vFW Closed Loop step-by-step

Material for running vFW closed loop

- ONAP.postman_collection.json: REST operations against ONAP component's endpoints;
- Powder lab ONAP.postman_environment.json: Environment file for Postman collection;
- vFWSNK.zip: Package that contains the Heat template and environment file for the vFirewall and vSink VNF components;
- vPKG.zip: Package that contains the Heat template and environment file for the vPacketGen VNF;
- vFWSNK_SDNC_preload.json: JSON file to upload to SDNC that overrides values in the environment file for the vFirewall and vSink VNF components;
- vPKG_SDNC_preload.json: JSON file to upload to SDNC that overrides values in the environment file for the vPacketGen VNF;
- VNF preload.xml: description of the VNF preload for SDNC

Powder lab.post...nvironment.json

vFWSNK_SDNC_preload.json

vFWSNK.zip

vPKG_SDNC_preload.json

vPKG.zip

VNF preload.xlsx

Setup the Environment

Modify /etc/hosts (UNIX) or C:\Windows\System32\Drivers\etc\hosts (Windows) by adding the following FQDNs:

155.98.37.45 portal.api.simpledemo.onap.org

155.98.37.35 policy.api.simpledemo.onap.org

155.98.37.34 sdc.api.simpledemo.onap.org

155.98.37.36 vid.api.simpledemo.onap.org

155.98.37.46 aai.api.simpledemo.onap.org

Create a Vendor Software Product

Designer - cs0008/demo123456!

Onboard -> Add License Model

- License key group
- Entitlement group
- Feature group
- License agreement
- Check in - Submit

Onboard -> Add Vendor Software Product (VSP)

- Compile form and save
- Click overview, then upload zip file
- Check in - Submit

Home

- Import VSP
- Create
- Submit for testing

Tester - jm0007/demo123456!

Home

- Click on the VSP ready for testing
- Start testing
- Accept

Create a Service

Designer - cs0008/demo123456!

Home

- Add service
- Fill the form and click Create to create the service
- Click on Composition
- Select Application L4+
- Drag the VSP and drop it into the canvas

Tester - jm0007/demo123456!

Home

- Click on the service ready for testing
- Start testing
- Accept

Governor - gv0001/demo123456!

Home

- Click on the service
- Approve for distribution

Operator - op0001/demo123456!

Home

- Click on the service

- Distribute

Instantiate a service

Admin - demo/demo123456!

VID

- Browse SDC model
- Deploy service

Preload A&AI (https://wiki.onap.org/display/DW/Tutorial_vIMS%3A+Create+AAI+cloud+account)

AAI Postman headers

- Basic Authentication: AAI/AAI
- Accept: application/json
- Content-Type: application/json
- X-FromAppId: AAI
- X-TransactionId: get_aai_subscr

Add a new service to A&AI

- Generate UUID https://www.uuidgenerator.net/ (use version 4), e.g.: e8cb8968-5411-478b-906a-f28747de72cd
- PUT the new service in A&AI: {{aai_ip}}:8443/aai/v11/service-design-and-creation/services/service/e8cb8968-5411-478b-906a-f28747de72cd

vFW Service

```
{

"service-id": "e8cb8968-5411-478b-906a-f28747de72cd",

"service-description": "vFW"

}
```

Check: GET (https) {{aai_ip}}:8443/aai/v11/service-design-and-creation/services

Create a new cloud region

PUT (https) {{aai_ip}}:8443/aai/v11/cloud-infrastructure/cloud-regions/cloud-region/OpenStack/RegionOne

```
{

    "cloud-owner": "OpenStack",

    "cloud-region-id": "RegionOne",

    "cloud-type": "openstack",

    "owner-defined-type": "owner type",

    "cloud-region-version": "v2.5",

    "cloud-zone": "cloud zone",

    "tenants": {

        "tenant": [{

            "tenant-id": "41d6d38489bd40b09ea8a6b6b852dcbd",

            "tenant-name": "Integration"


        }]
```

```
      }
}
```


Check: GET (https) {{aai_ip}}:8443/aai/v11/cloud-infrastructure/cloud-regions

Create a new customer

PUT (https) {{aai_ip}}:8443/aai/v11/business/customers/customer/Demonstration


```
{
  "global-customer-id": "Demonstration",
  "subscriber-name": "Demonstration",
  "subscriber-type": "INFRA",
  "service-subscriptions": {
    "service-subscription": [
      {
        "service-type": "vFW",
        "relationship-list": {
          "relationship": [{
            "related-to": "tenant",
            "relationship-data": [
              {"relationship-key": "cloud-region.cloud-owner", "relationship-value": "OpenStack"},
              {"relationship-key": "cloud-region.cloud-region-id", "relationship-value": "RegionOne"},
              {"relationship-key": "tenant.tenant-id", "relationship-value": "41d6d38489bd40b09ea8a6b6b852dcbd"}
            ]
          }]
        }
      }
    ]
  }
}
```


Check: GET (https) {{aai_ip}}:8443/aai/v11/business/customers

Create service instance and then VNF instance in VID (https://wiki.onap.org/display/DW/Tutorial+vIMS%3A+VID+Instantiate+the+VNF)

Preload VID

VID Postman headers

- Basic Authentication: demo/Kp8bJ4SXszM0WX
- Accept: application/json
- Content-Type: application/json
- USER_ID: demo
- X-TransactionId: robot-ete-bd65600d-8669-4903-8a14-af88203add38
- X-FromAppId: robot-ete

POST (http) {{vid_ip}}:{{vid_port}}/vid/maintenance/category_parameter/platform

```
{
  "options": ["Test-Platform"]
}
```

POST (http) {{vid_ip}}:{{vid_port}}/vid/maintenance/category_parameter/project

```
{
  "options": ["Test-Project"]
}
```

POST (http) {{vid_ip}}:{{vid_port}}/vid/maintenance/category_parameter/owningEntity

```
{
  "options": ["Test-Entity"]
}
```

POST (http) {{vid_ip}}:{{vid_port}}/vid/maintenance/category_parameter/lineOfBusiness

```
{
  "options": ["Test-Business"]
}
```

Preload SDNC (https://wiki.onap.org/display/DW/Tutorial_vIMS+%3A+SDNC+Updates)

- Create username and password: {{sdnc_ip}}:8843/signup
- Login: {{sdnc_ip}}:8843/login
- Preload topology information: {{sdnc_ip}}:8282/apidoc/explorer/index.html
  - Username/password: admin/Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U
  - POST /VNF-API/operations/VNF-API/preload-vnf-topology-operation

Instantiate VF Module via VID (https://wiki.onap.org/display/DW/Tutorial+vIMS%3A+VID+Instantiate+the+VNF)

Run heatbridge from Robot VM

- bash /opt/demo heatbridge <OPENSTACK_vFW_STACK_NAME> <Service_Instance_ID> <Service Type>
  - <OPENSTACK_vFW_STACK_NAME>: it's the base VF module name (and also the vFW VM name)
  - <Service_Instance_ID>: it's the service instance ID in the VID GUI
  - <Service Type>: vFW

Create Mount Point in APPC (https://wiki.onap.org/display/DW/Automatically+Creating+a+Netconf+Mount+in+APPC+from+SDNC)

PUT {{appc_ip}}:8282/restconf/config/network-topology:network-topology/topology/topology-netconf/node/${vpg_id}

- Username/password: admin/Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U
- Header: Content-type: application/xml
- ${prop.vpg_hostname} in the XML body is the VNF ID in the VID GUI (vPacketGen VNF Instance information button). Example of XML body:

```
<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">2831</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>

Check from APPC if the vPacketGen is mounted correctly

Connect to: {{appc_ip}}:8282/apidoc/explorer/index.html

- Username/password: admin/Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U
- Mounted Resources/PacketGen-vnf-id/sample-plugin(date)
    - The get operation should return the running streams: GET yang-ext:mount/sample-plugin:sample-plugin/pg-streams
- Logs in APPC VM:
    - /var/log/onap/appc/karaf.log
- Logs in Policy VM:
    - /var/log/onap/policy/pdpd/network.log
    - /var/log/onap/policy/pdpd/error.log
    - kubectl exec -it dev-drools-0 -n onap -- bash -c "tail -f /var/log/onap/policy/pdpd/network.log"

Update the Operational Policy

The Operational Policy needs to know the invariant UUID of the vPacketGen.

- Download the CSAR file of the vFW service from SDC
- Get the vPacketGen invariant UUID from {CSAR_HOME}/Definitions/service-VfirewallTest1106-template.yml or as model-invariant-id in the Generic VNF in AAI
    - VfirewallTest1106 is the name of the service in the SDC catalog
- Run the update-vfw-op-policy.sh script by providing:
    - IP of the Policy VM
    - vPacketGen invariant UUID
    - Path to the private key of the Policy VM

For OOM Beijing, policies must be loaded first (https://wiki.onap.org/display/DW/Policy+on+OOM):

- Login to PAP
- Copy push-policy.sh to a non read-only directory
    - cp /tmp/policy-install/config/push-policies.sh /tmp/policy-install
- Change vFW policy resourceID in /tmp/policy-install/push-policies.sh to reflect the real vPacketGen model-invariant-id, e.g.:
    - sed -i "s/Eace933104d443b496b8.nodes.heat.vpg/02c953b7-e626-4e16-9874-6191572949a0/g" push-policies.sh
- From Rancher VM, run: kubectl exec -it dev-pap-7ff989696d-s86wj -c pap -n onap -- bash -c "export PRELOAD_POLICIES=true; /tmp /policy-install/push-policies.sh"

Event monitoring

VES reporting: {{mr_ip}}:3904/events/unauthenticated.VES_MEASUREMENT_OUTPUT/mygroup/myid?timeout=5000

ONSET events to Policy: {{mr_ip}}:3904/events/unauthenticated.DCAE_CL_OUTPUT/mygroup/myid?timeout=5000

In OOM, the port number is 30227