Tutorial: VoLTE Example: Adding REST Node

Lets extend the example by adding a block of logic to setting up the data center connection to the third part local controllers between edge and core.

We will need to:

1. Create a Local Controller Emulator as a simple http server with basic auth that takes POST input, logs the data and returns succes-dci-connnect. jsons.
2. Create the Directed Graph
3. Load the l3-dci-connect template into the SDNC controller for the json body to the emulator
4. Use the SLI-API: Execute Graph function to test the new DG.

Create Local Controller Emulator

If you already have an http server available use it. For this demo we will install and run a lighttp docker container on the SDNC controller so that we have a self contained enivonment.

Lets use port 9000 for the outside port.

create a "http_test_server" directory on the SDNC VM under /opt and use that as the home directory for the container

scp this tar file http_test_server.tar into the /opt/http_test_server directory and untar it

cd docker

follow the notes in the README.txt to create the debian-lighttpd container

start the container

use testEmulator.sh to test that php is working

"docker exec -it debian-lighttpd bash" to log in to the emulator

"cd /var/log/lighttpd"

"tail emulator.log"

and you should see a test line like

"17:07:21 07:29:08 | name:test,value:testdata 3"

This emulator is setup to reply on "http://10.0.7.1:9000/l3-dci-connect.php" so we will use that in the Directed Graph.

Create a Directed Graph

This directed graph adds a block to deal with the setting of parametres and a set of REST API Call Nodes to the emulator so we can see the behavoir and experiment with different templates/sequences.

[{"id":"1c91b4f1.e45473","type":"service-logic","name":"VOLTE-API 0.0.2-DEMO","module":"VOLTE-API","version":"0.0.2-DEMO","comments":"","xml":"
<service-logic xmlns='http://www.onap.org/sdnc/svclogic' xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xsi:schemaLocation='http://www.onap.
org/sdnc/svclogic ./svclogic.xsd' module='VOLTE-API' version='0.0.2-DEMO'>","outputs":1,"x":317.1666564941406,"y":106.16667175292969,"z":"
30ac5fe5.ab4d2","wires":[["a28ec6bd.3832d8"]]},{"id":"40983a09.90a524","type":"dgstart","name":"DGSTART","outputs":1,"x":111,"y":81,"z":"30ac5fe5.
ab4d2","wires":[["1c91b4f1.e45473"]]},{"id":"a28ec6bd.3832d8","type":"method","name":"method volte-wan-activate","xml":"<method rpc='volte-wan-
activate' mode='sync'>\n","comments":"","outputs":1,"x":181.16668701171875,"y":170.16665649414062,"z":"30ac5fe5.ab4d2","wires":[["e53e6cd.afc3f1"]]},
{"id":"e53e6cd.afc3f1","type":"block","name":"block : atomic","xml":"<block atomic=\"true\">","atomic":"true","outputs":1,"x":389,"y":258,"z":"30ac5fe5.
ab4d2","wires":[["cdeda8e3.1218f","9794fca5.9de488","cd5f5503.b9f128","19f372e1.31b345","8ff8c47.cd19b38","edd83200.432db8"]]},{"id":"cdeda8e3.
1218f","type":"set","name":"set: order-status = Active","xml":"<set>\n\t<parameter name=\"service-data.oper-status.order-status\" value=\"Active\"/>","
comments":"","x":597,"y":603,"z":"30ac5fe5.ab4d2","wires":[]},{"id":"19f372e1.31b345","type":"returnSuccess","name":"return success","xml":"<return
status='success'>\n\t<parameter name=\"error-code\" value=\"200\" />\n\t<parameter name='error-message' value=\"`service-data.dcz-wanip=' + $service-
data.dcz-wanip`\"/>","comments":"","x":569,"y":790,"z":"30ac5fe5.ab4d2","wires":[]},{"id":"9794fca5.9de488","type":"set","name":"set: service-data = input","
xml":"<set>\n\t<parameter name=\"service-data.\" value=\"volte-wan-activate.\" />","comments":"","x":596.8888854980469,"y":704.388916015625,"z":"
30ac5fe5.ab4d2","wires":[]},{"id":"cd5f5503.b9f128","type":"set","name":"Set final indicator to Y","xml":"<set>\n<parameter name=\"ack-final\" value=\"Y\"
/>","comments":"","x":591.2063369750977,"y":748.1031608581543,"z":"30ac5fe5.ab4d2","wires":[]},{"id":"8ff8c47.cd19b38","type":"record","name":"
record","xml":"<record plugin=\"org.onap.ccsdk.sli.core.sli.recording.FileRecorder\">\n<parameter name=\"file\" value=\"/opt/opendaylight/current/data/log
/svclogic.log\"/>\n<parameter name=\"field1\" value=\"__TIMESTAMP__\"/>\n<parameter name=\"field2\" value=\"volte-wan-activate\"/>\n<parameter
name=\"field3\" value=\"`$volte-wan-activate.dca-wanip`\"/>\n<parameter name=\"field4\" value=\"`$volte-wan-activate.dcz-wanip`\"/>\n<parameter name=\"
field5\" value=\"`$volte-wan-activate.route_target`\"/>\n<parameter name=\"field6\" value=\"`$volte-wan-activate.route_distinguisher`\"/>","comments":"","
outputs":1,"x":559,"y":301,"z":"30ac5fe5.ab4d2","wires":[[]]},{"id":"c2486391.d9417","type":"execute","name":"excute REST DC-Connect-Create-A","xml":"
<execute plugin='org.onap.ccsdk.sli.plugins.restapicall.RestapiCallNode' method='sendRequest' >\n\t<parameter name=\"templateFileName\" value=\"
`$prop.restapi.templateDir + '/l3-dci-connect.json'`\"/>\n\t<!-- \n\t<parameter name=\"restapiUrl\" value=\"`$prop.controller_A_RestApi.url+ '/v2.0/l3-dci-
connects'`/>\n\t-->\n\t<parameter name=\"restapiUrl\" value=\"`$prop.controller_A_RestApi.url + '/l3-dci-connect.php'`\" />\n\t<parameter name=\"
restapiUser\" value=\"`$prop.controller_A_RestApi.user`\" />\n\t<parameter name=\"restapiPassword\" value=\"`$prop.controller_A_RestApi.password`\"
/>\n\t<parameter name=\"format\" value=\"xml\" />\n\t<parameter name=\"httpMethod\" value=\"post\" />\n\t<parameter name=\"responsePrefix\" value=\"
restapi-result\" />\n\t\n\t","comments":"","outputs":1,"x":998,"y":727,"z":"30ac5fe5.ab4d2","wires":[[]]},{"id":"852c391a.a7d2b","type":"execute","name":"
excute REST DC-Connect-Create-B","xml":"<execute plugin='org.onap.ccsdk.sli.plugins.restapicall.RestapiCallNode' method='sendRequest'
>\n\t<parameter name=\"templateFileName\" value=\"`$prop.restapi.templateDir + '/l3-dci-connect.json'`\"/>\n\t<!-- \n\t<parameter name=\"restapiUrl\"
value=\"`$prop.controller_A_RestApi.url+ '/v2.0/l3-dci-connects'`/>\n\t-->\n\t<parameter name=\"restapiUrl\" value=\"`$prop.controller_A_RestApi.url + '/l3-
dci-connect.php'`\" />\n\t<parameter name=\"restapiUser\" value=\"`$prop.controller_B_RestApi.user`\"/>\n\t<parameter name=\"restapiPassword\" value=\"
`$prop.controller_B_RestApi.password`\"/>\n\t<parameter name=\"format\" value=\"xml\" />\n\t<parameter name=\"httpMethod\" value=\"post\"
/>\n\t<parameter name=\"responsePrefix\" value=\"restapi-result\" />\n\t\n\t","comments":"","outputs":1,"x":998,"y":830,"z":"30ac5fe5.ab4d2","wires":[[]]},
{"id":"edd83200.432db8","type":"block","name":"block","xml":"<block>\n","atomic":"false","comments":"","outputs":1,"x":560,"y":370,"z":"30ac5fe5.ab4d2","
wires":[["f3cc8b33.0516c","6a3b097e.b896d8","a67a46fc.946838","c2486391.d9417","fbd78797.a6d598","852c391a.a7d2b","de93e82.c1dca18"]]},{"id":"
f3cc8b33.0516c","type":"set","name":"set A controller data (http emulator)","xml":"<set>\n<parameter name='prop.controller_A_RestApi.url' value='http://10.
0.7.1:9000' />\n<!-- 8181 when doing localhost -->\n<parameter name=\"prop.restapi.templateDir\" value=\"/opt/onap/sdnc/data\"/>\n<parameter name=\"
prop.controller_A_RestApi.user\" value=\"admin\"/>\n<parameter name=\"prop.controller_A_RestApi.password\" value=\"
Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U\"/>\n\n","comments":"","x":994,"y":455,"z":"30ac5fe5.ab4d2","wires":[]},{"id":"6a3b097e.b896d8","
type":"set","name":"set WAN data (l3_vni=5001)","xml":"<set>\n<parameter name='device.wan.l3_vni' value='5001' />\n","comments":"","x":976,"y":556,"
z":"30ac5fe5.ab4d2","wires":[]},{"id":"a67a46fc.946838","type":"set","name":"set DC A network data","xml":"<set>\n<!-- example uses device. \n   real
adapter might use device.l3_connection. or something else -->\n<!-- ARRAY parameters could be comma separated lists or leaf lists depending on adaptor
-->\n<parameter name='device.evpn_irts' value='1:5000'/>\n<parameter name='device.evpn_erts' value='1:5000'/>\n<parameter name='device.
local_subnet' value='\\\"8a41319d-87cf-4cd6-8957-00000000000A\\\"'/>\n<parameter name='device.id' value='CDD702C3-7719-4FE6-A5AD-
3A9C9E265309' />\n<parameter name='device.name' value='PODX-routerY' />\n<parameter name='device.description' value='VPC A connect VPC B'
/>\n<parameter name='device.router_id' value='CBB702C3-6789-1234-A5AD-00000000000A' />\n","comments":"","x":958,"y":675,"z":"30ac5fe5.ab4d2","
wires":[]},{"id":"fbd78797.a6d598","type":"set","name":"set DC B network data","xml":"<set>\n<!-- example uses device. \n   real adapter might use device.
l3_connection. or something else -->\n<!-- ARRAY parameters could be comma separated lists or leaf lists depending on adaptor -->\n<parameter
name='device.evpn_irts' value='1:5001'/>\n<parameter name='device.evpn_erts' value='1:5001'/>\n<parameter name='device.local_subnet' value='\\\"
8a41319d-87cf-4cd6-8957-00000000000B\\\"'/>\n<parameter name='device.id' value='CDD702C3-7719-4FE6-A5AD-3A9C9E265309' />\n<parameter
name='device.name' value='PODW-routerZ' />\n<parameter name='device.description' value='VPC B connect VPC A' />\n<parameter name='device.
router_id' value='CBB702C3-6789-1234-A5AD-00000000000B' />\n","comments":"","x":955,"y":782,"z":"30ac5fe5.ab4d2","wires":[]},{"id":"d5a70b78.3a1","
type":"comment","name":"Test block to describe logic - template uses \"device.\" for parameters","info":"","comments":"","x":1054,"y":378,"z":"30ac5fe5.
ab4d2","wires":[]},{"id":"a5a8ce82.7fa0d","type":"comment","name":"DC Set Nodes should map from Input parameters","info":"","comments":"","x":1043,"y":
618,"z":"30ac5fe5.ab4d2","wires":[]},{"id":"de93e82.c1dca18","type":"set","name":"set B controller data (http emulator)","xml":"<set>\n<parameter
name='prop.controller_B_RestApi.url' value='http://10.0.7.1:9000' />\n<!-- 8181 when doing localhost -->\n<parameter name=\"prop.restapi.templateDir\"
value=\"/opt/onap/sdnc/data\"/>\n<parameter name=\"prop.controller_B_RestApi.user\" value=\"admin\"/>\n<parameter name=\"prop.controller_B_RestApi.
password\" value=\"Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U\"/>\n\n","comments":"","x":998,"y":503,"z":"30ac5fe5.ab4d2","wires":[]}]

This DG is 0.0.2-demo so we will add this as another version to the sdnc. I will summarize the steps since the first part and the netconf node tutorial cover this with screen shots.

Go to the dbbuilder on port 3000

Create a new tab with the "+" symbol

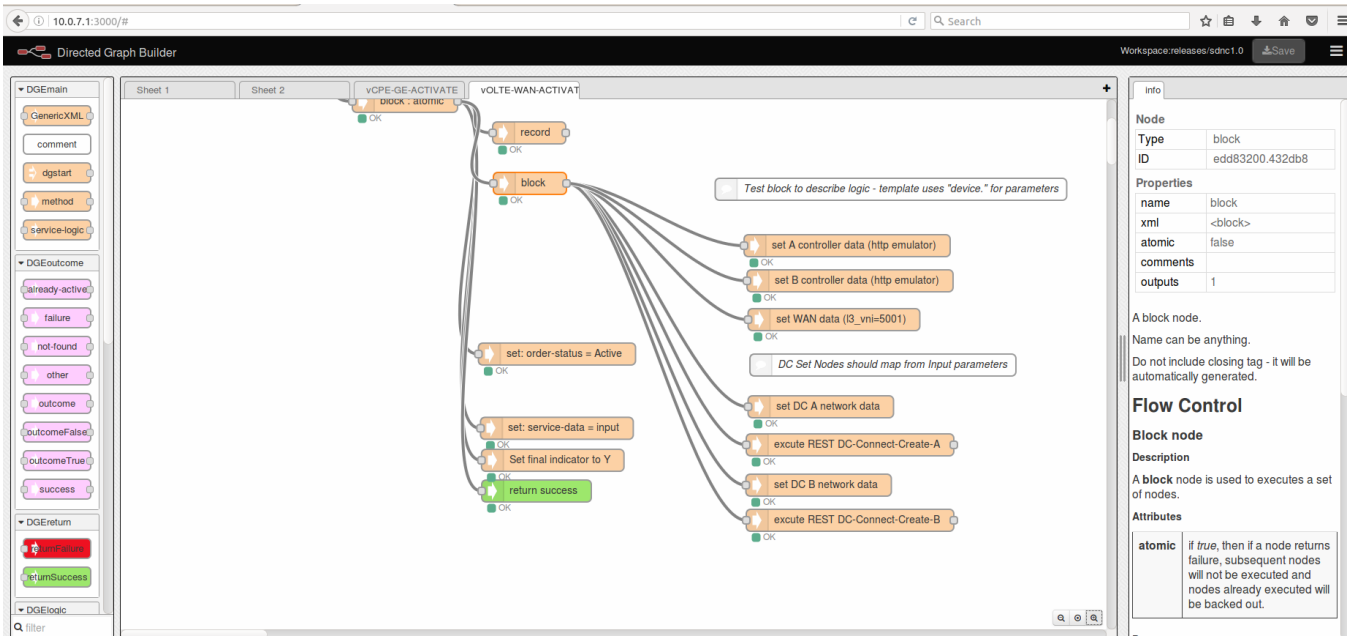
Copy the json string above to the clipboard

Select the menu import clipboard

Paste the json string into the dialog

Postition the DG on the pane

Click on the big red SAVE button



This directed graph handles both the A and B side local controller.

We use SET nodes for protoping but in the real DG we would use data form the input , or query A&AI or call resource allocate nodes to generate the data.

The WAN VNI is in its own set node but would be a resource allocation node in a real DG.

For each local controller we:

1. Set the url and credentials
2. Set the Data
3. Call the REST API execute node - note we are using the same template just changing the parameters.

Lets upload the DG

1.Click on the DGStart node

2.Click on UploadXML

3.Click on DGList

4.Click on Activate for the new DG version 0.0.2-demo

Load the l3-dci-connect template

This is basically the same flow as the Tutorial on setting up Netconf from SDNC to APPC.

Log into the SDNC VM and then log into the sdnc_controller_container

cd /opt/onap/sdnc/data

create the file "l3-dci-connect.json"

```
{
   "l3-dci-connect": {
           "id": "${device.id}",
           "name": "${device.name}",
           "description": "${device.description}",
           "router_id": "${device.router_id}",
           "local_subnets": ["${device.local_subnets}"],
           "local_network_all":false,
           "evpn_irts": ["${device.evpn_irts}"],
           "evpn_erts": ["${device.evpn_erts}"],
           "l3_vni": "${device.wan.l3_vni}"
   }
}
```

Note that the variables for the json payload are "device." or "device.wan." . The variable name branch would be a decision by the Adapter developer as to what model it would use. The DG maps variables into this device model.

The adapter developer may choose other model parameter names so this is just illustrative for the tutorial.

This template name, location and variable names must match what we have created in the REST API Call Node

```
<set>
<!-- example uses device.
      real adapter might use device.l3_connection. or something else -->
<!-- ARRAY parameters could be comma separated lists or leaf lists depending on adaptor -->
<parameter name='device.evpn_irts' value='1:5000'/>
<parameter name='device.evpn_erts' value='1:5000'/>
<parameter name='device.local_subnet' value='\"8a41319d-87cf-4cd6-8957-00000000000A\"'/>
<parameter name='device.id' value='CDD702C3-7719-4FE6-A5AD-3A9C9E265309' />
<parameter name='device.name' value='PODX-routerY' />
<parameter name='device.description' value='VPC A connect VPC B' />
<parameter name='device.router_id' value='CBB702C3-6789-1234-A5AD-00000000000A' />




<execute plugin='org.onap.ccsdk.sli.plugins.restapicall.RestapiCallNode' method='sendRequest' >
        <parameter name="templateFileName" value="`$prop.restapi.templateDir + '/l3-dci-connect.json'`"/>
        <!--
        <parameter name="restapiUrl" value="`$prop.controller_A_RestApi.url+ '/v2.0/l3-dci-connects'`/>
        -->
        <parameter name="restapiUrl" value="`$prop.controller_A_RestApi.url + '/l3-dci-connect.php'`" />
        <parameter name="restapiUser" value="`$prop.controller_A_RestApi.user`" />
        <parameter name="restapiPassword" value="`$prop.controller_A_RestApi.password`"/>
        <parameter name="format" value="xml" />
        <parameter name="httpMethod" value="post" />
        <parameter name="responsePrefix" value="restapi-result" />
```

Use the SLI-API: Execute Graph

Now that we have the emulator, the directed graph and the template loaded we can test.

Go to http://<sdnc_ip>:8282/apidoc/explorer/index.html

Go to the SLI-API:execute-graph operation and use the same input we used for the fist part of the tutorial.

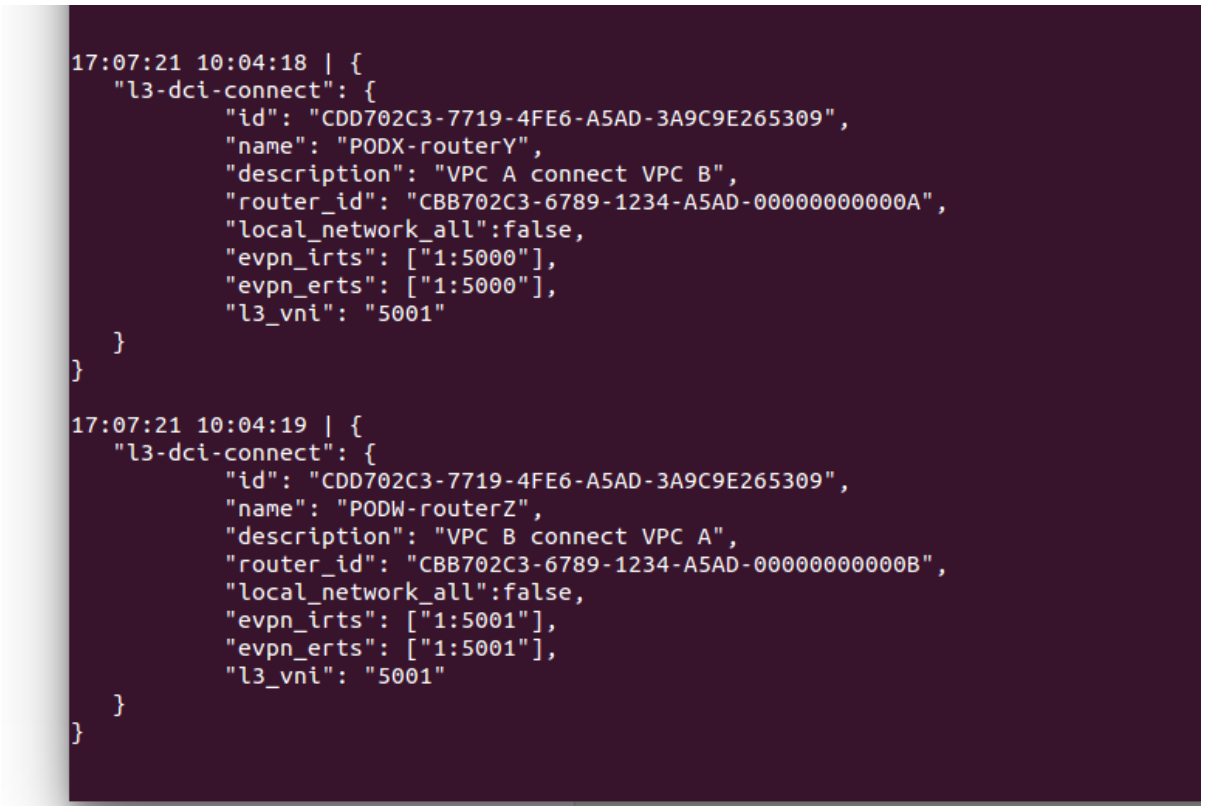
```
{
"input": {
    "module-name": "VOLTE-API",
    "rpc-name": "volte-wan-activate",
    "mode": "sync",
    "sli-parameter": [
        {
            "parameter-name": "volte-wan-activate.dca-wanip",
            "string-value": "10.1.20.2"
        },
        {
            "parameter-name": "volte-wan-activate.dcz-wanip",
            "string-value": "10.2.20.2"
        },
        {
        "parameter-name": "volte-wan-activate.wan_vni",
        "string-value": "101"
        },
        {
        "parameter-name": "volte-wan-activate.route_target",
        "string-value": "6020:201"
        },
                {
        "parameter-name": "volte-wan-activate.route_distinguisher",
        "string-value": "6020:101"
        }
        ]
    }
}
```

Click on "Try It Out!"

You should get

```
{
  "output": {
    "ack-final-indicator": "Y",
    "response-text": "service-data.dcz-wanip=10.2.20.2",
    "response-code": "200"
  }
}
```

Log in to the debian-lighttpd container and tail the /var/log/lighttpd/emulator.log and you should see the calls to emulator for local controller A and local controler B.



Next Steps

We should map from the input to the variable needed for the device in the set node instead of hard coding the value.

Once a local controller is available we can use this DG to test against it by changing the "set A controller data (http emulator)" nodes.