

Robot instantiateVFWCL

This page will walk through the robot framework automated creation of the vFirewall Closed Loop (vFWCL) use case to show how the overall vFWCL works and to high light the confirmation steps and training that can be derived from the robot automated test.

Preconditions:

- 1. The demonstration VNFs like vFWCL assume that the ONAP OAM network is setup in your environment as a 10.0.0.0/16 with a prefix cidr of 10.0. Robot will assign OAM IP addresses in this range.
- 2. The demonstration VNF heat templates assume that the PUBLIC neutron network (openStackPublicNetId) provided in the robot section of the override.yaml / values.yaml chart is setup so that new interfaces can be created via the VNF Heat template on that public interface. It need not be an external public network and many environments use a separate external network with floating IPs to assign an external IP to the VNFs external facing ip address/ports after instantiation if they have that setup.
- 3. ONAP has been installed
- 4. robot init has been run ((e.g., ./demo-k8s.sh onap init)) to setup the customer , cloud configurations, distribution of models and set the owning entity/project/line of business in VID for tracking the Service and VNF
- 5. Additionally a distribution should be run (./ete-k8s.sh onap healthdist) to confirm that model distribution succeeds

```
cd /root/onap/kubernetes/robot
./ete-k8s.sh onap health
./ete-k8s.sh onap healthdist
./demo-k8s.sh onap init
./demo-k8s.sh onap init_robot
```

The "init_robot" simply sets the password for the test account on the robot web page that we will be using.

<http://<k8s-node-ip>:30209/logs/> is the page to see the robot test cases results and from there we will walk through the results.

Select one of the test cases from the list and then open log.html (e.g., http://<k8host-ip>:30209/logs/0005_ete_instantiateVFWCL/log.html)

[TestsuitesTestLogvFWCL_Dublin.html](#)

10.12.5.150:30209/logs/0021_ete_instantiateVFWCL/log.html

Testsuites Test Log

Generated
20190710 11:47:07 GMT-05:00
30 minutes 3 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:14:47	<div></div>
All Tests	1	1	0	00:14:47	<div></div>

Statistics by Tag

Total	Pass	Fail	Elapsed	Pass / Fail	
InstantiateVFWCL	1	1	0	00:14:47	<div></div>

Statistics by Suite

Total	Pass	Fail	Elapsed	Pass / Fail	
Testsuites	1	1	0	00:14:48	<div></div>
Testsuites_Demo	1	1	0	00:14:48	<div></div>

Test Execution Log

SUITE Testsuites

Full Name: Testsuites

Source: /var/opt/ONAP/robot/testsuites

Start / End / Elapsed: 20190710 11:31:37.186 / 20190710 11:46:25.325 / 00:14:48.139

Status: 1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

SUITE Demo

./ete-k8s.sh onap instantiateVFWCL

root@onap-rancher:~/oom/kubernetes/robot# ./ete-k8s.sh onap instantiateVFWCL

++ export NAMESPACE=onap

++ NAMESPACE=onap

+++ kubectl --namespace onap get pods

```
+++ sed 's/ .*//'
+++ grep robot
++ POD=dev-robot-robot-7bf98dfb65-k2p65
++ TAGS='-i instantiateVFWCL'
++ ETEHOME=/var/opt/ONAP
+++ kubectrl --namespace onap exec dev-robot-robot-7bf98dfb65-k2p65 -- bash -c 'ls -1q /share/logs/ | wc -l'
++ export GLOBAL_BUILD_NUMBER=21
++ GLOBAL_BUILD_NUMBER=21
+++ printf %04d 21
++ OUTPUT_FOLDER=0021_ete_instantiateVFWCL
++ DISPLAY_NUM=111
++ VARIABLEFILES='-V /share/config/vm_properties.py -V /share/config/integration_robot_properties.py -V /share/config/integration_preload_parameters.py'
++ VARIABLES='-v GLOBAL_BUILD_NUMBER:12049'
++ kubectrl --namespace onap exec dev-robot-robot-7bf98dfb65-k2p65 -- /var/opt/ONAP/runTags.sh -V /share/config/vm_properties.py -V /share/config/integration_robot_properties.py -V /share/config/integration_preload_parameters.py -v GLOBAL_BUILD_NUMBER:12049 -d /share/logs/0021_ete_instantiateVFWCL -i instantiateVFWCL --display 111
Starting Xvfb on display :111 with res 1280x1024x24
Executing robot tests at log level TRACE

=====

Testsuites

=====

Testsuites.Demo :: Executes the VNF Orchestration Test cases including setu...

=====

Instantiate VFWCL
Downloaded:service-Vfwcl201907101631-csar.csar
Set VNF ProvStatus: e26ed2db-43a4-40bd-9880-65a02dbf734f to ACTIVE
Set VNF ProvStatus: fc482c95-3d58-45e5-b6c0-bc0321c049a1 to ACTIVE
Customer Name=DemoCust_aec5b5b9-6e62-4e07-bef6-88bddacd5132
VNF Module Name=Vfmodule_Ete_vFWCLvFWSNK_aec5b5b9_0
VNF Module Name=Vfmodule_Ete_vFWCLvPKG_aec5b5b9_1
Update old vFWCL Policy for ModelInvariantID=0b663b8a-2b4a-42d8-9ea6-f965602be1fc
Create vFWCL Monitoring Policy
Create vFWCL Operational Policy
{'u'content': 'u'controlLoop%3A%0A+++++version%3A+2.0.0%0A+++++controlLoopName%3A+ControlLoop-vFirewall-0b663b8a-2b4a-42d8-9ea6-f965602be1fc%0A+++++trigger_policy%3A+unique-policy-id-1-modifyConfig%0A+++++timeout%3A+1200%0A+++++abatment%3A+false%0A+polices%3A%0A+++++id%3A+unique-policy-id-1-modifyConfig%0A+++++name%3A+modify_packet_gen_config%0A+++++description%3A%0A+++++actor%3A+APPC%0A+++++recipe%3A+ModifyConfig%0A+++++target%3A%0A+++++resourceID%3A+0b663b8a-2b4a-42d8-9ea6-f965602be1fc%0A+++++type%3A+VNF%0A+++++payload%3A%0A+++++streams%3A+%27%7B%22active-streams%22%3A5%7D%27%0A+++++retry%3A+0%0A+++++timeout%3A+300%0A+++++success%3A+final_success%0A+++++failure%3A+final_failure%0A+++++failure_timeout%3A+final_failure_timeout%0A+++++failure_retries%3A+final_failure_retries%0A+++++failure_exception%3A+final_failure_exception%0A+++++failure_guard%3A+final_failure_guard%0A', 'u'policy-id': 'u'operational.modifyconfig'}
Push vFWCL To PDP Group
Validate vFWCL Policy
APPC Mount Point for VNF Module Name=Vfmodule_Ete_vFWCLvFWSNK_aec5b5b9_0
```

APPC Mount Point for VNF Module Name=Vfmodule_Ete_vFWCLvPKG_aec5b5b9_1

| PASS |

Testsuites.Demo :: Executes the VNF Orchestration Test cases inclu... | PASS |

1 critical test, 1 passed, 0 failed

1 test total, 1 passed, 0 failed

Testsuites | PASS |

1 critical test, 1 passed, 0 failed

1 test total, 1 passed, 0 failed

Output: /share/logs/0021_ete_instantiateVFWCL/output.xml

Log: /share/logs/0021_ete_instantiateVFWCL/log.html

Report: /share/logs/0021_ete_instantiateVFWCL/report.html

This one line will do all the steps required to instantiate and setup the Virtual Machines for closed loop control in about 20 minutes.

Either the instantiateVFWCL or the instantiateVFW can be used to confirm that your ONAP installation and configuration (particularly to Openstack) is correct.

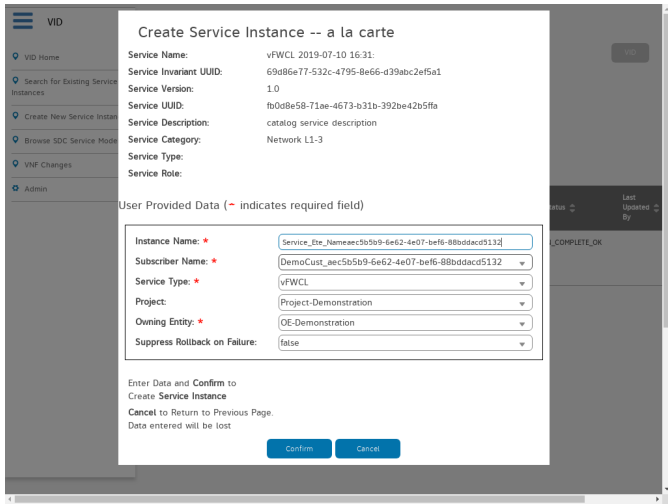
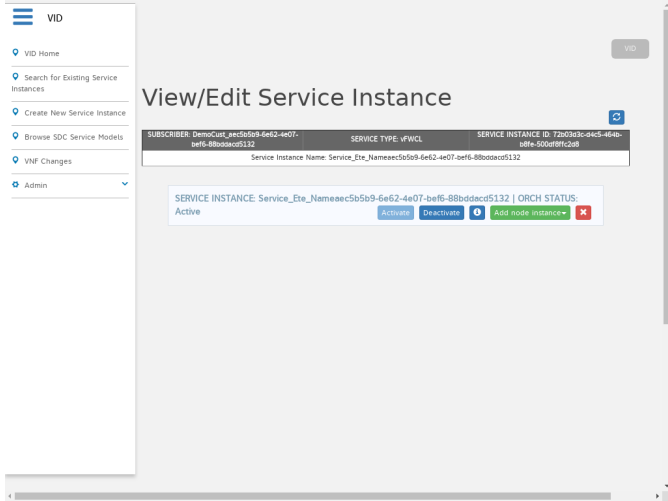
This table will cover the important steps in the log.html file to highlight items of note during the test case.

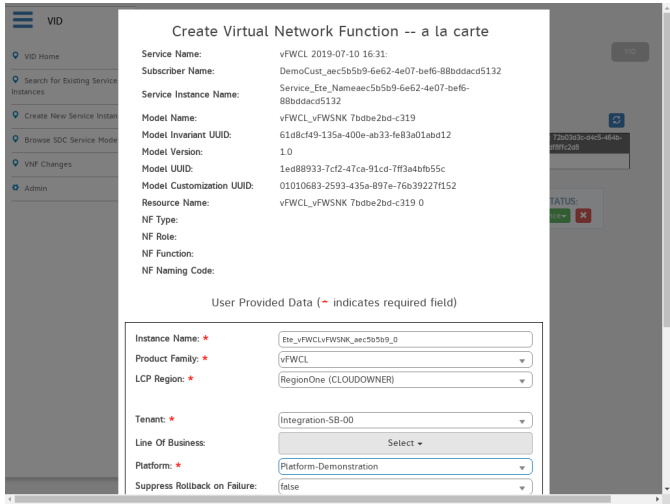
NOTE: The important difference between robot's use of the SDC internal API's is that the service is created first and items are attached to the service. In the GUI , the items are attached to a temporary structure and then a service is created by dragging the resources from the palette onto the service. The result is the same service and resource artifacts. References to ASDC are legacy tags that in the future will correctly be edited to be SDC.

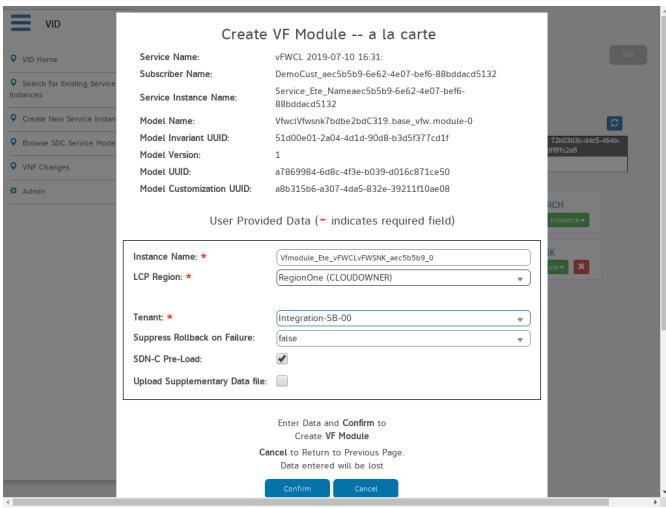
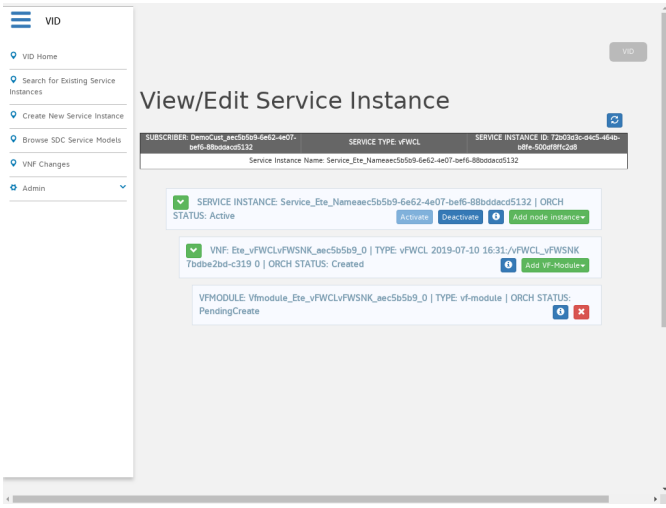
	Description (Keyword)	Comment
1.0	Setup Orchestrate VNF	These steps will pull data from openstack and create the data in A&AI if it does not exist for the tenant, region, network zone and complex.
	Initialize Tenant From Openstack	Set the tenant_name and tenant_id
	Initialize Regions From Openstack	Get the regions
	Inventory the Region If Not Exists	For all regions (RegionOne) under the tenant check A&AI and creates them if they do not exist
	Inventory Zones If Not Exists	Checks A&AI for the network zone (zone1) and creates it if it does not exist
	Inventory Complex If Not Exists	Checks A&AI for the complex (cli1) and creates it if it does not exist
2.0	Orchestrate the VNF	This will onboard the vendor software product (vFWCL) , create a service and distribute the service
	Get the Region	
	Generate a unique Customer Name	
	Generate a unique Service Name	

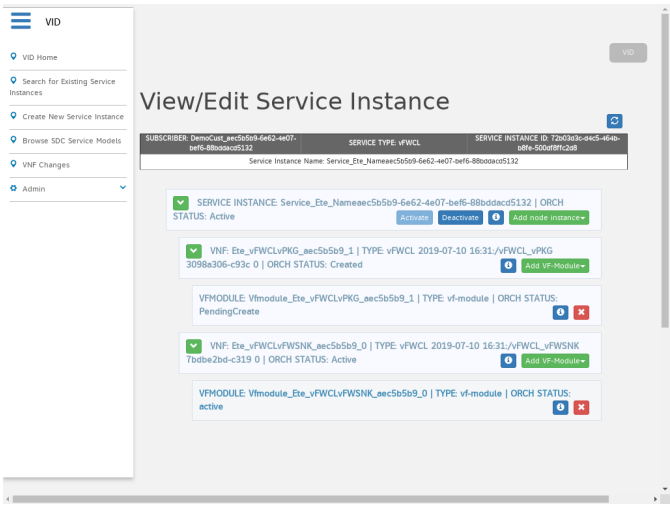
Model Distribution For Directory	<p>This step will onboard the vendor software products (heat templates) in the directory requested which in this case is for the vFWCL (heat templates are from the ONAP demo repository).</p> <p>For vFWCL there are two v VNFs in the service. vFWSNK (vFW and traffic sink) and vFWPKG (a packet generator).</p> <p>The VES telemetry will come from the vFW and the control loop will modify the packet generator to control the traffic.</p> <p>These steps are start the process of the service designer role as "cs0008"</p>
vFWCL/vFWSNK . Create Zip From Files In Directory	These steps will create a zip file with the .yaml, .env and the manifest from the vFWCL/vFWSNK directory
vFWCL/vPKG . Create Zip From Files In Directory	These steps will create a zip file with the .yaml, .env and the manifest from the vFWCL/vPKG directory
Distribute Model From ASDC	This will start to use the zip files just created to onboard the vendor software product
Add ASDC Catalog Service	Robot creates the service first to have a bucket in which to put artifacts. In the GUI , the service is created last.
vFWSNK: Set up ASDC Catalog Resource	for each VNF robot will create a license
Add ASDC License Model	
Add ASDC License Group	
Add ASDC Entitlement Pool	
Add ASDC Feature Group	
Add ASDC License Agreement	
Submit ASDC License Model	With all the parts of the license added the license model is submitted.
Add ASDC Software Product	The vendor software product meta data is created
Upload ASDC Heat Package	The zip file is uploaded
Validate ASDC Software Product	Ask SDC to parse the uploaded zip file
Submit ASDC Software Product	Submit the vendor software product
Package ASDC Software Product	
Add ASDC Catalog Resource	Turn the vendor software product into a catalog resource
Certify ASDC Catalog Resource	Make the catalog resource available for use in a service
vFWCL_vPKG:	
(repeat above steps from the vFWCL_vFWSNK) from Setup to Certify	
Checkin ASDC Catalog Service	Since robot already created the service and attach the catalog resource to it we only need to checkin the catalog service so that it is available for testing, governance approval and distribution
Request Certify ASDC Catalog Service	Last step by service designer (cs0008)

	Start Certify ASDC Catalog Service	Service tester role (jm0007) opens the service
	Certify ASDC Catalog Service	Service tester role certifies the service.
	Approve ASDC Catalog Service	Governance role approves the service (gv0001)
	Distribute ASDC Catalog Service	Operations role distributes the service (op0001)
		Distribution occurs to AAI, SDNC, SO and other components.
	Loop Over Check Catalog Service Distributed	This will loop over the get Distribution Details and look for the results of SO to mark the distribution complete when AAI, SDNC and SO have indicated DEPLOY_OK and no other component has raised a DEPLOY_ERROR in the watchdog timer interval (typically 5 minutes). VID queries AAI to see service in the DISTRIBUTION_COMPLETE_OK state in order to know to display the DEPLOY button.
3.0	Instantiate the Service (this is not a keyword but a change from SDC onboarding to using VID)	
	Create Customer For VNF	This is the first step on the VID screen. Robot creates a unique customer for each run of the test case to avoid conflicts and to make it easier to find the VNFs created in this test case. Robot creates the customer in A&AI via the REST interface
	Login To VID GUI	Robot got to the VID GUI directly
	Select From List By Label (VNF-API)	This test case uses the original VNF-API. You have to select this on the VID Home Screen
	Create VID Service Instance (Wait for Success)	
	Go To VID HOME	This is the starting point for robot VID interactions (the selected VNF-API is remembered in VID for this session)
	Go To VID Browse Service Models	The wait for is intended to handle delays in the model being available to VID from the AAI query for DISTRIBUTION_COMPLETE_OK models
	Filter on the service model name: FWCL 2019-06-24 15:49:	
	Wait Until Page Contains Element .. "DEPLOY"	
	Press Key (Click on Deploy button)	The create service "Instance Name" screen should now come up
	Select From List When Enabled //select [@prompt=Select Subscriber Name],	
	Select From List When Enabled //select [@prompt=Select Service Type],	
	Select From List When Enabled //select [@prompt=Select Project Name]	
	Select From List When Enabled //select [@prompt=Select Owning Entity]	

	<p>Input Text When Enabled, //input [@parameter-name='Instance Name']</p>	
	<p>Click on the Confirm button</p>	<p>VID will now submit the request to create the service to SO</p>
	<p>Poll MSO Get Request</p>	
	<p>Validate Service Instance</p>	<p>Query A&AI to make sure the service instance is created</p>
3.1	<p>vFWCLvFWSNK</p>	<p>For each VNF in the service create the VNF and the VF Module</p>
	<p>Create a unique VNF name</p>	
	<p>Create VID VNF</p>	
	<p>Click Element button=Add node instance</p>	<p>Select the drop down of VNFs in the service</p>
	<p>Click Element xpath=//a [contains(text(), '{\$vnf_type}')]</p>	<p>Select the vFWCLvFWSNK VNF This will cause the Create VNF Dialog Box to pop up</p>
	<p>Input Text xpath=//input [@parameter- id='instanceName']</p>	
	<p>Select From List By Label xpat h=//select[@parameter- id='productFamily'],</p>	

		S elect From List By Label xpath =//select[@parameter- id='lcpRegion'],	
		S elect From List By Label xpath =//select[@parameter- id='tenant'],	
		C lick Element xpath= //multiselect[@parameter- id='lineOfBusiness']	
		Select From List By Label xpat h=//select[@parameter- id='platform'],	
	Click Element button=Confirm	This will cause VID to send the Create VNF Instance request to SO	
	Wait for Success and Close Dialog		
Vnf (SDNC)	Preload	Now that a VNF is created we can set the data that will be used for the VFModules keyed by VNF instance name. Robot will use the REST API into SDNC to insert the preload data. This is a POST to submit and a GET to confirm.	
	Create VID VNF module (VID)		
	Wait for the Add VF Module button to be visible		
	Click Element xpath=//div[contains (.,'{vnf_name}')]/div/button [contains(.,'Add VF-Module')]		
	Click Element link=\${vnf_type}	This will cause the Add VF Module dialog to appear	
	Input Text xpath=//input [@parameter- id='instanceName'],		
	Select From List By Label xpath= //select[@parameter- id='lcpRegion']		

	Select From List By Label xpath= //select[@parameter- id='tenant']	
	Select Checkbox xpath=//input [@parameter- id='sdncPreload']	
	Click Element button=Confirm	<p>This will cause VID to send the Create VF Module to SO. SO will query SDNC for preload and then call the Openstack API's to create the stack</p> 
	Poll MSO Get Request	this can take minutes based on when the Openstack heat engine returns success to SO
		The VNF should now exist
	Validate Generic VNF	Query A&AI to see artifacts that SO and SDNC have written
	Execute Heatbridge	This will query openstack and update A&AI with items like the dhcp addresses assigned to the VM's It also does a reverse update of AAI with openstack data.
3.2	vFWCLvPKG :	

	<p>repeat the above steps for the second VNF from:</p> <p>Create a unique VNF name to Execute Heatbridge</p>	
4.0	Update vFWCL Policy	This will create and push the control loop policy to the DROOLS engine in policy
	Create vFirewall Monitoring Policy	This is a future capability that is not effective since DCAE doesn't pick up this policy in Dublin
	Create vFirewall Operational Policy	The Control Loop Name is created and the resourceID inserted into the policy so that Policy can find the VNF data during its processing for control loops
	Push vFirewall Policies To PDP Group	
	Validate the vFWCL Policy	
5.0	For each vfm module that has a defined variable for the mount $\{vf_module_name\} = Vfm_module_Ete_vFWCLvPKG_c7fca777_1$ has a vpg_name_0 variable (vFWSNK does not)	
	Create the APPC Mount Point	Create the NETCONF mount point in APPC for closed loop control
	Get the IP address from Openstack	
	Run APPC Put Request	
	<END>	<p>The Virtual Machines and networks are created from the two heat stack, A&AI has been updated and marked the service as Active, the policy has been pushed and the APPC netconf mount has been created.</p> <p>The DCAE TCA Policy needs to be updated with the matching Control Loop Name from the Operation</p> <p>vFW Closed Loop operation testing can begin as soon as the VNF finishes its application install and configuration.</p> <p>Traffic should be visible on port 667 of the traffic sink VM.</p>