Automatically Creating a Netconf Mount in APPC from SDNC

UNDER CONSTRUCTION

Introduction

Tutorials on how to use SDNC and the various nodes in the service logic interpreter would seem to be useful to the community.

The vFW demo use case in ONAP requires APPC to have a Netconf Mount to the traffic generator. That mount is created by a step in the robot framework but is actually an interesting way to demonstrate capabilities in SDNC since the data needed to create the netconf mount point all exists in SDNC. This also demonstrates the type of changes that can be done outside of an SDNC release with just Directed Graph changes.

This tutorial starts with a running ONAP platform configuration and would enhance the existing vFW use case. Instructions and background on the ONAP installation, use case, and interaction are at https://wiki.onap.org/display/DW/Installing+and+Running+the+ONAP+Demos.

The tutorial will cover a variety of topics as we create and implement a modified directed graph.

- Concepts based on ODL as controller framework
  - NetConf/Yang
    - Yang model loaded at run time
    - Operational/Config Tree
  - Karaf Container
    - Log Files
    - Log Level Commands
  - API Explorer
    - Execute graph RPC
    - Check netconf mounts
    - Json input/output
  - SLI Extensions to ODL
    - DG Language
      - Lexical Conventions
      - Context Memory
      - Variables
    - Builtin Node Types
      - FileRecorder
      - RestAPICall
        - XML Template
      - Record
      - Switch
      - Substr
      - Block
      - Set
      - For
    - Extended Node Types
      - Execute, Resource, ...
      - A&AI example
    - DB of DGs
      - View
      - Status
      - Activate
    - Log Files
    - DGBUILDER
      - Import Json
      - Edit
      - Save
      - Activate DG
  - ONAP Network and Application Controllers built on ODL
    - Base and Yang model driven APIs
      - vnf-topology
        - activate
        - delete
        - rollback
      - Modify Configuration
      - Netconf Mount

Specifically this tutorial will demonstrate how to modify the VNF Topology directed graphs to manage (create, delete) the netconf mount point between APPC and the vTrafficGenerator of the vFW VNF.

The steps will demonstrate activities on the DGBuilder as well as how to use both the FileRecorder node and the RestApiCallNode in the directed graph. None of the steps in the tutorial require a rebuild of SDNC or APPC.

Steps to remove the netconf mount created in the robot test framework are not shown but we may work with the robot framework team to show those as well as a later update.

Loading and Editing Directed Graphs

The first step is to import copies of the two directed graphs that we will want to modify into two tabs in dgbuilder.

The vnf-topology-operation API/Directed Graph uses the svc-action parameter (<switch test="`$vnf-topology-operation-input.sdnc-request-header.svc-action`">) to run either the activate or delete directed graph (a production version might want to look at the other svc-actions as well).

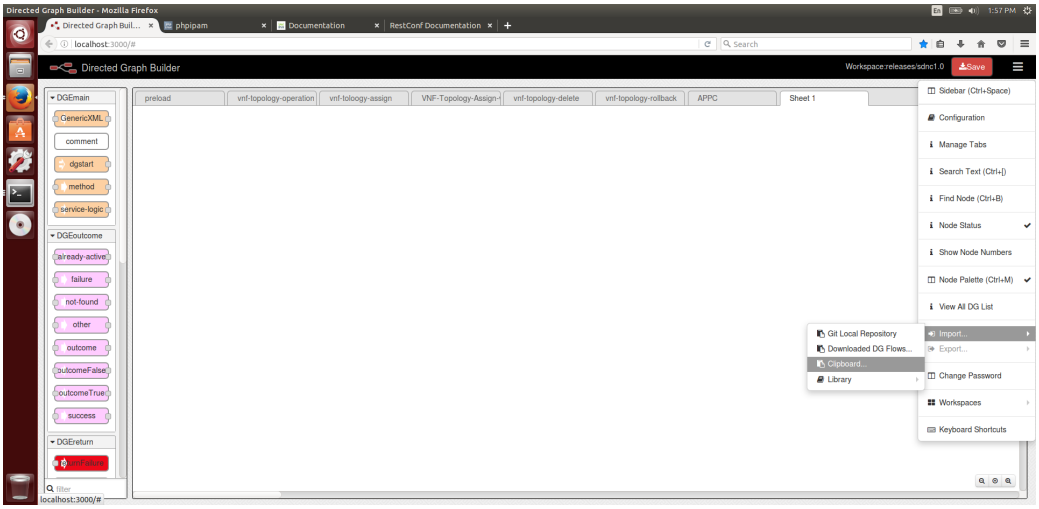
We need the two directed graphs; vnf-topology-activate and vnf-topology-delete to handle the two events that should affect the netconf mount on APPC (create after instantiating a VNF and delete after removing a VNF)

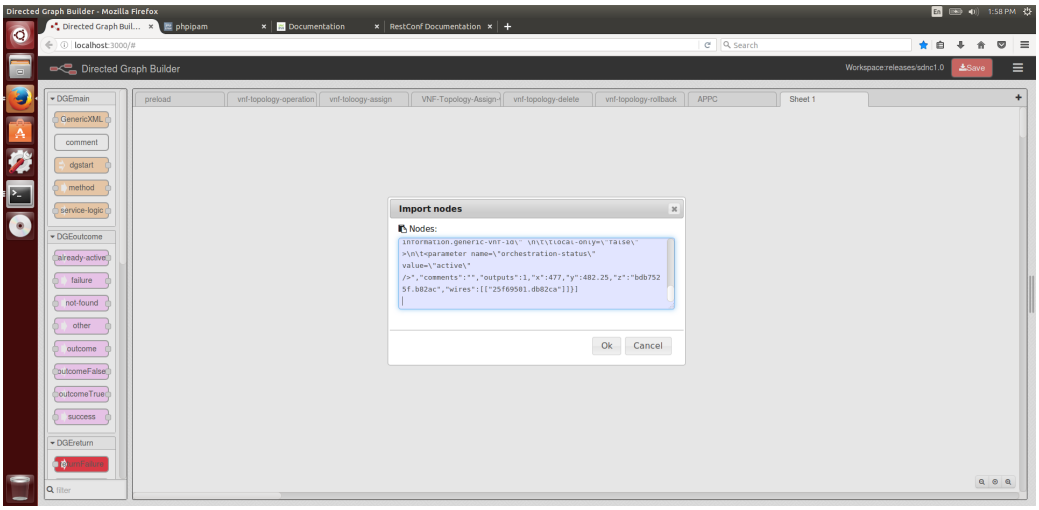
The graphs are in the SDNC repository under <ONAP_SRC>/sdnc/oam/platform-logic/vnfapi/src/main/json.

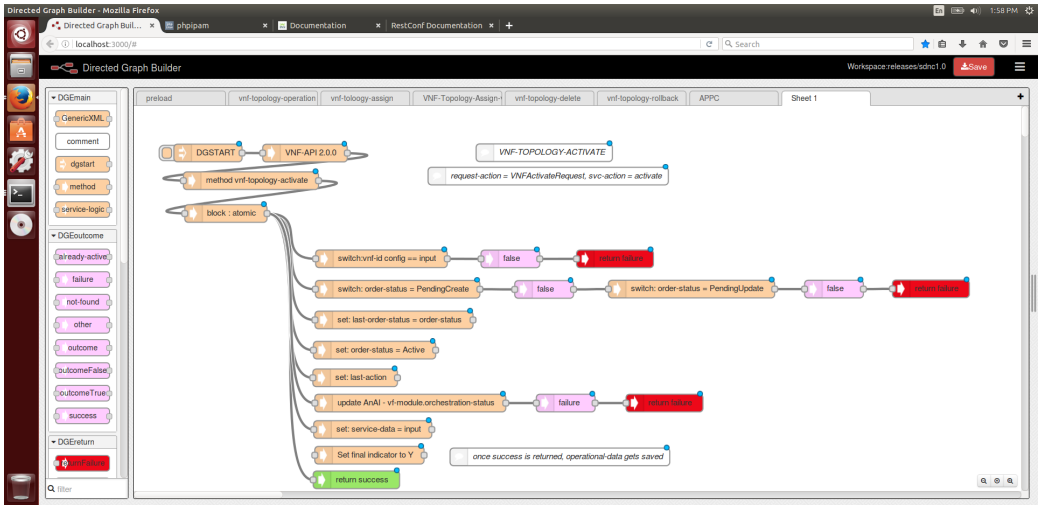
vnf-topology-activate.json	**https://gerrit.onap.org/r/gitweb?p=sdnc/oam.git;a=blob;f=platform-logic/vnfapi/src/main/json/vnf-topology-activate.json;h=87f11f2c256e9366a88715f33087db22ce7c9c9d;hb=refs/heads/release-1.0.0**
vnf-topology-delete.json	https://gerrit.onap.org/r/gitweb?p=sdnc/oam.git;a=blob;f=platform-logic/vnfapi/src/main/json/vnf-topology-delete.json;h=aee34c11968b42bcf2111ab29561038c653a0db3;hb=refs/heads/release-1.0.0

What you need to do is access the DGBuilder container, select a tab or hit "+" to create a new tab , copy the json string of the DG to your clipboard and then use the menu to import the json string from the clipboard.

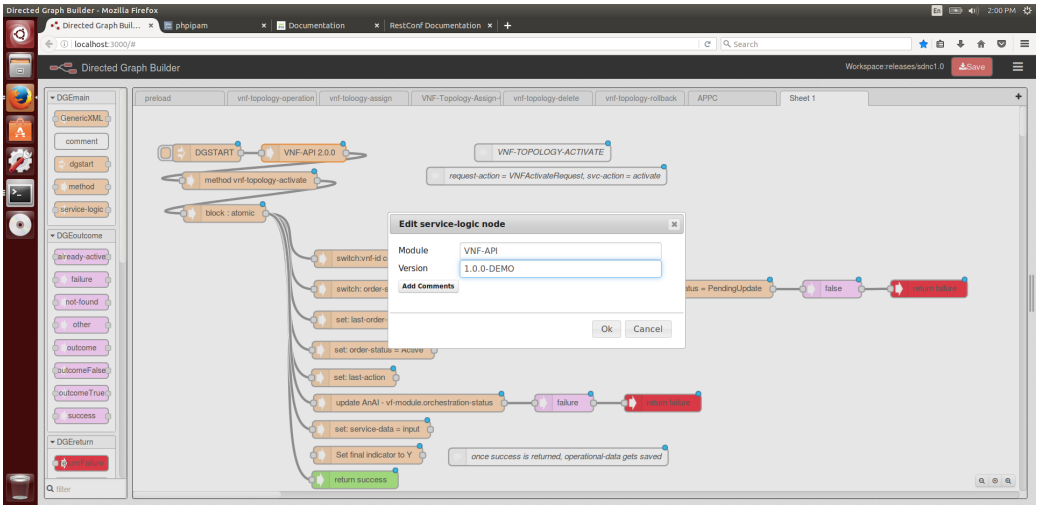
Access the DGBuilder on port 3000 on the SDNC VM (IP address will vary for your environment since you will be coming in on the public side of the controller not the private Net 10 address)



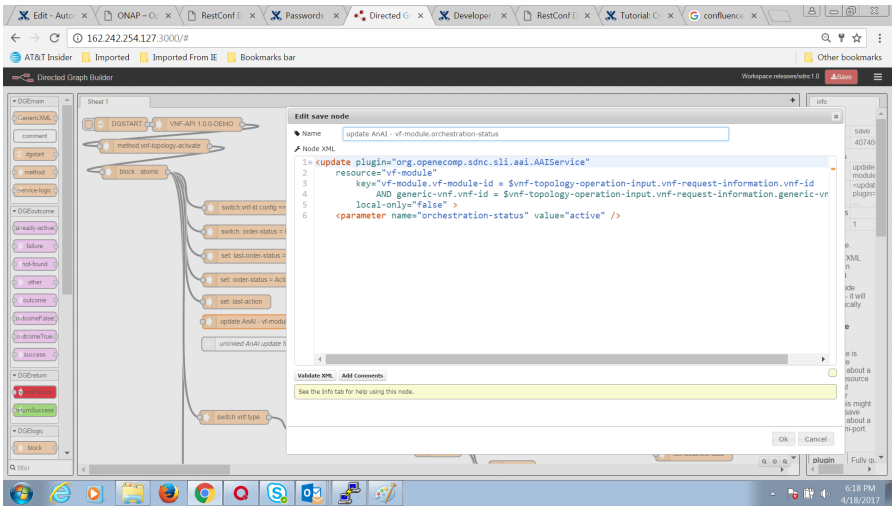




Change the version to 1.0.0-demo or something to distinguish it from the current active DG.



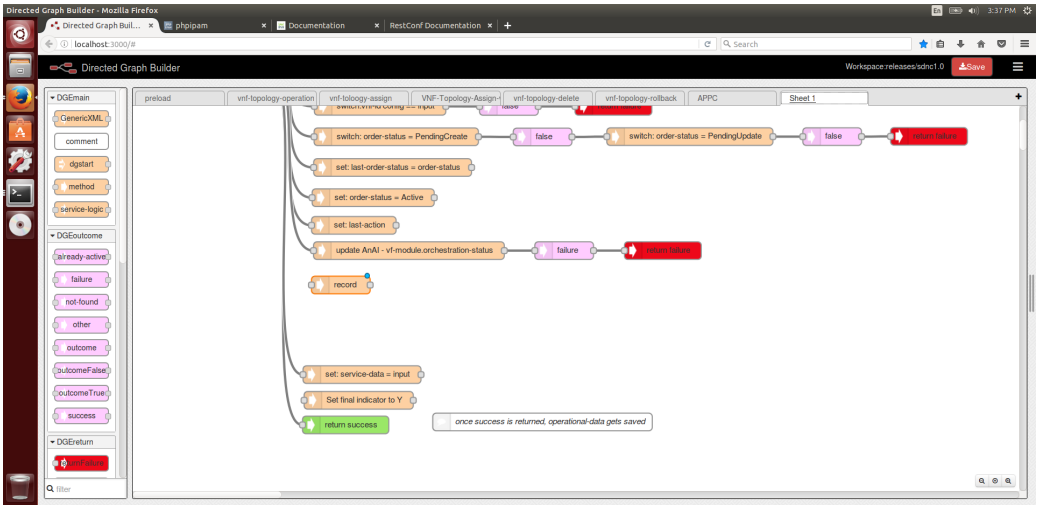
You also may need to change the execute node references for the AnAI update to refer to org.openecomp.sndc.sli.aai.AAIService instead of the com.att.sdnctl package.



Remember to hit the big red SAVE button in the upper left corner.

Now lets add a FileRecorder node. Move the nodes below the update AnAI node and paste in a Record node. (After testing you should move this node to the Block we are creating below for the base_vfw case so that you only record if it a vFW but its a handy debugging to have a file record at the start of our new steps)

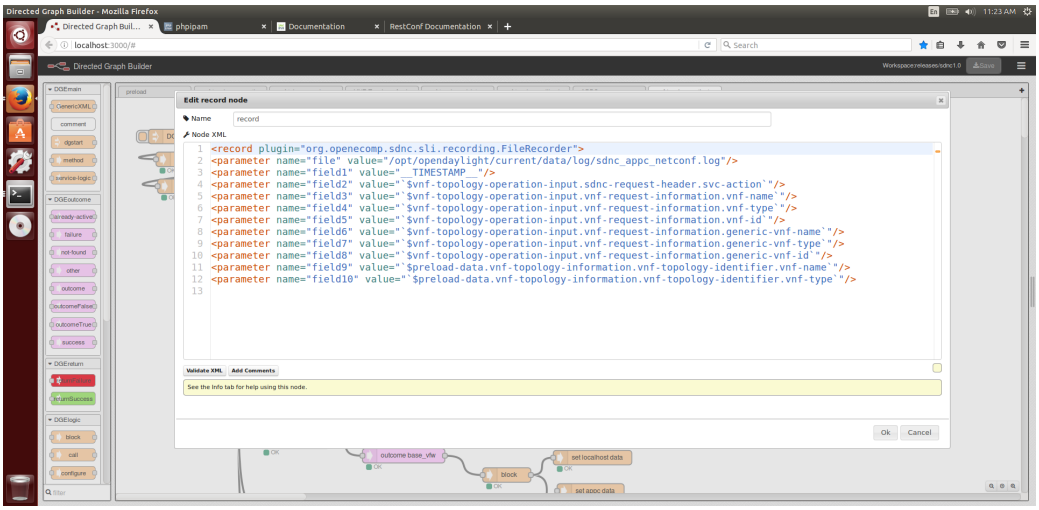
The easiest method to copy one from an existing DG like the vfn-topology-operation



Change the name of the file to "sdnc_appc_netconf.log" so that it is in a separate file from svclogic.log and karaf.log.

Delete attributes we dont care about and renumber so the fields are in order.

```
<record plugin="org.openecomp.sdnc.sli.recording.FileRecorder">
<parameter name="file" value="/opt/opendaylight/current/data/log/sdnc_appc_netconf.log"/>
<parameter name="field1" value="__TIMESTAMP__"/>
<parameter name="field2" value="`$vnf-topology-operation-input.sdnc-request-header.svc-action`"/>
<parameter name="field3" value="`$vnf-topology-operation-input.vnf-request-information.vnf-name`"/>
<parameter name="field4" value="`$vnf-topology-operation-input.vnf-request-information.vnf-type`"/>
<parameter name="field5" value="`$vnf-topology-operation-input.vnf-request-information.vnf-id`"/>
<parameter name="field6" value="`$vnf-topology-operation-input.vnf-request-information.generic-vnf-name`"/>
<parameter name="field7" value="`$vnf-topology-operation-input.vnf-request-information.generic-vnf-type`"/>
<parameter name="field8" value="`$vnf-topology-operation-input.vnf-request-information.generic-vnf-id`"/>
<parameter name="field9" value="`$preload-data.vnf-topology-information.vnf-topology-identifier.vnf-name`"/>
<parameter name="field10" value="`$preload-data.vnf-topology-information.vnf-topology-identifier.vnf-type`"/>
```

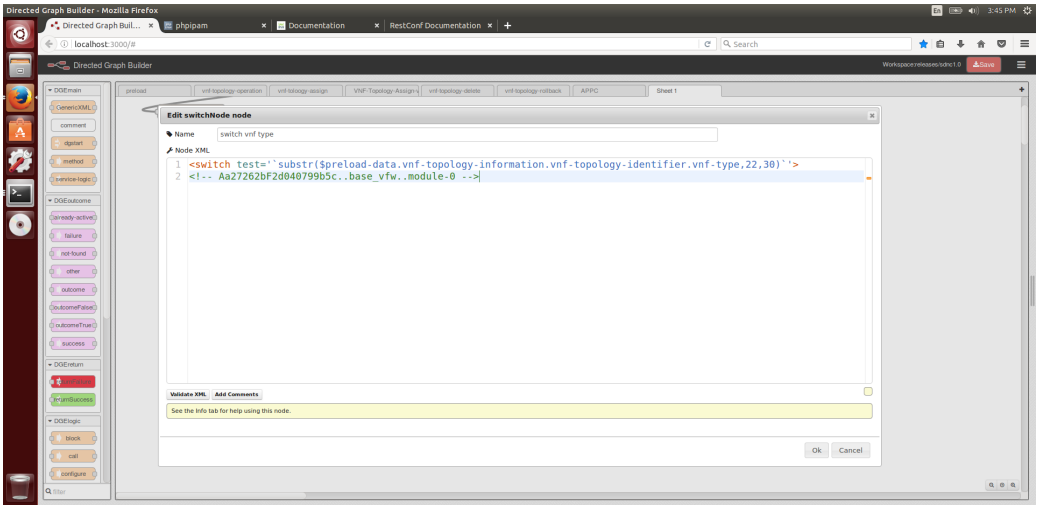


After the Record node add a Switch node so that we can filter on vFW VNF's only.

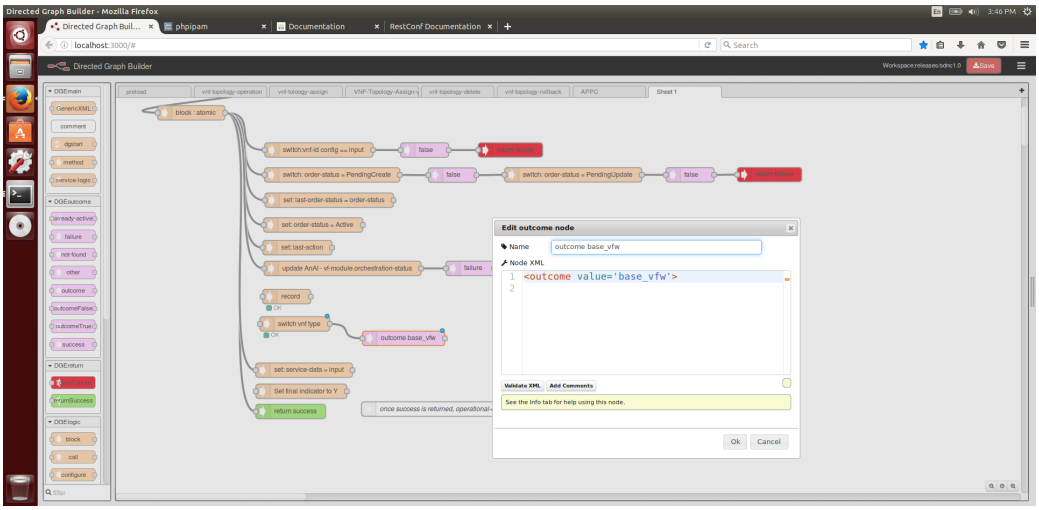
We will use the substr operator to pull out the 8 characters we care about in the vnf-type to detect when its a vfw.

There are probably better methods but this was easy to put in for our use case.

<switch test='`substr($preload-data.vnf-topology-information.vnf-topology-identifier.vnf-type,22,30)`'>
<!-- Aa27262bF2d040799b5c..base_vfw..module-0

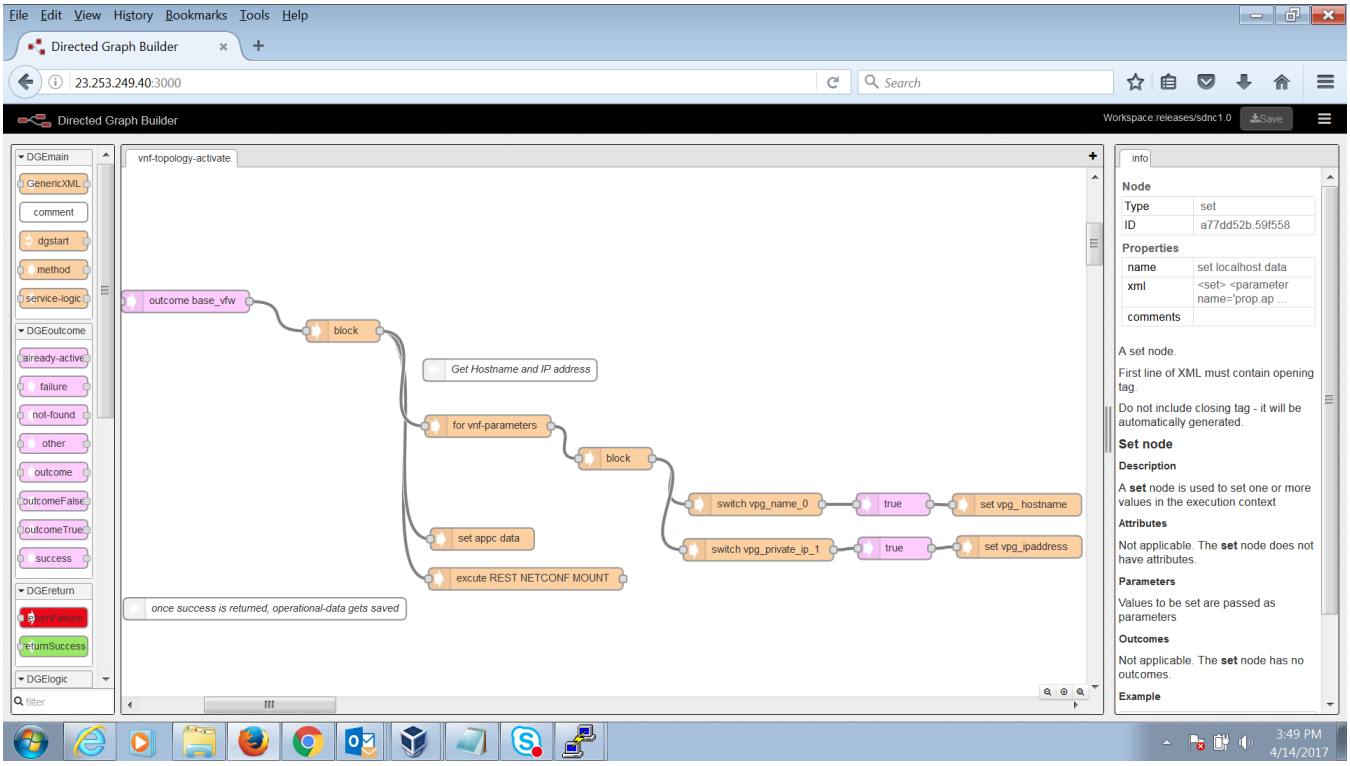


Next add outcome node with value="base_vfw" and link it to the switch node.



Next add a block node.

We are going to add some logic to find the vpg hostname and ip address from the preload-data , a set node so that we can set some properties and then the restapicall node..



FIND VPG Data

The first node in the block is a for loop that starts at 0 and ends with the length of the vnf-parameters array in the preload data.

<for index='k' start='0' end="`$preload-data.vnf-topology-information.vnf-parameters_length`" >

we then use another block node so that we can switch on the vnf-parameter-name of interest and pull out the vpg_hostname and vpg_ipaddress.

Pay special attention to the back ticking to refer to context memory variables.

SWITCH NODE:

<switch test="`$preload-data.vnf-topology-information.vnf-parameters[$k].vnf-parameter-name == 'vpg_name_0'`">

OUTCOME:

<outcome value='true'>

SET NODE

<set>
<parameter name='prop.vpg_hostname' value=`$preload-data.vnf-topology-information.vnf-parameters[$k].vnf-parameter-value`' />

SWITCH NODE:

<switch test="`$preload-data.vnf-topology-information.vnf-parameters[$k].vnf-parameter-name == 'vpg_private_ip_1'`">

OUTCOME:

<outcome value='true'>

SET NODE

<set>
<parameter name='prop.vpg_ipaddress' value=`$preload-data.vnf-topology-information.vnf-parameters[$k].vnf-parameter-value`' />

Now that we have the traffic generator data we can move on to the rest of the netconf call

SET NODE

The set node will set the parameters needed for the RestAPICallNode.

In a real example you would put this data into the properties file that is read by SDNC or put it into the mysql database but I wanted to demonstrate the set node as an expediant way to do things.

Not that you should replace the <APPC_RESTAPI_PASSWORD> with the correct string.

```
<set>
<parameter name='prop.appcRestApi.url' value='http://appc.api.simpledemo.openecomp.org:8282' />
<!-- 8181 when doing localhost -->
<parameter name="prop.restapi.templateDir" value="/opt/openecomp/sdnc/data"/>
<parameter name="prop.appcRestApi.sdncOdl.user" value="admin"/>
<parameter name="prop.appcRestApi.sdncOdl.password" value="<APPC_RESTAPI_PASSWORD>"/>
```

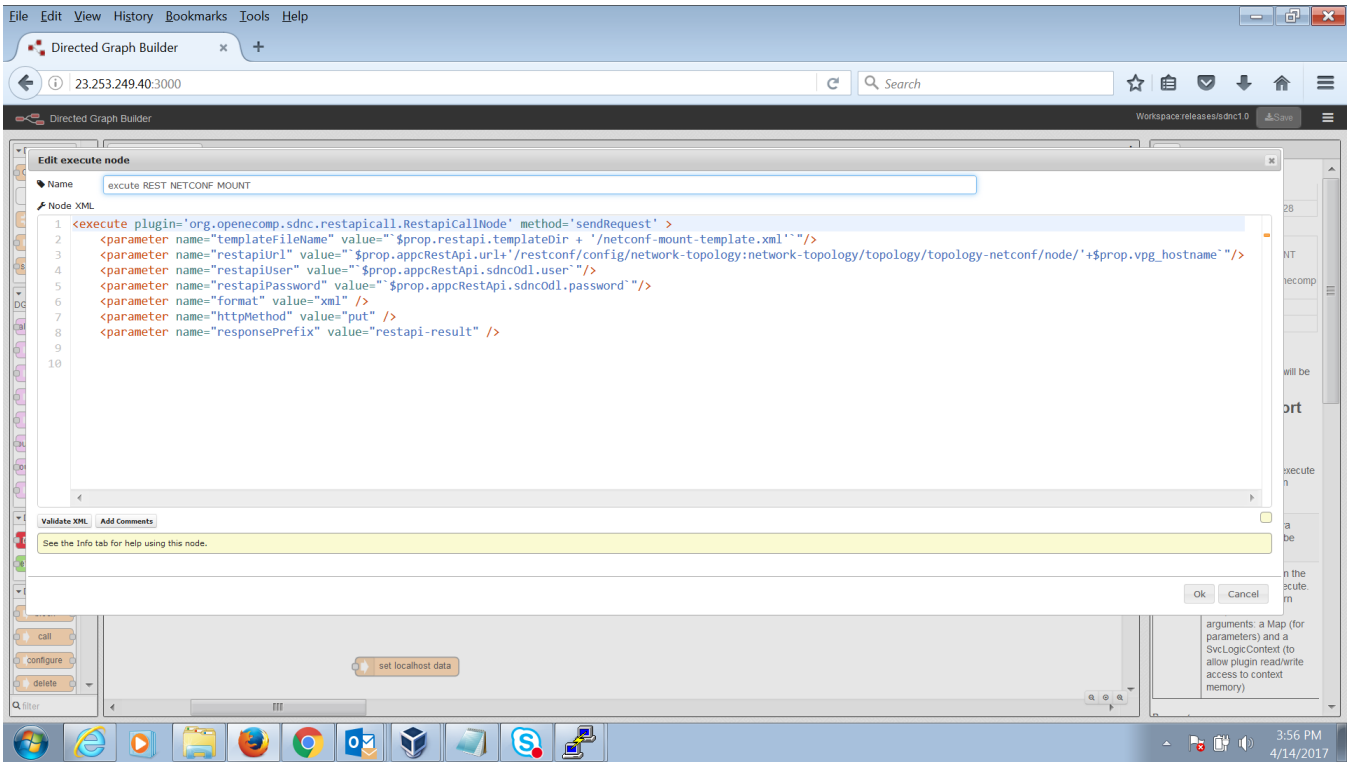
Next we add the Execute node to actually make the REST API Call to the block. Notice that it references the data from set node and template (netconf-mount-template.xml).

We will have to put the template in the directory specified in the prop.restpi.templateDir (/opt/openecom/sdnc/data) on the SDNC container.

The name of the node is the variable we set above $prop.vpg_hostname".

The method is a "PUT" for the creation of the netconf-mount.

```
<execute plugin='org.openecomp.sdnc.restapicall.RestapiCallNode' method='sendRequest' >
 <parameter name="templateFileName" value="`$prop.restapi.templateDir + '/netconf-mount-template.xml'`"/>
 <parameter name="restapiUrl" value="`$prop.appcRestApi.url+'/restconf/config/network-topology:network-topology
/topology/topology-netconf/node/'+$prop.vpg_hostname`"/>
 <parameter name="restapiUser" value="`$prop.appcRestApi.sdncOdl.user`"/>
 <parameter name="restapiPassword" value="`$prop.appcRestApi.sdncOdl.password`"/>
 <parameter name="format" value="xml" />
 <parameter name="httpMethod" value="put" />
 <parameter name="responsePrefix" value="restapi-result" />
```



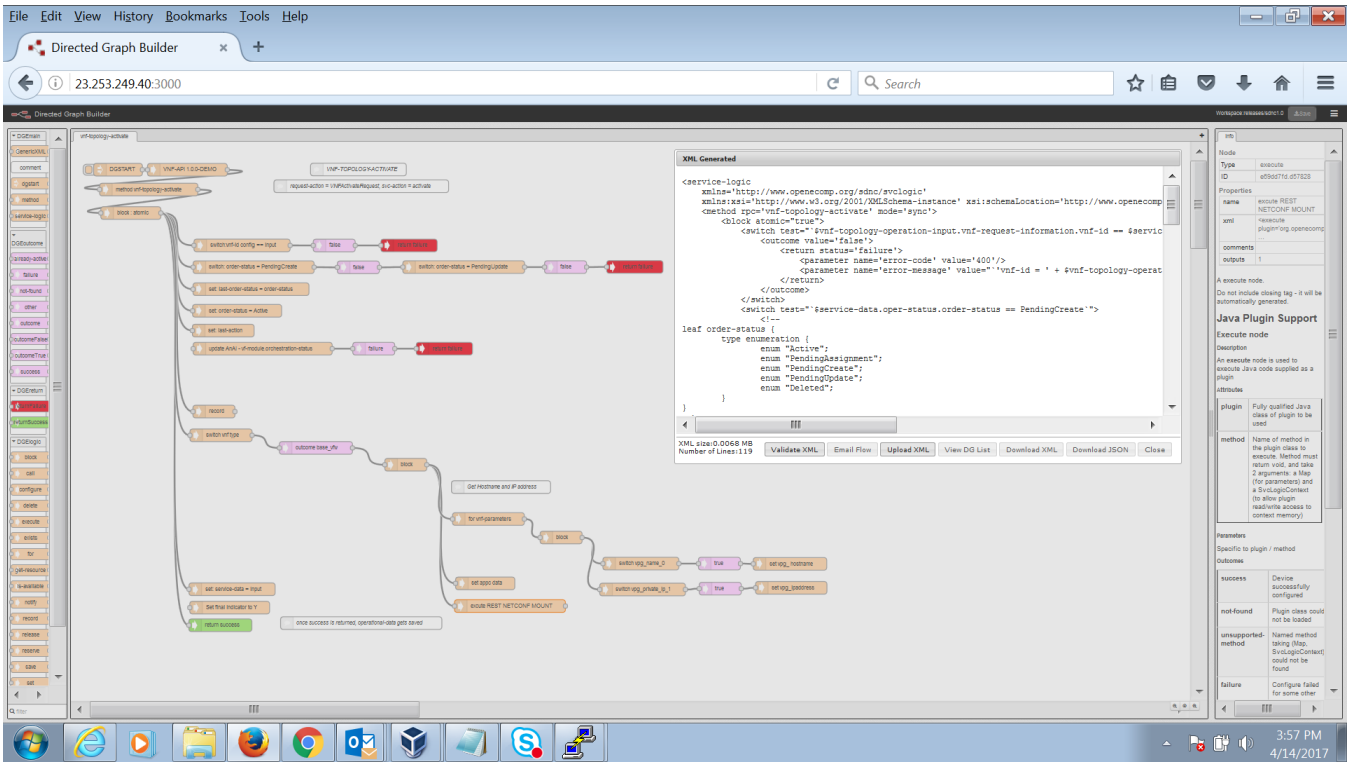
Link the "for" , "record" and "switch" nodes into the block node.

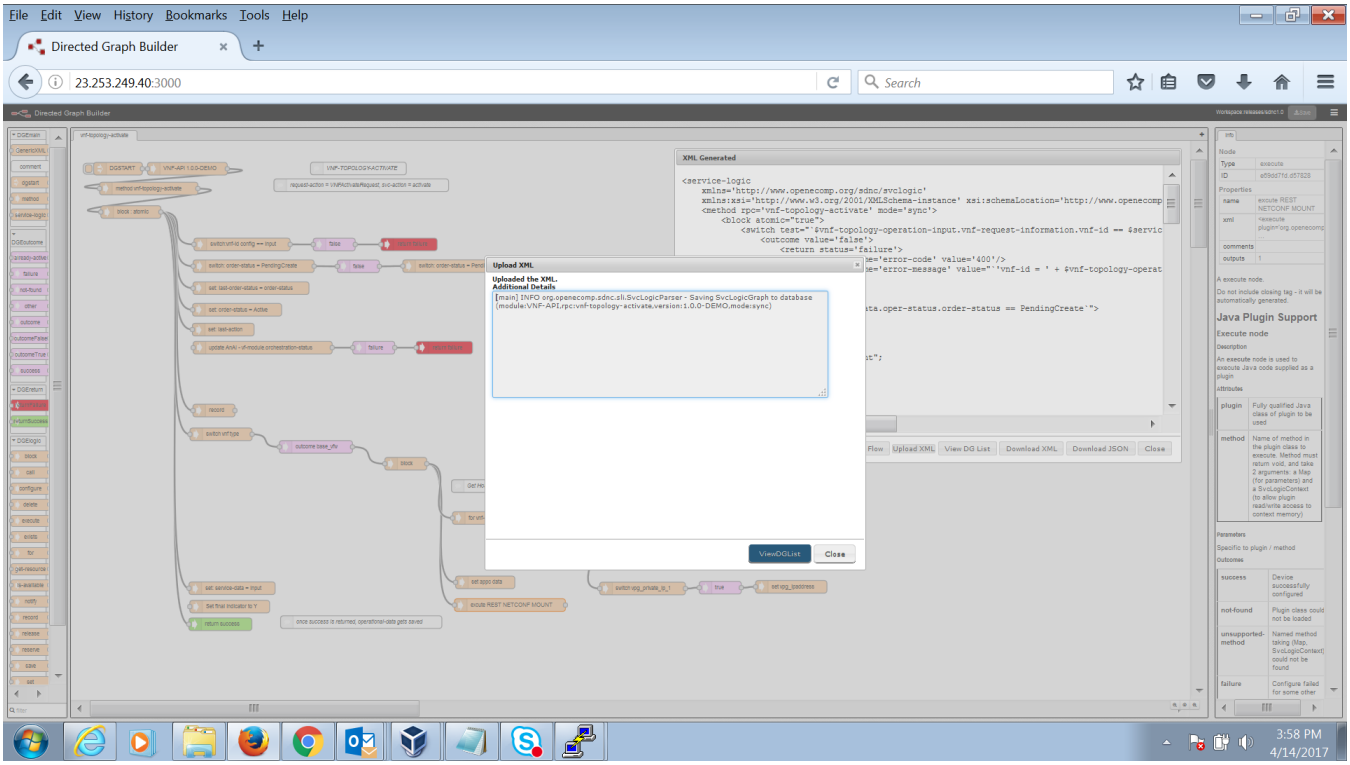
SAVE the DG

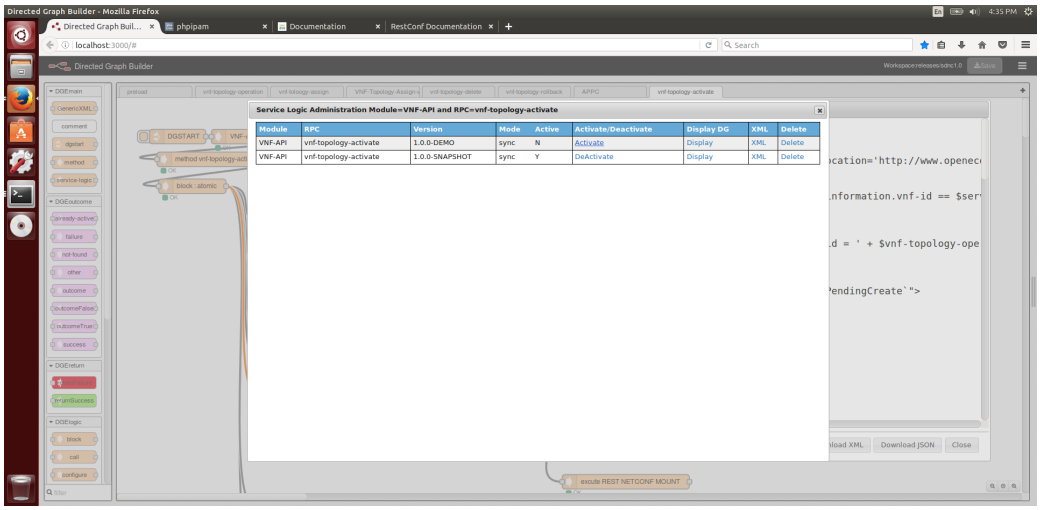
There is a full DG below and in gerrit in case you want to compare it to yours by importing it to another tab in dgbuilder.

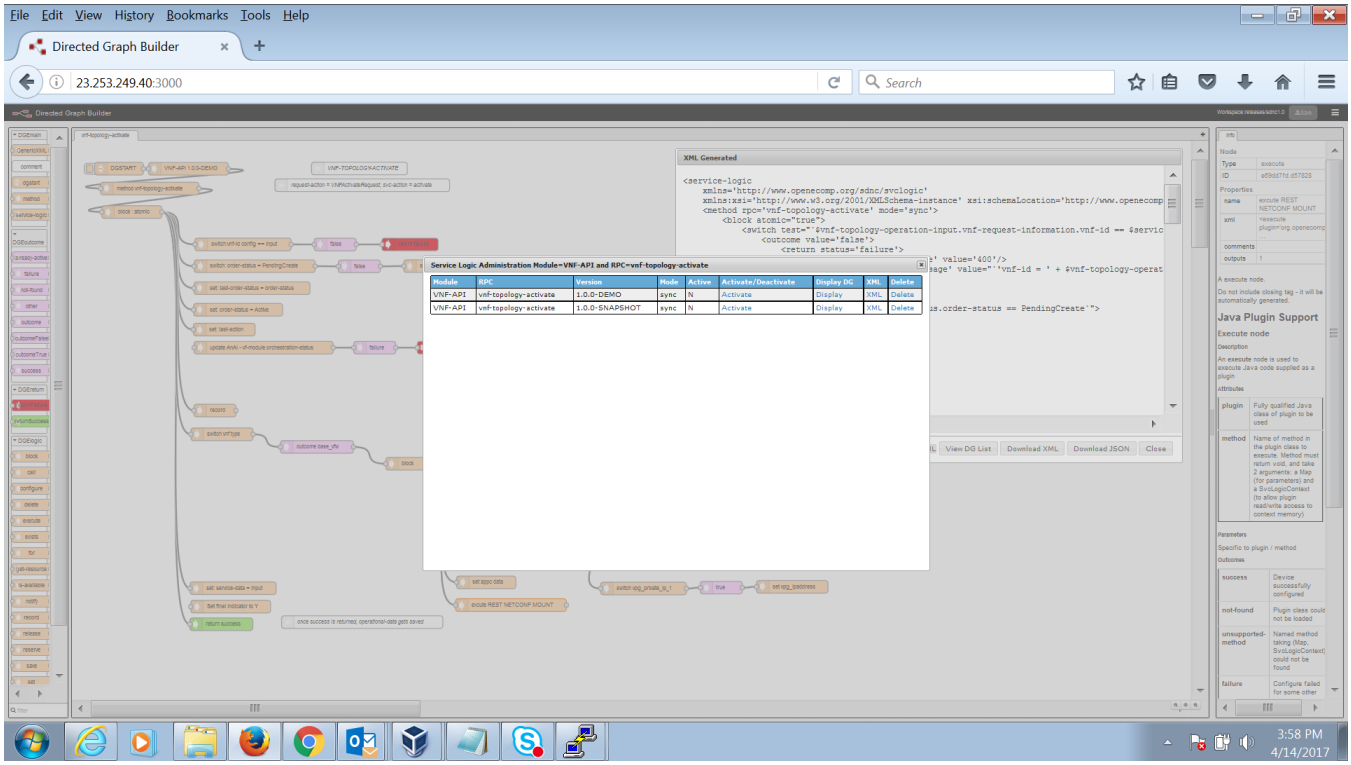
For testing using the SLI:ExecutGraph function I would unhook the AnAI update node from the DG (select and delete the connection between the block and the AnAI node since we will be testing before we have updated AnAI with the VNF data). You can see in the graphic below that the node is in the DG data but not connected to the tree so it won't get executed. We will re-connect the AnAI node to the block after we do the SLI:ExecuteGraph test.

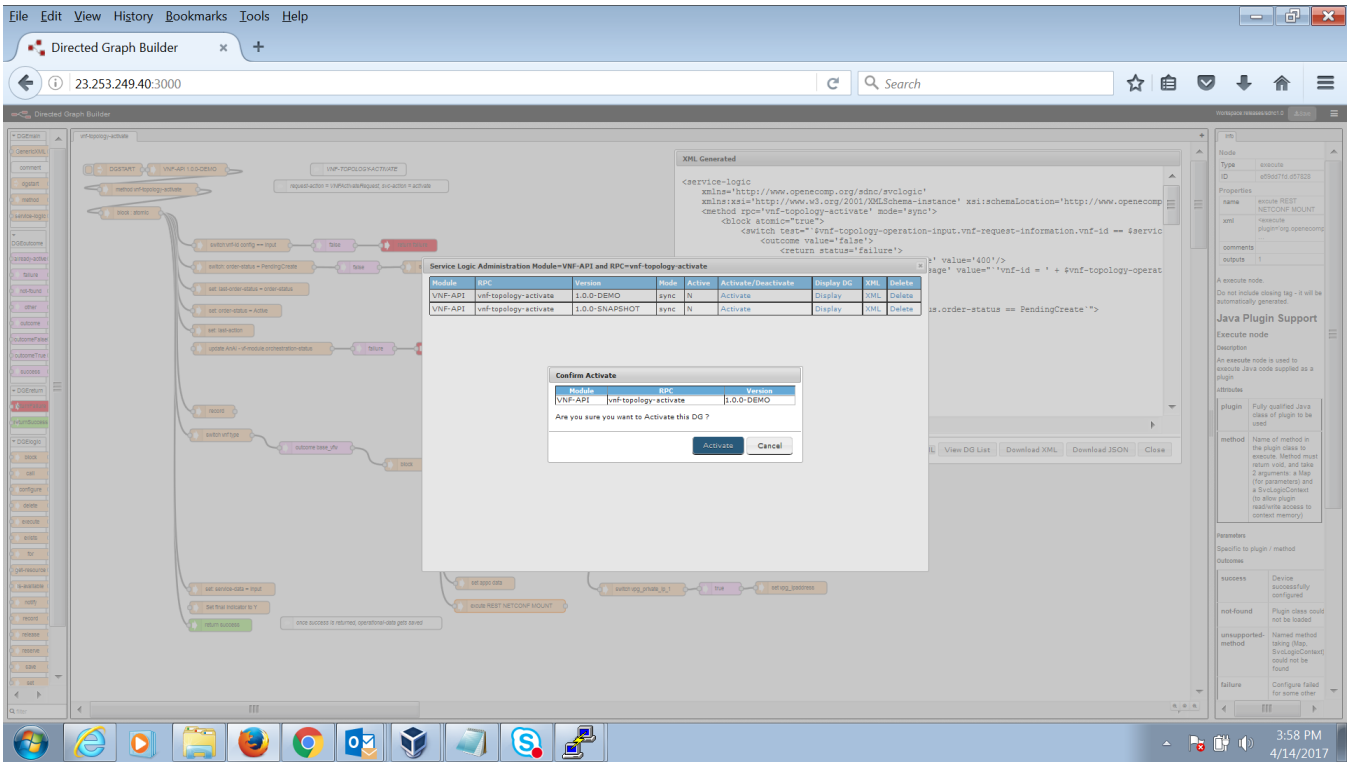
Upload the DG and Activate it.


File Edit View History Bookmarks Tools Help
Directed Graph Builder
23.253.249.40:3000
Search
XML Generated
Validate XML
Email Flow
Upload XML
View DG List
Download XML
Download JSON
Close
Java Plugin Support
Execute node
3:57 PM
4/14/2017


File Edit View History Bookmarks Tools Help
Directed Graph Builder
23.253.249.40:3000
Search
XML Generated
Upload XML
Uploaded the XML.
Additional Details
[main] INFO org.openecomp.sdnc.sli.SvcLogicParser - Saving SvcLogicGraph to database (module:VNF-API,rpc:vnf-topology-activate,version:1.0.0-DEMO,mode:sync)
ViewDGList
Close
Java Plugin Support
Execute node
3:58 PM
4/14/2017


Directed Graph Builder - Mozilla Firefox
Directed Graph Builder
Service Logic Administration Module=VNF-API and RPC=vnf-topology-activate
Module | RPC | Version | Mode | Active | Activate/Deactivate | Display DG | XML | Delete
VNF-API | vnf-topology-activate | 1.0.0-DEMO | sync | N | Activate | Display | XML | Delete
VNF-API | vnf-topology-activate | 1.0.0-SNAPSHOT | sync | Y | DeActivate | Display | XML | Delete
Download XML
Download JSON
Close


File Edit View History Bookmarks Tools Help
Directed Graph Builder
23.253.249.40:3000
Search
Service Logic Administration Module=VNF-API and RPC=vnf-topology-activate
Module | RPC | Version | Mode | Active | Activate/Deactivate | Display DG | XML | Delete
VNF-API | vnf-topology-activate | 1.0.0-DEMO | sync | N | Activate | Display | XML | Delete
VNF-API | vnf-topology-activate | 1.0.0-SNAPSHOT | sync | N | Activate | Display | XML | Delete
XML Generated
Java Plugin Support
3:58 PM
4/14/2017



Installing the REST API payload XML Template

Here are the steps to put the xml template on the sdnc controller docker container. Remember this will get removed if you re-install the docker container.

log into the container with "docker exec -it sdnc_controller_container bash"

cd /opt/openecomp/sdnc/data

and create the file netconf-mount-template.xml with the xml below (everything from <node> to </node> inclusive).

Note that the hostname (prop.vpg_hostname) and IP address (prop.vpg_ipaddress) of the vTrafficGenerator comes from the vnf preload data found during the DG processing.

The rest api call node uses ${variable_name} to indicate a variable from context memory that should be replaced in the template.

```
<node xmlns="urn:TBD:params:xml:ns:yang:network-topology">
 <node-id>${prop.vpg_hostname}</node-id>
 <host xmlns="urn:opendaylight:netconf-node-topology">${prop.vpg_ipaddress}</host>
 <port xmlns="urn:opendaylight:netconf-node-topology">2883</port>
 <username xmlns="urn:opendaylight:netconf-node-topology">admin</username>
 <password xmlns="urn:opendaylight:netconf-node-topology">admin</password>
 <tcp-only xmlns="urn:opendaylight:netconf-node-topology">false</tcp-only>
 <!-- non-mandatory fields with default values, you can safely remove these if you do not wish to override any of
these values-->
 <reconnect-on-changed-schema xmlns="urn:opendaylight:netconf-node-topology">false</reconnect-on-changed-schema>
 <connection-timeout-millis xmlns="urn:opendaylight:netconf-node-topology">20000</connection-timeout-millis>
 <max-connection-attempts xmlns="urn:opendaylight:netconf-node-topology">0</max-connection-attempts>
 <between-attempts-timeout-millis xmlns="urn:opendaylight:netconf-node-topology">2000</between-attempts-timeout-
millis>
 <sleep-factor xmlns="urn:opendaylight:netconf-node-topology">1.5</sleep-factor>
 <!-- keepalive-delay set to 0 turns off keepalives-->
 <keepalive-delay xmlns="urn:opendaylight:netconf-node-topology">120</keepalive-delay>
</node>
```

Testing the directed graph

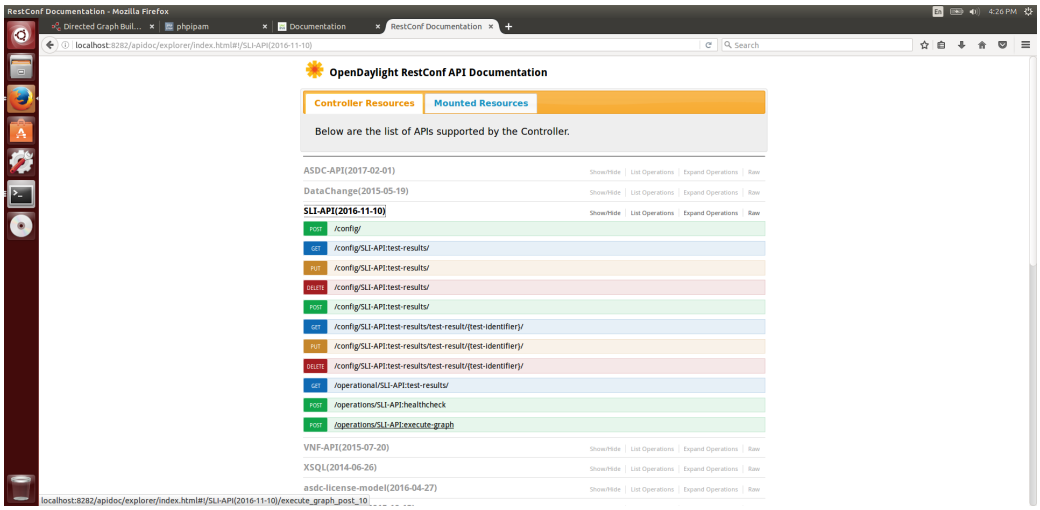
Access the sdnc controller container and turn on DEBUG for org.openecomp.sdnc

```
opendaylight-user@root>log:get all
Logger | Level
--------------
ROOT | INFO
opendaylight-user@root>log:set DEBUG org.openecomp.sdnc
opendaylight-user@root>log:get all
Logger | Level
-------------------------
ROOT | INFO
org.openecomp.sdnc | DEBUG
opendaylight-user@root>logout
```

Access the apidoc/explorer/index.html on port 8282 of the SDNC Controller

http://<SDNC_IP>:8282/apidoc/explorer/index.html

Under the SLI-API section we want the execute-graph RPC



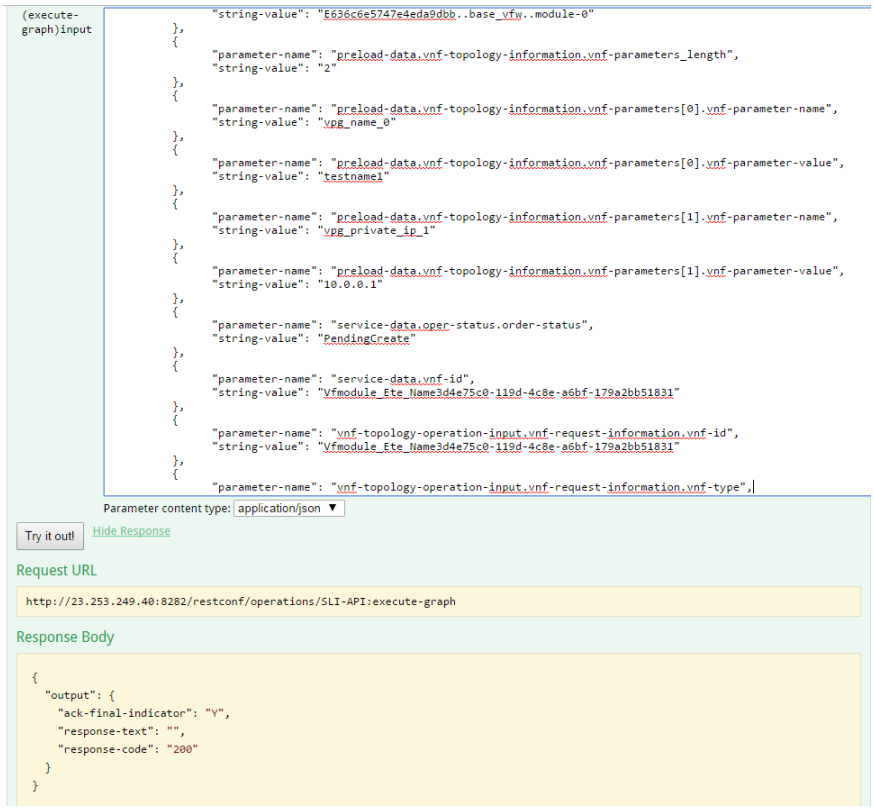
Open the execute-graph section and use the following input json to test the graph.

This sets up preload data including the two parameters of interest (and the length of the parametere array)

(use the data from the VNF-API GET to find matching valudes for the vnf-type, vnf-id, generic-vnf-id from your current preload data)

```
{
"input": {
"module-name": "VNF-API",
"rpc-name": "vnf-topology-activate",
"mode": "sync",
"sli-parameter": [
{
"parameter-name": "preload-data.vnf-topology-information.vnf-topology-identifier.vnf-type",
"string-value": "E636c6e5747e4eda9dbb..base_vfw..module-0"
},
{
"parameter-name": "preload-data.vnf-topology-information.vnf-parameters_length",
"string-value": "2"
},
{
"parameter-name": "preload-data.vnf-topology-information.vnf-parameters[0].vnf-parameter-name",
"string-value": "vpg_name_0"
},
{
"parameter-name": "preload-data.vnf-topology-information.vnf-parameters[0].vnf-parameter-value",
"string-value": "testname1"
},
{
"parameter-name": "preload-data.vnf-topology-information.vnf-parameters[1].vnf-parameter-name",
"string-value": "vpg_private_ip_1"
},
{
"parameter-name": "preload-data.vnf-topology-information.vnf-parameters[1].vnf-parameter-value",
"string-value": "10.0.0.1"
},
{
"parameter-name": "service-data.oper-status.order-status",
"string-value": "PendingCreate"
},
{
"parameter-name": "service-data.vnf-id",
"string-value": "Vfmodule_Ete_Name3d4e75c0-119d-4c8e-a6bf-179a2bb51831"
},
{
"parameter-name": "vnf-topology-operation-input.vnf-request-information.vnf-id",
"string-value": "Vfmodule_Ete_Name3d4e75c0-119d-4c8e-a6bf-179a2bb51831"
},
{
"parameter-name": "vnf-topology-operation-input.vnf-request-information.vnf-type",
"string-value": "E636c6e5747e4eda9dbb..base_vfw..module-0"
},
{
"parameter-name": "vnf-topology-operation-input.vnf-request-information.generic-vnf-id",
"string-value": "Vnf_Ete_Name3d4e75c0-119d-4c8e-a6bf-179a2bb51831"
}
]
}
}
```


Note that the input sets up both data that would come in on the RPC and data that is already in the controller.



Check to see the if the netconf mount is defined in the APPC

http://<appc_ip>:8282/apidoc/explorer/index.html

http://<appc_ip>:8282/apidoc/explorer/index.html#!/network-topology(2013-07-12)/GET_network_topology_get_1

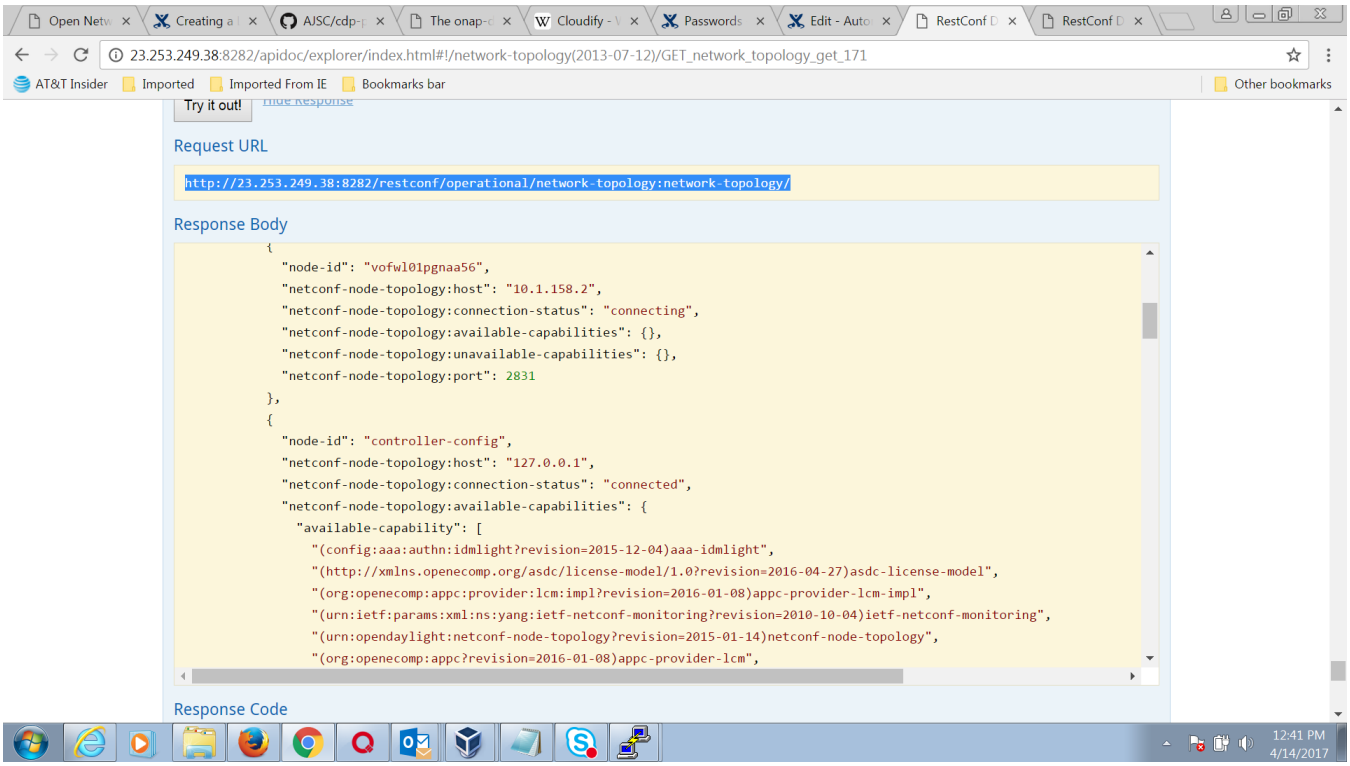
Here is an example for testname1 but your hostname and IP address will be different:

{
"node-id": "testname1",
"netconf-node-topology:sleep-factor": 1.5,
"netconf-node-topology:between-attempts-timeout-millis": 2000,
"netconf-node-topology:host": "10.0.0.1",
"netconf-node-topology:reconnect-on-changed-schema": false,
"netconf-node-topology:tcp-only": false,
"netconf-node-topology:keepalive-delay": 120,
"netconf-node-topology:max-connection-attempts": 0,
"netconf-node-topology:password": "admin",
"netconf-node-topology:username": "admin",
"netconf-node-topology:port": 2883,
"netconf-node-topology:connection-timeout-millis": 20000
},

If you go to operational part of the tree you can see the status of the netconf mounts

`http://<appc_ip>:8282/restconf/operational/network-topology:network-topology/`

The graphic below shows both one connecting (error) and one connected (success) with the yang models pulled over the netconf session. In this case with 127.0.0.1 that is from APPC itself.



If you want to delete the test netconf mount simply use the apidoc explorer or curl/POSTMAN to delete it. In this case for "testname1" this is the url.

DELETE http://<appc_ip>:8282/restconf/config/network-topology:network-topology/topology/topology-netconf/node/testname1/

Now if those test passed then go back to dgbuilder.

Reconnect the AnAI node to the Block.

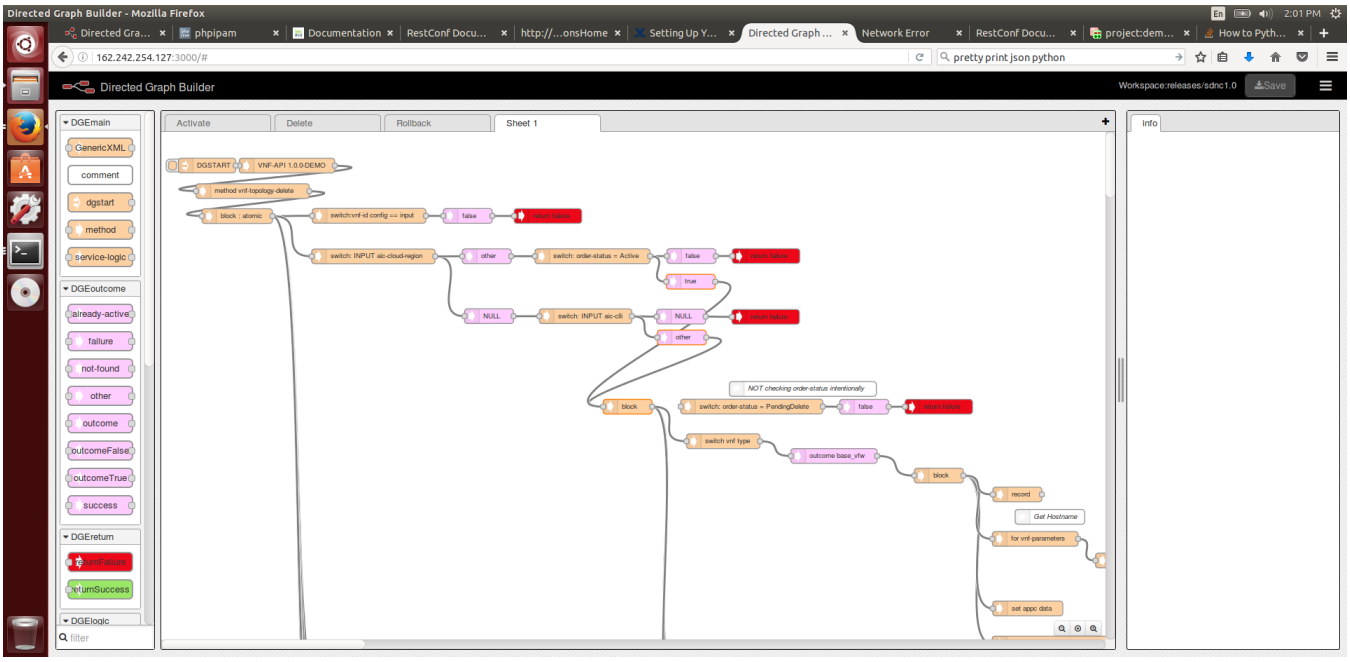
SAVE , Upload and Activate the DG.

You can run ETE tests from robot framework at this stage and you should see the netconf mount setup by SDNC (robot might try to set it up a second time be we will ignore that 🙂).

Now that you have the Activate you can proceed to do the same with the Delete Directed Graph.

This uses the DELETE method to the restapicall node and only need the vpg_hostname so it is a little simpler.

The current VFN-TOPOLOGY-DELETE graph assumes the MSO will be deleting the VNF so it doesnt remove the l3 network data that is still attached to the VNF in AnAI in some cases. You may want to tie the block for processing delete into both of the success branches from the order-status=Active switch test by adding a "true" outcome from the order status="Active" and linking that to the existing block node after the "other" outcome from the switch test of "aic-cllii. As shown below. This way the l3 network data would be removed from AnAI by SDNC since it was added by SDNC on the "activate". The VNF will be deleted in a few seconds after the delete runs but its probably something to consider.



The delete node you want to delete the responsePrefix and template parameter since they are not needed and will generate errors if included in an DELETE operation.

<execute plugin='org.openecomp.sdnc.restapicall.RestapiCallNode' method='sendRequest' >
<parameter name="restapiUrl" value="`$prop.appcRestApi.url+'/restconf/config/network-topology:network-topology/topology/topology-netconf/node/'+$prop.vpg_hostname`"/>
<parameter name="restapiUser" value="`$prop.appcRestApi.sdncOdl.user`"/>
<parameter name="restapiPassword" value="`$prop.appcRestApi.sdncOdl.password`"/>
<parameter name="format" value="xml" />
<parameter name="httpMethod" value="delete" />

The tutorial data will be in gerrit under the demo repository (demo/tutorials/CreateAppcNetconfMount)

Here is a string version for quick reference.

vnf_opertion_activate.json

[{"id":"9b515625.d22748","type":"dgstart","name":"DGSTART","outputs":1,"x":93.33332824707031,"y":36.33332824707031,"z":"669f9d98.2ac9f4","wires":[["80315e6e.63505"]]},{"id":"80315e6e.63505","type":"service-logic","name":"VNF-API 1.0.0-DEMO","module":"VNF-API","version":"1.0.0-DEMO","comments":"","xml":"<service-logic xmlns='http://www.openecomp.org/sdnc/svclogic' xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xsi:schemaLocation='http://www.openecomp.org/sdnc/svclogic ./svclogic.xsd' module='VNF-API' version='1.0.0-DEMO'>","outputs":1,"x":258.49998474121094,"y":36.5,"z":"669f9d98.2ac9f4","wires":[["3c8cef4d.49542"]]},{"id":"3c8cef4d.49542","type":"method","name":"method vnf-topology-activate","xml":"<method rpc='vnf-topology-activate' mode='sync'>\n","comments":"","outputs":1,"x":166.50001525878906,"y":83.49999237060547,"z":"669f9d98.2ac9f4","wires":[["6cd2edde.1d6bd4"]]},{"id":"6cd2edde.1d6bd4","type":"block","name":"block : atomic","xml":"<block atomic=\"true\">","atomic":"true","outputs":1,"x":123.83332824707031,"y":139.3333282470703,"z":"669f9d98.2ac9f4","wires":[["ab8c4330.26d2b","df0bd34b.bf0cf","d8650a48.f0bcd8","6e2d653a.a05bec","9bb6aa32.18f568","d92f94a6.8f1a28","b846b4fc.934e88","c3be1550.0013d8","40746641.fe81b8"]]},{"id":"ab8c4330.26d2b","type":"set","name":"set: order-status = Active","xml":"<set>\n\t<parameter name=\"service-data.oper-status.order-status\" value=\"Active\"/>","comments":"","x":376.83333587646484,"y":371.83331298828125,"z":"669f9d98.2ac9f4","wires":[]},{"id":"df0bd34b.bf0cf","type":"switchNode","name":"switch: order-status = PendingCreate","xml":"<switch test=\"`$service-data.oper-status.order-status == PendingCreate`\">\n\n<!--\nleaf order-status {\n\ttype enumeration {\n\t\tenum \"Active\";\n\t\tenum \"PendingAssignment\";\n\t\tenum \"PendingCreate\";\n\t\tenum \"PendingUpdate\";\n\t\tenum \"Deleted\";\n\t}\n}\n-->","comments":"","outputs":1,"x":416.16670989990234,"y":268.1665344238281,"z":"669f9d98.2ac9f4","wires":[["22236f98.35747"]]},{"id":"d8650a48.f0bcd8","type":"returnSuccess","name":"return success","xml":"<return status='success'>\n\t<parameter name=\"error-code\" value=\"200\" />","comments":"","x":336.9444580078125,"y":1119.5000267028809,"z":"669f9d98.2ac9f4","wires":[]},{"id":"6e2d653a.a05bec","type":"set","name":"set: last-order-status = order-status","xml":"<set>\n\t<parameter name=\"service-data.oper-status.last-order-status\" value=\"`$service-data.oper-status.order-status`\"/>","comments":"","x":409.94701385498047,"y":320.447021484375,"z":"669f9d98.2ac9f4","wires":[]},{"id":"9bb6aa32.18f568","type":"switchNode","name":"switch:vnf-id config == input","xml":"<switch test=\"`$vnf-topology-operation-input.vnf-request-information.vnf-id == $service-data.vnf-id`\">","comments":"","outputs":1,"x":388.60607147216797,"y":216.37876892089844,"z":"669f9d98.2ac9f4","wires":[["4c7aba25.1c7f44"]]},{"id":"d92f94a6.8f1a28","type":"set","name":"set: service-data = input","xml":"<set>\n\t<parameter name=\"service-data.\" value=\"vnf-topology-operation-input.\" />","comments":"","x":364.8333435058594,"y":1033.8889427185059,"z":"669f9d98.2ac9f4","wires":[]},{"id":"b846b4fc.934e88","type":"set","name":"set: last-action ","xml":"<set>\n\t<parameter name=\"service-data.oper-status.last-action\" value=\"`$service-data.request-information.request-action`\"/>","comments":"","x":343.83333587646484,"y":418.8888854980469,"z":"669f9d98.2ac9f4","wires":[]},{"id":"40746641.fe81b8","type":"save","name":"update AnAI - vf-module.orchestration-status","xml":"<update plugin=\"org.openecomp.sdnc.sli.aai.AAIService\" \n\tresource=\"vf-module\" \n\t\tkey=\"vf-module.vf-module-id = $vnf-topology-operation-input.vnf-request-information.vnf-id \n\t\t\tAND

generic-vnf.vnf-id = $vnf-topology-operation-input.vnf-request-information.generic-vnf-id\" \n\t\tlocal-only=\"false\" >\n\t<parameter name=\"orchestration-status\" value=\"active\" />","comments":"","outputs":1,"x":437.7221984863281,"y":461.8888816833496,"z":"669f9d98.2ac9f4","wires":[["738bd5ac.1e39cc"]]},{"id":"5a15cf5c.1fe94","type":"record","name":"record","xml":"<record plugin=\"org.openecomp.sdnc.sli.recording.FileRecorder\">\n<parameter name=\"file\" value=\"/opt/opendaylight/current/data/log/sdnc_appc_netconf.log\"/>\n<parameter name=\"field1\" value=\"__TIMESTAMP__\" />\n<parameter name=\"field2\" value=\"`$vnf-topology-operation-input.sdnc-request-header.svc-action`\"/>\n<parameter name=\"field3\" value=\"`$vnf-topology-operation-input.vnf-request-information.vnf-name`\"/>\n<parameter name=\"field4\" value=\"`$vnf-topology-operation-input.vnf-request-information.vnf-type`\"/>\n<parameter name=\"field5\" value=\"`$vnf-topology-operation-input.vnf-request-information.vnf-id`\"/>\n<parameter name=\"field6\" value=\"`$vnf-topology-operation-input.vnf-request-information.generic-vnf-name`\"/>\n<parameter name=\"field7\" value=\"`$vnf-topology-operation-input.vnf-request-information.generic-vnf-type`\"/>\n<parameter name=\"field8\" value=\"`$vnf-topology-operation-input.vnf-request-information.generic-vnf-id`\"/>\n<parameter name=\"field9\" value=\"`$preload-data.vnf-topology-information.vnf-topology-identifier.vnf-name`\"/>\n<parameter name=\"field10\" value=\"`$preload-data.vnf-topology-information.vnf-topology-identifier.vnf-type`\"/>\n","comments":"","outputs":1,"x":938.72216796875,"y":776.3888549804688,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"c3be1550.0013d8","type":"switchNode","name":"switch vnf type ","xml":"<switch test=`substr($preload-data.vnf-topology-information.vnf-topology-identifier.vnf-type,22,30)`>\n<!-- Aa27262bF2d040799b5c..base_vfw..module-0 -->","comments":"","outputs":1,"x":338.522216796875,"y":667.9055213928223,"z":"669f9d98.2ac9f4","wires":[["798dbf61.8ef5c"]]},{"id":"22236f98.35747","type":"other","name":"false","xml":"<outcome value='false'>","comments":"","outputs":1,"x":665.5000610351562,"y":268.4998435974121,"z":"669f9d98.2ac9f4","wires":[["b819ee7c.13f39"]]},{"id":"4c7aba25.1c7f44","type":"outcomeFalse","name":"false","xml":"<outcome value='false'>\n","comments":"","outputs":1,"x":608.060546875,"y":216.56058883666992,"z":"669f9d98.2ac9f4","wires":[["6309638d.9a29fc"]]},{"id":"738bd5ac.1e39cc","type":"failure","name":"failure","xml":"<outcome value='failure'>\n","comments":"","outputs":1,"x":702.2460556030273,"y":461.877384185791,"z":"669f9d98.2ac9f4","wires":[["91b545b4.961ef8"]]},{"id":"798dbf61.8ef5c","type":"outcome","name":"outcome base_vfw","xml":"<outcome value='base_vfw'>\n","comments":"","outputs":1,"x":564.5221862792969,"y":697.9055213928223,"z":"669f9d98.2ac9f4","wires":[["9aacbf5a.ed983"]]},{"id":"b819ee7c.13f39","type":"switchNode","name":"switch: order-status = PendingUpdate","xml":"<switch test=\"`$service-data.oper-status.order-status == PendingUpdate`\">\n\n<!--\nleaf order-status {\n\ttype enumeration {\n\t\tenum \"Active\";\n\t\tenum \"PendingAssignment\";\n\t\tenum \"PendingCreate\";\n\t\tenum \"PendingUpdate\";\n\t\tenum \"Deleted\";\n\t}\n}\n-->","comments":"","outputs":1,"x":916.4242553710938,"y":267.9242515563965,"z":"669f9d98.2ac9f4","wires":[["de1e0e32.44154"]]},{"id":"6309638d.9a29fc","type":"returnFailure","name":"return failure","xml":"<return status='failure'>\n\t<parameter name='error-code' value='400'/>\n\t<parameter name='error-message' value=\"`vnf-id = ' + $vnf-topology-operation-input.vnf-request-information.vnf-id + ' not found in config tree`\"/>","comments":"","x":786.1514892578125,"y":216.4696922302246,"z":"669f9d98.2ac9f4","wires":[]},{"id":"91b545b4.961ef8","type":"returnFailure","name":"return failure","xml":"<return status='failure'>\n\t<parameter name='error-code' value='500'/>\n\t<parameter name='error-message' value=\"`Encountered error while updating vf-module orchestration-status in AnAI with vnf-id = ' + $vnf-topology-operation-input.vnf-request-information.vnf-id + ' and generic-vnf-id = ' + $vnf-topology-operation-input.vnf-request-information.generic-vnf-id`\"/>","comments":"","x":870.3889846801758,"y":461.87745475769043,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"9aacbf5a.ed983","type":"block","name":"block","xml":"<block>\n","atomic":"false","comments":"","outputs":1,"x":778.5221862792969,"y":737.9055213928223,"z":"669f9d98.2ac9f4","wires":[["f6b5a926.ce9888","2c6e5e35.0760b2","c3f63154.8e521","5a15cf5c.1fe94"]]},{"id":"de1e0e32.44154","type":"other","name":"false","xml":"<outcome value='false'>","comments":"","outputs":1,"x":1158.878662109375,"y":267.92422103881836,"z":"669f9d98.2ac9f4","wires":[["5f574a63.b3a724"]]},{"id":"f6b5a926.ce9888","type":"execute","name":"excute REST NETCONF MOUNT ","xml":"<execute plugin='org.openecomp.sdnc.restapicall.RestapiCallNode' method='sendRequest' >\n\t<parameter name=\"templateFileName\" value=\"`$prop.restapi.templateDir + '/netconf-mount-template.xml`\" />\n\t<parameter name=\"restapiUrl\" value=\"`$prop.appcRestApi.url+'/restconf/config/network-topology:network-topology/topology/topology-netconf/node/'+$prop.vpg_hostname`\"/>\n\t<parameter name=\"restapiUser\" value=\"`$prop.appcRestApi.sdncOdl.user`\"/>\n\t<parameter name=\"restapiPassword\" value=\"`$prop.appcRestApi.sdncOdl.password`\"/>\n\t<parameter name=\"format\" value=\"xml\" />\n\t<parameter name=\"httpMethod\" value=\"put\" />\n\t<parameter name=\"responsePrefix\" value=\"restapi-result\" />\n\t\n\t","comments":"","outputs":1,"x":1026.5221099853516,"y":1073.9054565429688,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"2dff6fee.5d34e","type":"set","name":"set localhost data","xml":"<set>\n<parameter name='prop.appcRestApi.url' value='http://localhost:8181' />\n<!-- 8181 when doing localhost -->\n<parameter name=\"prop.restapi.templateDir\" value=\"/opt/openecomp/sdnc/data\"/>\n<parameter name=\"prop.appcRestApi.sdncOdl.user\" value=\"admin\"/>\n<parameter name=\"prop.appcRestApi.sdncOdl.password\" value=\"Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U\"/>\n","comments":"","x":1331.5221557617188,"y":746.905517578125,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"5f574a63.b3a724","type":"returnFailure","name":"return failure","xml":"<return status='failure'>\n\t<parameter name='error-code' value='400'/>\n\t<parameter name='error-message' value='Invalid last-order-status'/>","comments":"","x":1323.8331832885742,"y":267.4998950958252,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"e27e181e.227c48","type":"comment","name":"VNF-TOPOLOGY-ACTIVATE","info":"","comments":"","x":665.7221984863281,"y":35.88888168334961,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"2aacf238.e9652e","type":"comment","name":"once success is returned, operational-data gets saved","info":"","comments":"","x":672.3510437011719,"y":1114.4343528747559,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"75691490.243aac","type":"comment","name":"request-action = VNFActivateRequest, svc-action = activate","info":"","comments":"","x":672.4444122314453,"y":75.66666221618652,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"bcfdeb55.ac2538","type":"set","name":"Set final indicator to Y","xml":"<set>\n<parameter name=\"ack-final\" value=\"Y\"/>","comments":"","x":359.15079498291016,"y":1077.6031875610352,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"2c6e5e35.0760b2","type":"set","name":"set appc data","xml":"<set>\n<parameter name='prop.appcRestApi.url' value='http://appc.api.simpledemo.openecomp.org:8282' />\n<!-- 8181 when doing localhost -->\n<parameter name=\"prop.restapi.templateDir\" value=\"/opt/openecomp/sdnc/data\"/>\n<parameter name=\"prop.appcRestApi.sdncOdl.user\" value=\"admin\"/>\n<parameter name=\"prop.appcRestApi.sdncOdl.password\" value=\"Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U\"/>\n","comments":"","x":967.5221862792969,"y":1019.9054832458496,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"873d2429.978de8","type":"comment","name":"Get Hostname and IP address","info":"","comments":"","x":1005.2166748046875,"y":821.9387817382812,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"c3f63154.8e521","type":"for","name":"for vnf-parameters","xml":"<for index='k' start='0' end=\"`$preload-data.vnf-topology-information.vnf-parameters_length`\" >\n","comments":"","outputs":1,"x":975.2166442871094,"y":866.9388999938965,"z":"669f9d98.2ac9f4","wires":[["7112332f.7cf7ac"]]},{"id":"1c75c042.26d5d","type":"switchNode","name":"switch vpg_name_0","xml":"<switch test=\"`$preload-data.vnf-topology-information.vnf-parameters[$k].vnf-parameter-name == 'vpg_name_0`\">\n","comments":"","outputs":1,"x":1338.2167663574219,"y":972.9389305114746,"z":"669f9d98.2ac9f4","wires":[["f162e338.32bba"]]},{"id":"f162e338.32bba","type":"outcomeTrue","name":"true","xml":"<outcome value='true'>\n","comments":"","outputs":1,"x":1525.2167663574219,"y":972.9389305114746,"z":"669f9d98.2ac9f4","wires":[["fb2d23d2.1064d"]]},{"id":"fb2d23d2.1064d","type":"set","name":"set vpg_ hostname ","xml":"<set>\n<parameter name='prop.vpg_hostname' value=`$preload-data.vnf-topology-information.vnf-parameters[$k].vnf-parameter-value` />\n","comments":"","x":1693.2167663574219,"y":973.5389366149902,"z":"669f9d98.2ac9f4","wires":[[]]},{"id":"7112332f.7cf7ac","type":"block","name":"block","xml":"<block>\n","atomic":"false","comments":"","outputs":1,"x":1148.2167663574219,"y":910.9388999938965,"z":"669f9d98.2ac9f4","wires":[["1c75c042.26d5d","9dc9c622.a5ca58"]]},{"id":"9dc9c622.a5ca58","type":"switchNode","name":"switch vpg_private_ip_1","xml":"<switch test=\"`$preload-data.vnf-topology-information.vnf-parameters[$k].vnf-parameter-name == 'vpg_private_ip_1`\">\n","comments":"","outputs":1,"x":1342.2167053222656,"y":1034.538906097412,"z":"669f9d98.2ac9f4","wires":[["43a36baf.403124"]]},{"id":"43a36baf.403124","type":"outcomeTrue","name":"true","xml":"<outcome value='true'>\n","comments":"","outputs":1,"x":1527.2167053222656,"y":1032.538906097412,"z":"669f9d98.2ac9f4","wires":[["b4b98be7.2c0088"]]},{"id":"b4b98be7.2c0088","type":"set","name":"set vpg_ipaddress","xml":"<set>\n<parameter name='prop.vpg_ipaddress' value=`$preload-data.vnf-topology-information.vnf-parameters[$k].vnf-parameter-value` />\n","comments":"","x":1696.2167053222656,"y":1030.538906097412,"z":"669f9d98.2ac9f4","wires":[]},{"id":"54b07b39.58cfd4","type":"comment","name":"unlinked AnAI update for slitesting","info":"","comments":"","x":399.1666564941406,"y":509.8888816833496,"z":"669f9d98.2ac9f4","wires":[]}]