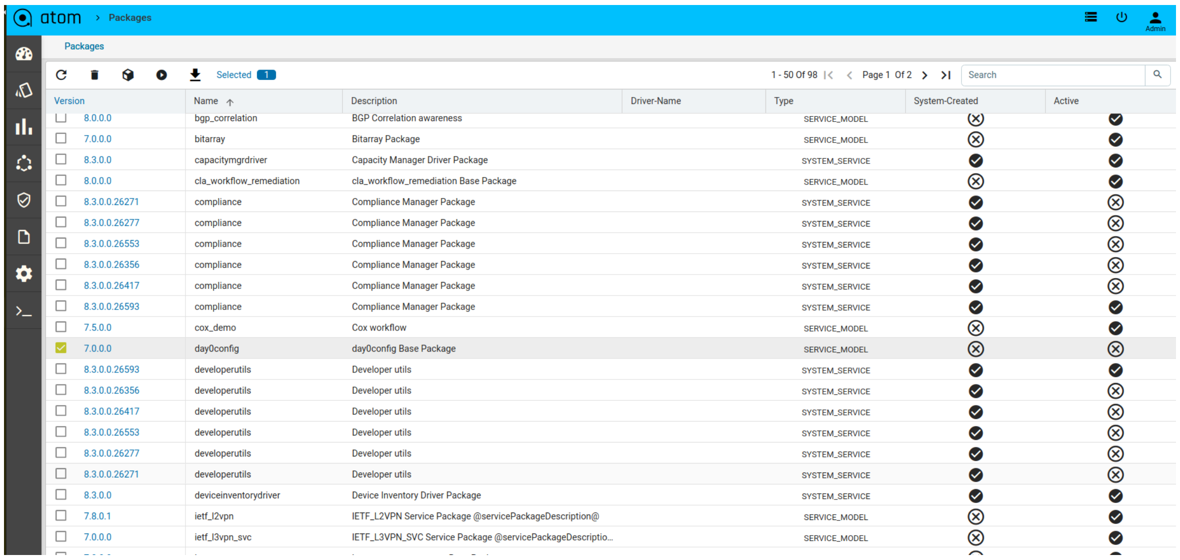


Version	Name	Description	Driver-Name	Type	System-Created	Active
8.0.0.0	bgp_correlation	BGP Correlation awareness		SERVICE_MODEL	⊗	✓
7.0.0.0	bitarray	Bitarray Package		SERVICE_MODEL	⊗	✓
8.3.0.0	capacitymgrdriver	Capacity Manager Driver Package		SYSTEM_SERVICE	✓	✓
8.0.0.0	cla_workflow_remediation	cla_workflow_remediation Base Package		SERVICE_MODEL	⊗	✓
8.3.0.0.26271	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26277	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26553	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26356	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26417	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26593	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	✓
7.5.0.0	cox_demo	Cox workflow		SERVICE_MODEL	⊗	✓

2. Click **Add package** and upload the **day0config.zip** and click the **OK** button.

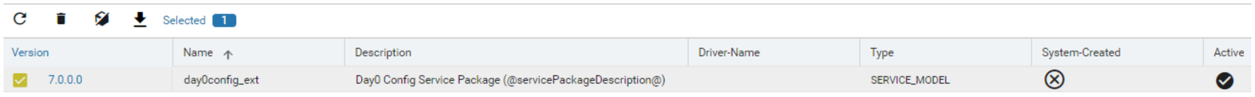


If any error occurs, verify if all the dependent packages are uploaded and registered in ATOM.



Version	Name	Description	Driver-Name	Type	System-Created	Active
8.0.0.0	bgp_correlation	BGP Correlation awareness		SERVICE_MODEL	⊗	✓
7.0.0.0	bitarray	Bitarray Package		SERVICE_MODEL	⊗	✓
8.3.0.0	capacitymgrdriver	Capacity Manager Driver Package		SYSTEM_SERVICE	✓	✓
8.0.0.0	cla_workflow_remediation	cla_workflow_remediation Base Package		SERVICE_MODEL	⊗	✓
8.3.0.0.26271	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26277	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26553	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26356	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26417	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26593	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	✓
7.5.0.0	cox_demo	Cox workflow		SERVICE_MODEL	⊗	✓
7.0.0.0	day0config	day0config Base Package		SERVICE_MODEL	⊗	⊗
8.3.0.0.26593	developerutils	Developer utils		SYSTEM_SERVICE	✓	✓
8.3.0.0.26356	developerutils	Developer utils		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26417	developerutils	Developer utils		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26553	developerutils	Developer utils		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26277	developerutils	Developer utils		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26271	developerutils	Developer utils		SYSTEM_SERVICE	✓	⊗
8.3.0.0	deviceinventorydriver	Device Inventory Driver Package		SYSTEM_SERVICE	✓	✓
7.8.0.1	ietf_l2vpn	IETF_L2VPN Service Package @servicePackageDescription@		SERVICE_MODEL	⊗	✓
7.0.0.0	ietf_l3vpn_svc	IETF_L3VPN_SVC Service Package @servicePackageDescriptio…		SERVICE_MODEL	⊗	✓

3. Load the Service Package
   To load **day0config.zip** in ATOM, select the package in the list and click the **Load** button and verify that the service package is in “Active” state (true) as shown below.



Version	Name	Description	Driver-Name	Type	System-Created	Active
7.0.0.0	day0config_ext	Day0 Config Service Package (@servicePackageDescription@)		SERVICE_MODEL	⊗	✓

4. All the above manual steps can be done using SDK gradle cmds.

   gradle load (to upload package to ATOM)

   gradle activate (to activate package, sets active = true)

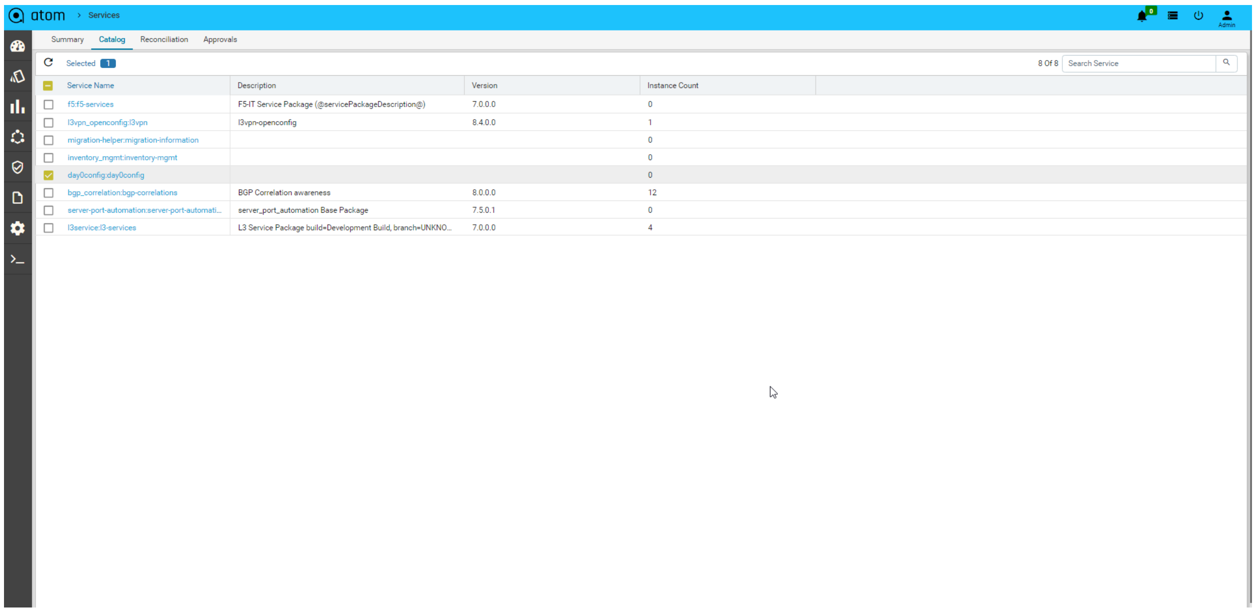
   gradle deactivate (to deactivate package, sets active = false)

5. For Extending/Migrating the existing **day0config-7.0.0.0.zip** to **day0config-7.0.1.0.zip,** refer to the section, "Upgrading Service packages"

Testing the Service Package

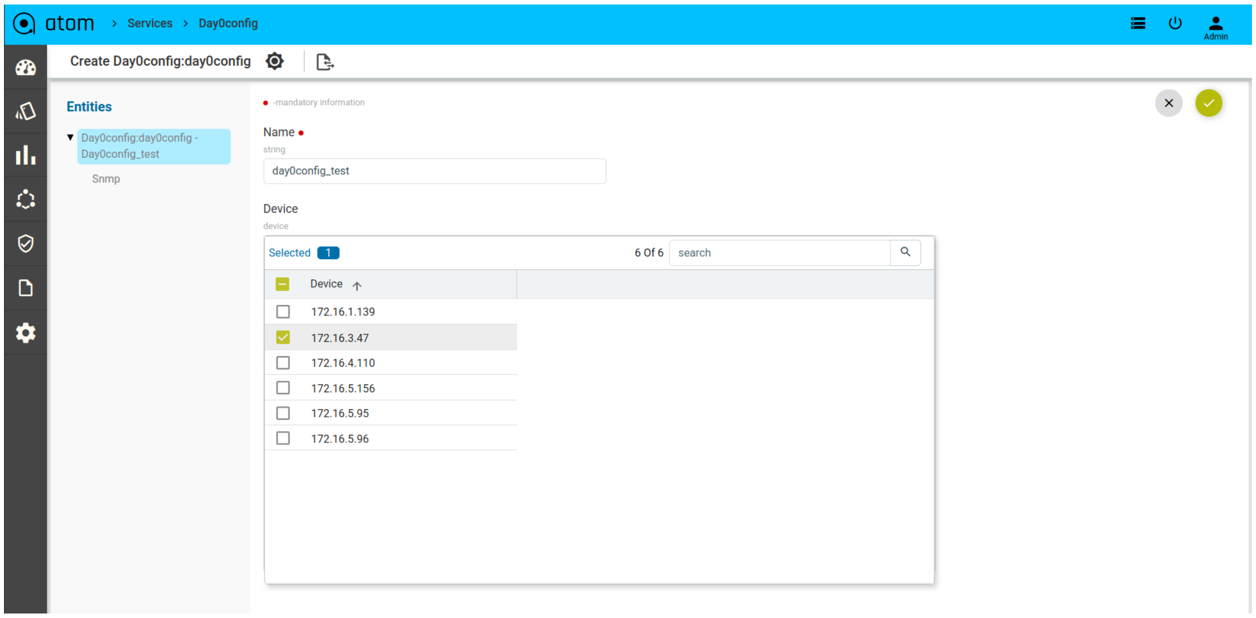
1. Navigate to **Automation > Services** to instantiate the day0service service.

   Click on day0services below



	Service Name	Description	Version	Instance Count
☐	f5:f5-services	F5-IT Service Package (@servicePackageDescription@)	7.0.0.0	0
☐	l3vpn_openconfig:l3vpn	l3vpn-openconfig	8.4.0.0	1
☐	migration-helper:migration-information			0
☐	inventory_mgmt:inventory-mgmt			0
✓	day0config:day0config			0
☐	bgp_correlation:bgp-correlations	BGP Correlation awareness	8.0.0.0	12
☐	server-port-automation:server-port-automati...	server_port_automation Base Package	7.5.0.1	0
☐	l3service:l3-services	L3 Service Package build=Development Build, branch=UNKNO...	7.0.0.0	4

2. Fill the form and click on the OK button


atom
Services
Day0config
Admin
Create Day0config:day0config
Entities
Day0config:day0config - Day0config_test
Snmp
Selected 1
6 Of 6
search

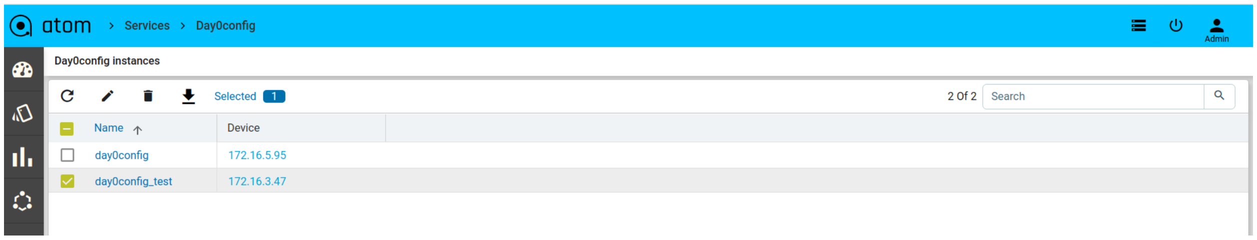
- mandatory information

Name ●
string
day0config_test

Device
device

	Device
☐	172.16.1.139
✓	172.16.3.47
☐	172.16.4.110
☐	172.16.5.156
☐	172.16.5.95
☐	172.16.5.96

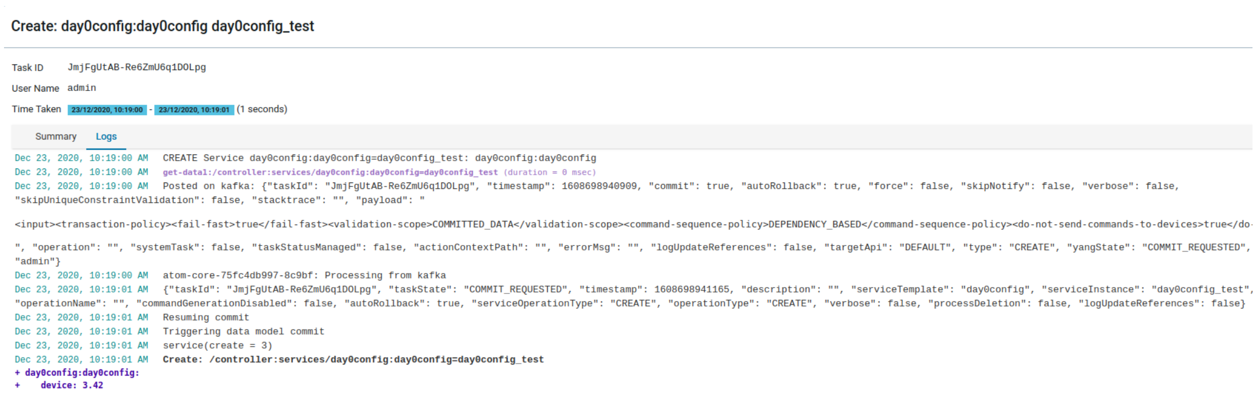
3. Successfully created day0 service entry can be seen below,



	Name	Device
☐	day0config	172.16.5.95
✓	day0config_test	172.16.3.47

4. To view details of the task associated with service creation, go to Task viewer and look for DataModel Create:day0service task. Select the task and click on the Details option to see the details of the service created. We will be able to see the details of what are sent

as input to day0service.



Create: day0config:day0config day0config_test

Task ID JmjFgUtAB-Re6ZmU6q1DOLpg

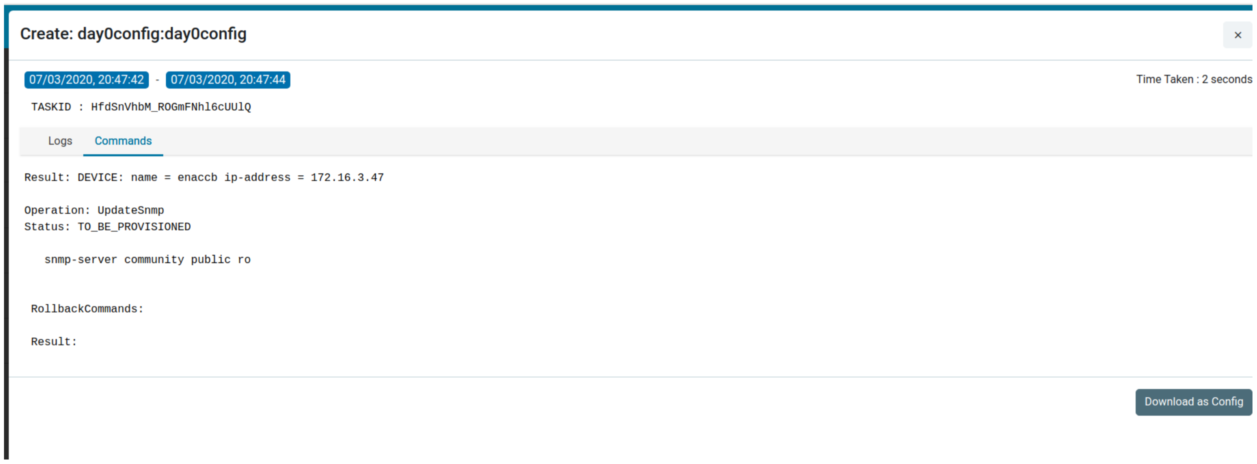
User Name admin

Time Taken 23/12/2020, 10:19:00 - 23/12/2020, 10:19:01 (1 seconds)

Summary Logs

```
Dec 23, 2020, 10:19:00 AM   CREATE Service day0config:day0config=day0config_test: day0config:day0config
Dec 23, 2020, 10:19:00 AM   get-data1:/controller:services/day0config:day0config=day0config_test (duration = 0 msec)
Dec 23, 2020, 10:19:00 AM   Posted on kafka: {"taskId": "JmjFgUtAB-Re6ZmU6q1DOLpg", "timestamp": 1608698940909, "commit": true, "autoRollback": true, "force": false, "skipNotify": false, "verbose": false,
"skipUniqueConstraintValidation": false, "stacktrace": "", "payload": "

<input><transaction-policy><fail-fast>true</fail-fast><validation-scope>COMMITTED_DATA</validation-scope><command-sequence-policy>DEPENDENCY_BASED</command-sequence-policy><do-not-send-commands-to-devices>true</do-

", "operation": "", "systemTask": false, "taskStatusManaged": false, "actionContextPath": "", "errorMsg": "", "logUpdateReferences": false, "targetApi": "DEFAULT", "type": "CREATE", "yangState": "COMMIT_REQUESTED",
"admin"}
Dec 23, 2020, 10:19:00 AM   atom-core-75fc4db997-8c9bf: Processing from kafka
Dec 23, 2020, 10:19:01 AM   {"taskId": "JmjFgUtAB-Re6ZmU6q1DOLpg", "taskState": "COMMIT_REQUESTED", "timestamp": 1608698941165, "description": "", "serviceTemplate": "day0config", "serviceInstance": "day0config_test",
"operationName": "", "commandGenerationDisabled": false, "autoRollback": true, "serviceOperationType": "CREATE", "operationType": "CREATE", "verbose": false, "processDeletion": false, "logUpdateReferences": false}
Dec 23, 2020, 10:19:01 AM   Resuming commit
Dec 23, 2020, 10:19:01 AM   Triggering data model commit
Dec 23, 2020, 10:19:01 AM   service(create = 3)
Dec 23, 2020, 10:19:01 AM   Create: /controller:services/day0config:day0config=day0config_test
+ day0config:day0config:
+    device: 3.42
```

5. Commands generated for day0service can also be viewed in the Task Details:



Create: day0config:day0config

07/03/2020, 20:47:42 - 07/03/2020, 20:47:44

Time Taken : 2 seconds

TASKID : HfdSnVhbM_ROGmFNhl6cUUlQ

Logs Commands

```
Result: DEVICE: name = enaccb ip-address = 172.16.3.47

Operation: UpdateSnmp
Status: TO_BE_PROVISIONED

   snmp-server community public ro


 RollbackCommands:

 Result:
```

Upgrading Service package

1. Service packages can be upgraded to newer versions without disrupting the ATOM.
2. Add the required modifications to the existing service package and tag changes with appropriate version number in the respective build.gradle file.

```
group 'com.anuta.ncx.packages'
version '7.0.1.0'
```

3. Create the zip of the required service package to be upgraded.

NOTE: Check that all the tasks running in ATOM are in 'COMPLETE' state before upgrading the package.

4. The package can be upgraded in the ATOM using below gradle task:

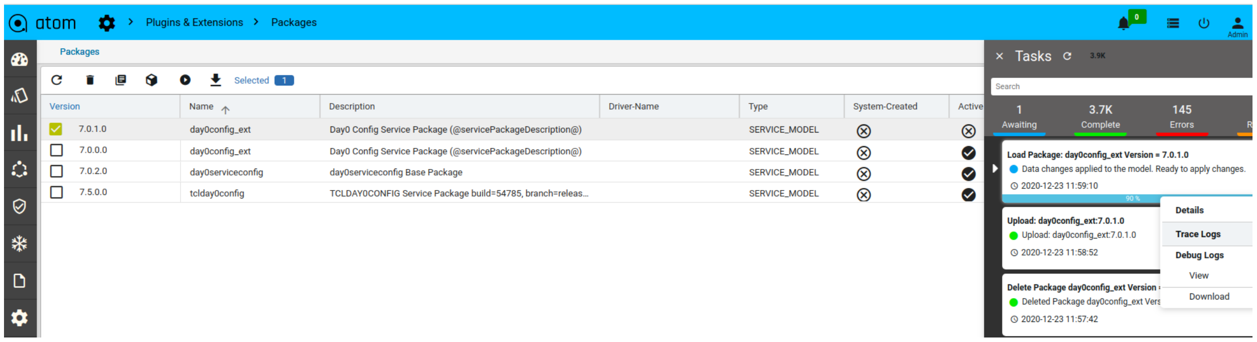
   gradle upgrade

   This **gradle upgrade** task will do following steps:

   1. Enable's maintenance mode on ATOM

2. Load the package need to be upgraded

3. Activates the Package and then disables maintenance mode.



atom > Plugins & Extensions > Packages

Packages

Selected 1

Version	Name	Description	Driver-Name	Type	System-Created	Active
✓ 7.0.1.0	day0config_ext	Day0 Config Service Package (@servicePackageDescription@)		SERVICE_MODEL	⊗	⊗
7.0.0.0	day0config_ext	Day0 Config Service Package (@servicePackageDescription@)		SERVICE_MODEL	⊗	✓
7.0.2.0	day0serviceconfig	day0serviceconfig Base Package		SERVICE_MODEL	⊗	✓
7.5.0.0	tclday0config	TCLDAY0CONFIG Service Package build=54785, branch=releas...		SERVICE_MODEL	⊗	✓

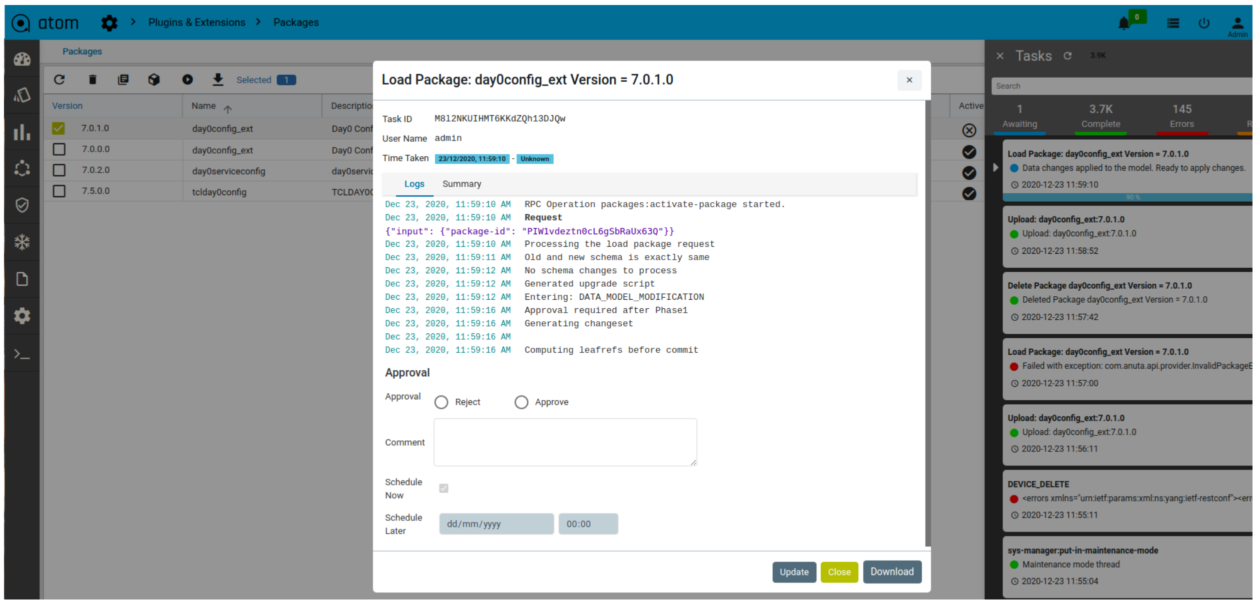
Tasks 3.9K

1 Awaiting | 3.7K Complete | 145 Errors

Load Package: day0config_ext Version = 7.0.1.0
Data changes applied to the model. Ready to apply changes.
2020-12-23 11:59:10
90 %

Upload: day0config_ext:7.0.1.0
Upload: day0config_ext:7.0.1.0
2020-12-23 11:58:52

Delete Package day0config_ext Version =
Deleted Package day0config_ext Vers
2020-12-23 11:57:42



Load Package: day0config_ext Version = 7.0.1.0

Task ID M812NKUIHMT6KKdZQh13DJQw
User Name admin
Time Taken 23/12/2020, 11:59:10 - Unknown

Logs | Summary

```
Dec 23, 2020, 11:59:10 AM   RPC Operation packages:activate-package started.
Dec 23, 2020, 11:59:10 AM   Request
{"input": {"package-id": "PIW1vdeztn0cL6gSbRaUx63Q"}}
Dec 23, 2020, 11:59:10 AM   Processing the load package request
Dec 23, 2020, 11:59:11 AM   Old and new schema is exactly same
Dec 23, 2020, 11:59:12 AM   No schema changes to process
Dec 23, 2020, 11:59:12 AM   Generated upgrade script
Dec 23, 2020, 11:59:12 AM   Entering: DATA_MODEL_MODIFICATION
Dec 23, 2020, 11:59:16 AM   Approval required after Phase1
Dec 23, 2020, 11:59:16 AM   Generating changeset
Dec 23, 2020, 11:59:16 AM
Dec 23, 2020, 11:59:16 AM   Computing leafrefs before commit
```

Approval

Approval: Reject / Approve

Comment

Schedule Now

Schedule Later: dd/mm/yyyy 00:00

Tasks 3.9K

1 Awaiting | 3.7K Complete | 145 Errors

Load Package: day0config_ext Version = 7.0.1.0
Data changes applied to the model. Ready to apply changes.
2020-12-23 11:59:10

Upload: day0config_ext:7.0.1.0
Upload: day0config_ext:7.0.1.0
2020-12-23 11:58:52

Delete Package day0config_ext Version = 7.0.1.0
Deleted Package day0config_ext Version = 7.0.1.0
2020-12-23 11:57:42

Load Package: day0config_ext Version = 7.0.1.0
Failed with exception: com.anuta.api.provider.InvalidPackageE
2020-12-23 11:57:00

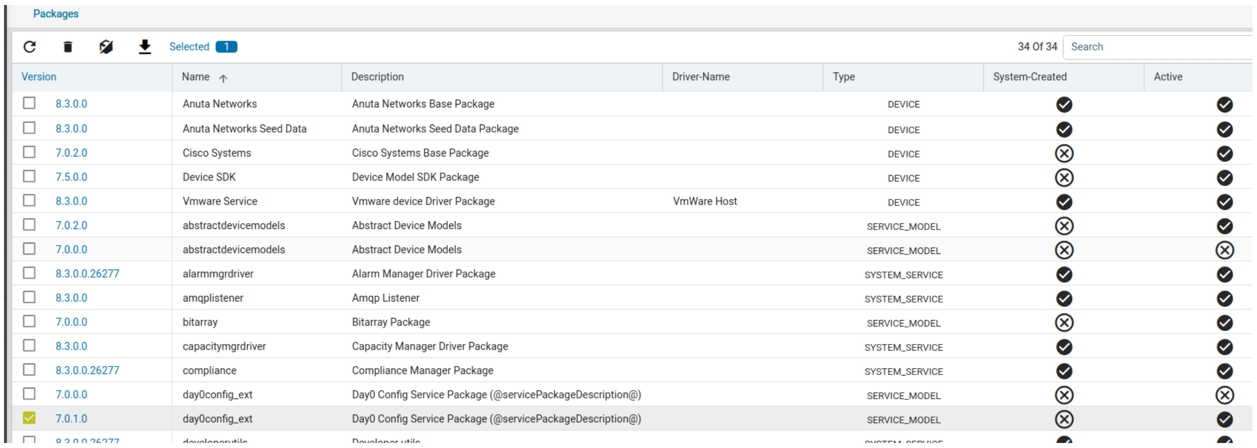
Upload: day0config_ext:7.0.1.0
Upload: day0config_ext:7.0.1.0
2020-12-23 11:56:11

DEVICE_DELETE
<errors xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf"><err
2020-12-23 11:55:11

sys-manager:put-in-maintenance-mode
Maintenance mode thread
2020-12-23 11:55:04

NOTE:
1. The command, gradle upgrade, will automatically put ATOM in maintenance mode, upload the modified package and disable the maintenance mode.
2. If required execute the command 'gradle clean' before upgrade.

5. Look for the 'Active' state status of the upgraded package.
6. Observe that the status of the Base package (version 7.0.0.0) is set to FALSE and the upgraded package status is set to TRUE(version 7.0.1.0).


Packages

Selected 1 34 Of 34 Search

	Version	Name	Description	Driver-Name	Type	System-Created	Active
☐	8.3.0.0	Anuta Networks	Anuta Networks Base Package		DEVICE	✓	✓
☐	8.3.0.0	Anuta Networks Seed Data	Anuta Networks Seed Data Package		DEVICE	✓	✓
☐	7.0.2.0	Cisco Systems	Cisco Systems Base Package		DEVICE	⊗	✓
☐	7.5.0.0	Device SDK	Device Model SDK Package		DEVICE	⊗	✓
☐	8.3.0.0	Vmware Service	Vmware device Driver Package	VmWare Host	DEVICE	✓	✓
☐	7.0.2.0	abstractdevicemodels	Abstract Device Models		SERVICE_MODEL	⊗	✓
☐	7.0.0.0	abstractdevicemodels	Abstract Device Models		SERVICE_MODEL	⊗	⊗
☐	8.3.0.0.26277	alarmmgrdriver	Alarm Manager Driver Package		SYSTEM_SERVICE	✓	✓
☐	8.3.0.0	amqplistener	Amqp Listener		SYSTEM_SERVICE	✓	✓
☐	7.0.0.0	bitarray	Bitarray Package		SERVICE_MODEL	⊗	✓
☐	8.3.0.0	capacitymgrdriver	Capacity Manager Driver Package		SYSTEM_SERVICE	✓	✓
☐	8.3.0.0.26277	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	✓
☐	7.0.0.0	day0config_ext	Day0 Config Service Package (@servicePackageDescription@)		SERVICE_MODEL	⊗	⊗
☑	7.0.1.0	day0config_ext	Day0 Config Service Package (@servicePackageDescription@)		SERVICE_MODEL	⊗	✓

Exercise: L2Edge Service Modeling

This section describes the procedure for creating a service package for L2Edge service. (in this example, l2edge.yang is modelled to configure an interface as L2 interface-types access/trunk with respective vlans)

1. Extract the provided atomsdk zip and create a service package environment by selecting option of type SERVICE_MODEL and package name as l2edge as per Creating a Package using SDK.
2. Create l2edge.yang under **src/main/model**
3. Put below content in l2edge.yang

```
module l2edge {
 yang-version 1.1;
 namespace "http://oneandone.net/l2edge";
 prefix l2edge;

 import controller { prefix ac;}
 import junos-conf-root {
   prefix jcr;
 }
 import junos-conf-interfaces {
   prefix jci;
 }
 import junos-conf-vlans {
   prefix jcv;
 }
 import ncx-extensions {
   prefix n-ext;
 }
```

```
  import sdk-extensions {
    prefix s-ext;
  }

  description
    "This module provides the l2-edge service";

  revision 2018-06-19 {
    description
      "Initial revision.";
  }

  augment "/ac:services" {
    list l2-edge {
      n-ext:ncx-service;
      n-ext:ncx-auto-update "ENTITY_LEVEL";
      description "Provides configuration for l2-edge ports.The l2-edge service provides network
connectivity for end hosts";

      key name;
      leaf name {
                    description "Name of the l2-edge service";
        type string;
      }
      leaf description {
          description "Description of the user of the service (port description)";
          type string;
      }

      container devices {
        description "Devices that should form an l2-edge service.";
        list device {
          description "Device the port should be configured on.";
          key name;
          unique "device-id interface-name";
          leaf name {
            type string;
          }
          leaf device-id {
```

```
          type leafref {
            path "/ac:devices/ac:device/ac:id";
          }
        }
        leaf unit {
          type uint32;
          description
            "unit value";
          n-ext:maps-to "/jcr:configuration/jci:interfaces/jci:interface/jci:unit/jci:name"{
            s-ext:device-platform "JUNOS";
          }
        }
        leaf variant {
          type enumeration {
            enum "trunk";
            enum "access";
          }
          default "trunk";
                n-ext:non-updatable;
          n-ext:maps-to
"/jcr:configuration/jci:interfaces/interface[jci:name=current()/../interface-name]/unit[jci:name=current()
/../unit]/family/ethernet-switching/port-mode"{
            s-ext:device-platform "JUNOS";
          }
          n-ext:maps-to
"/jcr:configuration/jci:interfaces/interface[jci:name=current()/../port-channel-id]/unit[jci:name=current()
/../unit]/family/ethernet-switching/port-mode"{
            s-ext:device-platform "JUNOS";
          }
        }
        leaf interface-name {
          description "Interface for the l2-edge service";
          type string {
              n-ext:atom-leafref-path
"/ac:devices/ac:device[ac:id=current()/../device-id]/jcr:configuration/jci:interfaces/jci:interface/jci:name"
;
          }
          mandatory true;
          n-ext:maps-to "/jcr:configuration/jci:interfaces/interface/name"{
            s-ext:device-platform "JUNOS";
```

```
      }
      n-ext:ncx-add-reference-false;
     }
     must
'(count(/ac:services/l2edge:l2-edge/devices/device[device-id=current()/device-id]/interface-name[interfa
ce-name=current()/interface-name]) < 2)' {
             error-message "Interface Name already used";
     }
     leaf native-vlan {
      description "Native VLAN that is configured when trunk ports are used";
      type int16 {
       range 1..4094;
      }
      mandatory true;
      when "../variant = 'trunk'";
      n-ext:maps-to "/jcr:configuration/jcv:vlans/vlan/vlan-id"{
         s-ext:device-platform "JUNOS";
      }
      n-ext:ncx-maps-to-expr "/jcr:configuration/jcv:vlans/vlan/name = Vlancurrent()"{
         s-ext:device-platform "JUNOS";
      }
      n-ext:maps-to
"/jcr:configuration/jci:interfaces/interface/unit/family/ethernet-switching/native-vlan-id"{
         s-ext:device-platform "JUNOS";
      }
     }
     container vlans {
      description "List of vlans permitted on the l2-edge ports";
       leaf-list vlan-id {
        type int16 {
         range 2..4094;
        }
        min-elements 1;
        n-ext:maps-to "/jcr:configuration/jcv:vlans/vlan/vlan-id"{
           s-ext:device-platform "JUNOS";
        }
        n-ext:ncx-maps-to-expr "/jcr:configuration/jcv:vlans/vlan/name = Vlancurrent()"{
           s-ext:device-platform "JUNOS";
        }
```

```
            }
            must
'(count(/ac:services/l2edge:l2-edge[name=current()/../../../name]/devices/device[name=current()/../na
me]/vlans/vlan-id) = 1 and ../variant = "access") or
(count(/ac:services/l2edge:l2-edge[name=current()/../../../name]/devices/device[name=current()/../nam
e]/vlans/vlan-id) >= 1 and ../variant = "trunk")' {
              error-message "vlan should be one if variant is access or vlan should be >= one if variant is
trunk";
            }
          }
        }
      }
    }
  }
}
```

4. Using ATOM sdk plugin task generate the service package code for l2edge.yang as below.

   Change to current directory i.e. the service package directory & run the commands

   gradle tasks --all (this will display all the tasks)

   gradle generateServicePackage(Runs the ServicePackage generate task)

5. Now the scripts folder has the auto-generated code. This code will have device mapping code as well since the yang has mappings between service leafs and device model leafs using maps-to statement.
6. Now we need to generate pybinds specific to device mappings:

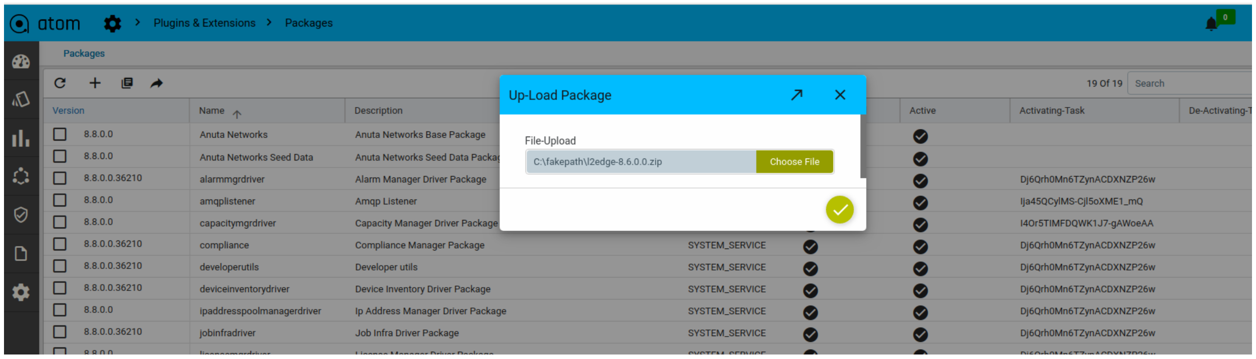
   gradle gDPB

7. Change to the current directory, i.e., servicepackage and run:

   gradle archive

8. Uploading l2edge service package zip to ATOM

   Navigate to **Administration** > **Plugins & Extensions** > **Packages > Add**


Plugins & Extensions > Packages

Packages

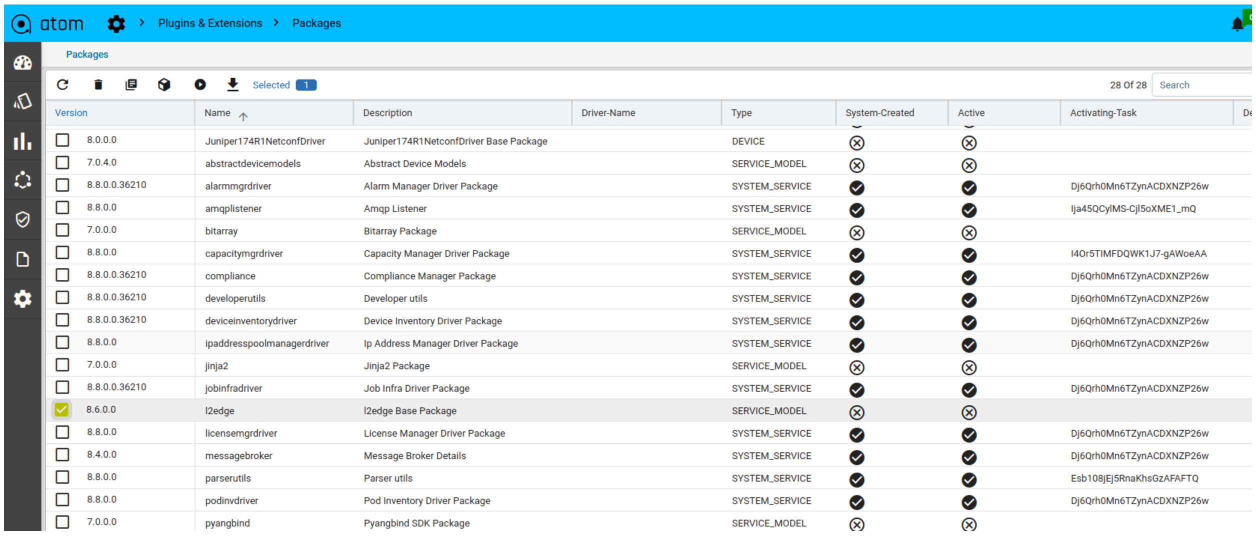
Up-Load Package

File-Upload: C:\fakepath\l2edge-8.6.0.0.zip

19 Of 19

Version	Name	Description			Active	Activating-Task	De-Activating-T
8.8.0.0	Anuta Networks	Anuta Networks Base Package			✓		
8.8.0.0	Anuta Networks Seed Data	Anuta Networks Seed Data Packag			✓		
8.8.0.0.36210	alarmmgrdriver	Alarm Manager Driver Package			✓	Dj6Qrh0Mn6TZynACDXNZP26w	
8.8.0.0	amqplistener	Amqp Listener			✓	Ija45QCylMS-Cjl5oXME1_mQ	
8.8.0.0	capacitymgrdriver	Capacity Manager Driver Package			✓	I4Or5TIMFDQWK1J7-gAWoeAA	
8.8.0.0.36210	compliance	Compliance Manager Package	SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w	
8.8.0.0.36210	developerutils	Developer utils	SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w	
8.8.0.0.36210	deviceinventorydriver	Device Inventory Driver Package	SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w	
8.8.0.0	ipaddresspoolmanagerdriver	Ip Address Manager Driver Package	SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w	
8.8.0.0.36210	jobinfradriver	Job Infra Driver Package	SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w	

On successful upload of the package, the following is displayed on the screen:

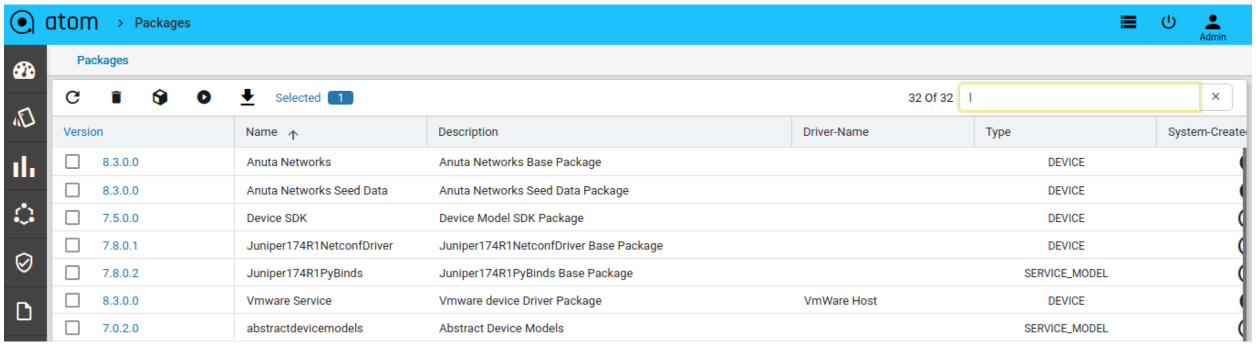

Plugins & Extensions > Packages

Packages

Selected 1 — 28 Of 28

Version	Name	Description	Driver-Name	Type	System-Created	Active	Activating-Task
8.0.0.0	Juniper174R1NetconfDriver	Juniper174R1NetconfDriver Base Package		DEVICE	⊗	⊗	
7.0.4.0	abstractdevicemodels	Abstract Device Models		SERVICE_MODEL	⊗	⊗	
8.8.0.0.36210	alarmmgrdriver	Alarm Manager Driver Package		SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w
8.8.0.0	amqplistener	Amqp Listener		SYSTEM_SERVICE	✓	✓	Ija45QCylMS-Cjl5oXME1_mQ
7.0.0.0	bitarray	Bitarray Package		SERVICE_MODEL	⊗	⊗	
8.8.0.0	capacitymgrdriver	Capacity Manager Driver Package		SYSTEM_SERVICE	✓	✓	I4Or5TIMFDQWK1J7-gAWoeAA
8.8.0.0.36210	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w
8.8.0.0.36210	developerutils	Developer utils		SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w
8.8.0.0.36210	deviceinventorydriver	Device Inventory Driver Package		SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w
8.8.0.0	ipaddresspoolmanagerdriver	Ip Address Manager Driver Package		SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w
7.0.0.0	jinja2	Jinja2 Package		SERVICE_MODEL	⊗	⊗	
8.8.0.0.36210	jobinfradriver	Job Infra Driver Package		SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w
8.6.0.0	l2edge	l2edge Base Package		SERVICE_MODEL	⊗	⊗	
8.8.0.0	licensemgrdriver	License Manager Driver Package		SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w
8.4.0.0	messagebroker	Message Broker Details		SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w
8.8.0.0	parserutils	Parser utils		SYSTEM_SERVICE	✓	✓	Esb108jEj5RnaKhsGzAFAFTQ
8.8.0.0	podinvdriver	Pod Inventory Driver Package		SYSTEM_SERVICE	✓	✓	Dj6Qrh0Mn6TZynACDXNZP26w
7.0.0.0	pyangbind	Pyangbind SDK Package		SERVICE_MODEL	⊗	⊗	

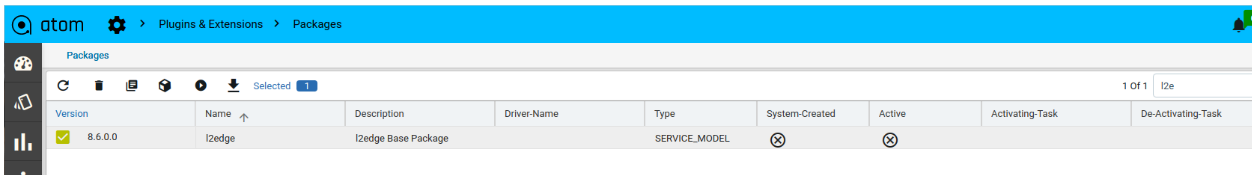
9. To Activate the l2edge package by selecting from list of packages using search box as below


Packages

Selected 1 — 32 Of 32

Version	Name	Description	Driver-Name	Type
8.3.0.0	Anuta Networks	Anuta Networks Base Package		DEVICE
8.3.0.0	Anuta Networks Seed Data	Anuta Networks Seed Data Package		DEVICE
7.5.0.0	Device SDK	Device Model SDK Package		DEVICE
7.8.0.1	Juniper174R1NetconfDriver	Juniper174R1NetconfDriver Base Package		DEVICE
7.8.0.2	Juniper174R1PyBinds	Juniper174R1PyBinds Base Package		SERVICE_MODEL
8.3.0.0	Vmware Service	Vmware device Driver Package	VmWare Host	DEVICE
7.0.2.0	abstractdevicemodels	Abstract Device Models		SERVICE_MODEL

After entering ‘l2edge’ key in search box hit the enter button to get the result


Plugins & Extensions > Packages

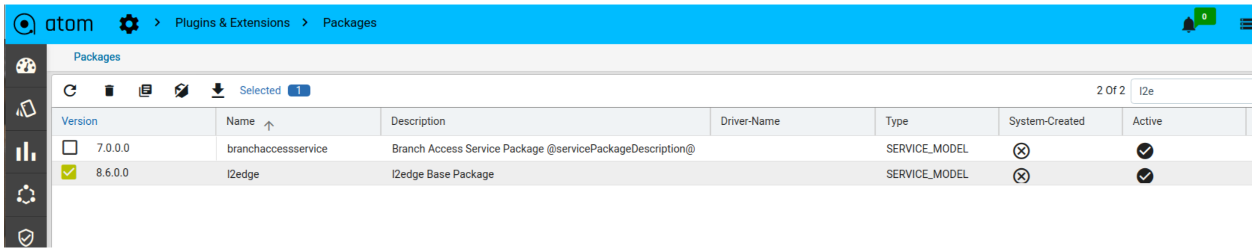
Packages

Selected 1 — 1 Of 1 — l2e

Version	Name	Description	Driver-Name	Type	System-Created	Active	Activating-Task	De-Activating-Task
8.6.0.0	l2edge	l2edge Base Package		SERVICE_MODEL	⊗	⊗		

Select the package and hit Activate button

After Active Package completion, notice Active state changed to true.



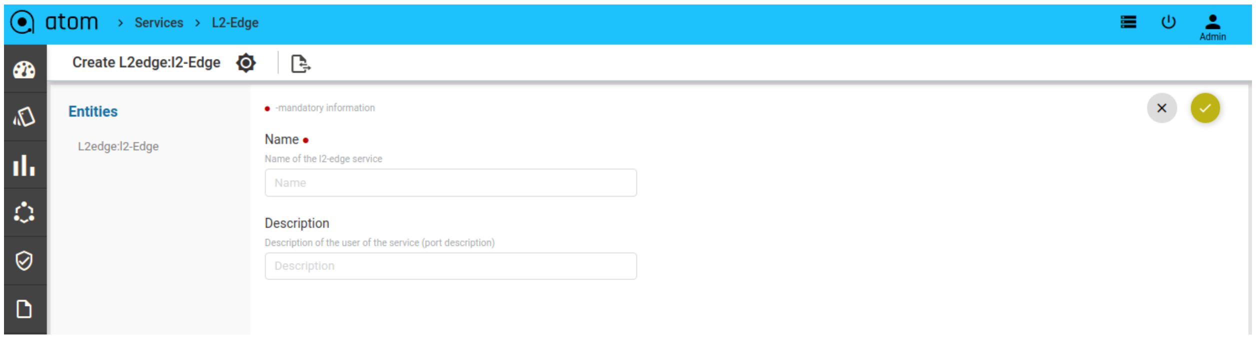
Version	Name	Description	Driver-Name	Type	System-Created	Active
☐ 7.0.0.0	branchaccessservice	Branch Access Service Package @servicePackageDescription@		SERVICE_MODEL	⊗	✓
✓ 8.6.0.0	l2edge	l2edge Base Package		SERVICE_MODEL	⊗	✓

10. Creating the l2edge through ATOM

Navigate to **Automation > Services**

In the Catalog pane, click the l2edge:l2-edge > click + to add the l2edge.

In the Create **l2edge**, fill the values in the properties displayed in the form:



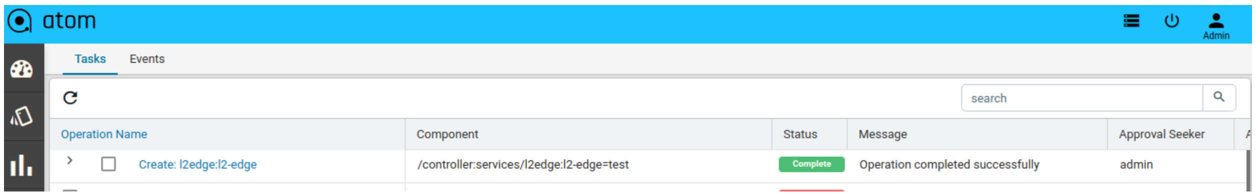
● -mandatory information

Name ●
Name of the l2-edge service
Name

Description
Description of the user of the service (port description)
Description

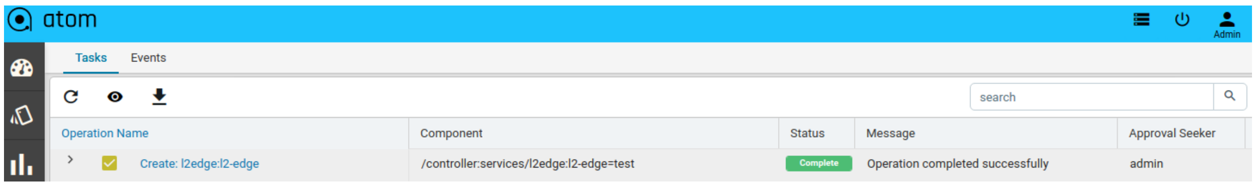
Click ✅ to instantiate the service.

To track the service, Go to **Administration -> Tasks and Events** and find for the tasks created:

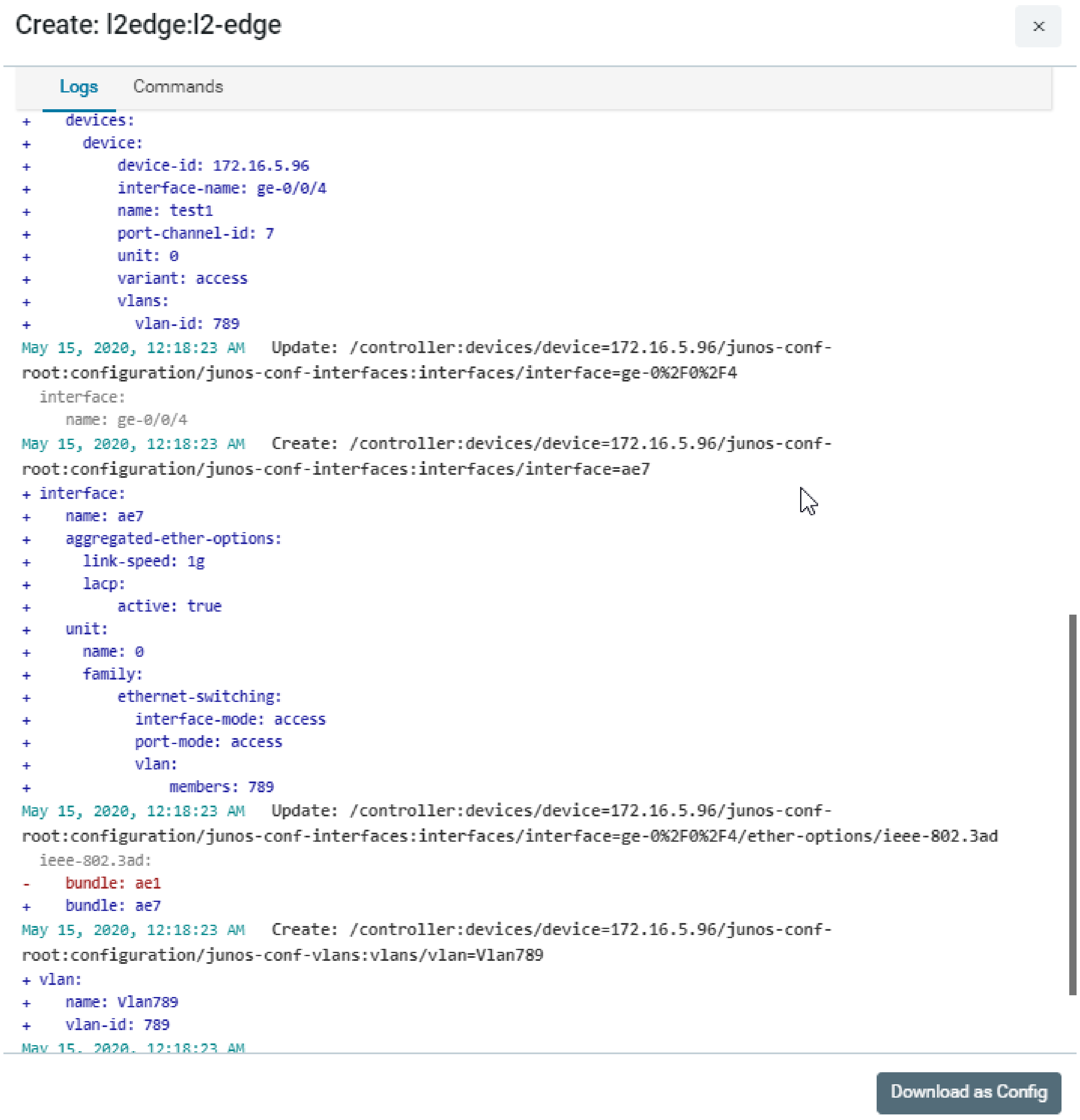


Operation Name	Component	Status	Message	Approval Seeker
Create: l2edge:l2-edge	/controller:services/l2edge:l2-edge=test	Complete	Operation completed successfully	admin

To view the commands generated for this service, click the **view** button



Operation Name	Component	Status	Message	Approval Seeker
✓ Create: l2edge:l2-edge	/controller:services/l2edge:l2-edge=test	Complete	Operation completed successfully	admin



```
+   devices:
+     device:
+         device-id: 172.16.5.96
+         interface-name: ge-0/0/4
+         name: test1
+         port-channel-id: 7
+         unit: 0
+         variant: access
+         vlans:
+           vlan-id: 789
May 15, 2020, 12:18:23 AM   Update: /controller:devices/device=172.16.5.96/junos-conf-
root:configuration/junos-conf-interfaces:interfaces/interface=ge-0%2F0%2F4
  interface:
    name: ge-0/0/4
May 15, 2020, 12:18:23 AM   Create: /controller:devices/device=172.16.5.96/junos-conf-
root:configuration/junos-conf-interfaces:interfaces/interface=ae7
+ interface:
+   name: ae7
+   aggregated-ether-options:
+     link-speed: 1g
+     lacp:
+         active: true
+   unit:
+     name: 0
+     family:
+         ethernet-switching:
+           interface-mode: access
+           port-mode: access
+           vlan:
+               members: 789
May 15, 2020, 12:18:23 AM   Update: /controller:devices/device=172.16.5.96/junos-conf-
root:configuration/junos-conf-interfaces:interfaces/interface=ge-0%2F0%2F4/ether-options/ieee-802.3ad
  ieee-802.3ad:
-   bundle: ae1
+   bundle: ae7
May 15, 2020, 12:18:23 AM   Create: /controller:devices/device=172.16.5.96/junos-conf-
root:configuration/junos-conf-vlans:vlans/vlan=Vlan789
+ vlan:
+   name: Vlan789
+   vlan-id: 789
May 15, 2020, 12:18:23 AM
```

Exercise: L3 Service Service Modeling

This section describes the procedure for creating service package for a L3 service (in this example, l3service.yang is modelled to configure an interface either as l3/sub/vlan and make it part of a vrf and assign IP)

1. Extract the provided atomsdk zip and create a service package environment by selecting the option of type SERVICE_MODEL and package name as l3service as per Creating a Package using SDK.

2. Create l3service.yang under **src/main/model**
3. Put below content in l3service.yang

```
module l3service {
 namespace "http://anutanetworks.com/l3service";
 prefix l3service;

 import ietf-inet-types {
  prefix inet;
 }
 import ncx-extensions {
  prefix n-ext;
 }
 import sdk-extensions {
  prefix s-ext;
 }
 import controller {
  prefix ac;
 }
 import interface {
  prefix ai;
 }
 import l2features {
  prefix l2;
 }
 import l3features {
  prefix l3;
 }
 import ncx-types {
  prefix nt;
 }

 organization
  "Anuta Networks";

 revision 2014-07-01 {
  description
   "Initial revision";
```

```
}

typedef interface-mode-type {
  type enumeration {
    enum "sub-interface";
    enum "l3-interface";
    enum "vlan";
  }
}

grouping l3service {
  leaf service-status {
    type string;
    description
      "string";
    config false;
    default "AVAILABLE";
  }
  leaf name {
    type string;
    description
      "string";
    mandatory true;
  }
  leaf device-id {
    type leafref {
      path "/ac:devices/ac:device/ac:id";
    }
    description
      "device-id";
    mandatory true;
  }
  leaf interface-mode {
    type interface-mode-type;
    description
      "sub-interface
       l3-interface
       vlan
      ";
     mandatory true;
     n-ext:maps-to
"/ac:devices/ac:device[ac:id=current()/../device-id]/ai:interfaces/interface[long-name=current()/../interfa
ce]/mode";
   }
   leaf interface {
     type leafref {
       path "/ac:devices/ac:device[ac:id=current()/../device-id]/ai:interfaces/ai:interface/ai:long-name";
     }
     description
       "interface";
     when "../interface-mode = 'sub-interface' or ../interface-mode = 'l3-interface' ";
     n-ext:ncx-maps-to-expr "/ac:devices/device/ai:interfaces/interface/long-name = current()";
     n-ext:ncx-maps-to-expr "/ac:devices/device/ai:interfaces/interface/name = current()";
     s-ext:ncx-maps-to-expr-when "../interface-mode = 'l3-interface'";
     n-ext:ncx-add-reference-when "../interface-mode = 'sub-interface' or ../interface-mode = 'vlan'";
     n-ext:non-updatable;
   }
   leaf description {
     type string;
     description
       "string";
     n-ext:maps-to "/ac:devices/ac:device[ac:id=current()/../device-id]/l2:vlans/l2:vlan/l2:name";
     n-ext:maps-to
"/ac:devices/ac:device[ac:id=current()/../device-id]/ai:interfaces/interface[long-name=current()/../interfa
ce]/description";
   }
   leaf vrf {
     type string;
     description
       "string";
     n-ext:maps-to "/ac:devices/ac:device[ac:id=current()/../device-id]/l3:vrfs/l3:vrf/l3:name";
     n-ext:maps-to
"/ac:devices/ac:device[ac:id=current()/../device-id]/ai:interfaces/interface[long-name=current()/../interfa
ce]/vrf";
   }
   leaf vlan-id {
     type uint32 {
       range "1..4096";
      }
      description
        "1..4096";
      mandatory true;
      n-ext:maps-to "/ac:devices/ac:device[ac:id=current()/../device-id]/l2:vlans/l2:vlan/l2:id";
      n-ext:maps-to
"/ac:devices/ac:device[ac:id=current()/../device-id]/ai:interfaces/interface[long-name=current()/../interfa
ce]/vlan";
      when "../interface-mode = 'sub-interface' or ../interface-mode = 'vlan' ";
      n-ext:ncx-maps-to-expr "/ac:devices/device/ai:interfaces/interface/long-name = Vlan+current()";
      n-ext:ncx-maps-to-expr "/ac:devices/device/ai:interfaces/interface/name = Vlan+current()";
      s-ext:ncx-maps-to-expr-when "../interface-mode = 'vlan'";
    }
    leaf ip-address {
      type inet:ipv4-address;
      description
        "Valid IPv4 Address (A.B.C.D for e.x: 172.16.1.1)";
      n-ext:maps-to
"/ac:devices/ac:device[ac:id=current()/../device-id]/ai:interfaces/interface[long-name=current()/../interfa
ce]/ip-address";
    }
    leaf netmask {
      type inet:ipv4-address;
      description
        "Valid IPv4 Address (A.B.C.D for e.x: 172.16.1.1)";
      n-ext:maps-to
"/ac:devices/ac:device[ac:id=current()/../device-id]/ai:interfaces/interface[long-name=current()/../interfa
ce]/netmask";
    }
    leaf ipv6-address {
      type inet:ipv6-address;
      description
        "Valid IPv6 Address (X::Y for e.x: 2001::1)";
      n-ext:maps-to
"/ac:devices/ac:device[ac:id=current()/../device-id]/ai:interfaces/interface[long-name=current()/../interfa
ce]/ipv6-address";
    }
    leaf ipv6-prefix-length {
      type nt:ipv6-prefix-length;
      description
        "IPv6 netmask in CIDR notation.";
```

```
     n-ext:maps-to
"/ac:devices/ac:device[ac:id=current()/../device-id]/ai:interfaces/interface[long-name=current()/../interfa
ce]/ipv6-prefix-length";
   }
   leaf vrf-definition-mode {
     type boolean;
     description
       "vrf-definition-mode: True/False";
     default "true";
     n-ext:maps-to
"/ac:devices/ac:device[ac:id=current()/../device-id]/ai:interfaces/interface[long-name=current()/../interfa
ce]/vrf-definition-mode";
     n-ext:maps-to
"/ac:devices/ac:device[ac:id=current()/../device-id]/l3:vrfs/l3:vrf[name=current()/../vrf]/l3:vrf-definition-
mode";
     config false;
   }
 }


 augment "/ac:services" {
   container l3-services {
     list l3-service {
       n-ext:ncx-service;
       key "name";
       n-ext:ncx-maps-to-expr "/ac:devices/device/ai:interfaces/interface/long-name =
$(interface).$(vlan-id)";
       n-ext:ncx-maps-to-expr "/ac:devices/device/ai:interfaces/interface/name = $(interface).$(vlan-id)";
       s-ext:ncx-maps-to-expr-when "../interface-mode = 'sub-interface'";
       uses l3service:l3service;
     }
   }
 }
}
```

4. Using ATOM sdk plugin task generate the service package code for l3service.yang as below

   Change to current directory i.e. the service package directory & run the commands

   gradle tasks --all (this will display all the tasks)

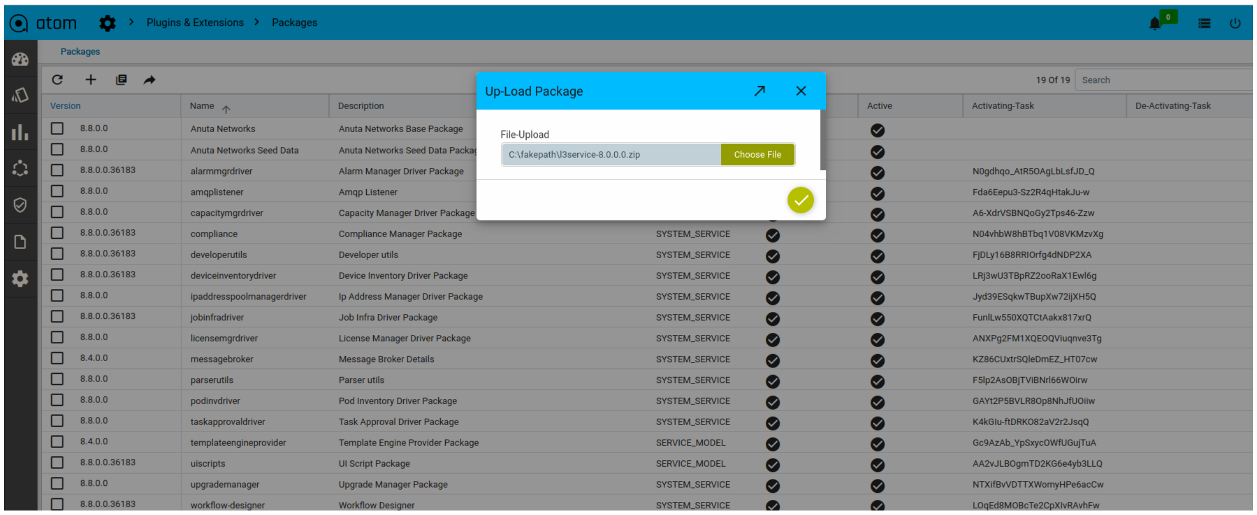
   gradle generateServicePackage(Runs the ServicePackage generate task)

5. Now the scripts folder has the auto-generated code. This code will have device mapping code as well since the yang has mappings between service leafs and device model leafs using maps-to statement.
6. Change to the current directory, i.e., servicepackage and run:

    gradle archive

7. Uploading l3service service package zip to ATOM

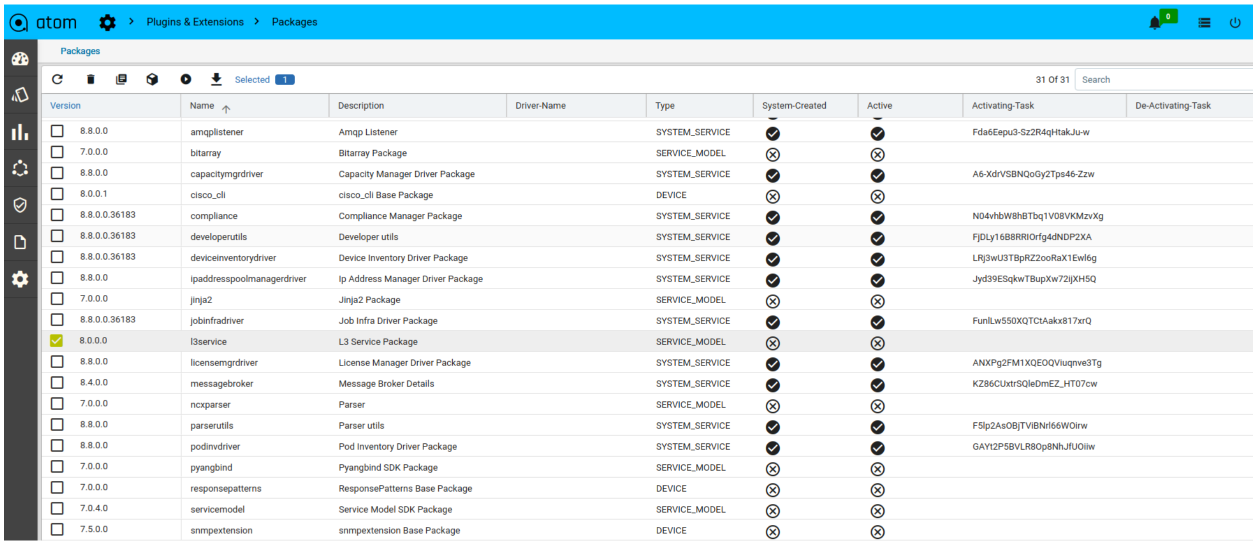
   Navigate to **Administration** > **Plugins & Extensions** > **Packages > Add**



C:\fakepath\l3service-8.0.0.0.zip

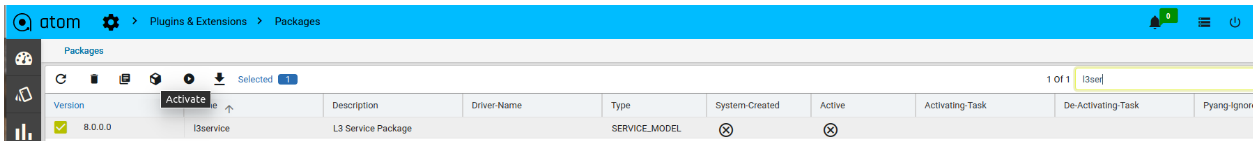
Version	Name	Description			Active	Activating-Task	De-Activating-Task
8.8.0.0	Anuta Networks	Anuta Networks Base Package			✓		
8.8.0.0	Anuta Networks Seed Data	Anuta Networks Seed Data Packag			✓		
8.8.0.0.36183	alarmmgrdriver	Alarm Manager Driver Package			✓	N0gdhqo_AtR5OAgLbLsfJD_Q	
8.8.0.0	amqplistener	Amqp Listener			✓	Fda6Eepu3-Sz2R4qHtakJu-w	
8.8.0.0	capacitymgrdriver	Capacity Manager Driver Package			✓	A6-XdrVSBNQoGy2Tps46-Zzw	
8.8.0.0.36183	compliance	Compliance Manager Package	SYSTEM_SERVICE	✓	✓	N04vhbW8hBTbq1V08VKMzvXg	
8.8.0.0.36183	developerutils	Developer utils	SYSTEM_SERVICE	✓	✓	FjDLy16B8RRIOrfg4dNDP2XA	
8.8.0.0.36183	deviceinventorydriver	Device Inventory Driver Package	SYSTEM_SERVICE	✓	✓	LRj3wU3TBpRZ2ooRaX1Ewl6g	
8.8.0.0	ipaddresspoolmanagerdriver	Ip Address Manager Driver Package	SYSTEM_SERVICE	✓	✓	Jyd39ESqkwTBupXw72ijXH5Q	
8.8.0.0.36183	jobinfradriver	Job Infra Driver Package	SYSTEM_SERVICE	✓	✓	FunlLw550XQTCtAakx817xrQ	
8.8.0.0	licensemgrdriver	License Manager Driver Package	SYSTEM_SERVICE	✓	✓	ANXPg2FM1XQEOQViuqnve3Tg	
8.4.0.0	messagebroker	Message Broker Details	SYSTEM_SERVICE	✓	✓	KZ86CUxtrSQleDmEZ_HT07cw	
8.8.0.0	parserutils	Parser utils	SYSTEM_SERVICE	✓	✓	F5lp2AsOBjTViBNrl66WOirw	
8.8.0.0	podinvdriver	Pod Inventory Driver Package	SYSTEM_SERVICE	✓	✓	GAYt2P5BVLR8Op8NhJfUOiiw	
8.8.0.0	taskapprovaldriver	Task Approval Driver Package	SYSTEM_SERVICE	✓	✓	K4kGlu-ftDRKO82aV2r2JsqQ	
8.4.0.0	templateengineprovider	Template Engine Provider Package	SERVICE_MODEL	✓	✓	Gc9AzAb_YpSxycOWfUGujTuA	
8.8.0.0.36183	uiscripts	UI Script Package	SERVICE_MODEL	✓	✓	AA2vJLBOgmTD2KG6e4yb3LLQ	
8.8.0.0	upgrademanager	Upgrade Manager Package	SYSTEM_SERVICE	✓	✓	NTXifBvVDTTXWomyHPe6acCw	
8.8.0.0.36183	workflow-designer	Workflow Designer	SYSTEM_SERVICE	✓	✓	LOqEd8MOBcTe2CpXIvRAvhFw	

On successful upload of the package, the following is displayed on the screen:



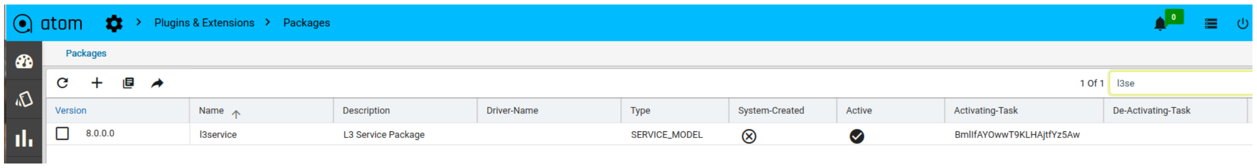
	Version	Name	Description	Driver-Name	Type	System-Created	Active	Activating-Task	De-Activating-Task
☐	8.8.0.0	amqplistener	Amqp Listener		SYSTEM_SERVICE	✓	✓	Fda6Eepu3-Sz2R4qHtakJu-w	
☐	7.0.0.0	bitarray	Bitarray Package		SERVICE_MODEL	⊗	⊗		
☐	8.8.0.0	capacitymgrdriver	Capacity Manager Driver Package		SYSTEM_SERVICE	✓	✓	A6-XdrVSBNQoGy2Tps46-Zzw	
☐	8.0.0.1	cisco_cli	cisco_cli Base Package		DEVICE	⊗	⊗		
☐	8.8.0.0.36183	compliance	Compliance Manager Package		SYSTEM_SERVICE	✓	✓	N04vhbW8hBTbq1V08VKMzvXg	
☐	8.8.0.0.36183	developerutils	Developer utils		SYSTEM_SERVICE	✓	✓	FjDLy16B8RRIOrfg4dNDP2XA	
☐	8.8.0.0.36183	deviceinventorydriver	Device Inventory Driver Package		SYSTEM_SERVICE	✓	✓	LRj3wU3TBpRZ2ooRaX1Ewl6g	
☐	8.8.0.0	ipaddresspoolmanagerdriver	Ip Address Manager Driver Package		SYSTEM_SERVICE	✓	✓	Jyd39ESqkwTBupXw72ijXH5Q	
☐	7.0.0.0	jinja2	Jinja2 Package		SERVICE_MODEL	⊗	⊗		
☐	8.8.0.0.36183	jobinfradriver	Job Infra Driver Package		SYSTEM_SERVICE	✓	✓	FunlLw550XQTCtAakx817xrQ	
☑	8.0.0.0	l3service	L3 Service Package		SERVICE_MODEL	⊗	⊗		
☐	8.8.0.0	licensemgrdriver	License Manager Driver Package		SYSTEM_SERVICE	✓	✓	ANXPg2FM1XQEOQViuqnve3Tg	
☐	8.4.0.0	messagebroker	Message Broker Details		SYSTEM_SERVICE	✓	✓	KZ86CUxtrSQleDmEZ_HT07cw	
☐	7.0.0.0	ncxparser	Parser		SERVICE_MODEL	⊗	⊗		
☐	8.8.0.0	parserutils	Parser utils		SYSTEM_SERVICE	✓	✓	F5lp2AsOBjTViBNrl66WOirw	
☐	8.8.0.0	podinvdriver	Pod Inventory Driver Package		SYSTEM_SERVICE	✓	✓	GAYt2P5BVLR8Op8NhJfUOiiw	
☐	7.0.0.0	pyangbind	Pyangbind SDK Package		SERVICE_MODEL	⊗	⊗		
☐	7.0.0.0	responsepatterns	ResponsePatterns Base Package		DEVICE	⊗	⊗		
☐	7.0.4.0	servicemodel	Service Model SDK Package		SERVICE_MODEL	⊗	⊗		
☐	7.5.0.0	snmpextension	snmpextension Base Package		DEVICE	⊗	⊗		

8. To Activate the l3service package by selecting from the list of packages using search box as below.After entering ‘l3service’ or ‘l3’ key in search box hit the enter button to give the result and Select the package and hit the Activate button.



Version	Name	Description	Driver-Name	Type	System-Created	Active	Activating-Task	De-Activating-Task	Pyang-Ignor
8.0.0.0	l3service	L3 Service Package		SERVICE_MODEL	⊗	⊗			

After Active Package completion, notice Active state changed to true.



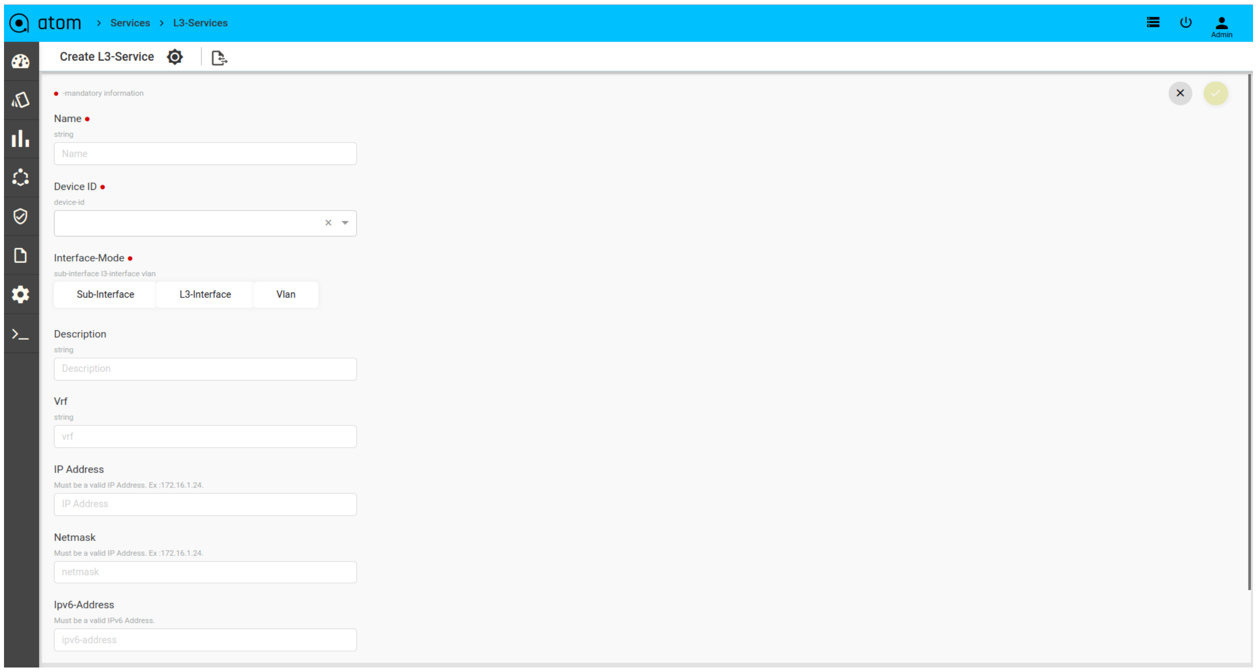
Version	Name	Description	Driver-Name	Type	System-Created	Active	Activating-Task	De-Activating-Task
8.0.0.0	l3service	L3 Service Package		SERVICE_MODEL	⊗	✓	BmlIfAYOwwT9KLHAjtfYz5Aw	

9. Creating the l3service through ATOM

   Navigate to **Automation > Services**

   In the Details pane, click the L3service icon > click Add L3service.

   In the Create **l3service** pane, fill the values in the properties displayed in the form:



● -mandatory information

Name ●
string
Name

Device ID ●
device-id

Interface-Mode ●
sub-interface l3-interface vlan
Sub-Interface | L3-Interface | Vlan

Description
string
Description

Vrf
string
vrf

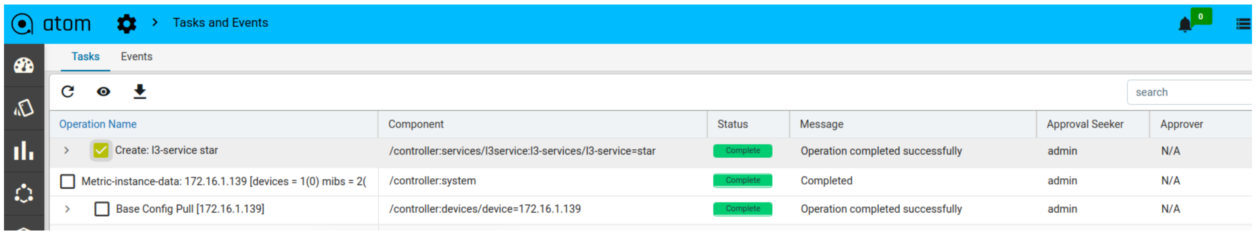
IP Address
Must be a valid IP Address. Ex :172.16.1.24.
IP Address

Netmask
Must be a valid IP Address. Ex :172.16.1.24.
netmask

Ipv6-Address
Must be a valid IPv6 Address.
ipv6-address

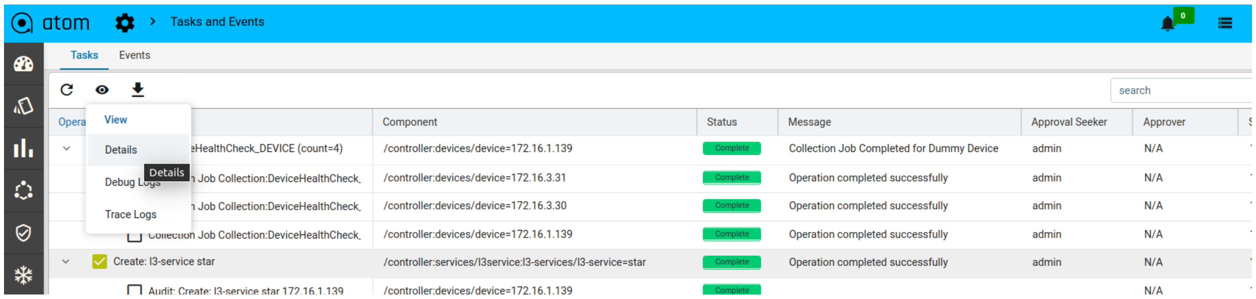
Click **OK** to instantiate the service.

To track the service, Go to **Administration -> Tasks and Events** and find for the tasks created:

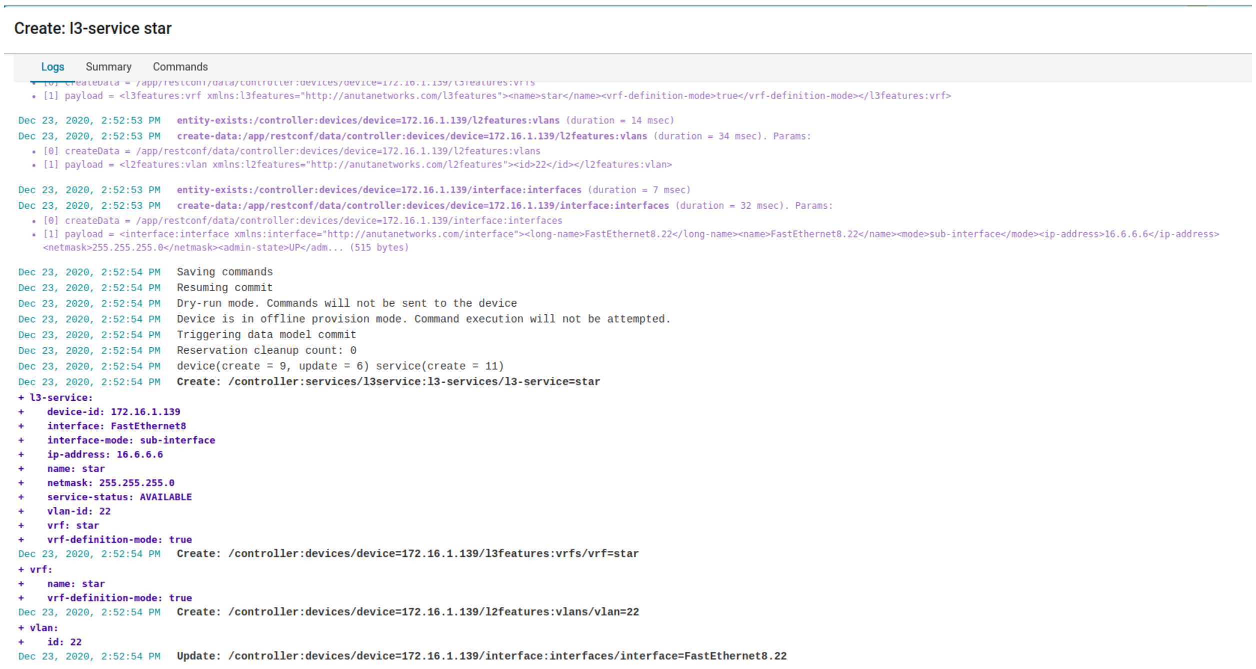


Operation Name	Component	Status	Message	Approval Seeker	Approver
✓ Create: l3-service star	/controller:services/l3service:l3-services/l3-service=star	Complete	Operation completed successfully	admin	N/A
Metric-instance-data: 172.16.1.139 [devices = 1(0) mibs = 2(	/controller:system	Complete	Completed	admin	N/A
Base Config Pull [172.16.1.139]	/controller:devices/device=172.16.1.139	Complete	Operation completed successfully	admin	N/A

To view the commands generated for this service, click the **view** button



Opera	Component	Status	Message	Approval Seeker	Approver
eHealthCheck_DEVICE (count=4)	/controller:devices/device=172.16.1.139	Complete	Collection Job Completed for Dummy Device	admin	N/A
n Job Collection:DeviceHealthCheck_	/controller:devices/device=172.16.3.31	Complete	Operation completed successfully	admin	N/A
n Job Collection:DeviceHealthCheck_	/controller:devices/device=172.16.3.30	Complete	Operation completed successfully	admin	N/A
Collection Job Collection:DeviceHealthCheck_	/controller:devices/device=172.16.1.139	Complete	Operation completed successfully	admin	N/A
✓ Create: l3-service star	/controller:services/l3service:l3-services/l3-service=star	Complete	Operation completed successfully	admin	N/A
Audit: Create: l3-service star 172.16.1.139	/controller:devices/device=172.16.1.139	Complete			N/A



Create: l3-service star

```
  • [0] createData = /app/restconf/data/controller:devices/device=172.16.1.139/l3features:vrfs
  • [1] payload = <l3features:vrf xmlns:l3features="http://anutanetworks.com/l3features"><name>star</name><vrf-definition-mode>true</vrf-definition-mode></l3features:vrf>

Dec 23, 2020, 2:52:53 PM   entity-exists:/controller:devices/device=172.16.1.139/l2features:vlans (duration = 14 msec)
Dec 23, 2020, 2:52:53 PM   create-data:/app/restconf/data/controller:devices/device=172.16.1.139/l2features:vlans (duration = 34 msec). Params:
  • [0] createData = /app/restconf/data/controller:devices/device=172.16.1.139/l2features:vlans
  • [1] payload = <l2features:vlan xmlns:l2features="http://anutanetworks.com/l2features"><id>22</id></l2features:vlan>

Dec 23, 2020, 2:52:53 PM   entity-exists:/controller:devices/device=172.16.1.139/interface:interfaces (duration = 7 msec)
Dec 23, 2020, 2:52:53 PM   create-data:/app/restconf/data/controller:devices/device=172.16.1.139/interface:interfaces (duration = 32 msec). Params:
  • [0] createData = /app/restconf/data/controller:devices/device=172.16.1.139/interface:interfaces
  • [1] payload = <interface:interface xmlns:interface="http://anutanetworks.com/interface"><long-name>FastEthernet8.22</long-name><name>FastEthernet8.22</name><mode>sub-interface</mode><ip-address>16.6.6.6</ip-address>
    <netmask>255.255.255.0</netmask><admin-state>UP</adm... (515 bytes)

Dec 23, 2020, 2:52:54 PM   Saving commands
Dec 23, 2020, 2:52:54 PM   Resuming commit
Dec 23, 2020, 2:52:54 PM   Dry-run mode. Commands will not be sent to the device
Dec 23, 2020, 2:52:54 PM   Device is in offline provision mode. Command execution will not be attempted.
Dec 23, 2020, 2:52:54 PM   Triggering data model commit
Dec 23, 2020, 2:52:54 PM   Reservation cleanup count: 0
Dec 23, 2020, 2:52:54 PM   device(create = 9, update = 6) service(create = 11)
Dec 23, 2020, 2:52:54 PM   Create: /controller:services/l3service:l3-services/l3-service=star
+ l3-service:
+    device-id: 172.16.1.139
+    interface: FastEthernet8
+    interface-mode: sub-interface
+    ip-address: 16.6.6.6
+    name: star
+    netmask: 255.255.255.0
+    service-status: AVAILABLE
+    vlan-id: 22
+    vrf: star
+    vrf-definition-mode: true
Dec 23, 2020, 2:52:54 PM   Create: /controller:devices/device=172.16.1.139/l3features:vrfs/vrf=star
+ vrf:
+    name: star
+    vrf-definition-mode: true
Dec 23, 2020, 2:52:54 PM   Create: /controller:devices/device=172.16.1.139/l2features:vlans/vlan=22
+ vlan:
+    id: 22
Dec 23, 2020, 2:52:54 PM   Update: /controller:devices/device=172.16.1.139/interface:interfaces/interface=FastEthernet8.22
```

Appendix

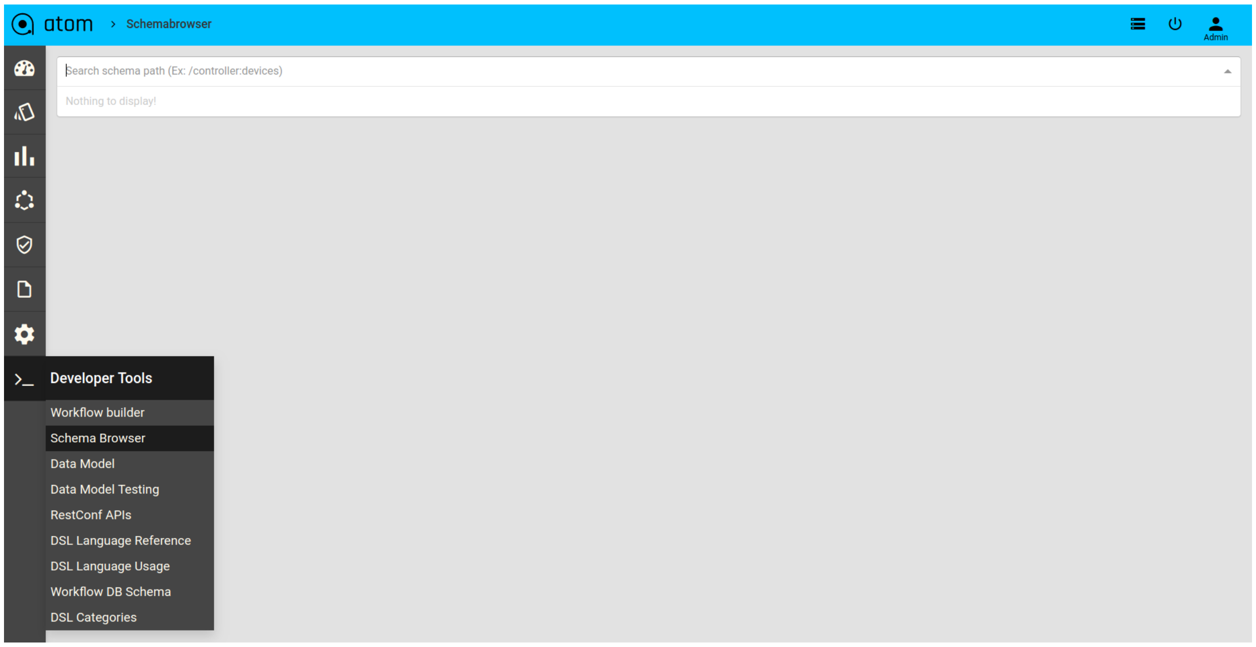
ATOM Schema Browser

Using the Schema Browser, you can browse the YANG schema (object tree) to derive the relative path for the modelled entities in ATOM. Using this utility, you can look at all the ATOM entities that have been modelled in YANG. Apart from the schema structure of the individual entities, you can also view the schema of any device or service package that is activated in ATOM.

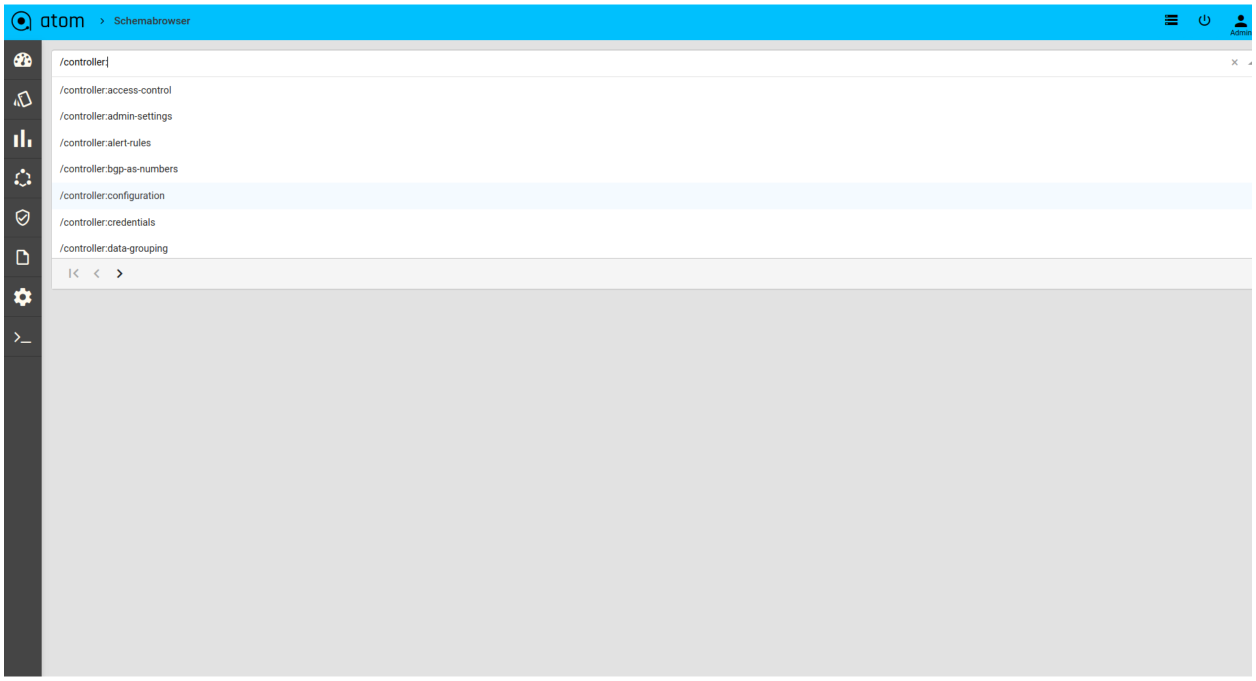
Example

In this example, let us explore how the schema for VRFs **/controller:devices/device/l3features:vrfs** can be obtained using the Schema Browser..

1. Go to **Administration > System > General settings.** Select the **Enable-Developer-Mode.**
2. Navigate to **Developer Tools** > **Schema Browser**



3. In the text box, type **/controller:**



/controller:

/controller:access-control

/controller:admin-settings

/controller:alert-rules

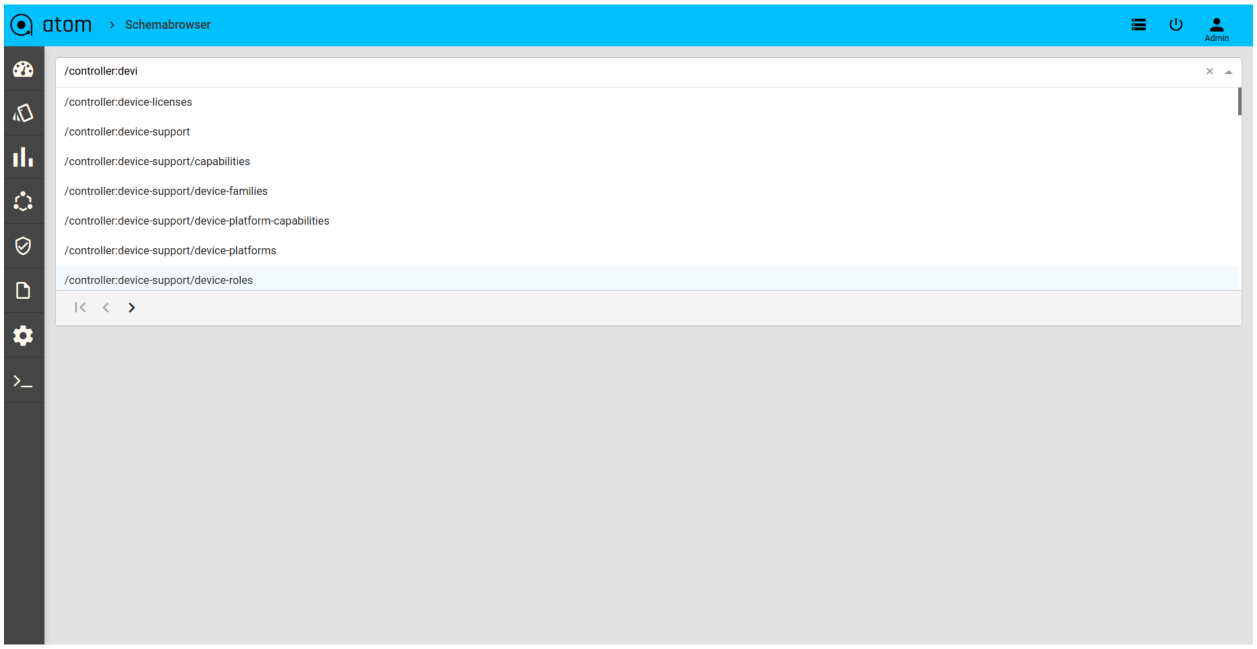
/controller:bgp-as-numbers

/controller:configuration

/controller:credentials

/controller:data-grouping

4. By typing the keyword **devi** in Schema Path (/controller:devi), all the entities starting with that start with devi as shown below:



/controller:devi

/controller:device-licenses

/controller:device-support

/controller:device-support/capabilities

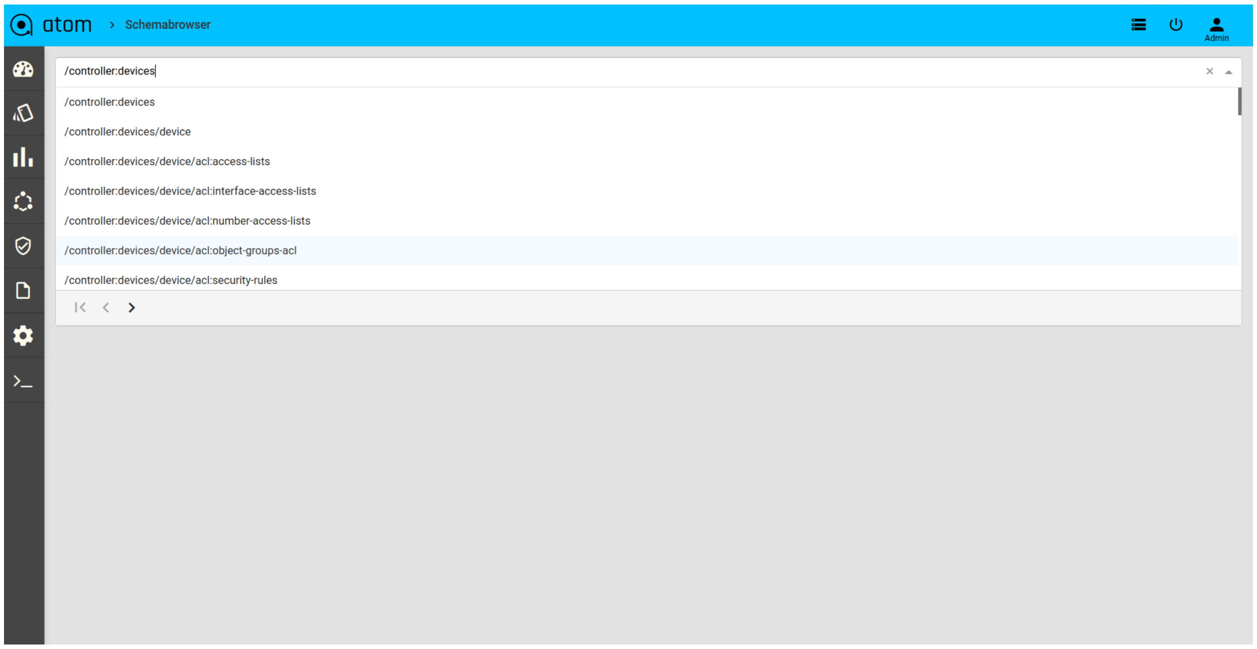
/controller:device-support/device-families

/controller:device-support/device-platform-capabilities

/controller:device-support/device-platforms

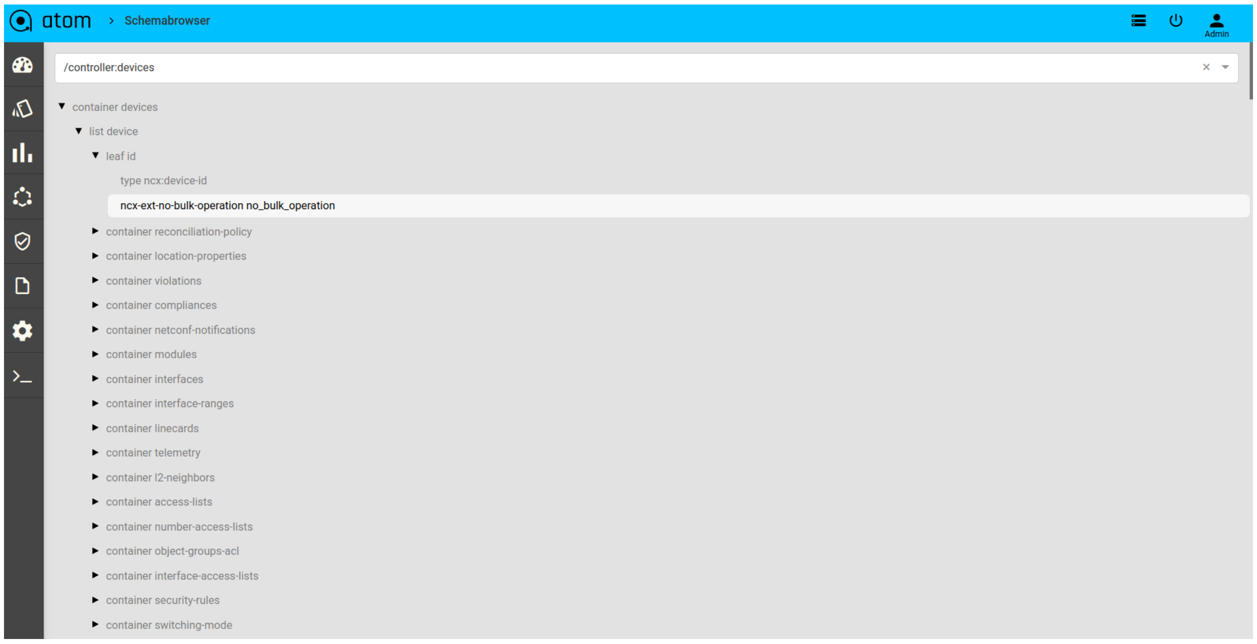
/controller:device-support/device-roles

5. As devices container is modeled under controller, xpath to fetch the devices schema present in ATOM is /**controller:devices**



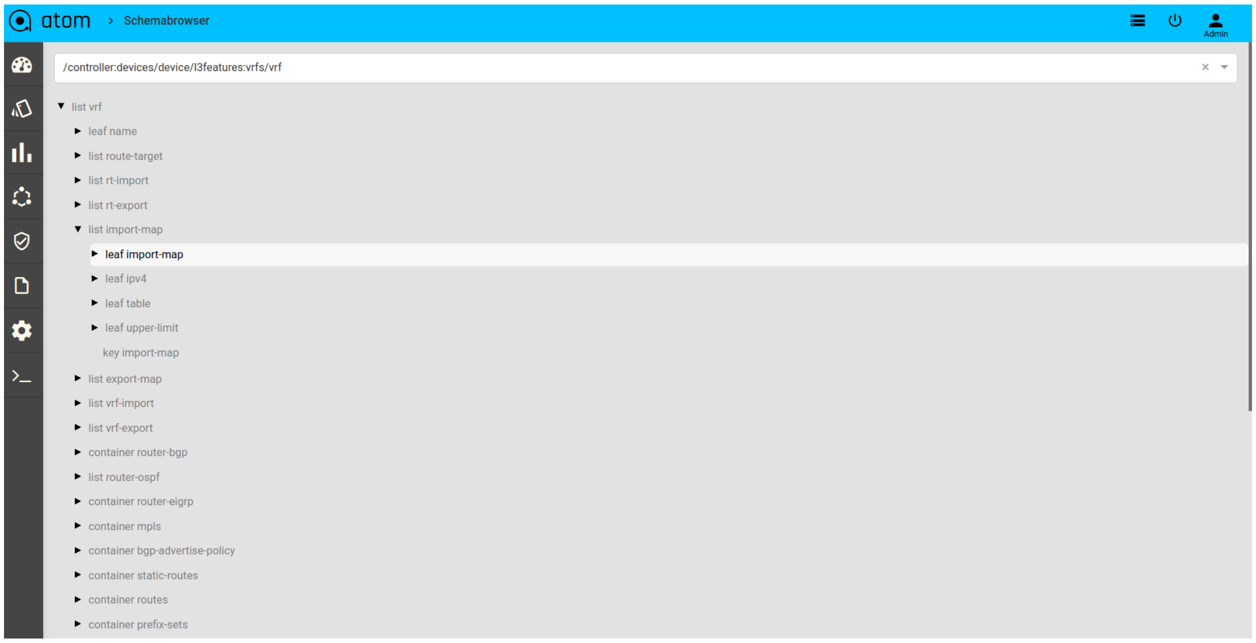
```
/controller:devices
/controller:devices
/controller:devices/device
/controller:devices/device/acl:access-lists
/controller:devices/device/acl:interface-access-lists
/controller:devices/device/acl:number-access-lists
/controller:devices/device/acl:object-groups-acl
/controller:devices/device/acl:security-rules
```

6. Hit the Enter button to get the devices schema



```
/controller:devices
▼ container devices
  ▼ list device
    ▼ leaf id
        type ncx:device-id
        ncx-ext-no-bulk-operation no_bulk_operation
    ▶ container reconciliation-policy
    ▶ container location-properties
    ▶ container violations
    ▶ container compliances
    ▶ container netconf-notifications
    ▶ container modules
    ▶ container interfaces
    ▶ container interface-ranges
    ▶ container linecards
    ▶ container telemetry
    ▶ container l2-neighbors
    ▶ container access-lists
    ▶ container number-access-lists
    ▶ container object-groups-acl
    ▶ container interface-access-lists
    ▶ container security-rules
    ▶ container switching-mode
```

The schema path for the VRFs: **/controller:devices/device/l3features:vrfs/vrf**



/controller:devices/device/l3features:vrfs/vrf

```
▾ list vrf
    ▸ leaf name
    ▸ list route-target
    ▸ list rt-import
    ▸ list rt-export
    ▾ list import-map
        ▸ leaf import-map
        ▸ leaf ipv4
        ▸ leaf table
        ▸ leaf upper-limit
          key import-map
    ▸ list export-map
    ▸ list vrf-import
    ▸ list vrf-export
    ▸ container router-bgp
    ▸ list router-ospf
    ▸ container router-eigrp
    ▸ container mpls
    ▸ container bgp-advertise-policy
    ▸ container static-routes
    ▸ container routes
    ▸ container prefix-sets
```

ATOM Extensions to YANG

Anuta Networks developed its own custom extensions to enhance the usage of YANG and these extensions are located in the following folders:

atomsdk/packages/Anuta/anuta/ncx/ncx-extensions.yang

atomsdk/packages/Anuta/anuta/ncx/ncx-ui-extensions.yang

atomsdk/packages/abstractdevicemodels/model/sdk-extensions.yang

Some of the commonly used UI extensions in ATOM are listed below:

1. **ncx-ext-seq-no:** This extension is used to indicate the display order of the property in the ATOM UI.

   E..g:
   ```
   leaf cpu-mhz {
           type uint32;
           n-ext-ui:ncx-ext-seq-no 56;
   }
   ```
2. **ncx-ext-hidden:** This extension can be used to hide a node from UI form.
3. **ncx-ext-multi-select:** This extension can be used to select multiple values in combobox.

Below are some of the extensions useful in service package code-generation and more can be found in the above mentioned files.

1. **maps-to**: This extension is used to define a mapping between service data node to device data node. The mapping can be an Xpath or RC path.
   E.g:

```
leaf auth-password {
  type string {
    length "1..8";
  }
  description
    "string";
  n-ext:maps-to "/controller:devices/device/if:interfaces/interface/hsrp:hsrp/auth-key";
}";
```

2. **ncx-maps-to-expr**: This extension is a superset of maps-to, we can write an expression about how it maps element(s) in service yang to element in device yang.
   E.g:
```
      list endpoints {
        n-ext:ncx-maps-to-expr '/ac:devices/device/ai:interfaces/interface/long-name' =
'$(interface-name).$(unit)';
        leaf unit {
          type uint8;
        }
        leaf interface-name {
          type string;
        }
      }
```

3. **device-platform**: Use this extension as a sub-statement of maps-to for specifying the platform to which this maps-to is applicable for.
   E.g:
```
        leaf native-vlan-id {
          type uint16 {
            range "1..4094";
          }
            mandatory true;
            when "../variant = 'trunk'";
            n-ext:maps-to
"/ac:devices/ac:device[ac:id=current()/../switch1-device-id]/if:interfaces/interface/allowed-vlans/if-ext:nativ
e-vlan"{
              sdk-ext:device-platform "JUNOS";
            }
        }
```

YANG Validations & Constraints

Below are few yang statements which will help to build intelligence(validations/ checks/constraints) in the model

1. **when**

   Nodes are valid only if the when condition is satisfied.

   E.g:

```
leaf variant {
    type enumeration {
      enum "trunk";
      enum "access";
    }
 }
leaf native-vlan {
 description "Native VLAN that is configured when trunk ports are used";
 type int16 {
   range 1..4094;
 }
 when "../variant = 'trunk'";
}
```

In the above example native-vlan will be used only if variant = trunk.

2. **must**

   It can be used on any data to have some constraints

   E.g:

```
leaf variant {
      type enumeration {
         enum "trunk";
         enum "access";
      }
 }
container vlans {
 description "List of vlans permitted on the l2-edge ports";
 list vlan {
   key vlan-id;
   leaf vlan-id {
     type int16 {
       range 2..4094;
     }
   }
   must
'(count(/ac:services/l2-edge:l2edge[name=current()/../../../../name]/devices/device[name=current()/../../nam
e]/vlans/vlan/id) = 1 and ../../variant = "access") or
(count(/ac:services/l2-edge:l2edge[name=current()/../../../../name]/devices/device[name=current()/../../name
]/vlans/vlan/id) >= 1 and ../../variant = "trunk")' {
     error-message "vlan should be one if variant is access or vlan should be >= one if variant is trunk";
   }
 }
}
```

Above must is used to validate vlan count to be exactly 1 if the variant is access or vlan count to be >=1 if variant is trunk

Library Utils for Service Modelling

For service modeling development Anuta provides library utils which are accessible at **atomsdk/packages/servicemodel/scripts.** These can be used in python based service logic written either manually or auto generated via SDK.

The python classes generated for the device yang models (Python Bindings for Device YANG Models) will also be present in the above directory which effectively makes the **servicemodel** package as a complete library for usage in service python logic.

servicemodel/scripts/device_abs_lib.py consists of various definitions which are basic CRUD operations performed on an object.

- create()- To create the object in ATOM using post operation
- update()- To edit the object in ATOM using put operation (overwrite)
- delete()- To delete the object in ATOM using delete operation.

The other definition **validate_inputs_form_payload** in this file will use the python bindings present in **servicemodel/scripts/controller** to validate the inputs and form the payload required for above CRUD operations.

Few other commonly used python modules in **servicemodel/scripts** are

1. yang.py
2. util.py
3. devicemgr.py

1. yang.py provides Sdk class and AbstractYangServiceHandler class

Sdk Class provides basic methods to do CRUD operations for service models.

- createData()- To create the object to ATOM using post operation
- updateData()- To edit the object of ATOM using put operation (overwrite)
- patchData()- To patch the object of ATOM using patch operation(extension)
- getData()- To get the object of ATOM using get operation
- deleteData()- To delete the object of ATOM using delete operation.

These methods are used for modeled entities only, not for RPC’s. To deal with RPC’s Sdk Class provides invokeRpc() method

CreateData Method

createData() method takes inputs as url, payload, yang_session, addReference and failOnExistingData arguments and posts the data to the server.

- **url** : target on which data to be posted
- **payload**: xml/json object to be posted
- **yang_session**: session object
- **addReference**: If AddReference is True, ATOM will create a reference for this object to keep track of this object.

Note: if addReference is True, no need to handle delete block for this object, as ATOM platform will take care of deletion of this object using reference.

```
@staticmethod
@util.wrappedmethod()
def createData(url, payload, yang_session, addReference=True, failOnExistingData=False):
  if payload == "" or payload is None:
     util.log_debug('payload is empty or none')
     return
  try:
    from com.anuta.model.base import YangSessionThreadLocal
  except ImportError:
    pass
  YangSessionThreadLocal.setDeviceAuditDisabled(True)
  try:
     Sdk.createDataWithTaskId(url, payload, yang_session, yang_session.getTaskId(), addReference,
failOnExistingData)
  finally:
     YangSessionThreadLocal.setDeviceAuditDisabled(False)
```

Example

In this example, addReference is True by default as it is not mentioned, So for this entity no need to handle delete code in service logic

```
uri = '%s/vrf=%s/router-bgp' % (dev.url, vrf_name)
print 'uri = %s, neighbor = %s' % (uri, neighbor.toXml())
yang.Sdk.createData(uri, neighbor.toXml(), ctx.getSession())
```

Similarly we have updateData, patchData and deleteData Methods in this Sdk class. Please refer yang.py module for more details in **atomsdk/packages/servicemodel/scripts**

invokeRpc() Method

This method takes inputs as rpcname and payload and provide respective output

- rpcname: Name of the rpc
- payload: payload object in xml/json

```
@staticmethod
# @util.wrappedmethod(detailed_log=True)
def invokeRpc(rpcname, payload, log = True):
```

```
if log:
    util.log_debug('rpcname = %s, payload = %s' % (rpcname, payload))
# FIXME: remove this in 5.7
origTaskId = YangSessionThreadLocal.getTaskId()
try:
    ret = Sdk.getInstance().restconf.invokeRpc(rpcname, payload)
finally:
    YangSessionThreadLocal.setTaskId(origTaskId)
return ret
```

If any entity in ATOM is not modeled then we can implement RPC for the entity and call when we require in service modeling using this method.

Example

In this example rpc-name is device-discovery and the payload we are forming with the help of kwargs dictionary, finally we will get output_xml object when invokeRpc is called.

```
def sl_device_discovery(**kwargs):
  slinput = sciencelogic_rpc.device_discovery.input.input()
  slinput.cidn = kwargs.get('cidn')
  slinput.source_system = kwargs.get('source_system')
  slinput.community  = kwargs.get('community')
  slinput.device_id = kwargs.get('device_id')
  slinput.collector = kwargs.get('collector')
  slinput.template = kwargs.get('template')
  if isinstance (kwargs.get('device_ip'), list):
    for devip in kwargs.get('device_ip'):
      sldips  = slinput.device_ips.add(devip)
  else:
      sldips  = slinput.device_ips.add(kwargs.get('device_ip'))
  payload = slinput.getxml(filter=True)
  log("create payload is:%s", payload)
  output_xml = yang.Sdk.invokeRpc('sciencelogic:device-discovery', payload)
  print "output_xml",output_xml
  return output_xml
```

AbstractYangServiceHandler class provide two basic methods

- **register() Method:** Register all the resource unit handlers. This method is called from the plugin.py module of the service package**.**

```
def register(self):
  """ Register all resource unit handlers. This is called from the plugin code
  when the plugin is started
  """
  for xpath in self.handler_map.keys():
     handler = self.handler_map[xpath]
     util.log_debug('Registering %s => %s' % (xpath, handler))
     registerServiceHandler(xpath, handler)
```

  This register method internally calls registerServiceHandler() class for details refer yang.py module

- **unregister() Method:** Unregister all the resource unit handlers. This method is called from the plugin.py module of the service package when unloading packages done from ATOM**.**

```
def unregister(self):
  """ Unregister all resource unit handlers. This is called from the plugin code
  when the plugin is stopped
  """
  for xpath in self.handler_map.keys():
    handler = self.handler_map[xpath]
    unregisterServiceHandler(xpath, self.global_map)
```

This unregister method internally calls unregisterServiceHandler() class, for details refer yang.py module

2. util.py provides utilities for service model development with the help of **IPPrefix class** and some other functions

IPPrefix class takes cidr as input and provides details about address, netmask and wild card etc., as output.

```
class IPPrefix(object):

  """ IP Prefix utility class
  """
  def __init__(self, prefix):
    self.prefix = prefix
    arr = prefix.split('/')
    if len(arr) > 1:
      self.address = arr[0]
      self.masklen = int(arr[1])
      mask = IPPrefix.get_mask_num(self.masklen)
      self.netmask = IPPrefix.to_ip_address(mask)
      self.wildcard = IPPrefix.to_ip_address(~mask)
      # FIXME: handle ipv6
      if self.masklen == 32:
        self.is_ipaddress = True
      else:
        self.is_ipaddress = False
    else:
      self.address = prefix
      self.netmask = '255.255.255.255'
      self.wildcard = '0.0.0.0'
      self.masklen = 32
      mask = ~0
      self.is_ipaddress = True
    # convert address to number
    addrnum = IPPrefix.ip2int(self.address) & mask
    self.network = IPPrefix.to_ip_address(addrnum)
```

Example of IPPrefix

In this example IPPrefix class takes cidr as input and provides netmask as output

```
cidr_obj = util.IPPrefix(inputdict['cidr'])
dest_mask = cidr_obj.netmask
```

Few frequently used methods in **Module util.py** are:

- **isEmpty() -** This will check if the object is empty or not. Code Snippet for isEmpty() Method

```
def isEmpty(val):
  """ Check weather val is empty

  Args:
  Val : Value need to check
  Returns:
  True: if the value is empty
  False: if the value is not empty
  """
  if(val == None):
     return True
  if isinstance(val, list):
     return len(val) == 0
  if Collection.isInstance(val):
     return val.isEmpty()
  if isinstance(val, str):
     return val.strip() == ''
  if isinstance(val, unicode):
     return str(val).strip() == ''

  return False
```

Example for isEmpty():

In this example checking if protocol is empty or not, if empty then we are return from there

```
def validate_protocol(self, ctx, protocol):
  if util.isEmpty(protocol):
     return
```

- **isNotEmpty() -** This will check if the object is not empty or not. Code Snippet for isNotEmpty() Method

```
def isNotEmpty(val):
  """ Check weather val is not empty

  Args:
  Val : Value need to check
  Returns:
  True: if the value is not empty
  False: if the value is empty
  """
  if isEmpty(val):
     return False
  return True
```

Example for isNotEmpty():

This example checks inputdict['name'] is not empty and then only proceeds further

```
#Start of Device binding with python bindings
interfaces_object = devices.device.interfaces.interfaces()
if util.isNotEmpty(inputdict['name']):
 interfaces_interface_object = interfaces_object.interface.add(long_name=inputdict['name'])
 interfaces_interface_object.name = inputdict['name']
```

3. devicemgr.py provides basic methods to get device obj from device ip or device id or device name etc.,

getDeviceByIp() Method

takes input as device management ip and provide device object to service model

Code Snippet for getDeviceByIp() Method:

```
def getDeviceByIp(ipAddress,validate_type = False,task_id=None):
  """
  fetch the device complete tree for given ip
  Args: device ip
  Return: device object which has device information
  """
  rcpaths = yang.Sdk.getRcPathListForXPathAndValue(
        '/controller:devices/device/mgmt-ip-address', ipAddress)
  if util.isEmpty(rcpaths):
     util.log_debug('rcpaths for this device = %s are empty' %(ipAddress))
     return None
  rcPath = rcpaths[0]
  if len(rcpaths) > 1:
     util.log_debug('WARN: got multiple rcpaths. count = %d' % (len(rcpaths)))
     util.log_debug('%s' % (rcpaths))
  xml = yang.Sdk.getData(rcPath, '', task_id, None)
  # util.log_debug('devicexml = %s' % (xml))
  if(xml == None):
        util.log_debug('No xml data. rcPath = %s' % (rcPath))
        return None
    xmlObj = util.parseXmlString(xml)
    dev = Device(xmlObj)

    if validate_type and util.isEmpty(dev.device.get_field_value('device_type')):
        raise Exception('Device type is empty for %s' % (ipAddress))

    return dev
```

Example

In this example getDeviceByIp() method takes input as ip(device-management-ip) and provides device_object for further actions

Note: If dev object is None then need to check whether device is onboarded or not (or) device is online or not.

```
def create(self, ip, os_type, sdata):
    print 'create ip = %s, ostype = %s' % (ip, os_type)
    dev = devicemgr.getDeviceByIp(ip)
    if(dev == None):
        print 'No device by ip: %s' % (ip)
        raise Exception('No device by ip: %s' % (ip))
```

Similarly we have few other frequently used methods

getDeviceById() method

getDeviceByName() method

getDeviceByUniqueName() method

getDeviceByInterfaceName() method

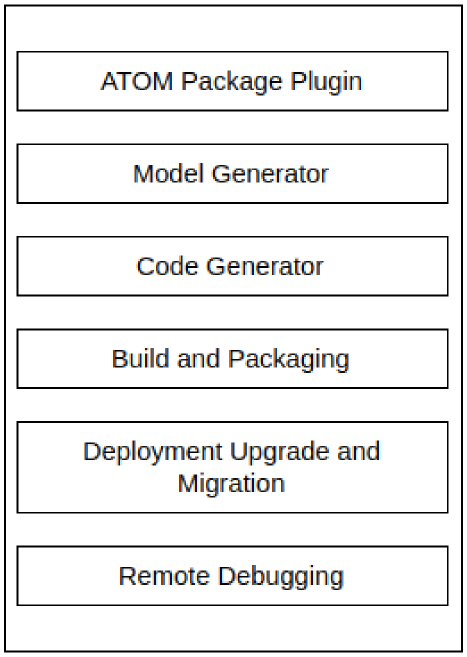
Please refer devicemgr.py module for more details at build/lib/servicemodel/scripts

ATOM SDK

Introduction

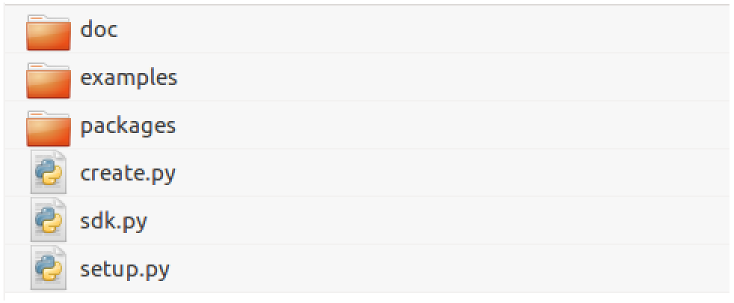
ATOM Software Development Kit (SDK) provides a gradle-based plugin **Package-Plugin jar** that serves as a backbone for package development in ATOM. ATOM SDK provides CLI and also integration into IDE like IntelliJ. The plugin enables you to perform the following tasks of Services/Drivers Development process in ATOM:

- Develop device packages
- Develop service packages
- Compile, validate, generate device and service packages
- Load Packages to ATOM
- Upgrade of Packages



Folder hierarchy

Unzip the contents of the ATOM SDK zip to view the following folder structure:.



```
doc
examples
packages
create.py
sdk.py
setup.py
```

- **doc** - This folder contains README and the plugin documentation.
- **examples -** This folder has package zip files for different types of packages.
- **packages** - The core Package Plugin jar is part of the packages folder, which also has a few more library and base dependency packages required for development of new device and service packages.
- *create.py*, *sdk.py*, *setup.py* - These are the python files required for setting up device and service packages environment.

Setting up the environment for ATOM Package Plugin

ATOM Package Plugin supports multiple gradle tasks that help create an environment suited for developing packages. These tasks can be triggered from an **IDE or CLI**.

For the plugin tasks to run, ensure that the prerequisites are met with.

Prerequisites

To setup the environment, you must ensure that the following software requirements are met:

1. Python (2.7.12/3.x)
2. Python setup tools
3. Python Pip and Python modules bitarray, cmd2, TAPI, XEGER
4. Pyang(1.7.8/2.5.2(for python3)).
5. JAVA (java 1.8 or greater)
6. Gradle

   For information about installing gradle in your environment, visit http://gradle.org.

Setting up the environment in Ubuntu

1. Execute the following commands:

```
sudo apt-get install python python-setuptools/sudo apt-get install python3 python-setuptools(for
python3)
sudo easy_install pip
sudo pip install bitarray
sudo pip install cmd2
sudo pip install tapi
sudo pip install xeger
sudo pip install requests
```

2. Install Oracle JDK for Linux and unzip it.

   Set the JAVA_HOME environment variable pointing to jdk directory.

3. Install gradle by executing the following command:

```
sudo apt-get install gradle
```

Setting up the environment in Windows

1. Download get-pip.py from https://bootstrap.pypa.io/get-pip.py

2. Execute the following command: python get-pip.py
3. Install Visual C++: https://www.microsoft.com/en-us/download/details.aspx?id=44266
4. Execute the following commands in the following order:

```
pip install setuptools --upgrade
pip install bitarray
pip install cmd2
pip install tapi
pip install xeger
pip install requests
```

5. Set the JAVA_HOME environment variable pointing to jdk directory.

   Example: **C:\Program Files\Java\jdk1.8.0_91**

   > NOTE: Proper installation of gradle can be verified by using the command *gradle -version.*

6 . Gradle Installation in windows

Step 1. https://gradle.org/releases/ get the latest Gradle distribution

Step 2. Unpack the distribution zip

Step 3. Configure your system environment Path variable

For e.x: C:\Gradle\gradle-4.10.2\bin.

Step 4. Verify your installation

Open a console (or a Windows command prompt) and run gradle -v to run gradle and verify the version, e.g.:

```
$ gradle -v
------------------------------------------------------------
Gradle 4.10.2
```

Setting up the environment in Centos

1. Execute the following commands:

```
sudo yum install python3 python-setuptools
sudo yum –y install python3-pip
sudo pip3 install bitarray
sudo pip3 install cmd2
sudo pip3 install tapi
sudo pip3 install xeger
sudo pip3 install requests
```

2. Install Oracle JDK for Centos and unzip it.

   Set the JAVA_HOME environment variable pointing to jdk directory.

6 . Gradle Installation in centos

Step 1. Start downloading the Gradle Binary-only zip file in the /tmp directory using the following wget command.

```
$ wget https://services.gradle.org/distributions/gradle-5.0-bin.zip -P /tmp
```

Step 2. Unpack the distribution zip with the following command.

```
$ sudo unzip -d /opt/gradle /tmp/gradle-5.0-bin.zip
```

Step 3. Configure your system environment Path variable

For e.x: C:\Gradle\gradle-5.0\bin.

Step 4. Verify your installation

Open a console (or a Windows command prompt) and run gradle -v to run gradle and verify the version, e.g.:

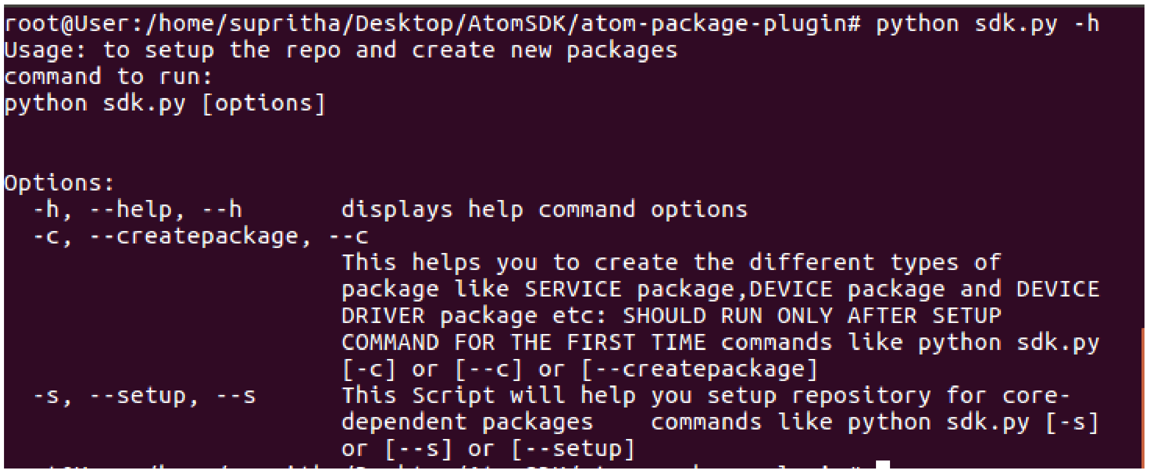
```
$ gradle -v
------------------------------------------------------------
Welcome to Gradle 5.0!


Here are the highlights of this release:
 - Kotlin DSL 1.0
 - Task timeouts
 - Dependency alignment aka BOM support
 - Interactive `gradle init`
```

```
For more details see https://docs.gradle.org/5.0/release-notes.html



------------------------------------------------------------

Gradle 5.0

------------------------------------------------------------


Build time:   2018-11-26 11:48:43 UTC

Revision:     7fc6e5abf2fc5fe0824aec8a0f5462664dbcd987


Kotlin DSL:   1.0.4

Kotlin:       1.3.10

Groovy:       2.5.4

Ant:          Apache Ant(TM) version 1.9.13 compiled on July 10 2018

JVM:          1.8.0_191 (Oracle Corporation 25.191-b12)

OS:           Linux 3.10.0-862.14.4.el7.x86_64 amd64
```

Setting up the repository for developing packages

In ATOM SDK, the ***sdk.py*** script sets up the SDK plugin environment for creating various packages.

To setup the repository of your choice, follow the steps as outlined below:



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin# python sdk.py -h
Usage: to setup the repo and create new packages
command to run:
python sdk.py [options]


Options:
  -h, --help, --h       displays help command options
  -c, --createpackage, --c
                        This helps you to create the different types of
                        package like SERVICE package,DEVICE package and DEVICE
                        DRIVER package etc: SHOULD RUN ONLY AFTER SETUP
                        COMMAND FOR THE FIRST TIME commands like python sdk.py
                        [-c] or [--c] or [--createpackage]
  -s, --setup, --s      This Script will help you setup repository for core-
                        dependent packages    commands like python sdk.py [-s]
                        or [--s] or [--setup]
```

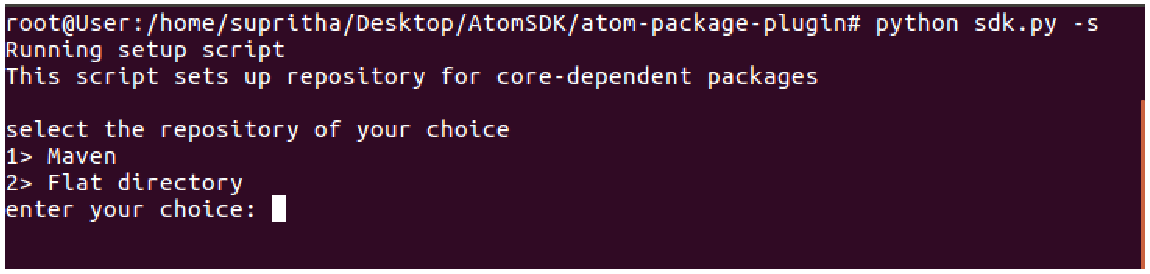
1. Run the command: python sdk.py -s

   This command runs the ***setup.py*** script which setups an environment for packages repository.

setup.py - This script is used to setup repositories for core-dependent packages. The core-dependent packages are present inside the "packages" folder and are necessary for developing new device and service packages.

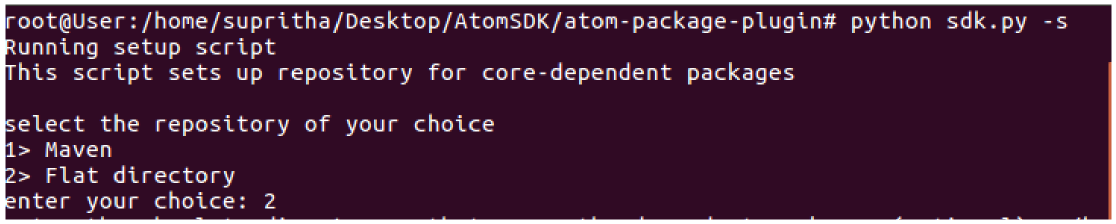
2. Select the repository of your choice.

   You can either setup a local repository or can publish the core-dependent packages to an artifact repository such as Nexus.



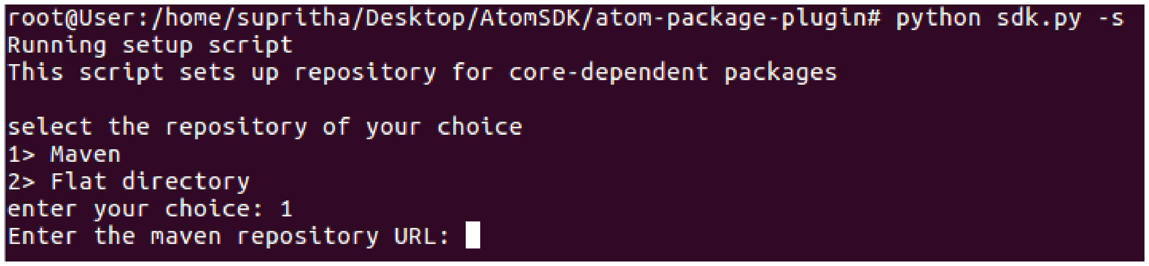
```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin# python sdk.py -s
Running setup script
This script sets up repository for core-dependent packages

select the repository of your choice
1> Maven
2> Flat directory
enter your choice: 
```

- **Local Repository (Flat Directory Structure)** : This option enables you to copy the core-dependent packages present in the *"packages"* folder to a flat directory.
- The absolute path of this particular flat directory, for example, '*/home/*' as shown below(verify that this folder is present already)



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin# python sdk.py -s
Running setup script
This script sets up repository for core-dependent packages

select the repository of your choice
1> Maven
2> Flat directory
enter your choice: 2
```

- **Maven Artifact Repository** : This option enables the user to copy the core-dependent packages in the "packages" folder uploaded to the artifact repository, for example Nexus.



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin# python sdk.py -s
Running setup script
This script sets up repository for core-dependent packages

select the repository of your choice
1> Maven
2> Flat directory
enter your choice: 1
Enter the maven repository URL: 
```

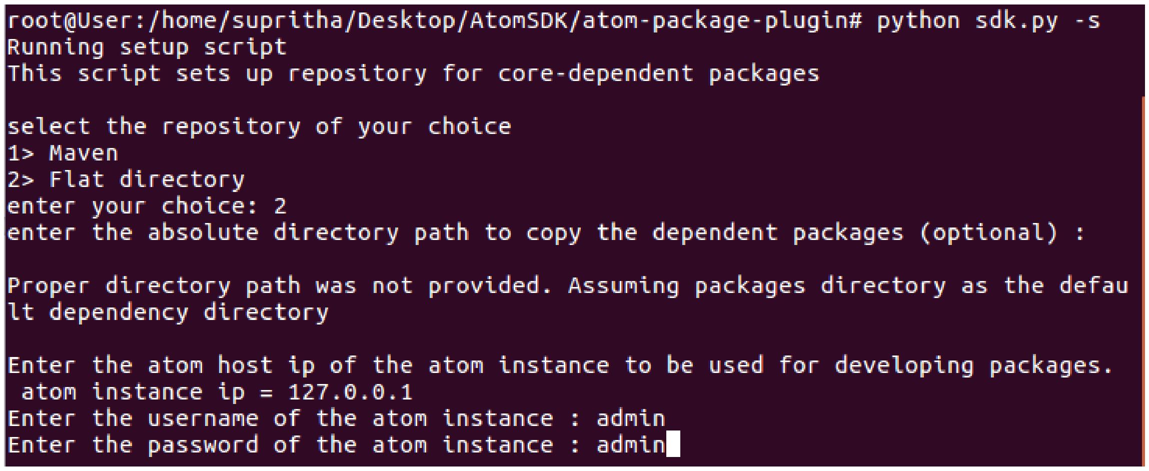
After setting up the repository, the script generates a ***config.xml*** file. This file contains two tags:

a) **repo-type** : Maven or Flat Directory
b) **repo-path** : The absolute path or URL of the directory.

The metadata present in the *config.xml* is important to run the subsequent scripts.

Let us take the example of the selected repository as the Flat Directory(a local repository) and the steps to be followed are illustrated below:

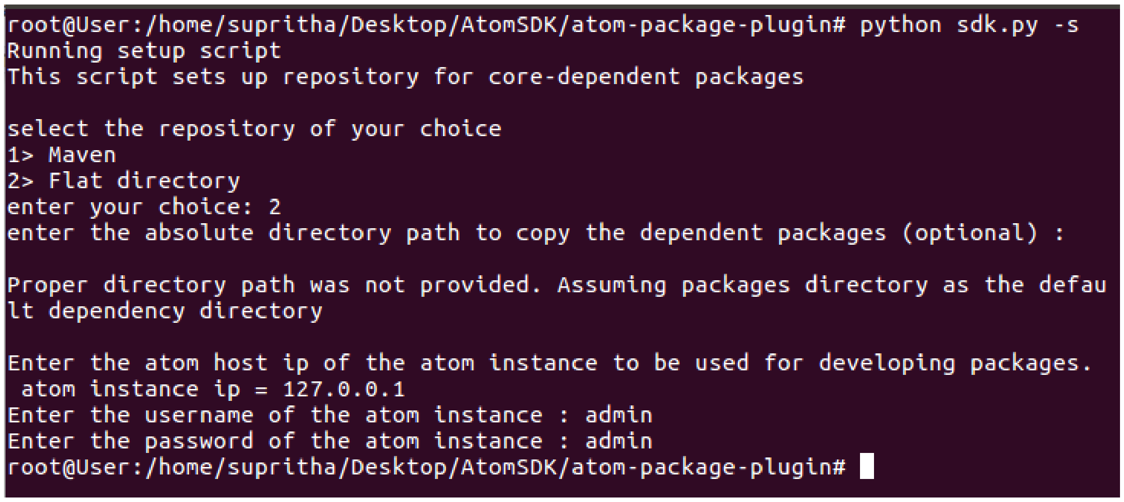
1. Enter the IP address of ATOM



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin# python sdk.py -s
Running setup script
This script sets up repository for core-dependent packages

select the repository of your choice
1> Maven
2> Flat directory
enter your choice: 2
enter the absolute directory path to copy the dependent packages (optional) :

Proper directory path was not provided. Assuming packages directory as the defau
lt dependency directory

Enter the atom host ip of the atom instance to be used for developing packages.
 atom instance ip = 127.0.0.1
Enter the username of the atom instance : admin
Enter the password of the atom instance : admin
```

If port is required for accessing the ATOM application then mention that as well. E.g: 172.16.1.10:30443, 127.0.0.1:8890

2. Enter the credentials to login into ATOM

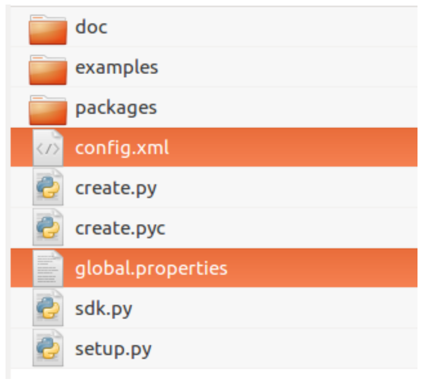


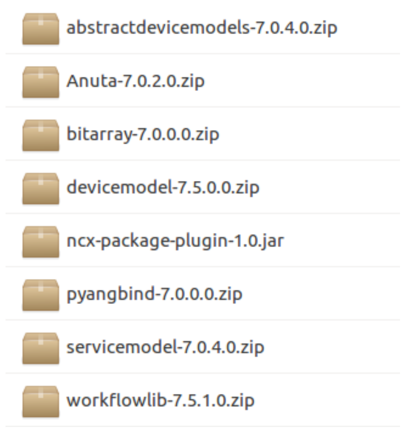
```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin# python sdk.py -s
Running setup script
This script sets up repository for core-dependent packages

select the repository of your choice
1> Maven
2> Flat directory
enter your choice: 2
enter the absolute directory path to copy the dependent packages (optional) :

Proper directory path was not provided. Assuming packages directory as the defau
lt dependency directory

Enter the atom host ip of the atom instance to be used for developing packages.
 atom instance ip = 127.0.0.1
Enter the username of the atom instance : admin
Enter the password of the atom instance : admin
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin#
```

After the successful setup process, the following files and folders are generated

- *global.properties* - contains the username, password and ATOM ip which will be used in package development process
- *config.xml* - contains the information of repo-type and path to dependencies.
- **dependencies** - The dependency packages for development of device and service models are copied to the destination folder of your choice.

IMPORTANT: Do not delete these files or folders.





Tasks for developing packages

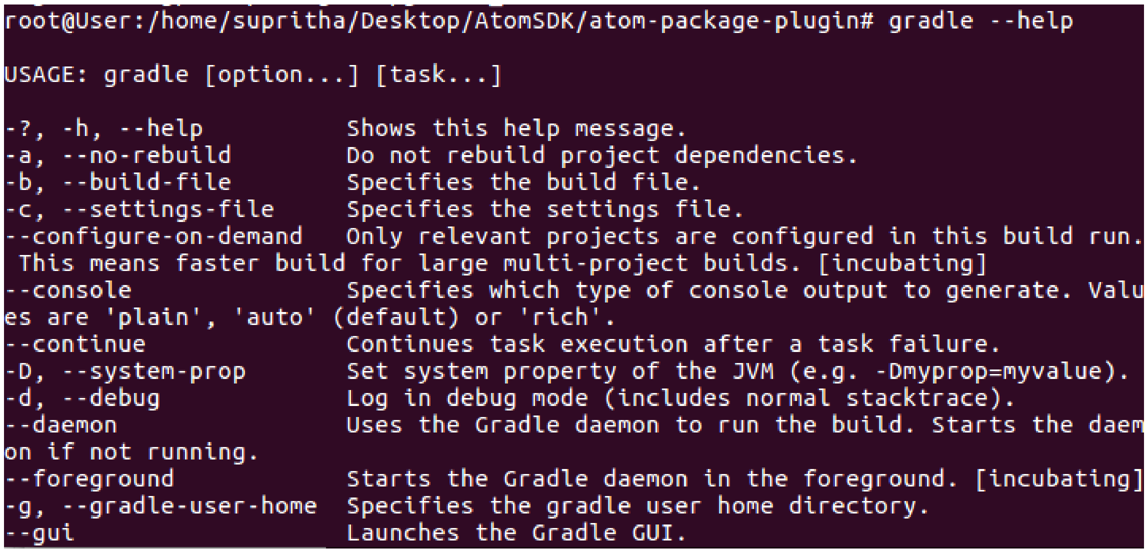
ATOM package plugin internally uses gradle for providing various options in package development.

General Gradle tasks

Command	Description
grade -help	All the commands are listed here
gradle tasks --all	All the gradle tasks are listed here

gradle -help

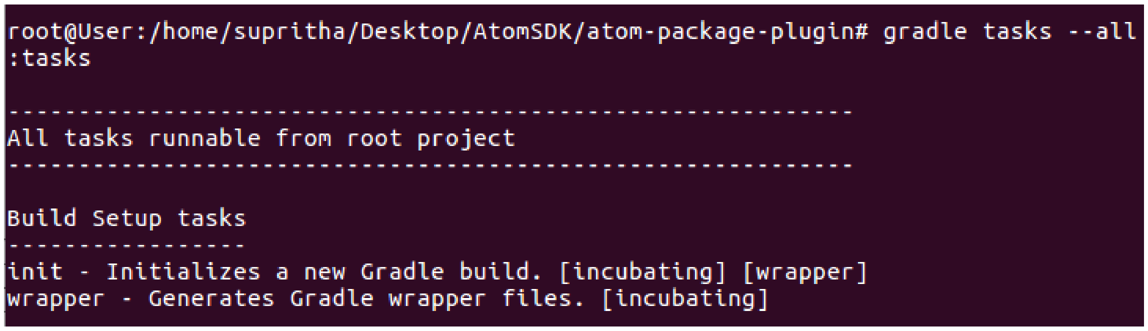
All the commands can be viewed as shown below:



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin# gradle --help

USAGE: gradle [option...] [task...]

-?, -h, --help          Shows this help message.
-a, --no-rebuild        Do not rebuild project dependencies.
-b, --build-file        Specifies the build file.
-c, --settings-file     Specifies the settings file.
--configure-on-demand   Only relevant projects are configured in this build run.
 This means faster build for large multi-project builds. [incubating]
--console               Specifies which type of console output to generate. Valu
es are 'plain', 'auto' (default) or 'rich'.
--continue              Continues task execution after a task failure.
-D, --system-prop       Set system property of the JVM (e.g. -Dmyprop=myvalue).
-d, --debug             Log in debug mode (includes normal stacktrace).
--daemon                Uses the Gradle daemon to run the build. Starts the daem
on if not running.
--foreground            Starts the Gradle daemon in the foreground. [incubating]
-g, --gradle-user-home  Specifies the gradle user home directory.
--gui                   Launches the Gradle GUI.
```

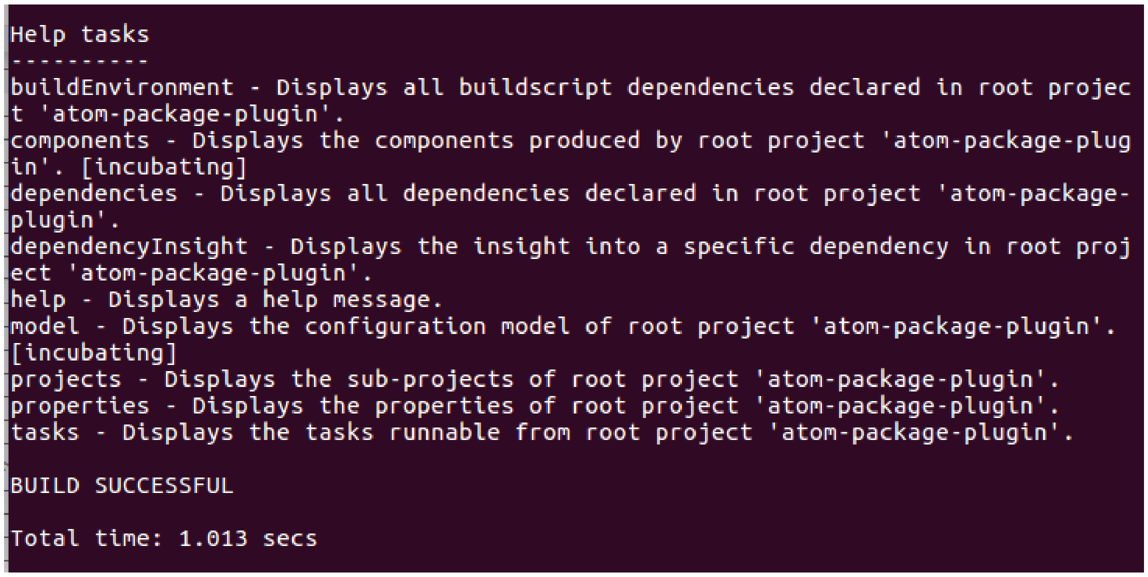
gradle tasks --all

Execute this command at root level as shown below:



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin# gradle tasks --all
:tasks

------------------------------------------------------------
All tasks runnable from root project
------------------------------------------------------------

Build Setup tasks
-----------------
init - Initializes a new Gradle build. [incubating] [wrapper]
wrapper - Generates Gradle wrapper files. [incubating]
```

Execution of these commands at the package level displays all the tasks that are available in the plugin, Build, Documentation



```
Help tasks
----------
buildEnvironment - Displays all buildscript dependencies declared in root projec
t 'atom-package-plugin'.
components - Displays the components produced by root project 'atom-package-plug
in'. [incubating]
dependencies - Displays all dependencies declared in root project 'atom-package-
plugin'.
dependencyInsight - Displays the insight into a specific dependency in root proj
ect 'atom-package-plugin'.
help - Displays a help message.
model - Displays the configuration model of root project 'atom-package-plugin'.
[incubating]
projects - Displays the sub-projects of root project 'atom-package-plugin'.
properties - Displays the properties of root project 'atom-package-plugin'.
tasks - Displays the tasks runnable from root project 'atom-package-plugin'.

BUILD SUCCESSFUL

Total time: 1.013 secs
```

ATOM specific tasks

Few of the Important Gradle Tasks, specific to ATOM, along with their descriptions are listed below:

Task	Description	Modeling Relevance
generateYin	This task is used to convert a YANG file to a YIN equivalent. The **generateYin** task internally uses python's PYANG tool which has been modified to support validating of the ATOM YANG models.	Device & Service Modelling
generateDeviceOperationTemplate	This task generates the *deviceoperation.xml* file containing the details of the create, delete, and update operations.	Device Modelling
verifyDeviceOperations	This task is used to verify the device operations defined for a device yang model. It creates a file that consists of warning statements of invalid device operation yang Xpath targets.	Device Modelling
generateDevicePackage	This task is used to generate device packages for a given **device.yang** being developed.	Device Modelling
generatePyBinds	This task is used to generate python class hierarchy for a YANG data model and its dependencies.	Device Modelling
generateDeviceDriverPackage	This task is used to generate a device driver package for a given **device-driver-yang** being developed.	Device Modelling
generateServicePackage	This task is used to generate a service package for a given *service.yang* being developed.	Service Modelling
generateDevicePybinds	This task is used to generate python class hierarchy for a YANG data model and its dependencies in a service package.	Service Modelling
Load	This task is used to upload a package to an ATOM instance.	Device & Service Modelling

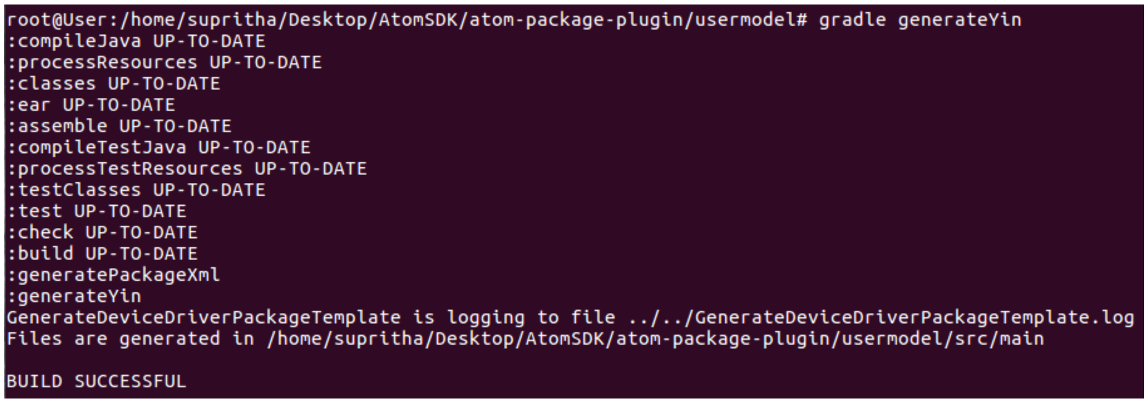
Activate	This task is used to activate a package present in an ATOM instance.	Device & Service Modelling
Deactivate	This task is used to deactivate a package loaded in an ATOM instance.	Device & Service Modelling
Delete	This task is used to delete a package present in an ATOM instance.	Device & Service Modelling
Upgrade	This task is used to upgrade an already existing package present in an ATOM instance.	Device & Service Modelling
Replace	Replace a package with new package content without any manual package upgrade steps	Device & Service Modelling
Purge	Clean all/specific data and its reference data under the package	Device & Service Modelling
cleanBuild	This task is a combination of two gradle tasks, 'clean' and ' build --refresh-dependencies'. This task first executes *gradle clean.* The clean task is defined by the java plugin and it removes the **buildDir** folder, thus cleaning previous builds' artifacts, which are no longer relevant.	Device & Service Modelling
copyToDependencies	This task internally runs the archive task (which generates the ready-to-be-uploadable zip). After generating the zip, it ascertains the repository type (whether maven/local) from *config.xml* file and the repository path.	Device & Service Modelling
enableMaintenanceMode	This task is used to enable maintenance mode on an ATOM instance. To run this task make sure all the necessary modifications are made in *gradle.properties* file, that have been explained in the *load* task.	Device & Service Modelling
disableMaintenanceMode	This task is used to disable maintenance mode on an ATOM instance. To run this task, ensure that all the necessary	Device & Service Modelling

	modifications are made in *gradle.properties* file, explained in the *load* task.	
restartServiceModelPluginAgent	This gradle task is used to stop and restart service model plugin for the agent.	Device & Service Modelling
restartServiceModelPluginServer	This gradle task is used to stop and restart service model plugin for server.	Device & Service Modelling

Below is a bit more detailed explanation of Gradle Tasks useful in Service Modelling.

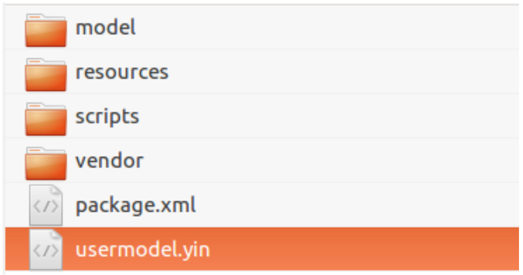
generateYin

1. Before running this task, ensure that the YANG model of the package is available in the path: **src\main\model**.
2. The generated yin file is created in the build\generated directory.



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel# gradle generateYin
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:ear UP-TO-DATE
:assemble UP-TO-DATE
:compileTestJava UP-TO-DATE
:processTestResources UP-TO-DATE
:testClasses UP-TO-DATE
:test UP-TO-DATE
:check UP-TO-DATE
:build UP-TO-DATE
:generatePackageXml
:generateYin
GenerateDeviceDriverPackageTemplate is logging to file ../../GenerateDeviceDriverPackageTemplate.log
Files are generated in /home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel/src/main

BUILD SUCCESSFUL
```

In the *gradle.properties* file, if the overwrite flag is set to 'true', the result is generated in the folder, **usermodel\src\main**, where 'usermodel' is the name of the package created.



If this flag is set to 'false', the file is generated in the path **usermodel\build\generated**

generateServicePackage

The service package generated contains the following entities:

1. The **model** folder contains the following:

<service_name>.yang file - Contains the schema of the service defined in YANG. This yang will be taken as input for the task to generate a service package basic service logic files.

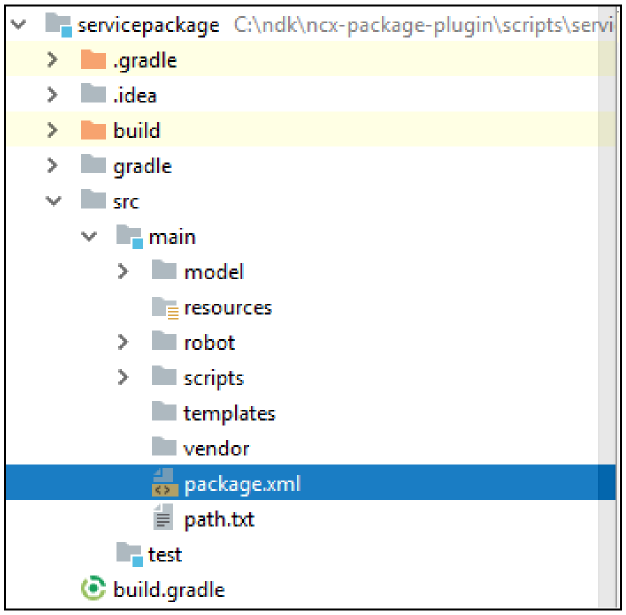
2. The **scripts** folder contains the following files:
    - *<service_name>.py* - Contains the logic binding the service to the device.
    - *plugin.py* - Code for adding the service as a new plug-in to ATOM
    - *_init_.py* - Required to make ATOM treat the directories as containing packages

If gradle.properties file autoupdate flag is set to true, it won't generate updates and delete pieces of code, whereas ATOM platform will handle it automatically.

If the driverimport flag is set to true it will take the driver name as the import for pybinds.

```
# Host of the atom instance to be used for developing the package.
atomHost=http://localhost:8890
# AuthToken for authentication. Auth Token is base 64 encoding of the string <username:password>
authToken=Basic YWRtaW46YWRtaW4=
# Flag to denote if the packages should be forced to upgrade while activating
forceUpgrade=false
# Flag to write files to src/main(when true) or build folder(when false).
overwrite=true
# Used to split python directories instead of generating under one directory
splitDir=true
# Used to strip code and compartmentalize in different python files
stripcode=False
# Used to generate device abstract library
devAbsLib=true
# Used to delete data from device when its true for Purge data
deleteTargetData = true
# If we set autoupdate flag to true it should not generate update piece of code
autoupdate = true
yangmount = false
```

If gradle.properties file overwrite flag is set to false, the result is generated in path **servicepackage/build/generated** else it is generated in **src/main** as shown below:

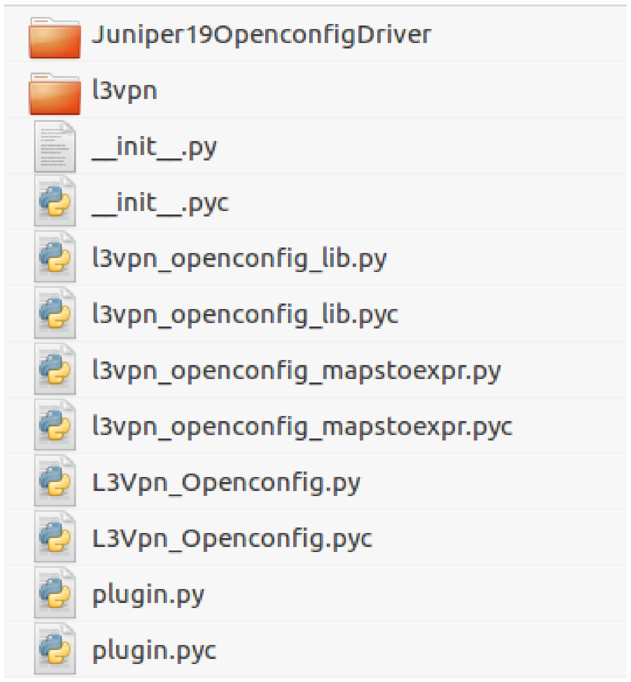


```
servicepackage  C:\ndk\ncx-package-plugin\scripts\servi
  .gradle
  .idea
  build
  gradle
  src
    main
      model
      resources
      robot
      scripts
      templates
      vendor
      package.xml
      path.txt
    test
  build.gradle
```

generateDevicePyBinds

By performing this task on device models of ATOM, the resulting python classes allow additional methods to be associated with the service modelling.

This generates with <devicedrivername> named library package which has python classes for the YANG data models mentioned as dependency.

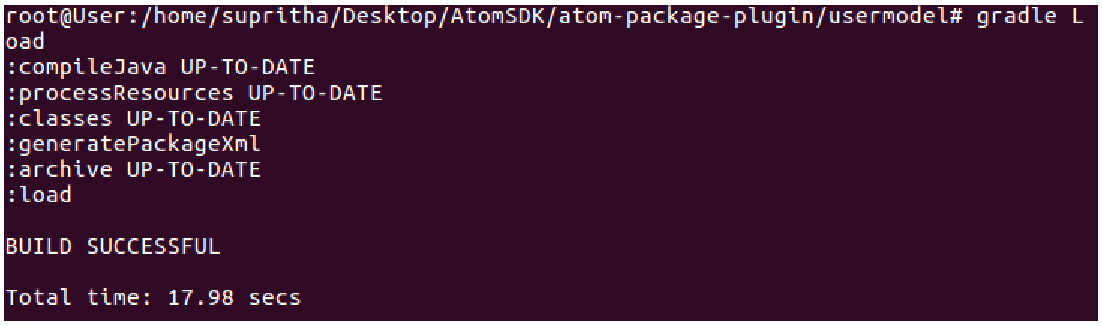


```
Juniper19OpenconfigDriver
l3vpn
__init__.py
__init__.pyc
l3vpn_openconfig_lib.py
l3vpn_openconfig_lib.pyc
l3vpn_openconfig_mapstoexpr.py
l3vpn_openconfig_mapstoexpr.pyc
L3Vpn_Openconfig.py
L3Vpn_Openconfig.pyc
plugin.py
plugin.pyc
```

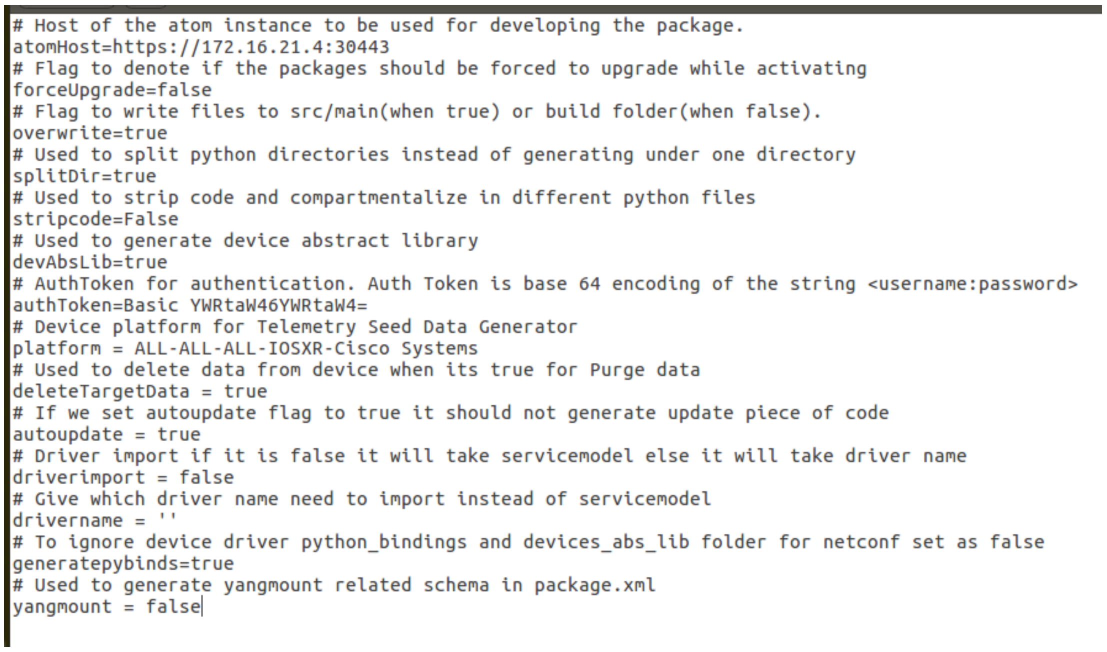
Load

This task is used to upload a package to an ATOM instance.

Before uploading the package to ATOM, make sure the parameter values in the *gradle.properties* file are as per ATOM instance you use.



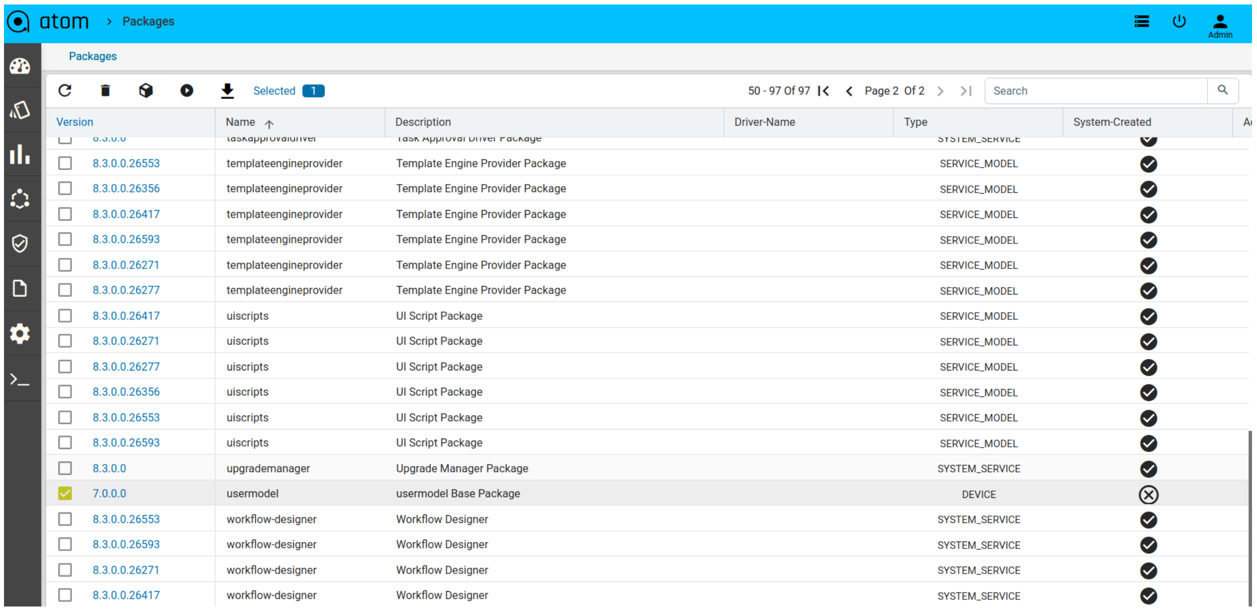
```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel# gradle L
oad
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:generatePackageXml
:archive UP-TO-DATE
:load

BUILD SUCCESSFUL

Total time: 17.98 secs
```



```
# Host of the atom instance to be used for developing the package.
atomHost=https://172.16.21.4:30443
# Flag to denote if the packages should be forced to upgrade while activating
forceUpgrade=false
# Flag to write files to src/main(when true) or build folder(when false).
overwrite=true
# Used to split python directories instead of generating under one directory
splitDir=true
# Used to strip code and compartmentalize in different python files
stripcode=False
# Used to generate device abstract library
devAbsLib=true
# AuthToken for authentication. Auth Token is base 64 encoding of the string <username:password>
authToken=Basic YWRtaW46YWRtaW4=
# Device platform for Telemetry Seed Data Generator
platform = ALL-ALL-ALL-IOSXR-Cisco Systems
# Used to delete data from device when its true for Purge data
deleteTargetData = true
# If we set autoupdate flag to true it should not generate update piece of code
autoupdate = true
# Driver import if it is false it will take servicemodel else it will take driver name
driverimport = false
# Give which driver name need to import instead of servicemodel
drivername = ''
# To ignore device driver python_bindings and devices_abs_lib folder for netconf set as false
generatepybinds=true
# Used to generate yangmount related schema in package.xml
yangmount = false
```

1. **atomHost** property should be provided with a valid IP address.
   If a port needs to be included for accessing applications. Then include that as well. E.g: https://172.16.16.177:30443
2. **authToken** should be provided.
   AuthToken is used for basic authorization. The format of the authtoken is, the keyword **Basic** followed by base64 encoding of the string <username>:<password>", as shown above. These are the default values that can be updated as required.

Upon Load gradle task being successful, the desired changes can be observed in the ATOM UI.

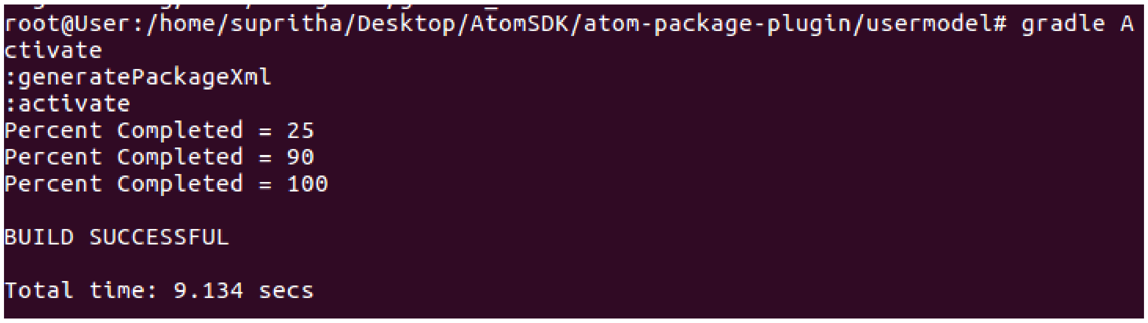


	Version	Name	Description	Driver-Name	Type	System-Created
☐	8.3.0.0	taskapprovaldriver	Task Approval Driver Package		SYSTEM_SERVICE	✓
☐	8.3.0.0.26553	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓
☐	8.3.0.0.26356	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓
☐	8.3.0.0.26417	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓
☐	8.3.0.0.26593	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓
☐	8.3.0.0.26271	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓
☐	8.3.0.0.26277	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓
☐	8.3.0.0.26417	uiscripts	UI Script Package		SERVICE_MODEL	✓
☐	8.3.0.0.26271	uiscripts	UI Script Package		SERVICE_MODEL	✓
☐	8.3.0.0.26277	uiscripts	UI Script Package		SERVICE_MODEL	✓
☐	8.3.0.0.26356	uiscripts	UI Script Package		SERVICE_MODEL	✓
☐	8.3.0.0.26553	uiscripts	UI Script Package		SERVICE_MODEL	✓
☐	8.3.0.0.26593	uiscripts	UI Script Package		SERVICE_MODEL	✓
☐	8.3.0.0	upgrademanager	Upgrade Manager Package		SYSTEM_SERVICE	✓
☑	7.0.0.0	usermodel	usermodel Base Package		DEVICE	⊗
☐	8.3.0.0.26553	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓
☐	8.3.0.0.26593	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓
☐	8.3.0.0.26271	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓
☐	8.3.0.0.26417	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓

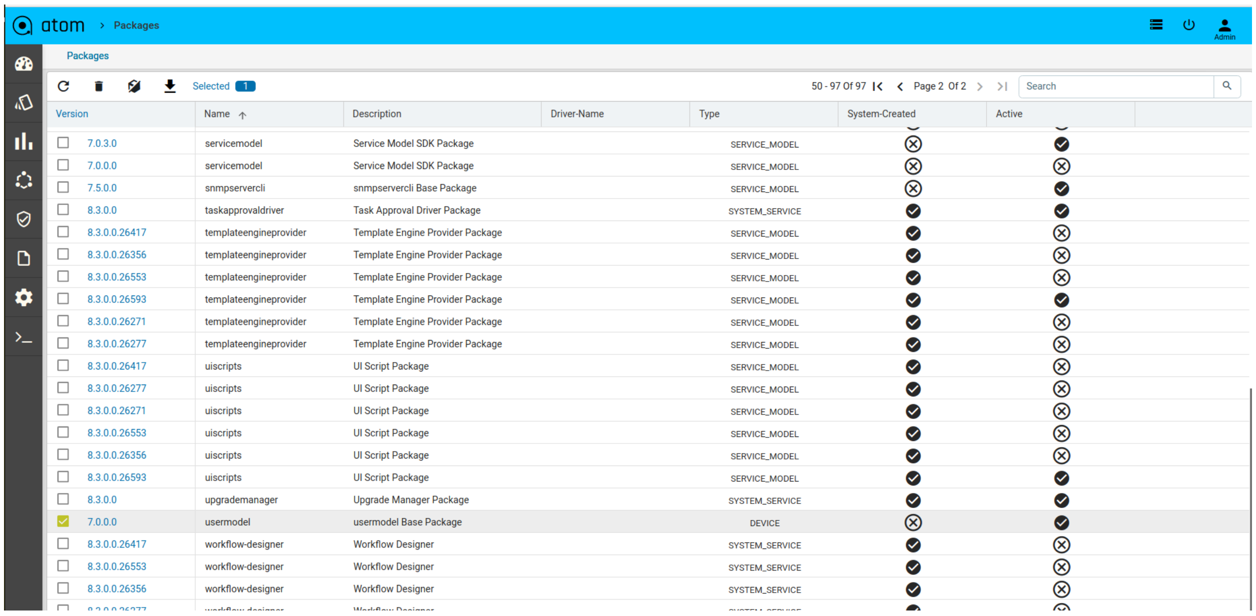
Activate

This task is used to activate a package available in an ATOM instance. To activate a package, ensure the *gradle.properties* file has the aforementioned properties (mentioned in **load** task).

This task changes the active flag of a package to ‘true’ in ATOM. Upon success, the desired changes can be observed in the ATOM UI.



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel# gradle A
ctivate
:generatePackageXml
:activate
Percent Completed = 25
Percent Completed = 90
Percent Completed = 100

BUILD SUCCESSFUL

Total time: 9.134 secs
```

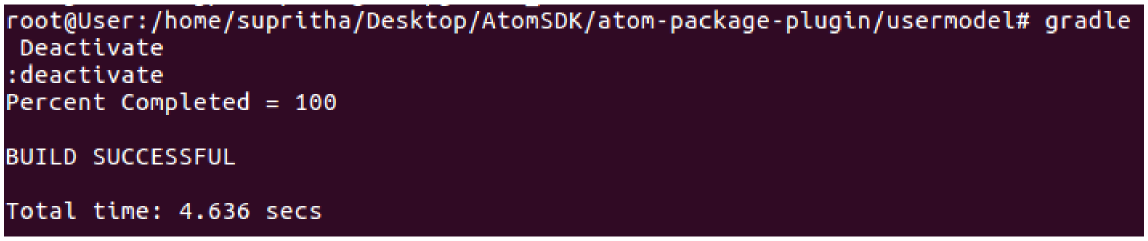


	Version	Name	Description	Driver-Name	Type	System-Created	Active
☐	7.0.3.0	servicemodel	Service Model SDK Package		SERVICE_MODEL	⊗	✓
☐	7.0.0.0	servicemodel	Service Model SDK Package		SERVICE_MODEL	⊗	⊗
☐	7.5.0.0	snmpservercli	snmpservercli Base Package		SERVICE_MODEL	⊗	✓
☐	8.3.0.0	taskapprovaldriver	Task Approval Driver Package		SYSTEM_SERVICE	✓	✓
☐	8.3.0.0.26417	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26356	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26553	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26593	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	✓
☐	8.3.0.0.26271	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26277	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26417	uiscripts	UI Script Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26277	uiscripts	UI Script Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26271	uiscripts	UI Script Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26553	uiscripts	UI Script Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26356	uiscripts	UI Script Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26593	uiscripts	UI Script Package		SERVICE_MODEL	✓	✓
☐	8.3.0.0	upgrademanager	Upgrade Manager Package		SYSTEM_SERVICE	✓	✓
☑	7.0.0.0	usermodel	usermodel Base Package		DEVICE	⊗	✓
☐	8.3.0.0.26417	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗
☐	8.3.0.0.26553	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗
☐	8.3.0.0.26356	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗

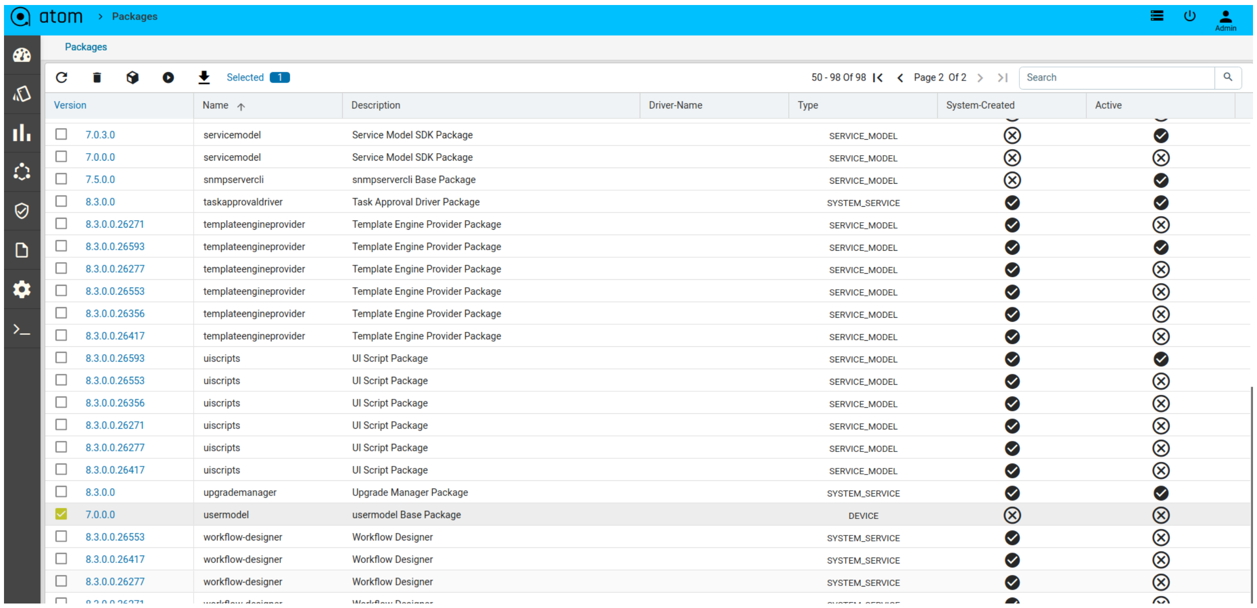
Deactivate

This task is used to deactivate a package loaded in an ATOM instance. To deactivate a package ensure that the *gradle.properties* file has the aforementioned properties (mentioned in **load** task). This task changes the active flag of a package to ‘false’.

Upon success, the desired changes can be observed in the ATOM UI.



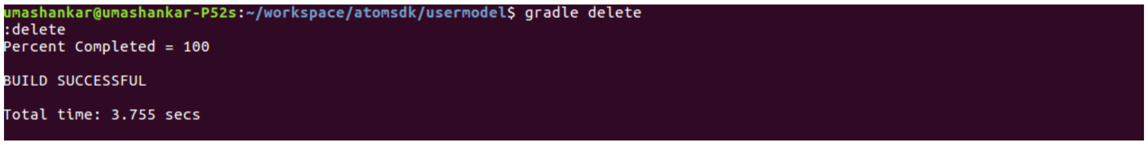
```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel# gradle
 Deactivate
:deactivate
Percent Completed = 100

BUILD SUCCESSFUL

Total time: 4.636 secs
```



	Version	Name	Description	Driver-Name	Type	System-Created	Active
☐	7.0.3.0	servicemodel	Service Model SDK Package		SERVICE_MODEL	⊗	✓
☐	7.0.0.0	servicemodel	Service Model SDK Package		SERVICE_MODEL	⊗	✓
☐	7.5.0.0	snmpservercli	snmpservercli Base Package		SERVICE_MODEL	⊗	✓
☐	8.3.0.0	taskapprovaldriver	Task Approval Driver Package		SYSTEM_SERVICE	✓	✓
☐	8.3.0.0.26271	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26593	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	✓
☐	8.3.0.0.26277	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26553	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26356	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26417	templateengineprovider	Template Engine Provider Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26593	uiscripts	UI Script Package		SERVICE_MODEL	✓	✓
☐	8.3.0.0.26553	uiscripts	UI Script Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26356	uiscripts	UI Script Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26271	uiscripts	UI Script Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26277	uiscripts	UI Script Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0.26417	uiscripts	UI Script Package		SERVICE_MODEL	✓	⊗
☐	8.3.0.0	upgrademanager	Upgrade Manager Package		SYSTEM_SERVICE	✓	✓
☑	7.0.0.0	usermodel	usermodel Base Package		DEVICE	⊗	⊗
☐	8.3.0.0.26553	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗
☐	8.3.0.0.26417	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗
☐	8.3.0.0.26277	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗

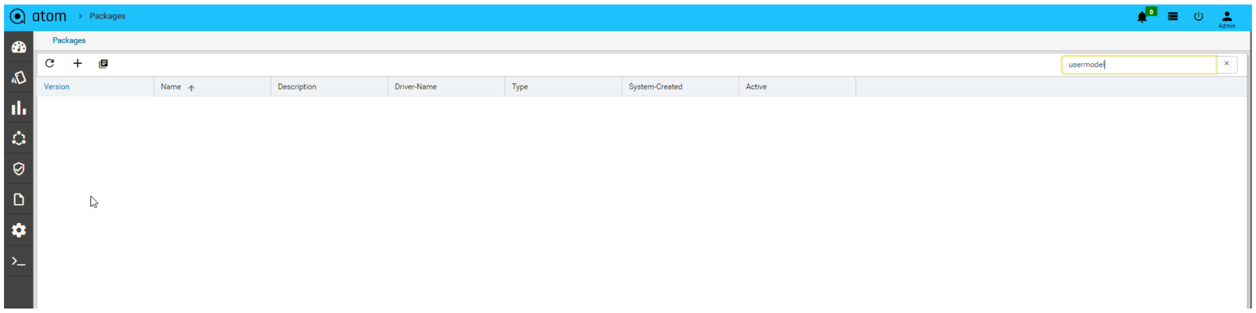
Delete

This task is used to delete a package from an ATOM instance.



```
umashankar@umashankar-P52s:~/workspace/atomsdk/usermodel$ gradle delete
:delete
Percent Completed = 100

BUILD SUCCESSFUL

Total time: 3.755 secs
```

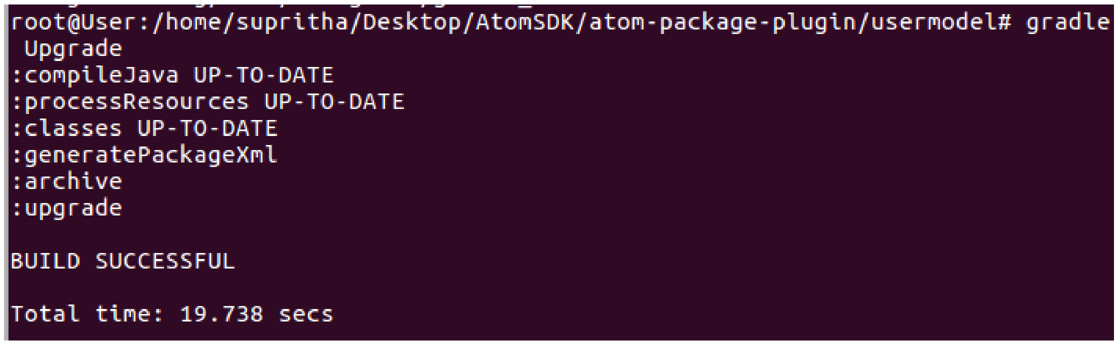
Upon success, the desired changes can be observed in the ATOM UI.



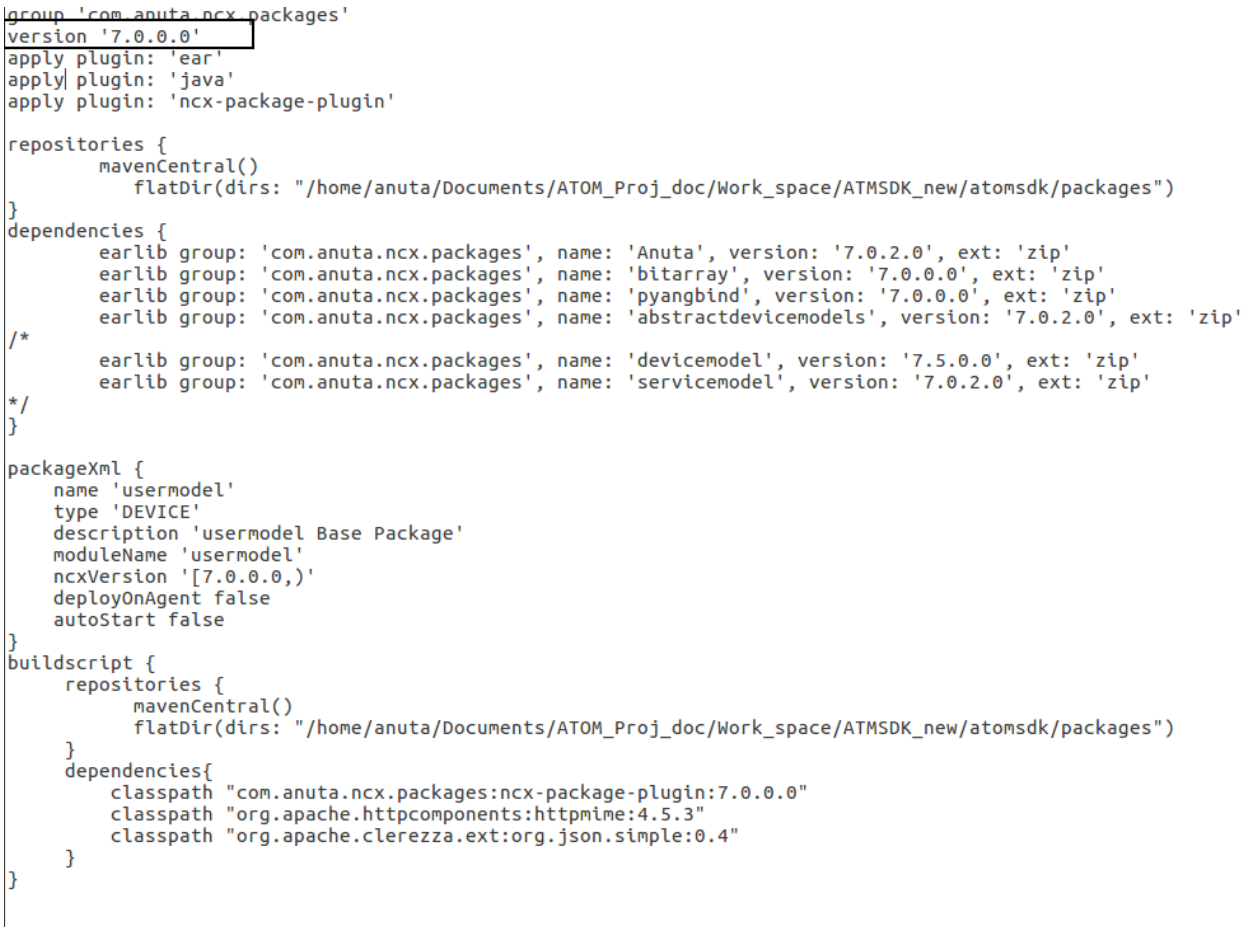
Upgrade

This task is used to upgrade an already existing package present in an ATOM instance. To upgrade a package ensure that the *gradle.properties* file has the properties mentioned in the **load** task. The 'to be upgraded' package should have a different version than the existing package in ATOM.

The version of the package can be changed in the *build.gradle* file as shown below.

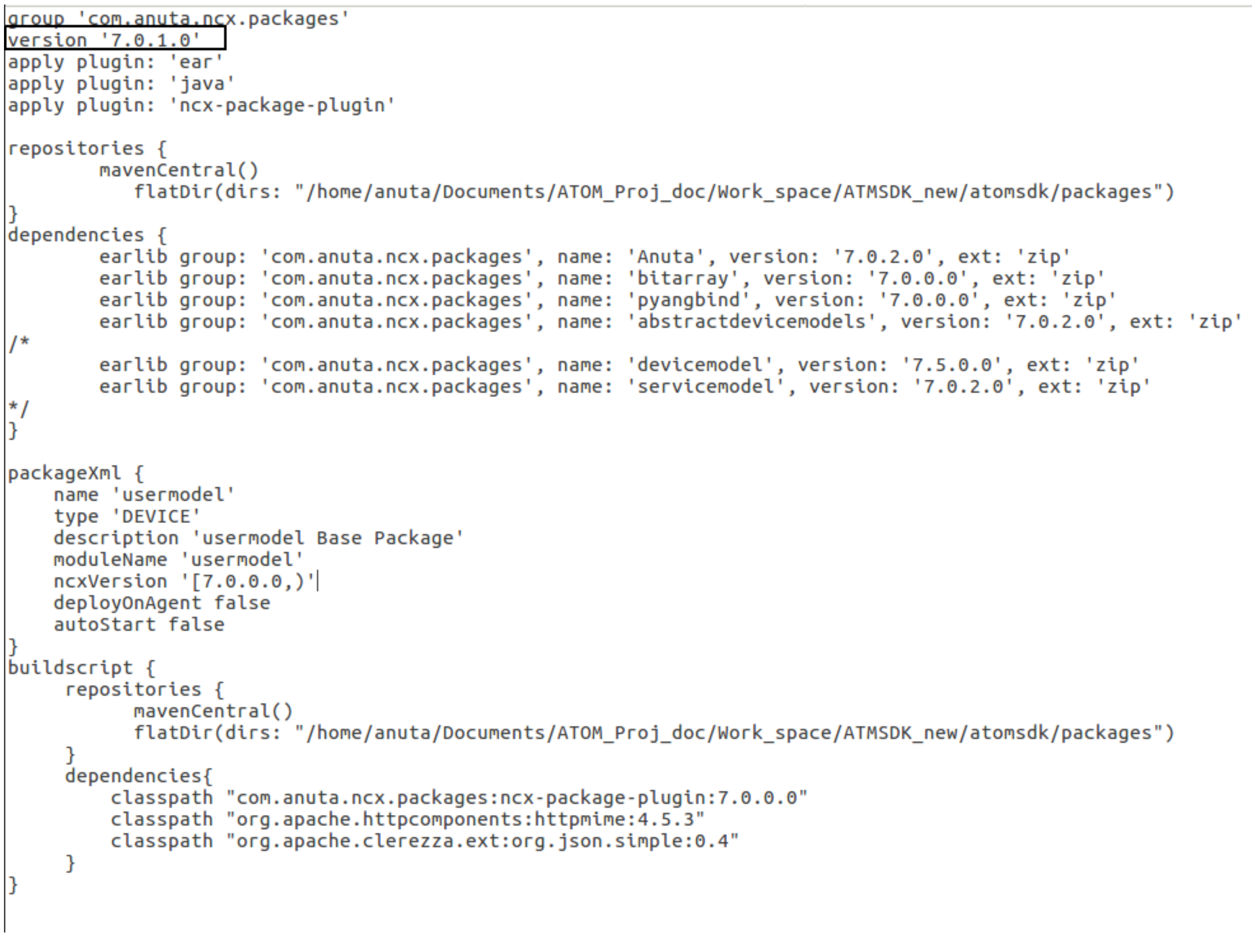


```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel# gradle
 Upgrade
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:generatePackageXml
:archive
:upgrade

BUILD SUCCESSFUL

Total time: 19.738 secs
```



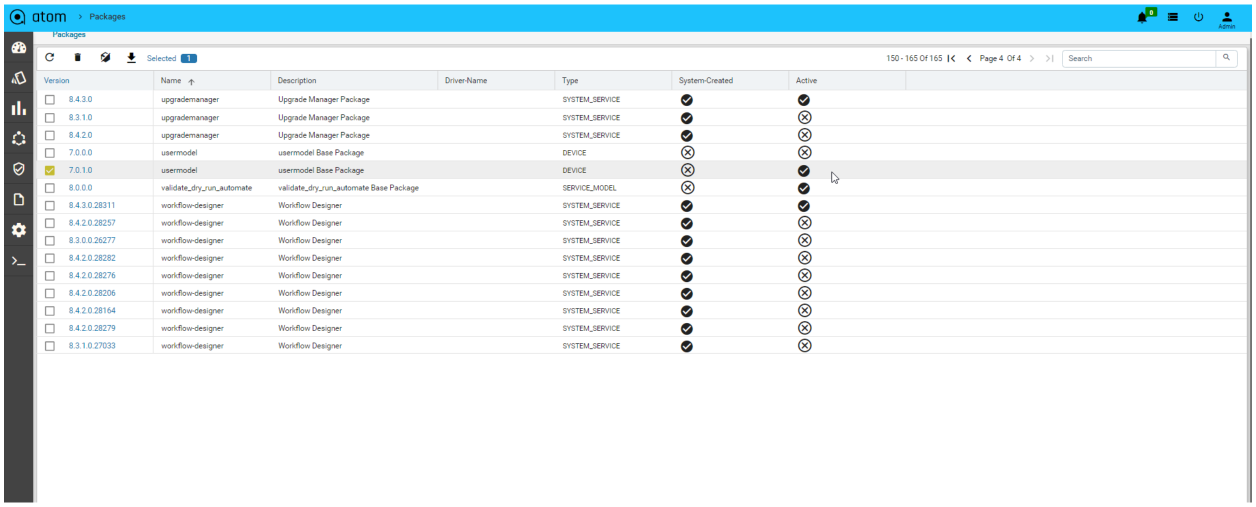
```
group 'com.anuta.ncx.packages'
version '7.0.0.0'
apply plugin: 'ear'
apply plugin: 'java'
apply plugin: 'ncx-package-plugin'

repositories {
        mavenCentral()
           flatDir(dirs: "/home/anuta/Documents/ATOM_Proj_doc/Work_space/ATMSDK_new/atomsdk/packages")
}
dependencies {
        earlib group: 'com.anuta.ncx.packages', name: 'Anuta', version: '7.0.2.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'bitarray', version: '7.0.0.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'pyangbind', version: '7.0.0.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'abstractdevicemodels', version: '7.0.2.0', ext: 'zip'
/*
        earlib group: 'com.anuta.ncx.packages', name: 'devicemodel', version: '7.5.0.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'servicemodel', version: '7.0.2.0', ext: 'zip'
*/
}

packageXml {
    name 'usermodel'
    type 'DEVICE'
    description 'usermodel Base Package'
    moduleName 'usermodel'
    ncxVersion '[7.0.0.0,)'
    deployOnAgent false
    autoStart false
}
buildscript {
     repositories {
           mavenCentral()
           flatDir(dirs: "/home/anuta/Documents/ATOM_Proj_doc/Work_space/ATMSDK_new/atomsdk/packages")
     }
     dependencies{
         classpath "com.anuta.ncx.packages:ncx-package-plugin:7.0.0.0"
         classpath "org.apache.httpcomponents:httpmime:4.5.3"
         classpath "org.apache.clerezza.ext:org.json.simple:0.4"
     }
}
```

Version being changed to ‘*7.0.1.0’* from ‘*7.0.0.0’*



```
group 'com.anuta.ncx.packages'
version '7.0.1.0'
apply plugin: 'ear'
apply plugin: 'java'
apply plugin: 'ncx-package-plugin'

repositories {
        mavenCentral()
           flatDir(dirs: "/home/anuta/Documents/ATOM_Proj_doc/Work_space/ATMSDK_new/atomsdk/packages")
}
dependencies {
        earlib group: 'com.anuta.ncx.packages', name: 'Anuta', version: '7.0.2.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'bitarray', version: '7.0.0.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'pyangbind', version: '7.0.0.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'abstractdevicemodels', version: '7.0.2.0', ext: 'zip'
/*
        earlib group: 'com.anuta.ncx.packages', name: 'devicemodel', version: '7.5.0.0', ext: 'zip'
        earlib group: 'com.anuta.ncx.packages', name: 'servicemodel', version: '7.0.2.0', ext: 'zip'
*/
}

packageXml {
    name 'usermodel'
    type 'DEVICE'
    description 'usermodel Base Package'
    moduleName 'usermodel'
    ncxVersion '[7.0.0.0,)'
    deployOnAgent false
    autoStart false
}
buildscript {
     repositories {
           mavenCentral()
           flatDir(dirs: "/home/anuta/Documents/ATOM_Proj_doc/Work_space/ATMSDK_new/atomsdk/packages")
     }
     dependencies{
         classpath "com.anuta.ncx.packages:ncx-package-plugin:7.0.0.0"
         classpath "org.apache.httpcomponents:httpmime:4.5.3"
         classpath "org.apache.clerezza.ext:org.json.simple:0.4"
     }
}
```

On completion of the upgrade task, the latest version of the package is updated to 'Active: true' and the older version of the same package is set to 'Active:False' as shown below:



150 - 165 Of 165 Page 4 Of 4

Version	Name	Description	Driver-Name	Type	System-Created	Active
8.4.3.0	upgrademanager	Upgrade Manager Package		SYSTEM_SERVICE	✓	✓
8.3.1.0	upgrademanager	Upgrade Manager Package		SYSTEM_SERVICE	✓	⊗
8.4.2.0	upgrademanager	Upgrade Manager Package		SYSTEM_SERVICE	✓	⊗
7.0.0.0	usermodel	usermodel Base Package		DEVICE	⊗	⊗
7.0.1.0	usermodel	usermodel Base Package		DEVICE	⊗	✓
8.0.0.0	validate_dry_run_automate	validate_dry_run_automate Base Package		SERVICE_MODEL	⊗	✓
8.4.3.0.28311	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	✓
8.4.2.0.28257	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗
8.3.0.0.26277	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗
8.4.2.0.28282	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗
8.4.2.0.28276	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗
8.4.2.0.28206	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗
8.4.2.0.28164	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗
8.4.2.0.28279	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗
8.3.1.0.27033	workflow-designer	Workflow Designer		SYSTEM_SERVICE	✓	⊗

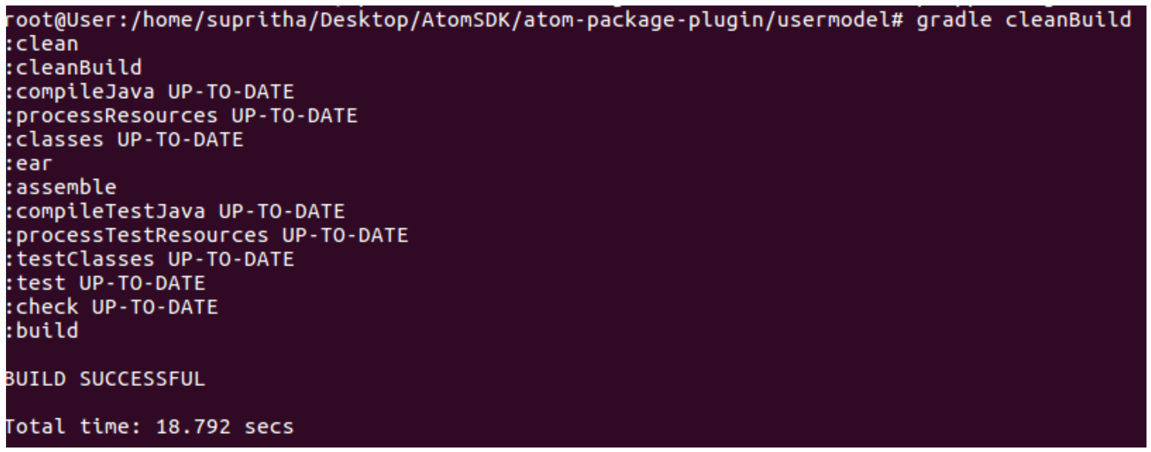
cleanBuild

This task is a combination of two gradle tasks, 'clean' and 'build --refresh-dependencies'. This task first executes *gradle clean.* The clean task is defined by the java plugin and it removes the **buildDir** folder, thus cleaning previous builds' artifacts, which are no longer relevant.

After cleaning, this task runs ***gradle build --refresh-dependencies***.

The --refresh-dependencies option - Enables Gradle to ignore all cached entries for resolved modules and artifacts. A fresh resolve will be performed against all configured repositories, with dynamic versions recalculated, modules refreshed, and artifacts downloaded.

To run this task, enter: **gradle cleanBuild**



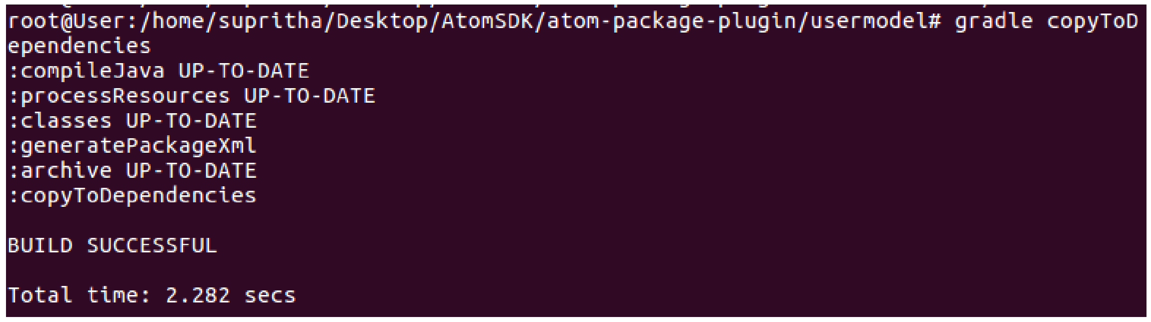
```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel# gradle cleanBuild
:clean
:cleanBuild
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:ear
:assemble
:compileTestJava UP-TO-DATE
:processTestResources UP-TO-DATE
:testClasses UP-TO-DATE
:test UP-TO-DATE
:check UP-TO-DATE
:build

BUILD SUCCESSFUL

Total time: 18.792 secs
```

copyToDependencies

This task internally runs the archive task (which generates the ready-to-be-uploadable zip). After generating the zip, it ascertains the repository type (whether maven/local) from *config.xml* file and the repository path.

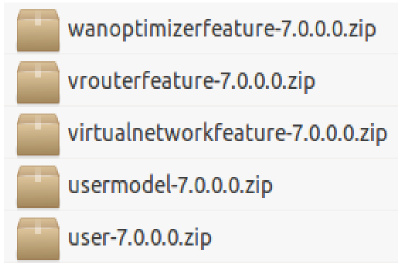
Based on the information gathered, it either publishes the artifact repository in case the repository is maven or copies the dependencies to the local dependencies directory in case the repository is flat Directory.

To run this task, enter: **gradle copyToDependencies**



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel# gradle copyToD
ependencies
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:generatePackageXml
:archive UP-TO-DATE
:copyToDependencies

BUILD SUCCESSFUL

Total time: 2.282 secs
```

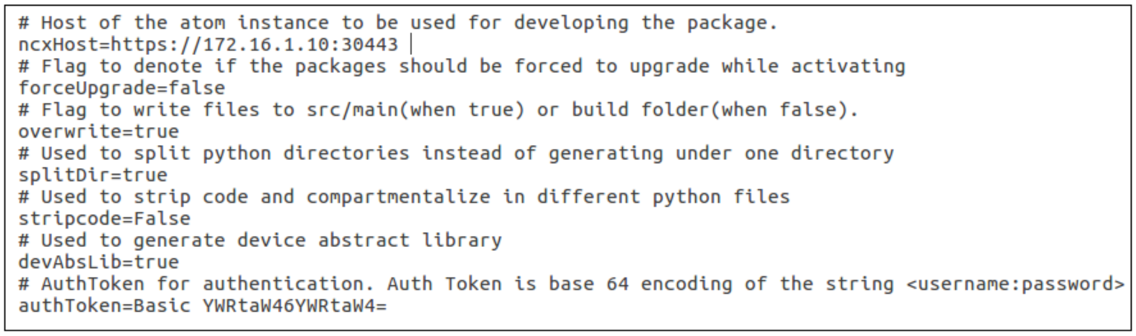
Above task copied from usermodel zip into package dependency folder so that usermodel package can serve as a dependency for any other package development.



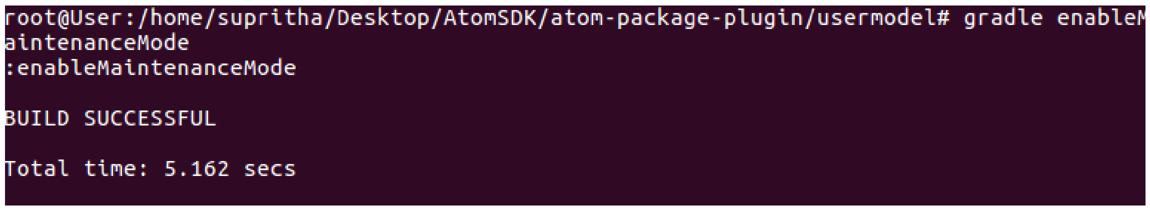
wanoptimizerfeature-7.0.0.0.zip
vrouterfeature-7.0.0.0.zip
virtualnetworkfeature-7.0.0.0.zip
usermodel-7.0.0.0.zip
user-7.0.0.0.zip

enableMaintenanceMode

This task is used to enable maintenance mode on an ATOM instance. To run this task make sure all the necessary modifications are made in *gradle.properties* file, that have been explained in the *load* task.

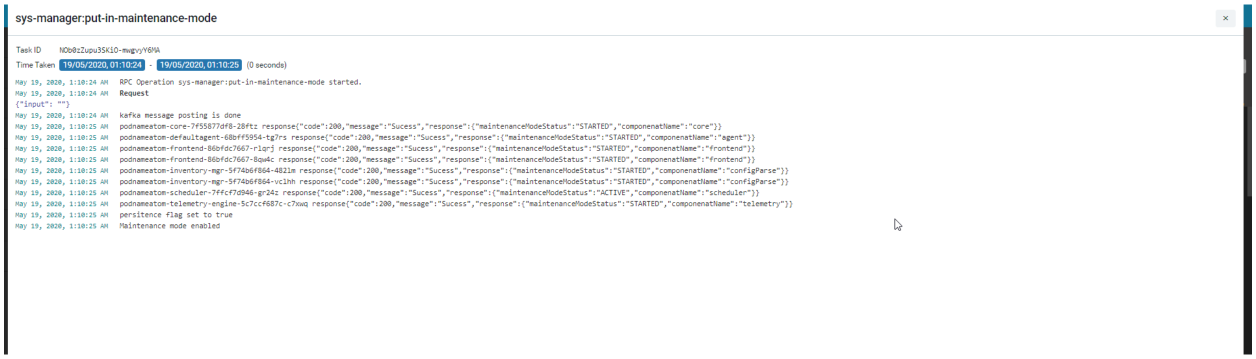


```
# Host of the atom instance to be used for developing the package.
ncxHost=https://172.16.1.10:30443
# Flag to denote if the packages should be forced to upgrade while activating
forceUpgrade=false
# Flag to write files to src/main(when true) or build folder(when false).
overwrite=true
# Used to split python directories instead of generating under one directory
splitDir=true
# Used to strip code and compartmentalize in different python files
stripcode=False
# Used to generate device abstract library
devAbsLib=true
# AuthToken for authentication. Auth Token is base 64 encoding of the string <username:password>
authToken=Basic YWRtaW46YWRtaW4=
```



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel# gradle enableM
aintenanceMode
:enableMaintenanceMode

BUILD SUCCESSFUL

Total time: 5.162 secs
```

Upon success, the desired changes can be observed in the ATOM UI.



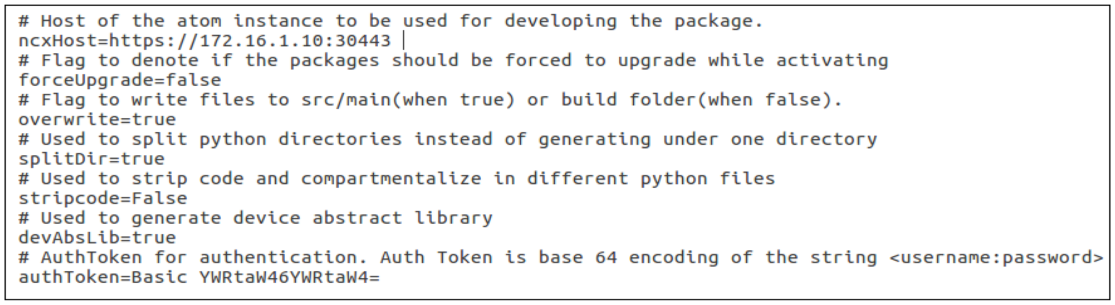
sys-manager:put-in-maintenance-mode

Task ID NOb0zZupu3SKiO-mwgvyY6MA
Time Taken 19/05/2020, 01:10:24 - 19/05/2020, 01:10:25 (0 seconds)

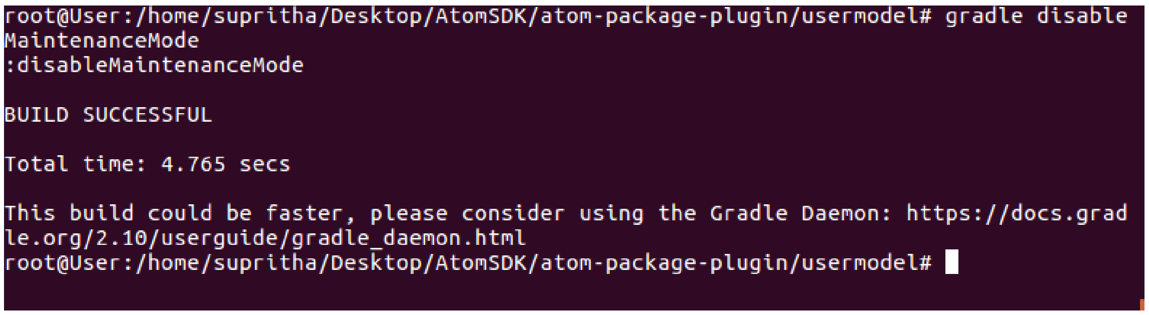
```
May 19, 2020, 1:10:24 AM   RPC Operation sys-manager:put-in-maintenance-mode started.
May 19, 2020, 1:10:24 AM   Request
{"input": ""}
May 19, 2020, 1:10:24 AM   kafka message posting is done
May 19, 2020, 1:10:25 AM   podnameatom-core-7f55877df8-28ftz response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"STARTED","componenatName":"core"}}
May 19, 2020, 1:10:25 AM   podnameatom-defaultagent-68bff5954-tg7rs response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"STARTED","componenatName":"agent"}}
May 19, 2020, 1:10:25 AM   podnameatom-frontend-86bfdc7667-rlqrj response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"STARTED","componenatName":"frontend"}}
May 19, 2020, 1:10:25 AM   podnameatom-frontend-86bfdc7667-8qw4c response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"STARTED","componenatName":"frontend"}}
May 19, 2020, 1:10:25 AM   podnameatom-inventory-mgr-5f74b6f864-482lm response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"STARTED","componenatName":"configParse"}}
May 19, 2020, 1:10:25 AM   podnameatom-inventory-mgr-5f74b6f864-vclhh response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"STARTED","componenatName":"configParse"}}
May 19, 2020, 1:10:25 AM   podnameatom-scheduler-7ffcf7d946-gr24z response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"ACTIVE","componenatName":"scheduler"}}
May 19, 2020, 1:10:25 AM   podnameatom-telemetry-engine-5c7ccf687c-c7xwq response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"STARTED","componenatName":"telemetry"}}
May 19, 2020, 1:10:25 AM   persitence flag set to true
May 19, 2020, 1:10:25 AM   Maintenance mode enabled
```

disableMaintenanceMode

This task is used to disable maintenance mode on an ATOM instance. To run this task , ensure that all the necessary modifications are made in *gradle.properties* file, explained in the *load* task.

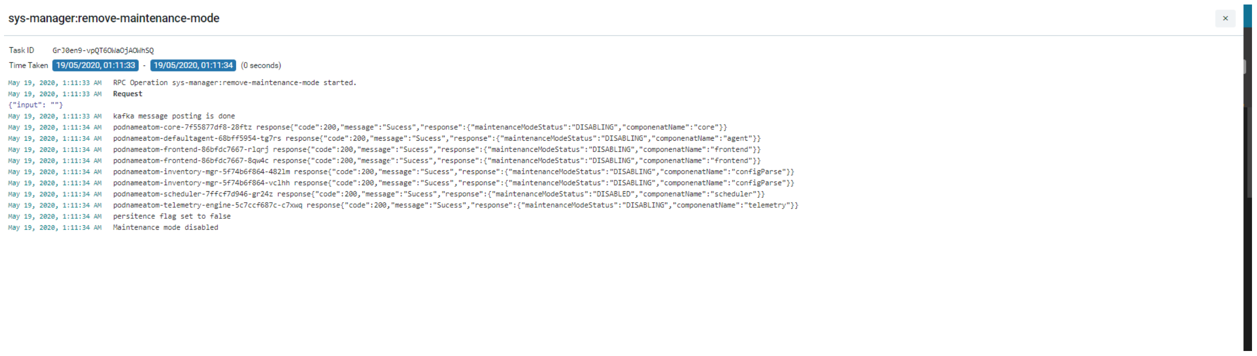


```
# Host of the atom instance to be used for developing the package.
ncxHost=https://172.16.1.10:30443
# Flag to denote if the packages should be forced to upgrade while activating
forceUpgrade=false
# Flag to write files to src/main(when true) or build folder(when false).
overwrite=true
# Used to split python directories instead of generating under one directory
splitDir=true
# Used to strip code and compartmentalize in different python files
stripcode=False
# Used to generate device abstract library
devAbsLib=true
# AuthToken for authentication. Auth Token is base 64 encoding of the string <username:password>
authToken=Basic YWRtaW46YWRtaW4=
```



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel# gradle disable
MaintenanceMode
:disableMaintenanceMode

BUILD SUCCESSFUL

Total time: 4.765 secs

This build could be faster, please consider using the Gradle Daemon: https://docs.grad
le.org/2.10/userguide/gradle_daemon.html
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel#
```

Upon success, the desired changes can be observed in the ATOM UI.



sys-manager:remove-maintenance-mode

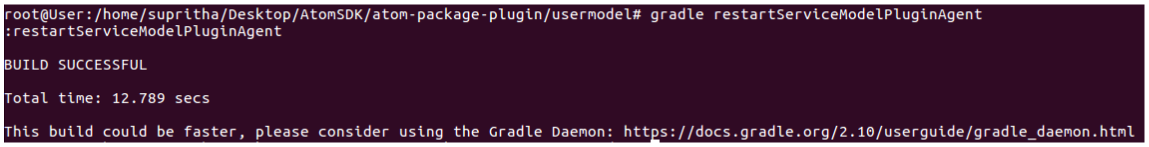
Task ID GrJ0en9-vpQT6OWaOjAOWhSQ

Time Taken 19/05/2020, 01:11:33 - 19/05/2020, 01:11:34 (0 seconds)

```
May 19, 2020, 1:11:33 AM   RPC Operation sys-manager:remove-maintenance-mode started.
May 19, 2020, 1:11:33 AM   Request
{"input": ""}
May 19, 2020, 1:11:33 AM   kafka message posting is done
May 19, 2020, 1:11:34 AM   podnameatom-core-7f55877df8-28ftz response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"DISABLING","componenatName":"core"}}
May 19, 2020, 1:11:34 AM   podnameatom-defaultagent-68bff5954-tg7rs response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"DISABLING","componenatName":"agent"}}
May 19, 2020, 1:11:34 AM   podnameatom-frontend-86bfdc7667-rlqrj response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"DISABLING","componenatName":"frontend"}}
May 19, 2020, 1:11:34 AM   podnameatom-frontend-86bfdc7667-8qw4c response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"DISABLING","componenatName":"frontend"}}
May 19, 2020, 1:11:34 AM   podnameatom-inventory-mgr-5f74b6f864-482lm response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"DISABLING","componenatName":"configParse"}}
May 19, 2020, 1:11:34 AM   podnameatom-inventory-mgr-5f74b6f864-vclhh response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"DISABLING","componenatName":"configParse"}}
May 19, 2020, 1:11:34 AM   podnameatom-scheduler-7ffcf7d946-gr24z response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"DISABLED","componenatName":"scheduler"}}
May 19, 2020, 1:11:34 AM   podnameatom-telemetry-engine-5c7ccf687c-c7xwq response{"code":200,"message":"Sucess","response":{"maintenanceModeStatus":"DISABLING","componenatName":"telemetry"}}
May 19, 2020, 1:11:34 AM   persitence flag set to false
May 19, 2020, 1:11:34 AM   Maintenance mode disabled
```

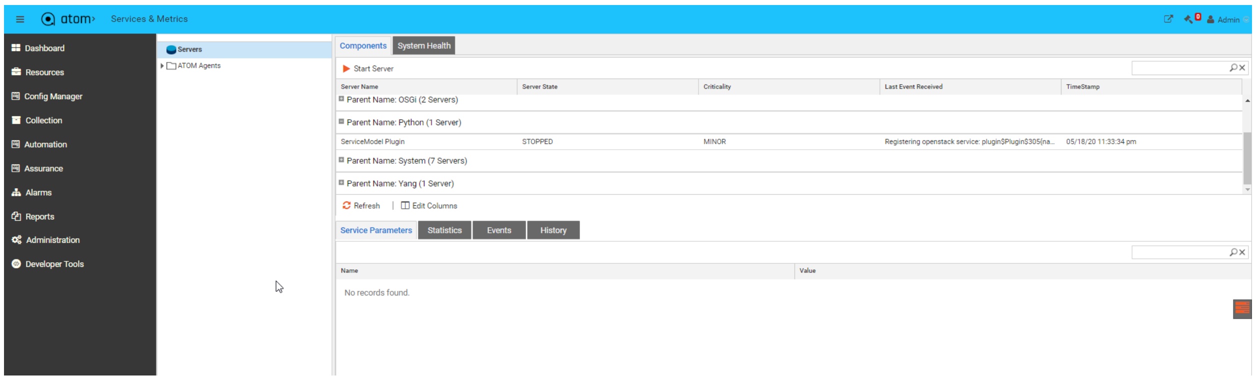
restartServiceModelPluginAgent

This gradle task is used to stop and restart service model plugin for the agent.



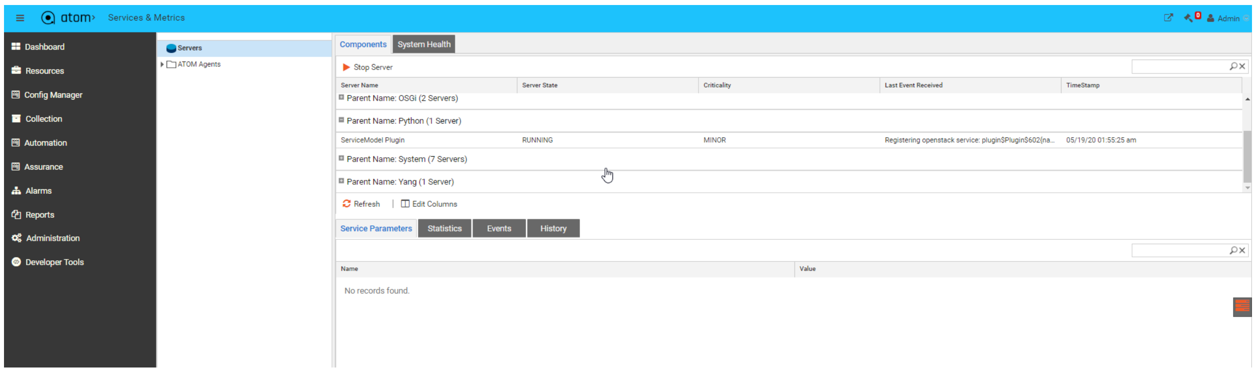
```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel# gradle restartServiceModelPluginAgent
:restartServiceModelPluginAgent

BUILD SUCCESSFUL

Total time: 12.789 secs

This build could be faster, please consider using the Gradle Daemon: https://docs.gradle.org/2.10/userguide/gradle_daemon.html
```

During the task execution, the desired changes can be observed in the ATOM UI.



Server Name	Server State	Criticality	Last Event Received	TimeStamp
Parent Name: OSGi (2 Servers)				
Parent Name: Python (1 Server)				
ServiceModel Plugin	STOPPED	MINOR	Registering openstack service: plugin$Plugin$305(na...	05/18/20 11:33:34 pm
Parent Name: System (7 Servers)				
Parent Name: Yang (1 Server)				

Name	Value
No records found.	

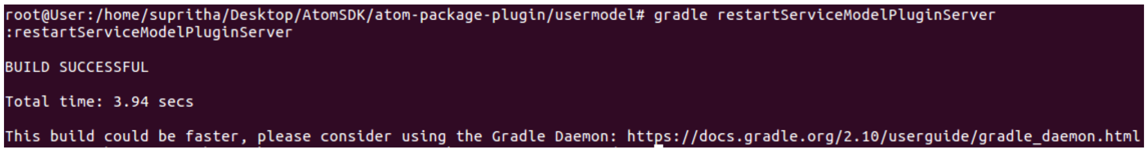


Server Name	Server State	Criticality	Last Event Received	TimeStamp
Parent Name: OSGi (2 Servers)				
Parent Name: Python (1 Server)				
ServiceModel Plugin	RUNNING	MINOR	Registering openstack service: plugin$Plugin$602(na...	05/19/20 01:55:25 am
Parent Name: System (7 Servers)				
Parent Name: Yang (1 Server)				

Name	Value
No records found.	

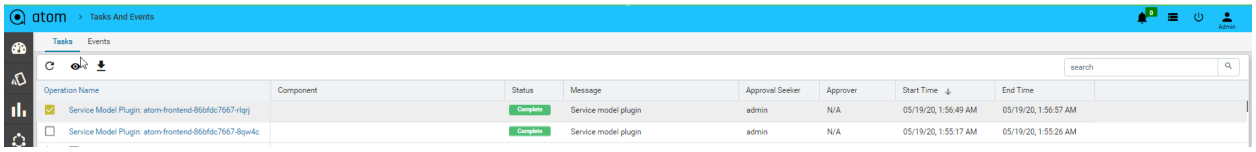
restartServiceModelPluginServer

This gradle task is used to stop and restart service model plugin for server.



```
root@User:/home/supritha/Desktop/AtomSDK/atom-package-plugin/usermodel# gradle restartServiceModelPluginServer
:restartServiceModelPluginServer

BUILD SUCCESSFUL

Total time: 3.94 secs

This build could be faster, please consider using the Gradle Daemon: https://docs.gradle.org/2.10/userguide/gradle_daemon.html
```

Upon success, the desired changes can be observed in the ATOM Tasks UI.



Operation Name	Component	Status	Message	Approval Seeker	Approver	Start Time	End Time
✓ Service Model Plugin: atom-frontend-86bfdc7667-rlqrj		Complete	Service model plugin	admin	N/A	05/19/20, 1:56:49 AM	05/19/20, 1:56:57 AM
Service Model Plugin: atom-frontend-86bfdc7667-8qw4c		Complete	Service model plugin	admin	N/A	05/19/20, 1:55:17 AM	05/19/20, 1:55:26 AM

Running ATOM Package Plugin Tasks

The ATOM Package Plugin gradle tasks available for package development as described in the section, "Tasks for developing packages" can be executed from an IDE (For reference, Intellij is used as an IDE) or directly from the CLI as described below.

Running Tasks in IDE

1. Install Intellij from https://www.jetbrains.com/idea/download/
2. Select **File** > **New** > **Project** from existing sources.
3. To import the ATOM-package plugin into IntelliJ, select the **build.gradle** file from the package being developed as discussed in Setting up ATOM Package Plugin Environment.
4. Select **View** > **Tool windows** > **gradle**

   Gradle support is displayed on the right-hand side.

5. Select the "Tasks" list. Execute the following operations:

   a) build > clean

   b) build

6. Run the ATOM plugin tasks based on your requirements as discussed in ATOM specific tasks

   Example: gradle generateYin

Running Tasks in CLI

1. Follow the procedure as described in the section, "Setting up the repository for package development", locate the new package being developed and execute the following commands:

```
gradlew build –refresh-dependencies (Windows)
./gradlew build –refresh-dependencies (Linux)
```

2. To view all the available tasks in gradle, enter the command: - gradle tasks --all
3. To clean the latest build, enter : gradle tasks clean
4. To build the project, enter: gradle tasks build
5. To run a required ATOM plugin task based on your requirement as discussed in ATOM specific tasks, enter : gradle <task-name>

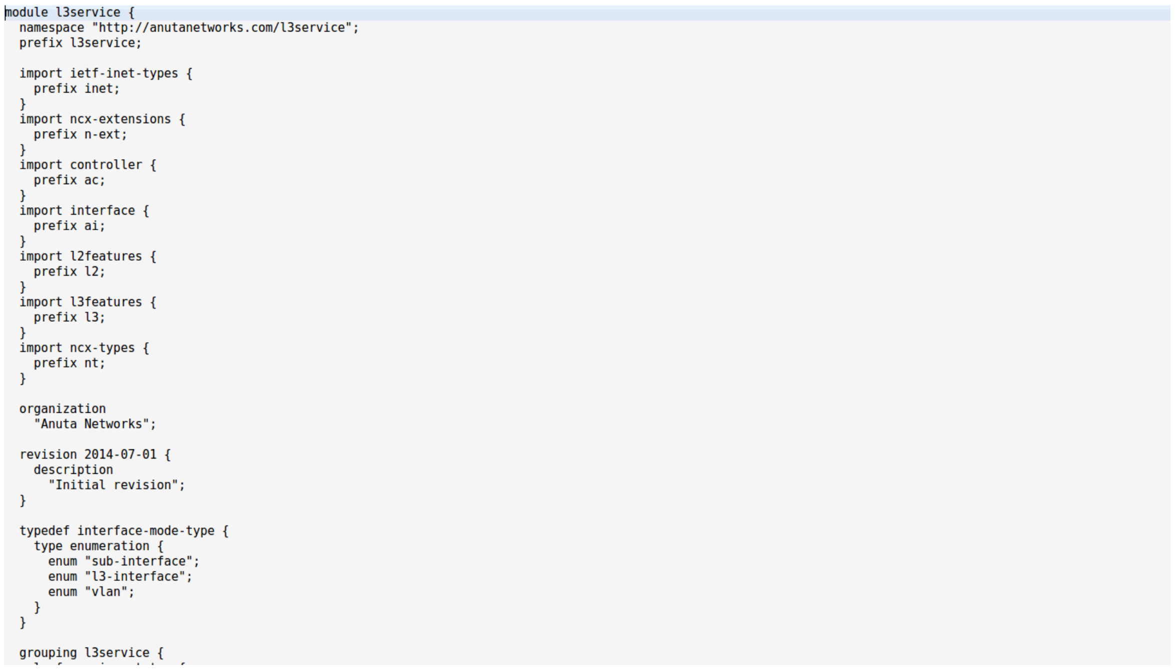
   E.g: gradle genarateYin

Troubleshoot & FAQs - Service Modelling

Errors during package upload into ATOM

Package Dependency Error

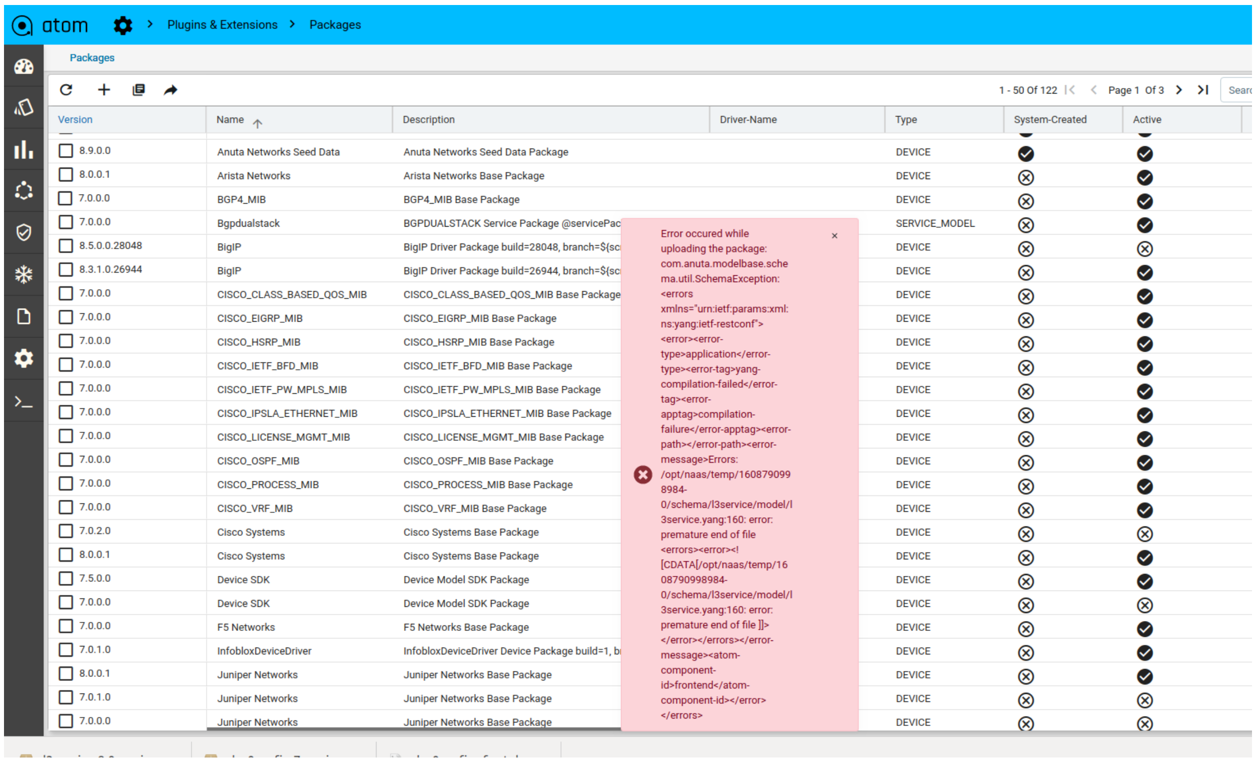
Make sure all the packages mentioned in import statements of your service yang are present in the ATOM already. Those dependency packages if not uploaded then ATOM will throw package dependency Errors. For E.g in below snippet, make sure all import statements yang files are present.



```
module l3service {
  namespace "http://anutanetworks.com/l3service";
  prefix l3service;

  import ietf-inet-types {
    prefix inet;
  }
  import ncx-extensions {
    prefix n-ext;
  }
  import controller {
    prefix ac;
  }
  import interface {
    prefix ai;
  }
  import l2features {
    prefix l2;
  }
  import l3features {
    prefix l3;
  }
  import ncx-types {
    prefix nt;
  }

  organization
    "Anuta Networks";

  revision 2014-07-01 {
    description
      "Initial revision";
  }

  typedef interface-mode-type {
    type enumeration {
      enum "sub-interface";
      enum "l3-interface";
      enum "vlan";
    }
  }

  grouping l3service {
```

Example exception:

```
Upload: l3service:8.0.0.0 -
applicationyang-compilation-failedcompilation-failure/opt/naas/temp/1559629884671-0/schema/model/l3service.yang
Errors:
Failed to convert file l3service.yang
# read /data/naas/DevicePackages/Anuta/model/ncx-ui-component-state.yang
# READ /data/naas/DevicePackages/Anuta/model/ncx-ui-component-state.yang
# read /data/naas/DevicePackages/Anuta/model/interface.yang
# READ /data/naas/DevicePackages/Anuta/model/interface.yang
# read /data/naas/DevicePackages/Anuta/model/if-type.yang
# READ /data/naas/DevicePackages/Anuta/model/if-type.yang
/opt/naas/temp/1559629884671-0/schema/model/l3service.yang:11: error: module "sdk-extensions" not found in search
path
/opt/naas/temp/1559629884671-0/schema/model/l3service.yang:20: error: module "l2features" not found in search path
/opt/naas/temp/1559629884671-0/schema/model/l3service.yang:20: warning: imported module l2features not used
/opt/naas/temp/1559629884671-0/schema/model/l3service.yang:23: error: module "l3features" not found in search path
/opt/naas/temp/1559629884671-0/schema/model/l3service.yang:23: warning: imported module l3features not used
/data/naas/DevicePackages/Anuta/model/ietf-netconf-acm@2012-02-22.yang:105: warning: the escape sequence "\*" is
unsafe in double quoted strings - pass the flag --lax-quote-checks to avoid this warning
/data/naas/DevicePackages/Anuta/model/ietf-netconf-acm@2012-02-22.yang:146: warning: the escape sequence "\*" is
unsafe in double quoted strings - pass the flag --lax-quote-checks to avoid this warning
/opt/naas/temp/1559629884671-0/schema/model/l3service.yang
```

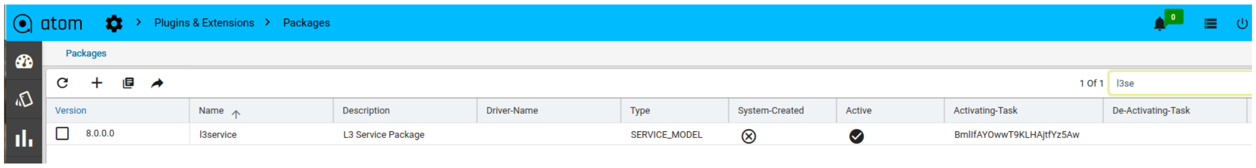


Version	Name	Description	Driver-Name	Type	System-Created	Active
8.9.0.0	Anuta Networks Seed Data	Anuta Networks Seed Data Package		DEVICE	✓	✓
8.0.0.1	Arista Networks	Arista Networks Base Package		DEVICE	⊗	✓
7.0.0.0	BGP4_MIB	BGP4_MIB Base Package		DEVICE	⊗	✓
7.0.0.0	Bgpdualstack	BGPDUALSTACK Service Package @servicePac		SERVICE_MODEL	⊗	✓
8.5.0.0.28048	BigIP	BigIP Driver Package build=28048, branch=${sc		DEVICE	⊗	⊗
8.3.1.0.26944	BigIP	BigIP Driver Package build=26944, branch=${sc		DEVICE	⊗	✓
7.0.0.0	CISCO_CLASS_BASED_QOS_MIB	CISCO_CLASS_BASED_QOS_MIB Base Package		DEVICE	⊗	✓
7.0.0.0	CISCO_EIGRP_MIB	CISCO_EIGRP_MIB Base Package		DEVICE	⊗	✓
7.0.0.0	CISCO_HSRP_MIB	CISCO_HSRP_MIB Base Package		DEVICE	⊗	✓
7.0.0.0	CISCO_IETF_BFD_MIB	CISCO_IETF_BFD_MIB Base Package		DEVICE	⊗	✓
7.0.0.0	CISCO_IETF_PW_MPLS_MIB	CISCO_IETF_PW_MPLS_MIB Base Package		DEVICE	⊗	✓
7.0.0.0	CISCO_IPSLA_ETHERNET_MIB	CISCO_IPSLA_ETHERNET_MIB Base Package		DEVICE	⊗	✓
7.0.0.0	CISCO_LICENSE_MGMT_MIB	CISCO_LICENSE_MGMT_MIB Base Package		DEVICE	⊗	✓
7.0.0.0	CISCO_OSPF_MIB	CISCO_OSPF_MIB Base Package		DEVICE	⊗	✓
7.0.0.0	CISCO_PROCESS_MIB	CISCO_PROCESS_MIB Base Package		DEVICE	⊗	✓
7.0.0.0	CISCO_VRF_MIB	CISCO_VRF_MIB Base Package		DEVICE	⊗	✓
7.0.2.0	Cisco Systems	Cisco Systems Base Package		DEVICE	⊗	⊗
8.0.0.1	Cisco Systems	Cisco Systems Base Package		DEVICE	⊗	✓
7.5.0.0	Device SDK	Device Model SDK Package		DEVICE	⊗	✓
7.0.0.0	Device SDK	Device Model SDK Package		DEVICE	⊗	⊗
7.0.0.0	F5 Networks	F5 Networks Base Package		DEVICE	⊗	✓
7.0.1.0	InfobloxDeviceDriver	InfobloxDeviceDriver Device Package build=1, br		DEVICE	⊗	✓
8.0.0.1	Juniper Networks	Juniper Networks Base Package		DEVICE	⊗	✓
7.0.1.0	Juniper Networks	Juniper Networks Base Package		DEVICE	⊗	⊗
7.0.0.0	Juniper Networks	Juniper Networks Base Package		DEVICE	⊗	⊗

```
Error occured while
uploading the package:
com.anuta.modelbase.sche
ma.util.SchemaException:
<errors
xmlns="urn:ietf:params:xml:
ns:yang:ietf-restconf">
<error><error-
type>application</error-
type><error-tag>yang-
compilation-failed</error-
tag><error-
apptag>compilation-
failure</error-apptag><error-
path></error-path><error-
message>Errors:
/opt/naas/temp/160879099
8984-
0/schema/l3service/model/l
3service.yang:160: error:
premature end of file
<errors><error><!
[CDATA[/opt/naas/temp/16
08790998984-
0/schema/l3service/model/l
3service.yang:160: error:
premature end of file ]]>
</error></errors></error-
message><atom-
component-
id>frontend</atom-
component-id></error>
</errors>
```

Solution:

First **upload** and **load** all dependency packages, after that upload the service package.

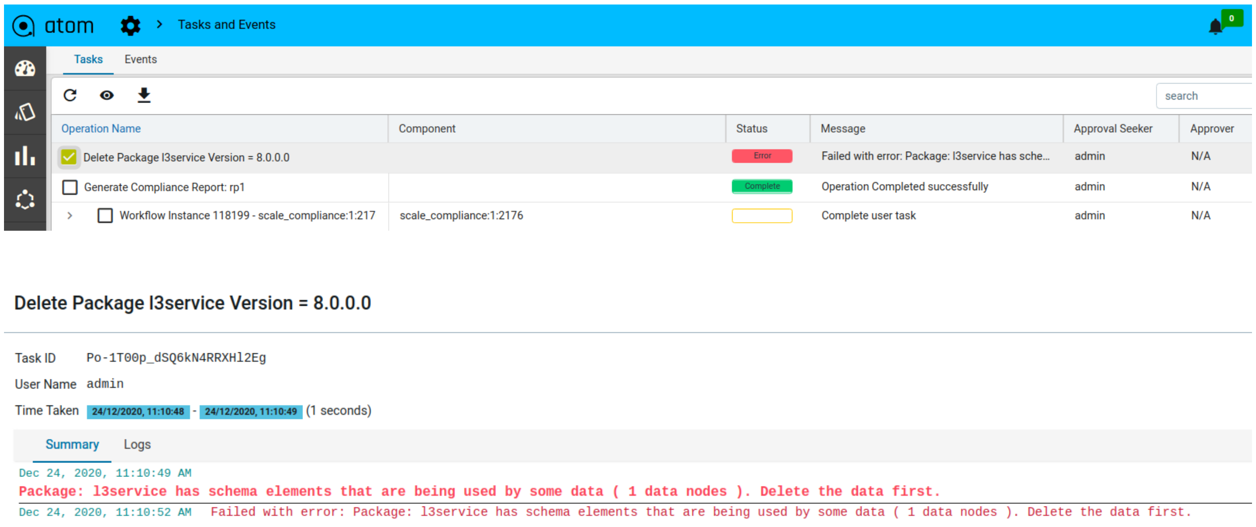


Version	Name	Description	Driver-Name	Type	System-Created	Active	Activating-Task	De-Activating-Task
8.0.0.0	l3service	L3 Service Package		SERVICE_MODEL	⊗	✓	BmlIfAYOwwT9KLHAjtfYz5Aw	

Package Deletion Error

Users trying to delete the package from **Administration > Plugins & Extensions > Packages,** but the services related to this package still exist in the Services tab, then the following exception will be seen.

```
Error: com.anuta.api.DataIntegrityException
```



Operation Name	Component	Status	Message	Approval Seeker	Approver
Delete Package l3service Version = 8.0.0.0		Error	Failed with error: Package: l3service has sche...	admin	N/A
Generate Compliance Report: rp1		Complete	Operation Completed successfully	admin	N/A
Workflow Instance 118199 - scale_compliance:1:217	scale_compliance:1:2176		Complete user task	admin	N/A

Delete Package l3service Version = 8.0.0.0

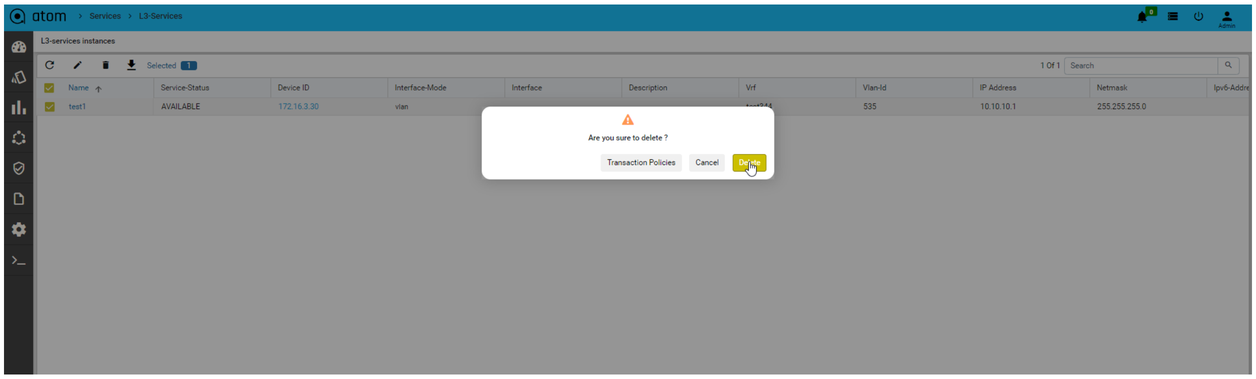
Task ID Po-1T00p_dSQ6kN4RRXHl2Eg
User Name admin
Time Taken 24/12/2020, 11:10:48 - 24/12/2020, 11:10:49 (1 seconds)

Summary Logs

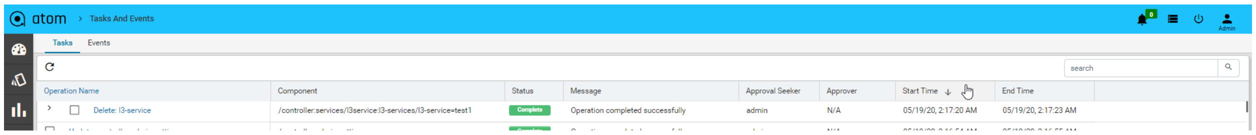
```
Dec 24, 2020, 11:10:49 AM
Package: l3service has schema elements that are being used by some data ( 1 data nodes ). Delete the data first.
Dec 24, 2020, 11:10:52 AM  Failed with error: Package: l3service has schema elements that are being used by some data ( 1 data nodes ). Delete the data first.
```

Solution:
Ensure that the services instantiated using the service package are deleted. Before the deletion of the service package, do the following:

1. Go to the **Automation -> Services** tab and delete all the services.



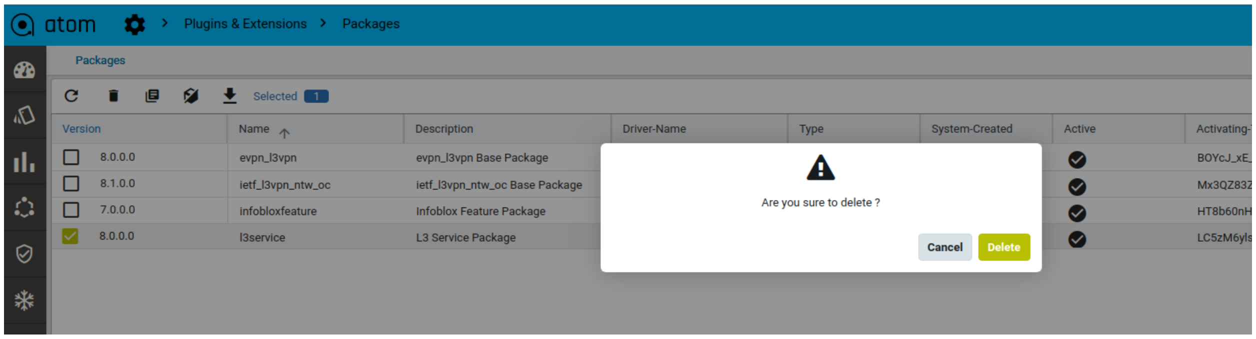
Name	Service-Status	Device ID	Interface-Mode	Interface	Description	Vrf	Vlan-Id	IP Address	Netmask	Ipv6-Addre
test1	AVAILABLE	172.16.3.30	vlan			test344	535	10.10.10.1	255.255.255.0	


atom > Tasks And Events

Tasks | Events

Operation Name	Component	Status	Message	Approval Seeker	Approver	Start Time	End Time
Delete: l3-service	/controller:services/l3service:l3-services/l3-service=test1	Complete	Operation completed successfully	admin	N/A	05/19/20, 2:17:20 AM	05/19/20, 2:17:23 AM

2. Navigate to the packages tab, Unload the package first and after that delete package.


atom > Plugins & Extensions > Packages

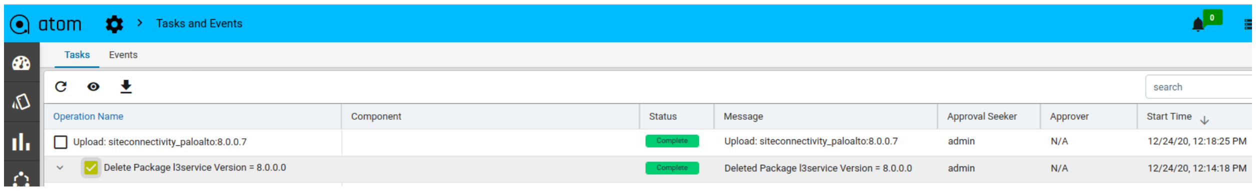
Packages

Selected 1

Version	Name	Description	Driver-Name	Type	System-Created	Active	Activating-
8.0.0.0	evpn_l3vpn	evpn_l3vpn Base Package				✓	BOYcJ_xE_
8.1.0.0	ietf_l3vpn_ntw_oc	ietf_l3vpn_ntw_oc Base Package				✓	Mx3QZ83Z
7.0.0.0	infobloxfeature	Infoblox Feature Package				✓	HT8b60nH
✓ 8.0.0.0	l3service	L3 Service Package				✓	LC5zM6yls

Are you sure to delete ?

Cancel | Delete


atom > Tasks and Events

Tasks | Events

Operation Name	Component	Status	Message	Approval Seeker	Approver	Start Time
Upload: siteconnectivity_paloalto:8.0.0.7		Complete	Upload: siteconnectivity_paloalto:8.0.0.7	admin	N/A	12/24/20, 12:18:25 PM
Delete Package l3service Version = 8.0.0.0		Complete	Deleted Package l3service Version = 8.0.0.0	admin	N/A	12/24/20, 12:14:18 PM

Logging Level for Task Logs

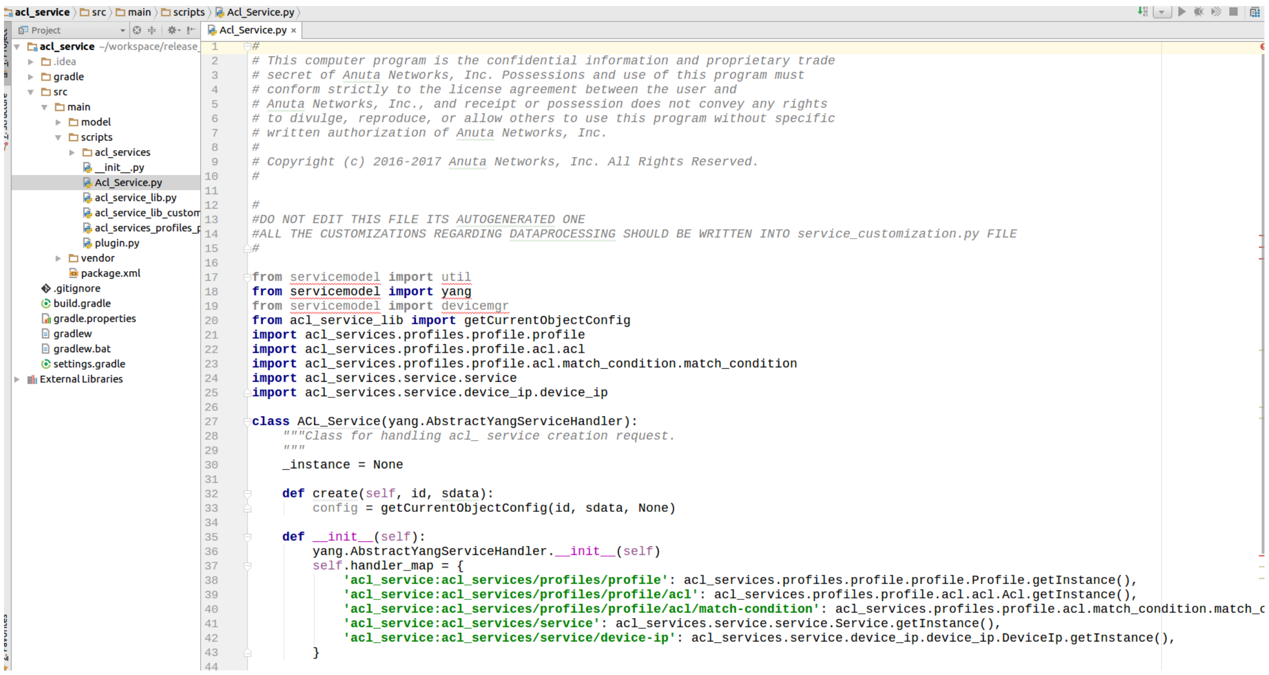
Log_info or Log_debug statements can be added in python code to get those debug or info messages in the Task Logs downloaded from ATOM.

E.x:

```
import util
util.log_info('calling register config provider for cisco')
util.log_debug("Exception seen with message: %s\n"%(str(e)))
```

Handler maps

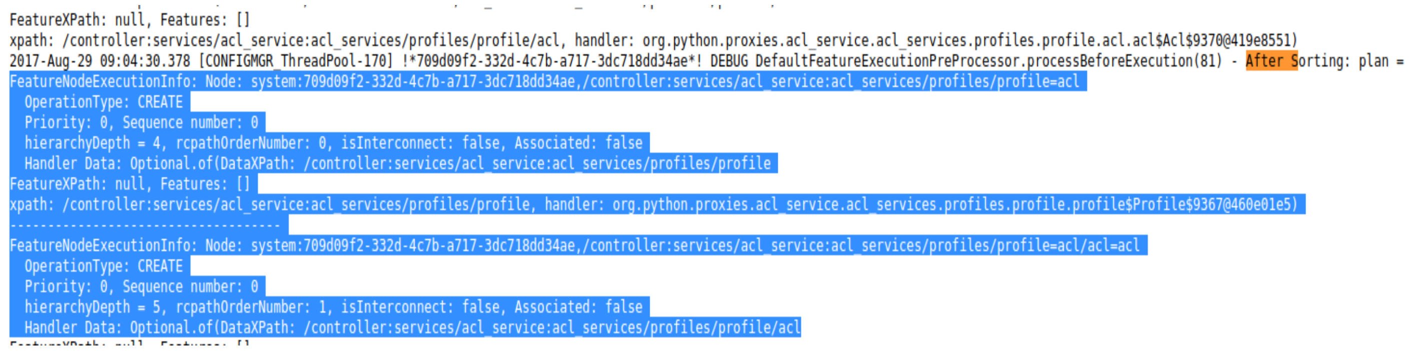
To debug issues of python code whether it entered into each module or not, we need to check if the required handler map is triggered or not. Handler map is to know which handles of service code are getting triggered first when service is triggered. Generally these will be present in {module name}.py file (ex: suppose module name is acl_service, file in service package will be with the name of Acl_Service.py).



```
#
# This computer program is the confidential information and proprietary trade
# secret of Anuta Networks, Inc. Possessions and use of this program must
# conform strictly to the license agreement between the user and
# Anuta Networks, Inc., and receipt or possession does not convey any rights
# to divulge, reproduce, or allow others to use this program without specific
# written authorization of Anuta Networks, Inc.
#
# Copyright (c) 2016-2017 Anuta Networks, Inc. All Rights Reserved.
#


#
#DO NOT EDIT THIS FILE ITS AUTOGENERATED ONE
#ALL THE CUSTOMIZATIONS REGARDING DATAPROCESSING SHOULD BE WRITTEN INTO service_customization.py FILE
#

from servicemodel import util
from servicemodel import yang
from servicemodel import devicemgr
from acl_service_lib import getCurrentObjectConfig
import acl_services.profiles.profile.profile
import acl_services.profiles.profile.acl.acl
import acl_services.profiles.profile.acl.match_condition.match_condition
import acl_services.service.service
import acl_services.service.device_ip.device_ip

class ACL_Service(yang.AbstractYangServiceHandler):
    """Class for handling acl_ service creation request.
    """
    _instance = None

    def create(self, id, sdata):
        config = getCurrentObjectConfig(id, sdata, None)

    def __init__(self):
        yang.AbstractYangServiceHandler.__init__(self)
        self.handler_map = {
            'acl_service:acl_services/profiles/profile': acl_services.profiles.profile.profile.Profile.getInstance(),
            'acl_service:acl_services/profiles/profile/acl': acl_services.profiles.profile.acl.acl.Acl.getInstance(),
            'acl_service:acl_services/profiles/profile/acl/match-condition': acl_services.profiles.profile.acl.match_condition.match_c
            'acl_service:acl_services/service': acl_services.service.service.Service.getInstance(),
            'acl_service:acl_services/service/device-ip': acl_services.service.device_ip.device_ip.DeviceIp.getInstance(),
        }
```

Verification In tasklog

Look for keyword After Sorting which shows what are the handles of service being invoked

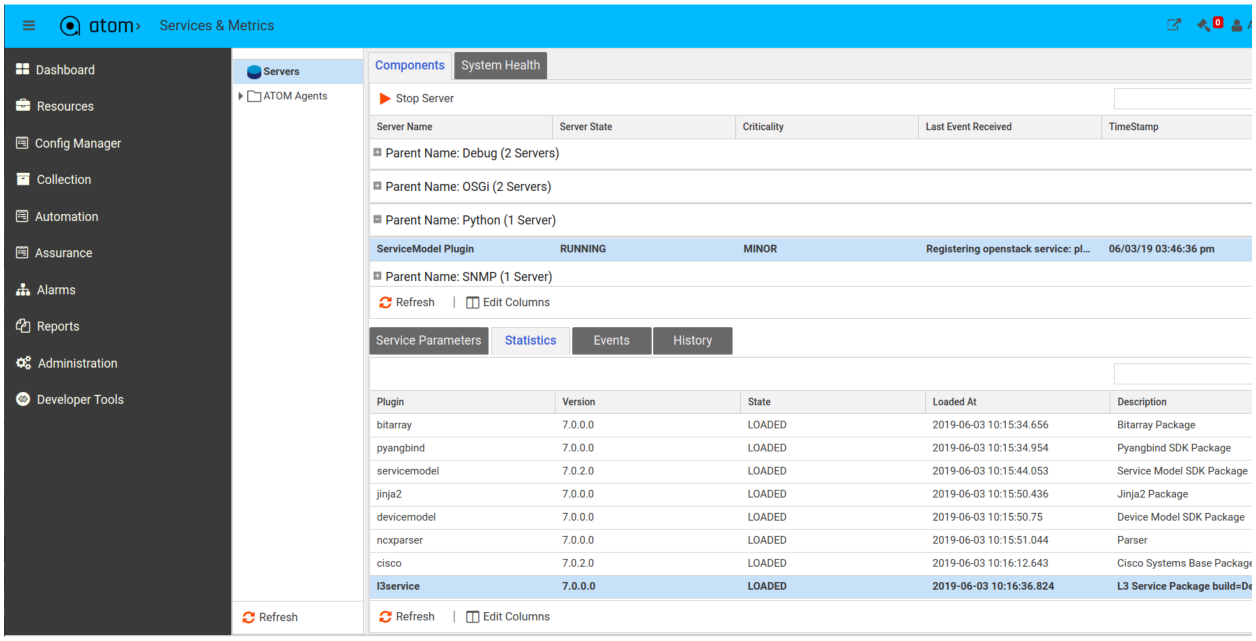


```
FeatureXPath: null, Features: []
xpath: /controller:services/acl_service:acl_services/profiles/profile/acl, handler: org.python.proxies.acl_service.acl_services.profiles.profile.acl.acl$Acl$9370@419e8551)
2017-Aug-29 09:04:30.378 [CONFIGMGR_ThreadPool-170] !*709d09f2-332d-4c7b-a717-3dc718dd34ae*! DEBUG DefaultFeatureExecutionPreProcessor.processBeforeExecution(81) - After Sorting: plan =
FeatureNodeExecutionInfo: Node: system:709d09f2-332d-4c7b-a717-3dc718dd34ae,/controller:services/acl_service:acl_services/profiles/profile=acl
  OperationType: CREATE
  Priority: 0, Sequence number: 0
  hierarchyDepth = 4, rcpathOrderNumber: 0, isInterconnect: false, Associated: false
  Handler Data: Optional.of(DataXPath: /controller:services/acl_service:acl_services/profiles/profile
FeatureXPath: null, Features: []
xpath: /controller:services/acl_service:acl_services/profiles/profile, handler: org.python.proxies.acl_service.acl_services.profiles.profile.profile$Profile$9367@460e01e5)
----------------------------------
FeatureNodeExecutionInfo: Node: system:709d09f2-332d-4c7b-a717-3dc718dd34ae,/controller:services/acl_service:acl_services/profiles/profile=acl/acl=acl
  OperationType: CREATE
  Priority: 0, Sequence number: 0
  hierarchyDepth = 5, rcpathOrderNumber: 1, isInterconnect: false, Associated: false
  Handler Data: Optional.of(DataXPath: /controller:services/acl_service:acl_services/profiles/profile/acl
```

Registering the Service Package with ATOM

After the service package is loaded successfully into ATOM, in the **Services tab,** user can check if the python code with respect to yang is registered with ATOM or not. Users should cross check like below.

1. Click **Administration** > **Troubleshoot>Services & Metrics** > **Servers** > **Components** > **Python** > **ServiceModelPlugin** > click the **Statistics** tab
2. Check the status of the service package.
3. In the **Statistics** tab, if LOADED is displayed in the **State** column, the service package is loaded into the ATOM system successfully.
4. If State column is **FAILED** for uploaded service package, then service package contains some errors
5. Check for the exceptions in **Service Model Plugin** server log, fix the exception in respective python module and upload again.



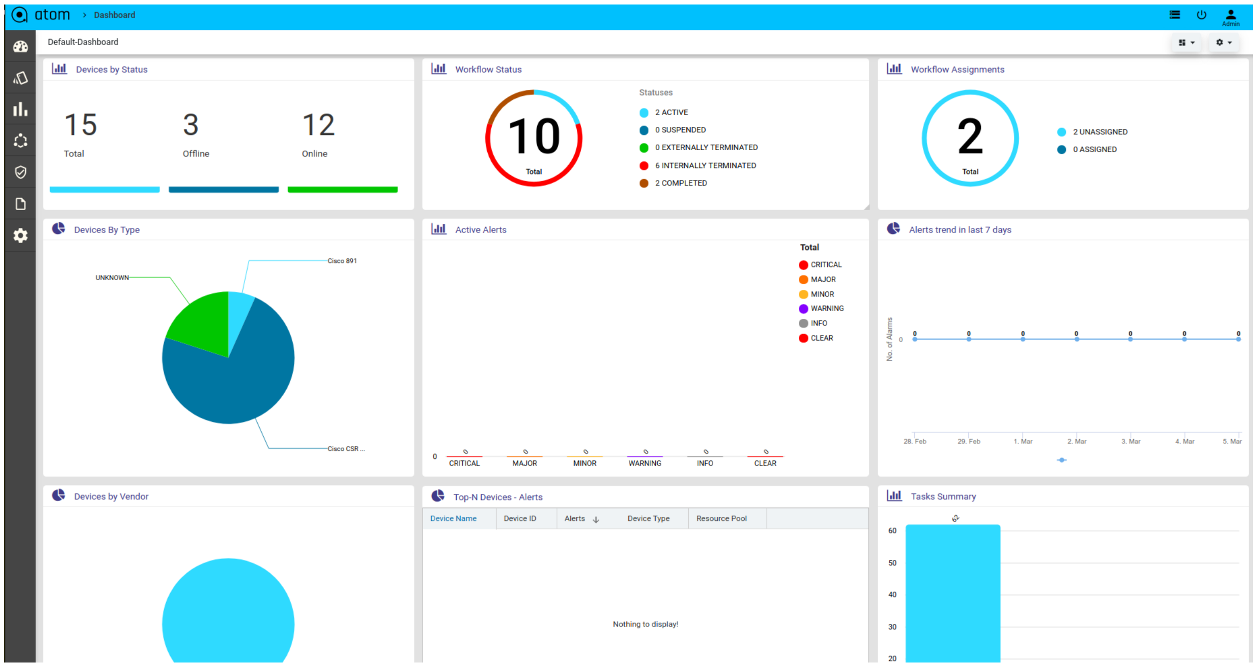
Server Name	Server State	Criticality	Last Event Received	TimeStamp
Parent Name: Debug (2 Servers)				
Parent Name: OSGi (2 Servers)				
Parent Name: Python (1 Server)				
ServiceModel Plugin	RUNNING	MINOR	Registering openstack service: pl...	06/03/19 03:46:36 pm
Parent Name: SNMP (1 Server)				

Plugin	Version	State	Loaded At	Description
bitarray	7.0.0.0	LOADED	2019-06-03 10:15:34.656	Bitarray Package
pyangbind	7.0.0.0	LOADED	2019-06-03 10:15:34.954	Pyangbind SDK Package
servicemodel	7.0.2.0	LOADED	2019-06-03 10:15:44.053	Service Model SDK Package
jinja2	7.0.0.0	LOADED	2019-06-03 10:15:50.436	Jinja2 Package
devicemodel	7.0.0.0	LOADED	2019-06-03 10:15:50.75	Device Model SDK Package
ncxparser	7.0.0.0	LOADED	2019-06-03 10:15:51.044	Parser
cisco	7.0.2.0	LOADED	2019-06-03 10:16:12.643	Cisco Systems Base Package
l3service	7.0.0.0	LOADED	2019-06-03 10:16:36.824	L3 Service Package build=De

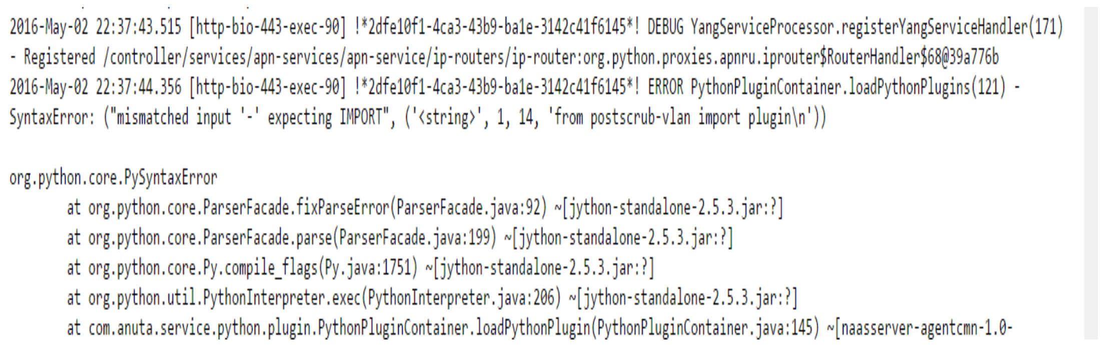
Binding of the logic with ATOM

After uploading, registering the service package with ATOM successfully, syntactical and semantic errors in the python glue logic can also cause issues that need to be resolved.

Sample Failure to upload Python Plugin files



1. Navigate to the **Task Viewer**, download or view the **Service Model Plugin** logs and look for the exception in the task log as shown below:



```
2016-May-02 22:37:43.515 [http-bio-443-exec-90] !*2dfe10f1-4ca3-43b9-ba1e-3142c41f6145*! DEBUG YangServiceProcessor.registerYangServiceHandler(171)
- Registered /controller/services/apn-services/apn-service/ip-routers/ip-router:org.python.proxies.apnru.iprouter$RouterHandler$68@39a776b
2016-May-02 22:37:44.356 [http-bio-443-exec-90] !*2dfe10f1-4ca3-43b9-ba1e-3142c41f6145*! ERROR PythonPluginContainer.loadPythonPlugins(121) -
SyntaxError: ("mismatched input '-' expecting IMPORT", ('<string>', 1, 14, 'from postscrub-vlan import plugin\n'))

org.python.core.PySyntaxError
        at org.python.core.ParserFacade.fixParseError(ParserFacade.java:92) ~[jython-standalone-2.5.3.jar:?]
        at org.python.core.ParserFacade.parse(ParserFacade.java:199) ~[jython-standalone-2.5.3.jar:?]
        at org.python.core.Py.compile_flags(Py.java:1751) ~[jython-standalone-2.5.3.jar:?]
        at org.python.util.PythonInterpreter.exec(PythonInterpreter.java:206) ~[jython-standalone-2.5.3.jar:?]
        at com.anuta.service.python.plugin.PythonPluginContainer.loadPythonPlugin(PythonPluginContainer.java:145) ~[naasserver-agentcmn-1.0-
```

Due to the hyphen (-) in the **postscrub-vlan.py** file, the package python code registering into ATOM did not happen.

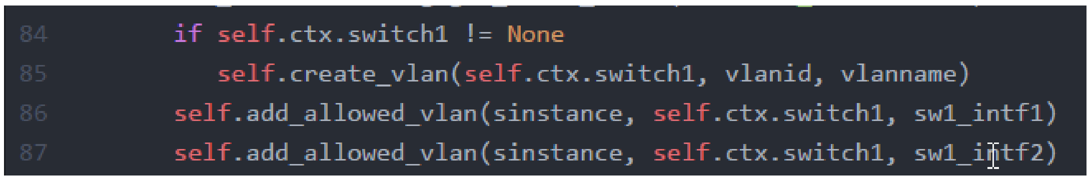
Exception:

```
2016-Apr-15 00:23:08.186 [http-bio-443-exec-73] !*f2861c70-2050-4488-b48a-e14173c43727*! ERROR
PythonPluginContainer.loadPythonPlugins(121) - SyntaxError: ("mismatched input '-' expecting IMPORT",
('<string>', 1, 14, 'from postscrub-vlan import plugin\n'))
```

Solution:

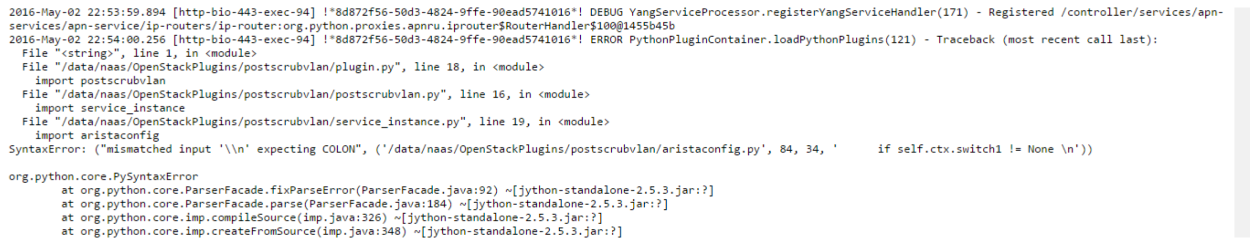
1. Rename the **postscrub-vlan.py** to **postscrubvlan.py**, similarly rename yang and yin files

Syntax Errors in Python plugin file



```
84        if self.ctx.switch1 != None
85           self.create_vlan(self.ctx.switch1, vlanid, vlanname)
86        self.add_allowed_vlan(sinstance, self.ctx.switch1, sw1_intf1)
87        self.add_allowed_vlan(sinstance, self.ctx.switch1, sw1_intf2)
```

Error



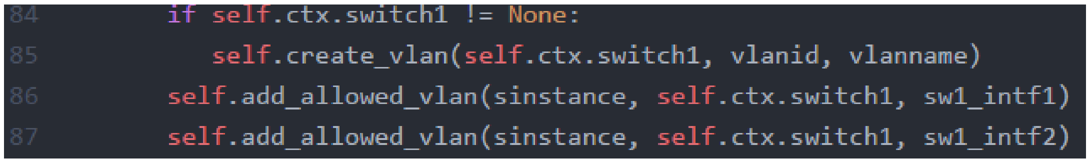
```
2016-May-02 22:53:59.894 [http-bio-443-exec-94] !*8d872f56-50d3-4824-9ffe-90ead5741016*! DEBUG YangServiceProcessor.registerYangServiceHandler(171) - Registered /controller/services/apn-
services/apn-service/ip-routers/ip-router:org.python.proxies.apnru.iprouter$RouterHandler$100@1455b45b
2016-May-02 22:54:00.256 [http-bio-443-exec-94] !*8d872f56-50d3-4824-9ffe-90ead5741016*! ERROR PythonPluginContainer.loadPythonPlugins(121) - Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/data/naas/OpenStackPlugins/postscrubvlan/plugin.py", line 18, in <module>
    import postscrubvlan
  File "/data/naas/OpenStackPlugins/postscrubvlan/postscrubvlan.py", line 16, in <module>
    import service_instance
  File "/data/naas/OpenStackPlugins/postscrubvlan/service_instance.py", line 19, in <module>
    import aristaconfig
SyntaxError: ("mismatched input '\\n' expecting COLON", ('/data/naas/OpenStackPlugins/postscrubvlan/aristaconfig.py', 84, 34, '      if self.ctx.switch1 != None \n'))

org.python.core.PySyntaxError
        at org.python.core.ParserFacade.fixParseError(ParserFacade.java:92) ~[jython-standalone-2.5.3.jar:?]
        at org.python.core.ParserFacade.parse(ParserFacade.java:184) ~[jython-standalone-2.5.3.jar:?]
        at org.python.core.imp.compileSource(imp.java:326) ~[jython-standalone-2.5.3.jar:?]
        at org.python.core.imp.createFromSource(imp.java:348) ~[jython-standalone-2.5.3.jar:?]
```

Exception

```
SyntaxError: ("mismatched input '\\n' expecting COLON",
('/data/naas/OpenStackPlugins/postscrubvlan/aristaconfig.py', 84, 34, '      if self.ctx.switch1 !=
None \n'))
```

Solution:

Correct it like below



```
        if self.ctx.switch1 != None:
            self.create_vlan(self.ctx.switch1, vlanid, vlanname)
        self.add_allowed_vlan(sinstance, self.ctx.switch1, sw1_intf1)
        self.add_allowed_vlan(sinstance, self.ctx.switch1, sw1_intf2)
```

Now service package is free of syntactical errors and is loaded successfully into ATOM

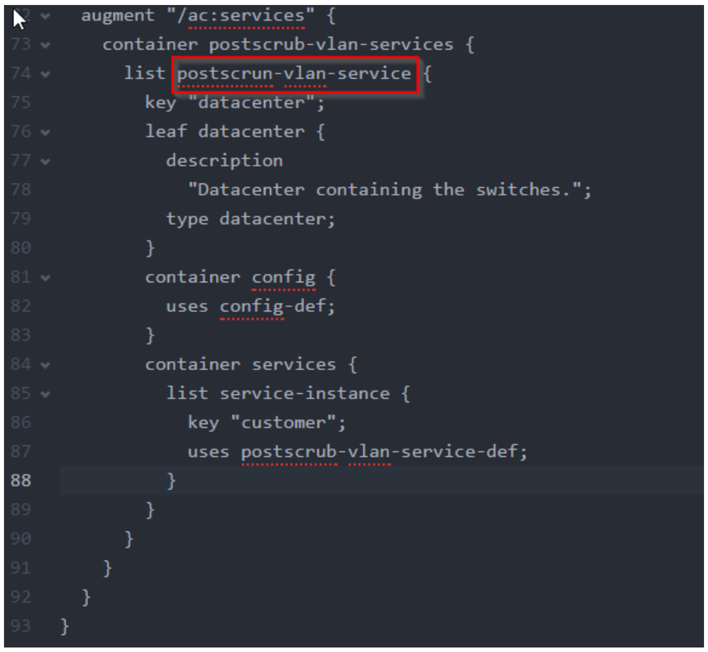
Semantic Errors in the Service package files

Even if the syntactically yang and python files are correct, there might be some issues due to typos in the YANG files or URLs used in python, glue logic

Though these errors are not displayed in the log, check for the following:

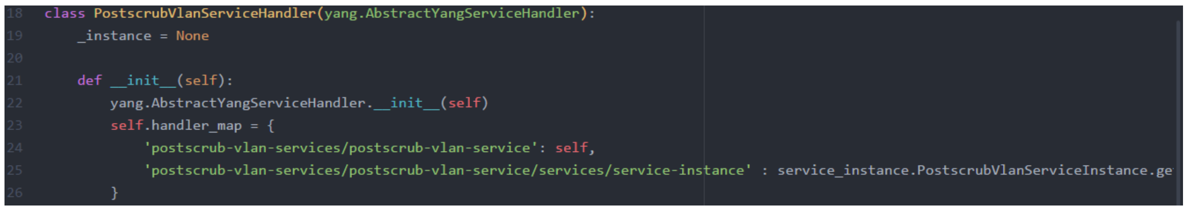
1. The used URLs are appropriate
2. Typos in the yang files or python modules

Typo in Yang File



```
  augment "/ac:services" {
    container postscrub-vlan-services {
      list postscrun-vlan-service {
        key "datacenter";
        leaf datacenter {
          description
            "Datacenter containing the switches.";
          type datacenter;
        }
        container config {
          uses config-def;
        }
        container services {
          list service-instance {
            key "customer";
            uses postscrub-vlan-service-def;
          }
        }
      }
    }
  }
}
```

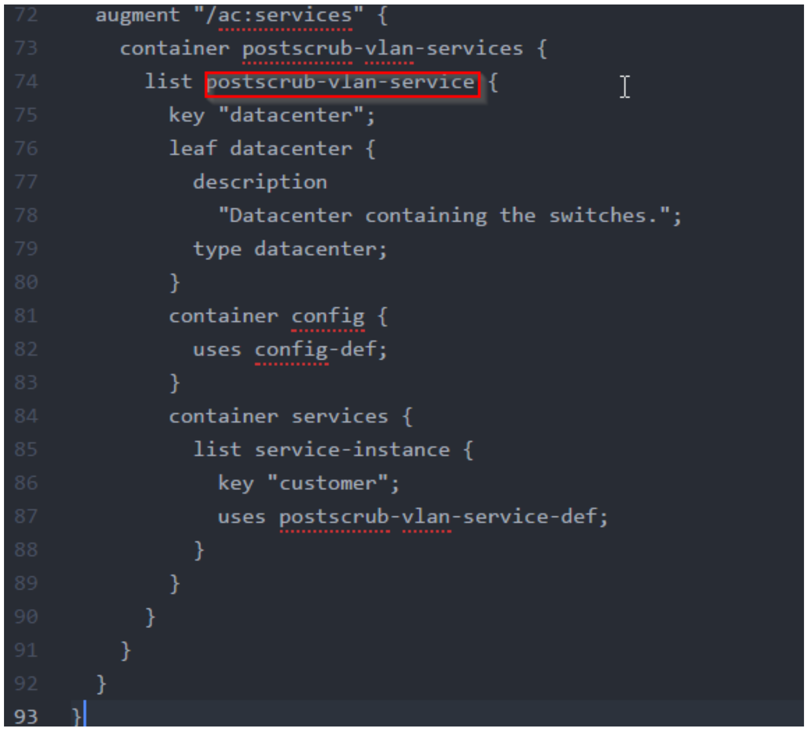
In glue logic, the path is described as shown in the following snippet:



```
class PostscrubVlanServiceHandler(yang.AbstractYangServiceHandler):
    _instance = None

    def __init__(self):
        yang.AbstractYangServiceHandler.__init__(self)
        self.handler_map = {
            'postscrub-vlan-services/postscrub-vlan-service': self,
            'postscrub-vlan-services/postscrub-vlan-service/services/service-instance' : service_instance.PostscrubVlanServiceInstance.ge
        }
```

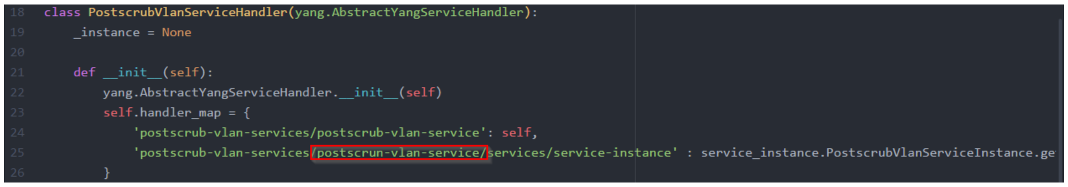
Solution

Correct the typo present in yang and yin files



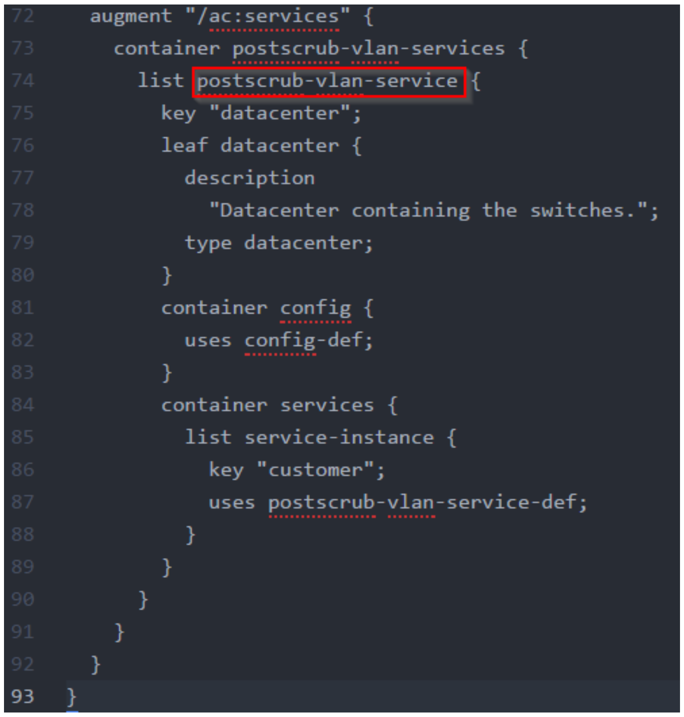
```
  augment "/ac:services" {
    container postscrub-vlan-services {
      list postscrub-vlan-service {
        key "datacenter";
        leaf datacenter {
          description
            "Datacenter containing the switches.";
          type datacenter;
        }
        container config {
          uses config-def;
        }
        container services {
          list service-instance {
            key "customer";
            uses postscrub-vlan-service-def;
          }
        }
      }
    }
  }
}
```

Typos in Python Module



```
class PostscrubVlanServiceHandler(yang.AbstractYangServiceHandler):
    _instance = None

    def __init__(self):
        yang.AbstractYangServiceHandler.__init__(self)
        self.handler_map = {
            'postscrub-vlan-services/postscrub-vlan-service': self,
            'postscrub-vlan-services/postscrun-vlan-service/services/service-instance' : service_instance.PostscrubVlanServiceInstance.ge
        }
```

Defined in the YANG module is as shown below:

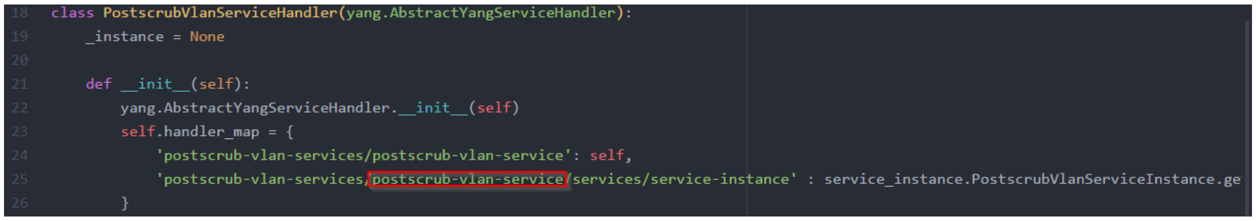


```
  augment "/ac:services" {
    container postscrub-vlan-services {
      list postscrub-vlan-service {
        key "datacenter";
        leaf datacenter {
          description
            "Datacenter containing the switches.";
          type datacenter;
        }
        container config {
          uses config-def;
        }
        container services {
          list service-instance {
            key "customer";
            uses postscrub-vlan-service-def;
          }
        }
      }
    }
  }
}
```

Mismatch in the yang and python module due to typos leads to improper call invocation.

Solution

Correct the typos in the python module as shown below:



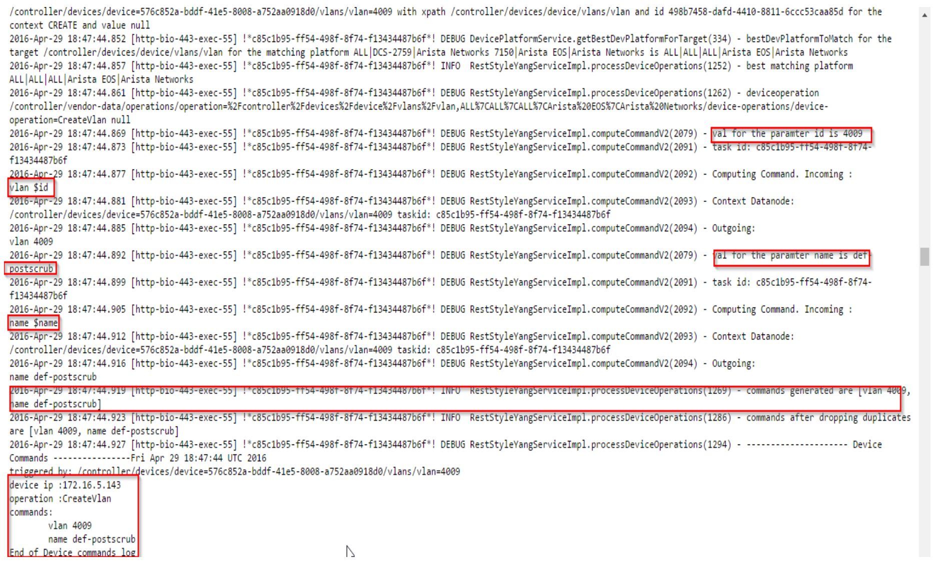
```
class PostscrubVlanServiceHandler(yang.AbstractYangServiceHandler):
    _instance = None

    def __init__(self):
        yang.AbstractYangServiceHandler.__init__(self)
        self.handler_map = {
            'postscrub-vlan-services/postscrub-vlan-service': self,
            'postscrub-vlan-services/postscrub-vlan-service/services/service-instance' : service_instance.PostscrubVlanServiceInstance.ge
        }
```

Commands not being generated in ATOM

1. Go to the **naas server.log** or the task log.
2. Look for the pattern “best match platform”.

```
2016-Apr-29 18:47:44.852 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG
DevicePlatformService.getBestDevPlatformForTarget(334) - bestDevPlatformToMatch for the target
/controller/devices/device/vlans/vlan for the matching platform ALL|DCS-2759|Arista Networks 7150|Arista
EOS|Arista Networks is ALL|ALL|ALL|Arista EOS|Arista Networks

2016-Apr-29 18:47:44.857 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! INFO
RestStyleYangServiceImpl.processDeviceOperations(1252) - best matching platform ALL|ALL|ALL|Arista EOS|Arista
Networks
```

3. For operation best match platform found and respective command conditions are validated and **$ variables** are replaced by values provided by the end user.
4. The detailed highlights are marked in red as shown below:

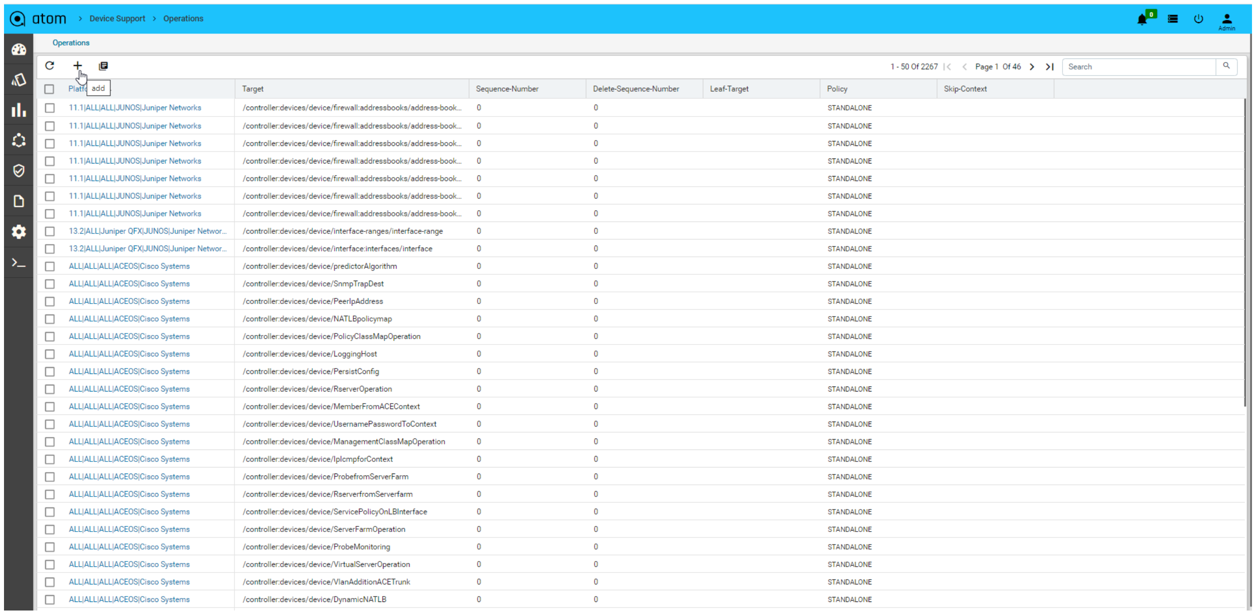


```
/controller/devices/device=576c852a-bddf-41e5-8008-a752aa0918d0/vlans/vlan=4009 with xpath /controller/devices/device/vlans/vlan and id 498b7458-dafd-4410-8811-6ccc53caa85d for the
context CREATE and value null
2016-Apr-29 18:47:44.852 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG DevicePlatformService.getBestDevPlatformForTarget(334) - bestDevPlatformToMatch for the
target /controller/devices/device/vlans/vlan for the matching platform ALL|DCS-2759|Arista Networks 7150|Arista EOS|Arista Networks is ALL|ALL|ALL|Arista EOS|Arista Networks
2016-Apr-29 18:47:44.857 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! INFO  RestStyleYangServiceImpl.processDeviceOperations(1252) - best matching platform
ALL|ALL|ALL|Arista EOS|Arista Networks
2016-Apr-29 18:47:44.861 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.processDeviceOperations(1262) - deviceoperation
/controller/vendor-data/operations/operation=%2Fcontroller%2Fdevices%2Fdevice%2Fvlans%2Fvlan,ALL%7CALL%7CALL%7CArista%20EOS%7CArista%20Networks/device-operations/device-
operation=CreateVlan null
2016-Apr-29 18:47:44.869 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.computeCommandV2(2079) - val for the paramter id is 4009
2016-Apr-29 18:47:44.873 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.computeCommandV2(2091) - task id: c85c1b95-ff54-498f-8f74-
f13434487b6f
2016-Apr-29 18:47:44.877 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.computeCommandV2(2092) - Computing Command. Incoming :
vlan $id
2016-Apr-29 18:47:44.881 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.computeCommandV2(2093) - Context Datanode:
/controller/devices/device=576c852a-bddf-41e5-8008-a752aa0918d0/vlans/vlan=4009 taskid: c85c1b95-ff54-498f-8f74-f13434487b6f
2016-Apr-29 18:47:44.885 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.computeCommandV2(2094) - Outgoing:
vlan 4009
2016-Apr-29 18:47:44.892 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.computeCommandV2(2079) - val for the paramter name is def-
postscrub
2016-Apr-29 18:47:44.899 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.computeCommandV2(2091) - task id: c85c1b95-ff54-498f-8f74-
f13434487b6f
2016-Apr-29 18:47:44.905 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.computeCommandV2(2092) - Computing Command. Incoming :
name $name
2016-Apr-29 18:47:44.912 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.computeCommandV2(2093) - Context Datanode:
/controller/devices/device=576c852a-bddf-41e5-8008-a752aa0918d0/vlans/vlan=4009 taskid: c85c1b95-ff54-498f-8f74-f13434487b6f
2016-Apr-29 18:47:44.916 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.computeCommandV2(2094) - Outgoing:
name def-postscrub
2016-Apr-29 18:47:44.919 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! INFO  RestStyleYangServiceImpl.processDeviceOperations(1269) - commands generated are [vlan 4009,
name def-postscrub]
2016-Apr-29 18:47:44.923 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! INFO  RestStyleYangServiceImpl.processDeviceOperations(1286) - commands after dropping duplicates
are [vlan 4009, name def-postscrub]
2016-Apr-29 18:47:44.927 [http-bio-443-exec-55] !*c85c1b95-ff54-498f-8f74-f13434487b6f*! DEBUG RestStyleYangServiceImpl.processDeviceOperations(1294) - -------------------- Device
Commands ---------------Fri Apr 29 18:47:44 UTC 2016
triggered by: /controller/devices/device=576c852a-bddf-41e5-8008-a752aa0918d0/vlans/vlan=4009
device ip :172.16.5.143
operation :CreateVlan
commands:
        vlan 4009
        name def-postscrub
End of Device commands log
```

5. If the best match platform is null, check if the device operations are defined in the vendor-data.
6. In the ATOM UI, navigate to **Administration >Plugins & Extensions > Device Support > Operations**
7. If the Operations are defined in the Device support and the commands are not being generated still, check with Anuta Networks
8. However, if the operations have not been defined, create an operation in the UI.

Solution

If device operation is not defined for the platform, define it by adding the **Create**, **Update,** and **Delete** operations for that platform as illustrated below:



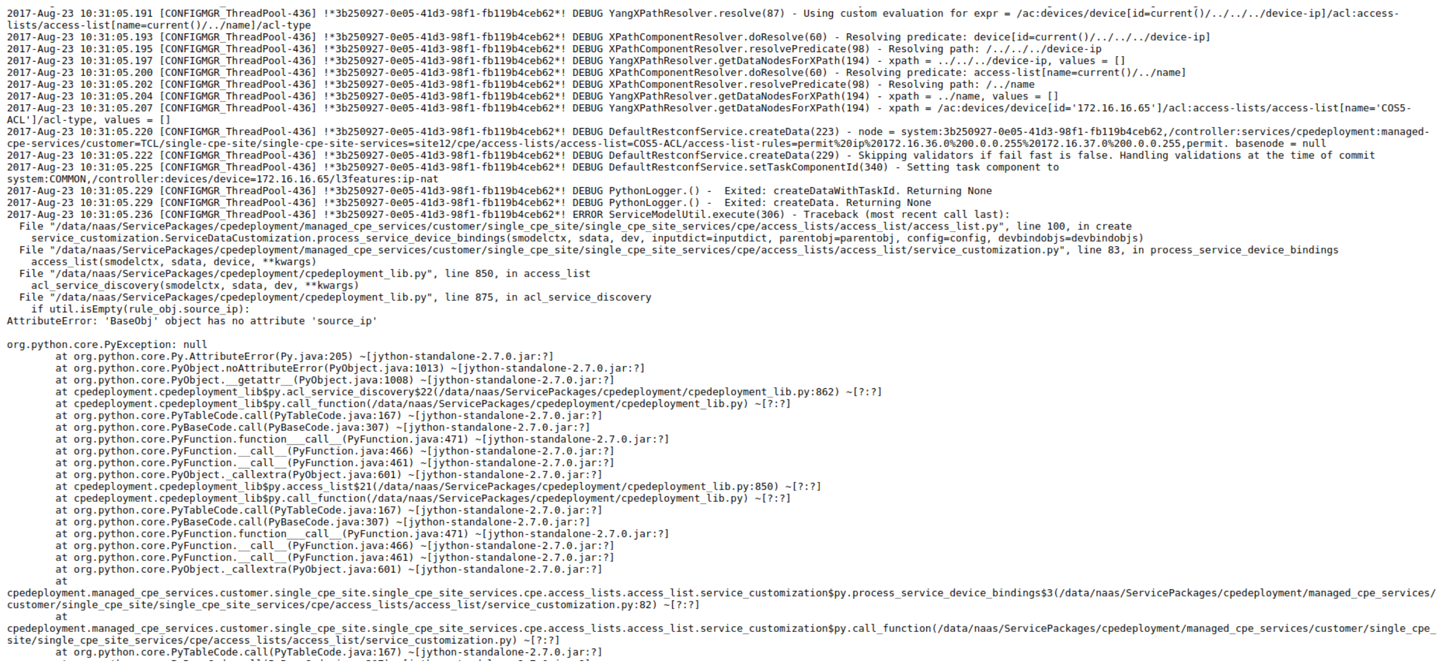
Platf	Target	Sequence-Number	Delete-Sequence-Number	Leaf-Target	Policy	Skip-Context
11.1|ALL|ALL|JUNOS|Juniper Networks	/controller:devices/device/firewall:addressbooks/address-book...	0	0		STANDALONE	
11.1|ALL|ALL|JUNOS|Juniper Networks	/controller:devices/device/firewall:addressbooks/address-book...	0	0		STANDALONE	
11.1|ALL|ALL|JUNOS|Juniper Networks	/controller:devices/device/firewall:addressbooks/address-book...	0	0		STANDALONE	
11.1|ALL|ALL|JUNOS|Juniper Networks	/controller:devices/device/firewall:addressbooks/address-book...	0	0		STANDALONE	
11.1|ALL|ALL|JUNOS|Juniper Networks	/controller:devices/device/firewall:addressbooks/address-book...	0	0		STANDALONE	
11.1|ALL|ALL|JUNOS|Juniper Networks	/controller:devices/device/firewall:addressbooks/address-book...	0	0		STANDALONE	
11.1|ALL|ALL|JUNOS|Juniper Networks	/controller:devices/device/firewall:addressbooks/address-book...	0	0		STANDALONE	
13.2|ALL|Juniper QFX|JUNOS|Juniper Networ...	/controller:devices/device/interface-ranges/interface-range	0	0		STANDALONE	
13.2|ALL|Juniper QFX|JUNOS|Juniper Networ...	/controller:devices/device/interface:interfaces/interface	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/predictorAlgorithm	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/SnmpTrapDest	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/PeerIpAddress	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/NATLBpolicymap	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/PolicyClassMapOperation	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/LoggingHost	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/PersistConfig	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/RserverOperation	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/MemberFromACEContext	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/UsernamePasswordToContext	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/ManagementClassMapOperation	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/IplcmpforContext	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/ProbefromServerFarm	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/RserverfromServerfarm	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/ServicePolicyOnLBInterface	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/ServerFarmOperation	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/ProbeMonitoring	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/VirtualServerOperation	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/VlanAdditionACETrunk	0	0		STANDALONE	
ALL|ALL|ALL|ACEOS|Cisco Systems	/controller:devices/device/DynamicNATLB	0	0		STANDALONE	

Attribute Error

Error:

When the leaf is defined in yang and at the time of service creation the user gives an empty value to that leaf, below error will be seen.

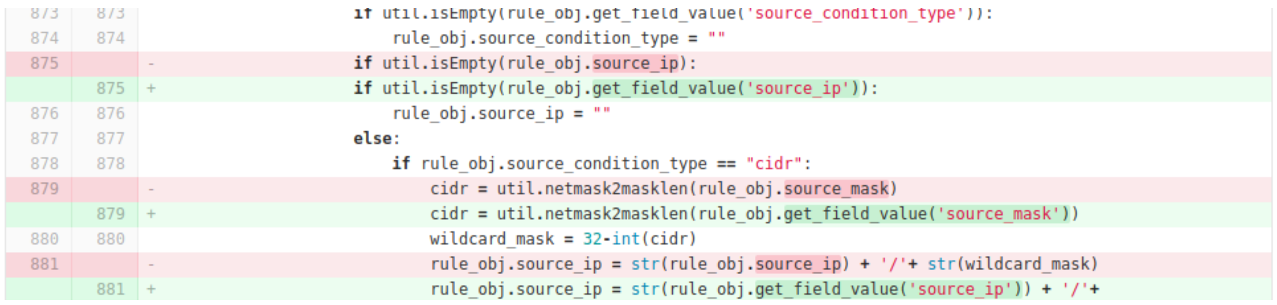
Base-Object has no Attribute/object



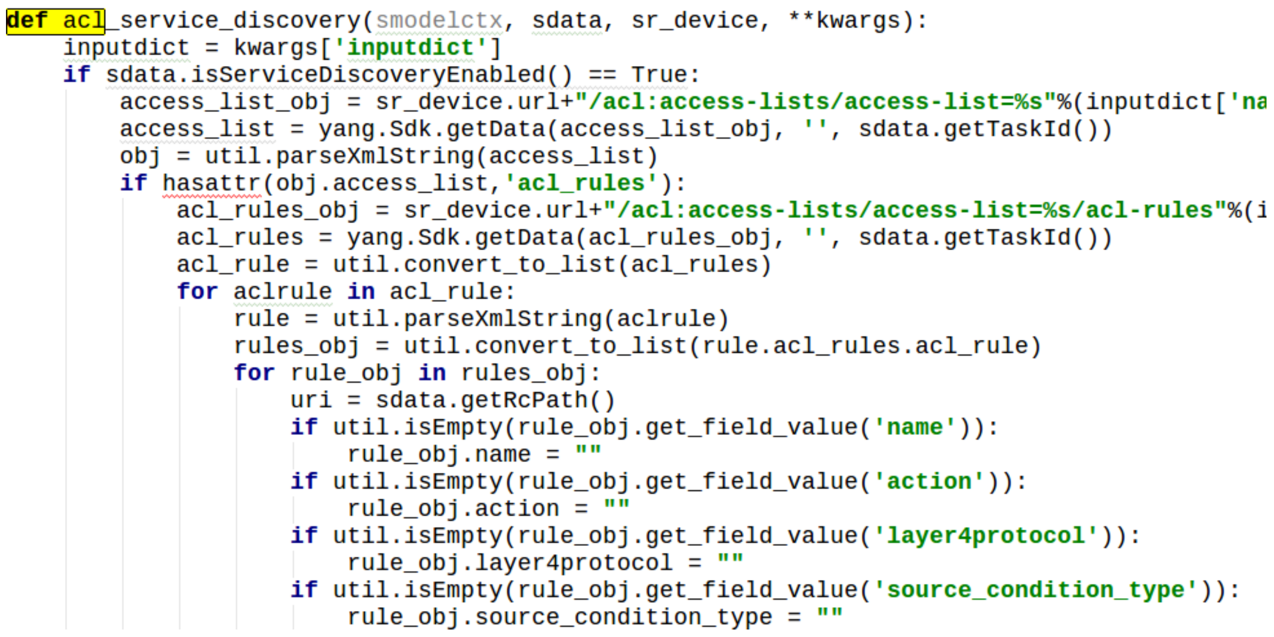
```
2017-Aug-23 10:31:05.191 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG YangXPathResolver.resolve(87) - Using custom evaluation for expr = /ac:devices/device[id=current()/../../../device-ip]/acl:access-
lists/access-list[name=current()/../name]/acl-type
2017-Aug-23 10:31:05.193 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG XPathComponentResolver.doResolve(60) - Resolving predicate: device[id=current()/../../../device-ip]
2017-Aug-23 10:31:05.195 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG XPathComponentResolver.resolvePredicate(98) - Resolving path: /../../../device-ip
2017-Aug-23 10:31:05.197 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG YangXPathResolver.getDataNodesForXPath(194) - xpath = ../../../device-ip, values = []
2017-Aug-23 10:31:05.200 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG XPathComponentResolver.doResolve(60) - Resolving predicate: access-list[name=current()/../name]
2017-Aug-23 10:31:05.202 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG XPathComponentResolver.resolvePredicate(98) - Resolving path: /../name
2017-Aug-23 10:31:05.204 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG YangXPathResolver.getDataNodesForXPath(194) - xpath = ../name, values = []
2017-Aug-23 10:31:05.207 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG YangXPathResolver.getDataNodesForXPath(194) - xpath = /ac:devices/device[id='172.16.16.65']/acl:access-lists/access-list[name='COS5-
ACL']/acl-type, values = []
2017-Aug-23 10:31:05.220 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG DefaultRestconfService.createData(223) - node = system:3b250927-0e05-41d3-98f1-fb119b4ceb62,/controller:services/cpedeployment:managed-
cpe-services/customer=TCL/single-cpe-site/single-cpe-site-services=site12/cpe/access-lists/access-list=COS5-ACL/access-list-rules=permit%20ip%20172.16.36.0%200.0.0.255%20172.16.37.0%200.0.0.255,permit. basenode = null
2017-Aug-23 10:31:05.222 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG DefaultRestconfService.createData(229) - Skipping validators if fail fast is false. Handling validations at the time of commit
2017-Aug-23 10:31:05.225 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG DefaultRestconfService.setTaskComponentId(340) - Setting task component to
system:COMMON,/controller:devices/device=172.16.16.65/l3features:ip-nat
2017-Aug-23 10:31:05.229 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG PythonLogger.() -  Exited: createDataWithTaskId. Returning None
2017-Aug-23 10:31:05.229 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! DEBUG PythonLogger.() -  Exited: createData. Returning None
2017-Aug-23 10:31:05.236 [CONFIGMGR_ThreadPool-436] !*3b250927-0e05-41d3-98f1-fb119b4ceb62*! ERROR ServiceModelUtil.execute(306) - Traceback (most recent call last):
  File "/data/naas/ServicePackages/cpedeployment/managed_cpe_services/customer/single_cpe_site/single_cpe_site_services/cpe/access_lists/access_list/access_list.py", line 100, in create
    service_customization.ServiceDataCustomization.process_service_device_bindings(smodelctx, sdata, dev, inputdict=inputdict, parentobj=parentobj, config=config, devbindobjs=devbindobjs)
  File "/data/naas/ServicePackages/cpedeployment/managed_cpe_services/customer/single_cpe_site/single_cpe_site_services/cpe/access_lists/access_list/service_customization.py", line 83, in process_service_device_bindings
    access_list(smodelctx, sdata, device, **kwargs)
  File "/data/naas/ServicePackages/cpedeployment/cpedeployment_lib.py", line 850, in access_list
    acl_service_discovery(smodelctx, sdata, dev, **kwargs)
  File "/data/naas/ServicePackages/cpedeployment/cpedeployment_lib.py", line 875, in acl_service_discovery
    if util.isEmpty(rule_obj.source_ip):
AttributeError: 'BaseObj' object has no attribute 'source_ip'

org.python.core.PyException: null
        at org.python.core.Py.AttributeError(Py.java:205) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyObject.noAttributeError(PyObject.java:1013) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyObject.__getattr__(PyObject.java:1008) ~[jython-standalone-2.7.0.jar:?]
        at cpedeployment.cpedeployment_lib$py.acl_service_discovery$22(/data/naas/ServicePackages/cpedeployment/cpedeployment_lib.py:862) ~[?:?]
        at cpedeployment.cpedeployment_lib$py.call_function(/data/naas/ServicePackages/cpedeployment/cpedeployment_lib.py) ~[?:?]
        at org.python.core.PyTableCode.call(PyTableCode.java:167) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyBaseCode.call(PyBaseCode.java:307) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyFunction.function___call__(PyFunction.java:471) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyFunction.__call__(PyFunction.java:466) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyFunction.__call__(PyFunction.java:461) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyObject._callextra(PyObject.java:601) ~[jython-standalone-2.7.0.jar:?]
        at cpedeployment.cpedeployment_lib$py.access_list$21(/data/naas/ServicePackages/cpedeployment/cpedeployment_lib.py:850) ~[?:?]
        at cpedeployment.cpedeployment_lib$py.call_function(/data/naas/ServicePackages/cpedeployment/cpedeployment_lib.py) ~[?:?]
        at org.python.core.PyTableCode.call(PyTableCode.java:167) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyBaseCode.call(PyBaseCode.java:307) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyFunction.function___call__(PyFunction.java:471) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyFunction.__call__(PyFunction.java:466) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyFunction.__call__(PyFunction.java:461) ~[jython-standalone-2.7.0.jar:?]
        at org.python.core.PyObject._callextra(PyObject.java:601) ~[jython-standalone-2.7.0.jar:?]
        at
cpedeployment.managed_cpe_services.customer.single_cpe_site.single_cpe_site_services.cpe.access_lists.access_list.service_customization$py.process_service_device_bindings$3(/data/naas/ServicePackages/cpedeployment/managed_cpe_services/
customer/single_cpe_site/single_cpe_site_services/cpe/access_lists/access_list/service_customization.py:82) ~[?:?]
        at
cpedeployment.managed_cpe_services.customer.single_cpe_site.single_cpe_site_services.cpe.access_lists.access_list.service_customization$py.call_function(/data/naas/ServicePackages/cpedeployment/managed_cpe_services/customer/single_cpe_
site/single_cpe_site_services/cpe/access_lists/access_list/service_customization.py) ~[?:?]
        at org.python.core.PyTableCode.call(PyTableCode.java:167) ~[jython-standalone-2.7.0.jar:?]
```

Solution

To overcome the above exception, make sure that the user should add the **get_field_value** like shown in below snippet to fetch the value from service yang and assign it to create call.



```
873  873          if util.isEmpty(rule_obj.get_field_value('source_condition_type')):
874  874              rule_obj.source_condition_type = ""
875       -       if util.isEmpty(rule_obj.source_ip):
     875  +       if util.isEmpty(rule_obj.get_field_value('source_ip')):
876  876              rule_obj.source_ip = ""
877  877          else:
878  878              if rule_obj.source_condition_type == "cidr":
879       -               cidr = util.netmask2masklen(rule_obj.source_mask)
     879  +               cidr = util.netmask2masklen(rule_obj.get_field_value('source_mask'))
880  880                  wildcard_mask = 32-int(cidr)
881       -               rule_obj.source_ip = str(rule_obj.source_ip) + '/'+ str(wildcard_mask)
     881  +               rule_obj.source_ip = str(rule_obj.get_field_value('source_ip')) + '/'+
```



```
def acl_service_discovery(smodelctx, sdata, sr_device, **kwargs):
    inputdict = kwargs['inputdict']
    if sdata.isServiceDiscoveryEnabled() == True:
        access_list_obj = sr_device.url+"/acl:access-lists/access-list=%s"%(inputdict['na
        access_list = yang.Sdk.getData(access_list_obj, '', sdata.getTaskId())
        obj = util.parseXmlString(access_list)
        if hasattr(obj.access_list,'acl_rules'):
            acl_rules_obj = sr_device.url+"/acl:access-lists/access-list=%s/acl-rules"%(i
            acl_rules = yang.Sdk.getData(acl_rules_obj, '', sdata.getTaskId())
            acl_rule = util.convert_to_list(acl_rules)
            for aclrule in acl_rule:
                rule = util.parseXmlString(aclrule)
                rules_obj = util.convert_to_list(rule.acl_rules.acl_rule)
                for rule_obj in rules_obj:
                    uri = sdata.getRcPath()
                    if util.isEmpty(rule_obj.get_field_value('name')):
                        rule_obj.name = ""
                    if util.isEmpty(rule_obj.get_field_value('action')):
                        rule_obj.action = ""
                    if util.isEmpty(rule_obj.get_field_value('layer4protocol')):
                        rule_obj.layer4protocol = ""
                    if util.isEmpty(rule_obj.get_field_value('source_condition_type')):
                        rule_obj.source_condition_type = ""
```

Sorting of Create/Delete commands:

After service instantiated sometimes commands will be generated in the wrong order in task details, due to this operations can fail to execute on device. To overcome the failure follow below procedure.

Example code :

Suppose devices will accept **'CreateQPolicyMap'** command operations first and after that **'CreateInterface'**, **'UpdateInterface'** command operations but commands generated in reverse order in service task details, we need to write sorting for the Create/Delete commands like below.

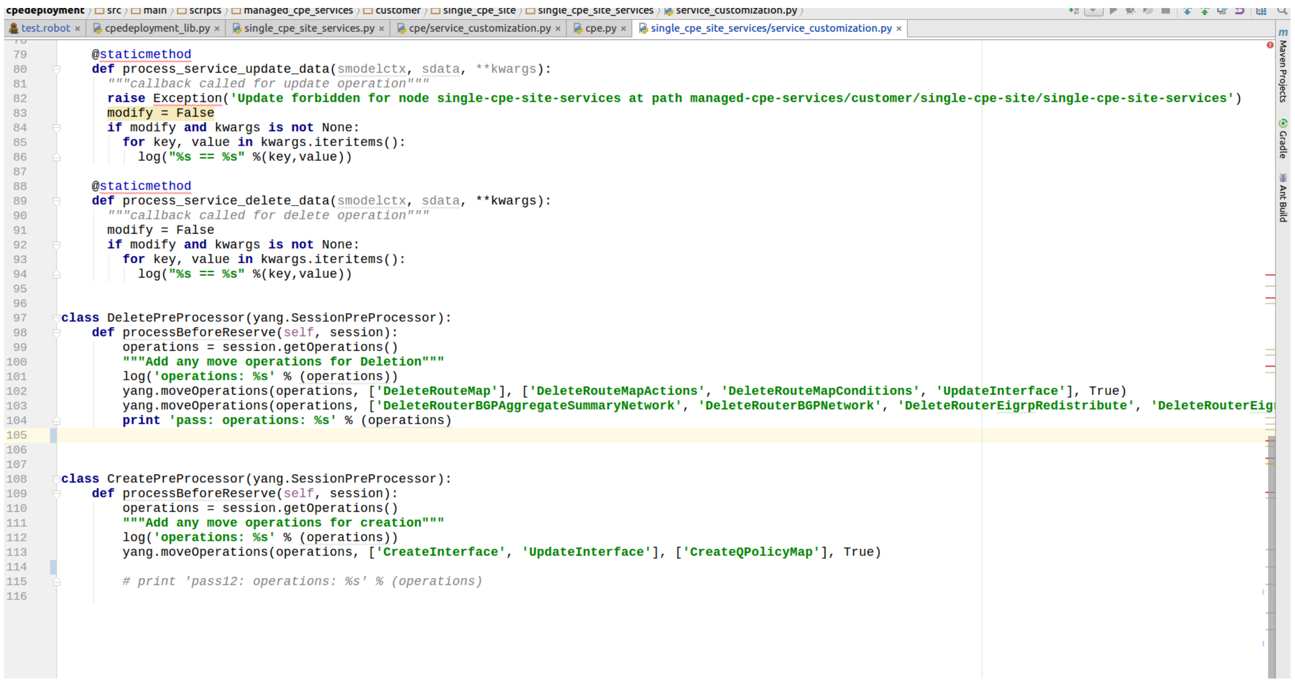
This class is already present in respective service_customization files. User needs to add the move or delete operation like shown below.

```
class CreatePreProcessor(yang.SessionPreProcessor):
  def processBeforeReserve(self, session):
    operations = session.getOperations()
    """Add any move operations for creation"""
    log('operations: %s' % (operations))
    yang.moveOperations(operations, ['CreateInterface', 'UpdateInterface'], ['CreateQPolicyMap'], True)
```

Explanation for example code:

In above code **CreateQPolicyMap** command will come before **CreateInterface or UpdateInterface** command.

For deletion commands also same procedure.



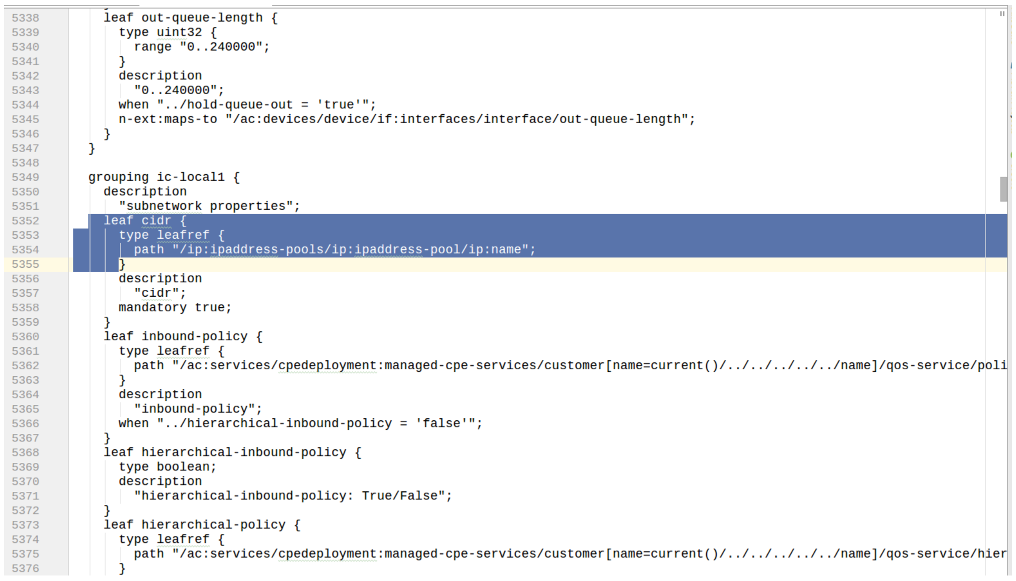
cpedeployment 〉 src 〉 main 〉 scripts 〉 managed_cpe_services 〉 customer 〉 single_cpe_site 〉 single_cpe_site_services 〉 service_customization.py 〉

test.robot × | cpedeployment_lib.py × | single_cpe_site_services.py × | cpe/service_customization.py × | cpe.py × | single_cpe_site_services/service_customization.py ×

```
    @staticmethod
    def process_service_update_data(smodelctx, sdata, **kwargs):
      """callback called for update operation"""
      raise Exception('Update forbidden for node single-cpe-site-services at path managed-cpe-services/customer/single-cpe-site/single-cpe-site-services')
      modify = False
      if modify and kwargs is not None:
        for key, value in kwargs.iteritems():
          log("%s == %s" %(key,value))

    @staticmethod
    def process_service_delete_data(smodelctx, sdata, **kwargs):
      """callback called for delete operation"""
      modify = False
      if modify and kwargs is not None:
        for key, value in kwargs.iteritems():
          log("%s == %s" %(key,value))


class DeletePreProcessor(yang.SessionPreProcessor):
    def processBeforeReserve(self, session):
        operations = session.getOperations()
        """Add any move operations for Deletion"""
        log('operations: %s' % (operations))
        yang.moveOperations(operations, ['DeleteRouteMap'], ['DeleteRouteMapActions', 'DeleteRouteMapConditions', 'UpdateInterface'], True)
        yang.moveOperations(operations, ['DeleteRouterBGPAggregateSummaryNetwork', 'DeleteRouterBGPNetwork', 'DeleteRouterEigrpRedistribute', 'DeleteRouterEig
        print 'pass: operations: %s' % (operations)



class CreatePreProcessor(yang.SessionPreProcessor):
    def processBeforeReserve(self, session):
        operations = session.getOperations()
        """Add any move operations for creation"""
        log('operations: %s' % (operations))
        yang.moveOperations(operations, ['CreateInterface', 'UpdateInterface'], ['CreateQPolicyMap'], True)

        # print 'pass12: operations: %s' % (operations)

```

IPAM Pools integration with services:

To integrate the IP address pools with services, make sure that user can add code like below in services.yang



```
      leaf out-queue-length {
        type uint32 {
          range "0..240000";
        }
        description
          "0..240000";
        when "../hold-queue-out = 'true'";
        n-ext:maps-to "/ac:devices/device/if:interfaces/interface/out-queue-length";
      }
    }

    grouping ic-local1 {
      description
        "subnetwork properties";
      leaf cidr {
        type leafref {
          path "/ip:ipaddress-pools/ip:ipaddress-pool/ip:name";
        }
        description
          "cidr";
        mandatory true;
      }
      leaf inbound-policy {
        type leafref {
          path "/ac:services/cpedeployment:managed-cpe-services/customer[name=current()/../../../../../name]/qos-service/poli
        }
        description
          "inbound-policy";
        when "../hierarchical-inbound-policy = 'false'";
      }
      leaf hierarchical-inbound-policy {
        type boolean;
        description
          "hierarchical-inbound-policy: True/False";
      }
      leaf hierarchical-policy {
        type leafref {
          path "/ac:services/cpedeployment:managed-cpe-services/customer[name=current()/../../../../../name]/qos-service/hier
        }
```

After adding the leaf in yang, user will generate the code by using SDK, but in codegen bindings of IPAM related methods will not be generated automatically if extensions are not used properly. Users can add below code in some lib.py file and import those definitions wherever needed.

- Below method is used to get the used ips from ipaddresspool.

```
def get_used_ip_list_from_ippool(ipaddress_pool_name, sdata):
    print "inside get_used_ip_list_from_ippool"
    ipaddress_pool_name = util.make_interfacename(ipaddress_pool_name)
    ipaddress_pool_name = ipaddress_pool_name.replace(' ', '%20')
    ip_used_list = []
    get_ipaddress_pool_url = "/app/restconf/data/ipam:ipaddress-pools/ipaddress-pool=%s"
%(ipaddress_pool_name)
    pool = yang.Sdk.getData(get_ipaddress_pool_url, '', sdata.getTaskId())
    pool = util.parseXmlString(pool)
    if hasattr(pool.ipaddress_pool, 'ipaddress_entries'):
        get_ipaddress_pool_entries_url =
"/app/restconf/data/ipam:ipaddress-pools/ipaddress-pool=%s/ipaddress-entries" %(ipaddress_pool_name)
        entries = yang.Sdk.getData(get_ipaddress_pool_entries_url, '', sdata.getTaskId())
        entries = util.parseXmlString(entries)
        #print "list of ip_address_pool_entries is:", entries

        if hasattr(entries.ipaddress_entries, 'ipaddress_entry'):
            for entry in util.convert_to_list(entries.ipaddress_entries.ipaddress_entry):
                ip_used_list.append(entry.ipaddress)

    return ip_used_list
```

- Below method is used to add the ipaddress entries under ipaddress pools.

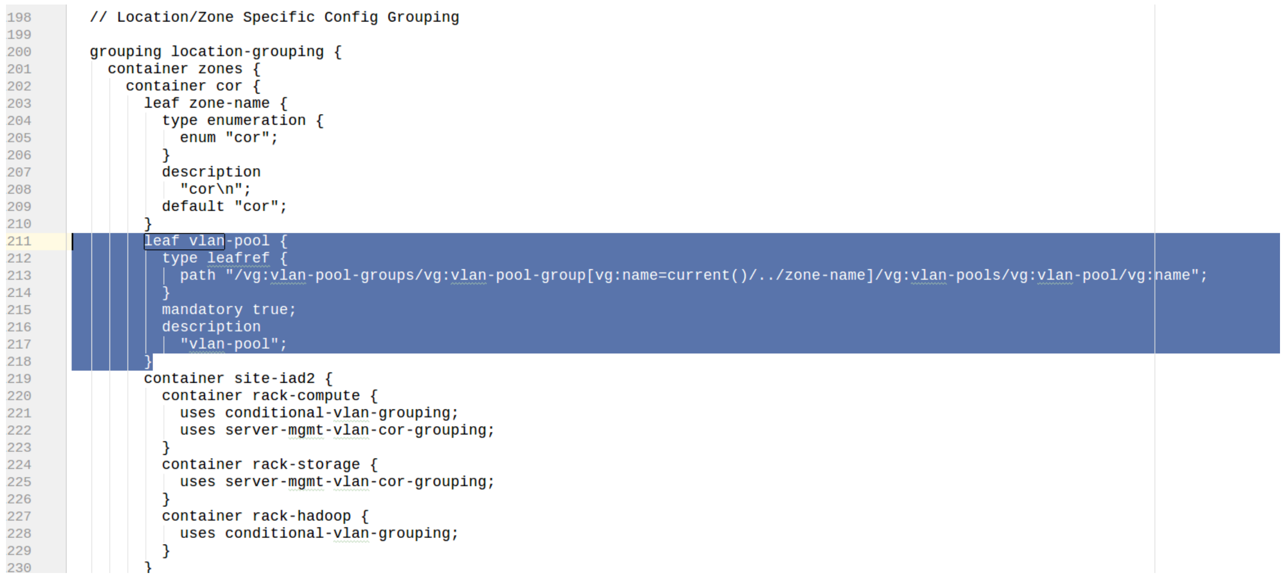
```
def add_ipaddress_entries(ipaddress_pool_name, ip_address,sdata):
    payload = '<ipaddress-entries/>'
    ipaddress_pool_name = util.make_interfacename(ipaddress_pool_name)
    ipaddress_pool_name = ipaddress_pool_name.replace(' ', '%20')
    url = '/app/restconf/data/ipam:ipaddress-pools/ipaddress-pool=%s'%(ipaddress_pool_name)
    yang.Sdk.createData(url, payload, sdata.getSession(), False)

    ## Update used count
    payload_ippool = '''<ipaddress-entry>
                <ipaddress>'''+ip_address+'''</ipaddress>
                <name>'''+ipaddress_pool_name+'_'+ip_address+'''</name>
                </ipaddress-entry>'''
    yang.Sdk.createData("/app/restconf/data/ipam:ipaddress-pools/ipaddress-pool=%s/ipaddress-entries"
%(ipaddress_pool_name), payload_ippool, sdata.getSession())
```

- Below method is used to get the free ips from cidr.

```
def get_freeip_from_cidr(cidr, used_list):
    print "inside get_freeip_from_cidr"
    cidr_obj = util.IPPrefix(cidr)
    gateway_ip = cidr_obj.gateway_ip()
    #used_list.sort()
    #ip_address = used_list[0]
    #last_ip_address = used_list[used_list.__len__()-1]

    network_given = IPNetwork(cidr)
    (addrStr, cidrStr) = cidr.split('/')
    addr = addrStr.split('.')
    cidr = int(cidrStr)
    mask = [0, 0, 0, 0]
    for i in range(cidr):
        mask[i/8] = mask[i/8] + (1 << (7 - i % 8))
    net = []
    for i in range(4):
        net.append(int(addr[i]) & mask[i])
    network = ".".join(map(str, net))
    ip_address = network

    print "gateway_ip for /32 cidr", gateway_ip
    print "cidr_obj.masklen is", cidr_obj.masklen
    if str(cidr_obj.masklen) == str(32):
        return gateway_ip
    else:
        gateway_ip = util.next_ip_address(ip_address)
        print "gateway_ip is:", gateway_ip
        while (True):
            if gateway_ip not in used_list:
                break
            else:
                gateway_ip = util.next_ip_address(gateway_ip)
        print "final gateway_ip is :", gateway_ip
        ip = IPAddress(gateway_ip)
        if not network_given.Contains(ip):
            raise Exception('Invalid IP address for this cidr')
        return gateway_ip
```

VLAN Pools integration with services:

To integrate the VLAN pools with services, make sure that user should add code like below in services.yang



```
// Location/Zone Specific Config Grouping

grouping location-grouping {
  container zones {
    container cor {
      leaf zone-name {
        type enumeration {
          enum "cor";
        }
        description
          "cor\n";
        default "cor";
      }
      leaf vlan-pool {
        type leafref {
          path "/vg:vlan-pool-groups/vg:vlan-pool-group[vg:name=current()/../zone-name]/vg:vlan-pools/vg:vlan-pool/vg:name";
        }
        mandatory true;
        description
          "vlan-pool";
      }
      container site-iad2 {
        container rack-compute {
          uses conditional-vlan-grouping;
          uses server-mgmt-vlan-cor-grouping;
        }
        container rack-storage {
          uses server-mgmt-vlan-cor-grouping;
        }
        container rack-hadoop {
          uses conditional-vlan-grouping;
        }
      }
```

After adding the leaf in yang, user will generate the code by using SDK, but in codegen bindings of VLAN pools related methods may not be generated automatically based on the extension used. Users can add below code in some lib.py file and call those definitions when needed.

Below method is used to allocate the VLANs in a VLAN pool.

```
def allocate_vlan(dev, obj, sdata, pool, group, count=1):
  print "poolname %s groupname %s"%(pool, group)
  if util.isNotEmpty(count):
   if util.isEmpty(pool):
     raise Exception('Vlan pool cannot be empty')


   allocated_list = []
   for i in range(0,int(count)):
     if i == 0:
       addr = vlanpool.allocate_vlan(group, pool)
     else:
       addr = vlanpool.allocate_vlan(group, pool, addr+1)
     allocated_list.append(addr)
     vlans_object = devices.device.vlans.vlans()
     vlans_vlan_object = vlans_object.vlan.add(id=addr)

     vlans_object_payload = vlans_object.getxml(filter=True)
     log('vlans_object_payload: %s' % (vlans_object_payload))
   return allocated_list
```