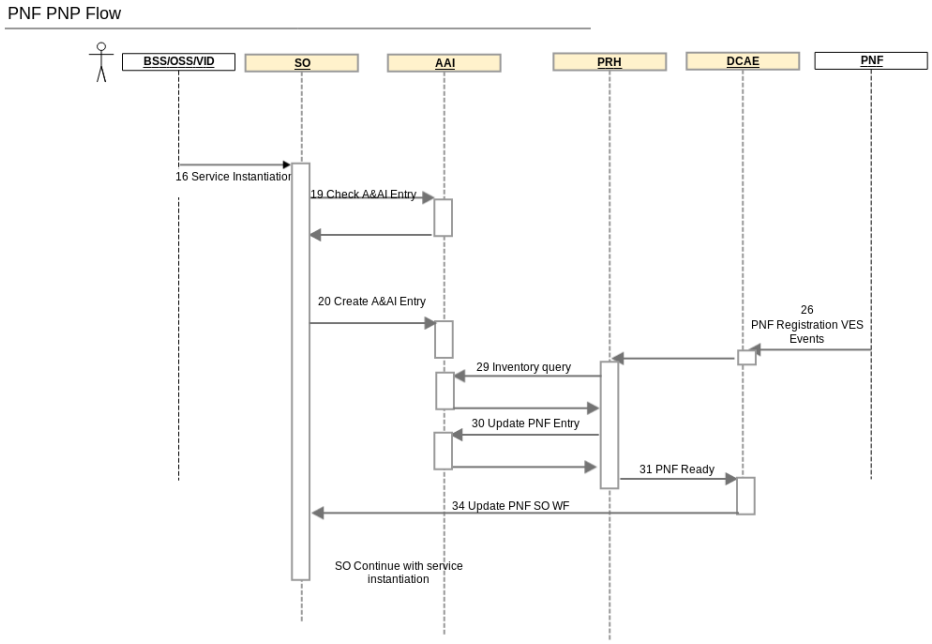
5G - PNF PnP - Integration Test Cases

Test update **in progress** for Guilin release and SO building block flow

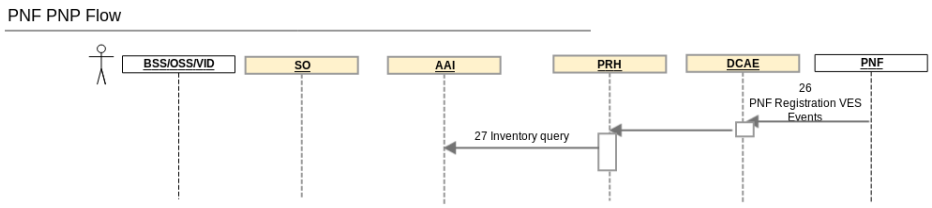


Link to specification:

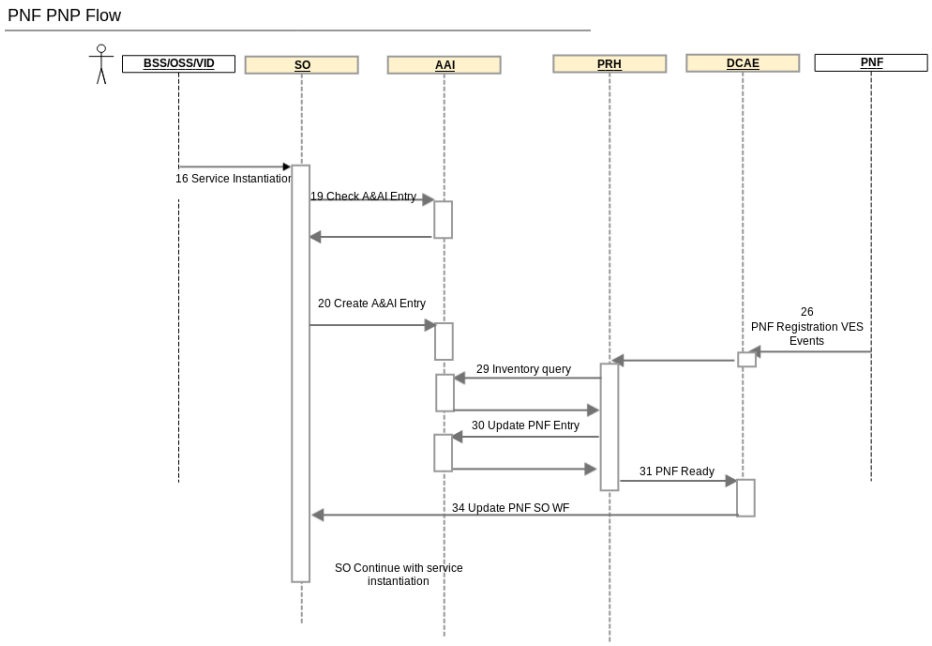
5G - PNF Plug and Play

PNF PNP message flow diagram

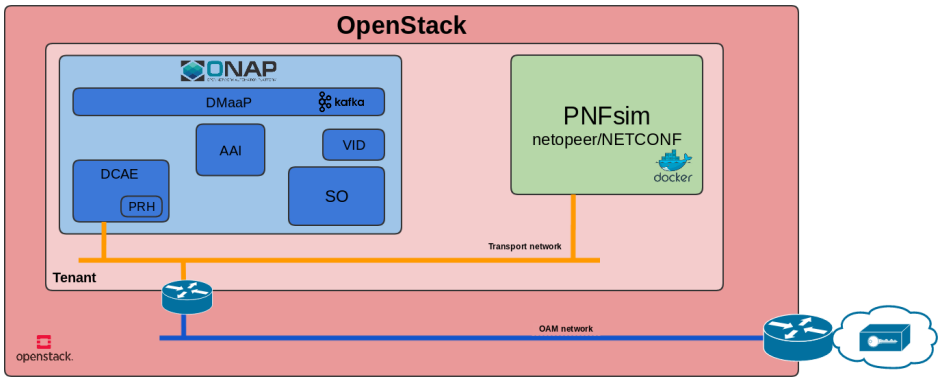
Rainy day scenario - PNF is sending registration request to ONAP where required AAI entry is not present:



Sunny day scenario - PNF is sending registration request to ONAP where required AAI entry is prepared by SO workflow:



PNF PNP deployment diagram



PNF PNP hardware requirements

- ONAP - **standard ONAP instance located in Wind River lab**
- **4 Virtual machine**s dedicated for PNF Simulator:
  - **2 VCPU**

- **8 GB of RAM**
- **32 GB of HDD**

PNF PNP test cases

High-Level descriptions

Id.	Test Case Name	Test Case Description
T01	Create and distribute service which contains PNF based on imported VSP	Verification if in VID is present PNF enabled service.
T02	PNF registration accepting when AAI entry created in advance	Verification if PNF resource registration is done properly when correct AAI record (based on *correlationID*) is present before first *InventoryQuery* is done by PRH. Verification if AAI entries: *ipaddress-v4-oam* and *ipaddress-v6-oam* are updated correctly based on *pnfRegistration* message contents.
T03	Delete pnf service and pnf resource	
T04	Delete pnf service instance and reasign pnf resource to another service instance	
T05	PNF registration rejected	Verification if PRH drops the *PnfRegistration* request when no AAI entry exists for the *corr elationID*. AAI entries shall not be created by PRH.
T06	PNF registration accepted when AAI entry is created using AAI API (without SO instantiation)	Verification if PNF resource registration is done properly when correct AAI record (based on *correlationID*) is present - created using AAI API Verification if AAI entries: *ipaddress-v4-oam* and *ipaddress-v6-oam* are updated correctly based on *correlationID.*

Detailed descriptions

Test Case ID	T01
Test Case Name	**Create and distribute service which contains PNF based on imported VSP**
Description	Verification if in VID is present PNF enabled service. Test case covers following steps from message flow in 5G - PNF Plug and Play:
Release	Frankfurt/Guilin

Preconditions	1. Users with roles: Designer (Carlos Santana - cs0008), Admin (demo) should be available 2. Robot init sucesfully executed. In intsalation server followin script shoulkd be sucesfully ececuted. **~/oom/kubernetes/robot/demo-k8s.sh onap init**

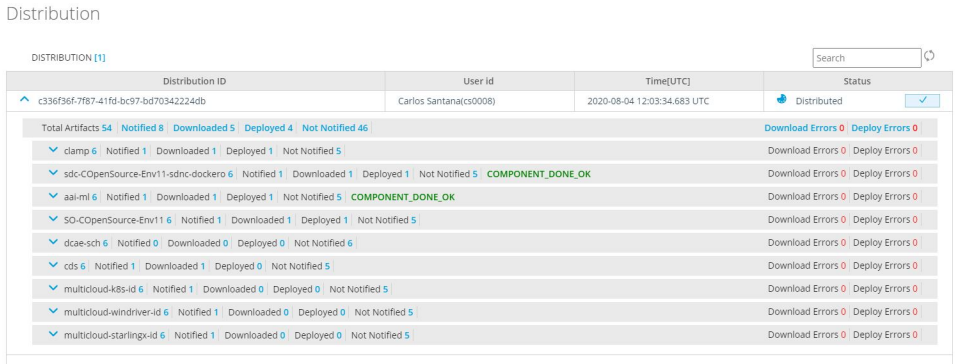
```
ubuntu@onap-5915-rke-node:~$ ~/oom/kubernetes/robot/demo-k8s.sh onap init
Number of parameters:
2
KEY:
init
++ kubectl --namespace onap get pods
++ sed 's/ .*//'
++ grep robot
+ POD=dev-robot-6d67844b64-pgbx5
++ dirname ./demo-k8s.sh
+ DIR=.
+ SCRIPTDIR=scripts/demoscript
+ ETEHOME=/var/opt/ONAP
+ '[' ']'
++ kubectl --namespace onap exec dev-robot-6d67844b64-pgbx5 -- bash -c 'ls -1q /share/logs/ | wc -l'
+ export GLOBAL_BUILD_NUMBER=0
+ GLOBAL_BUILD_NUMBER=0
++ printf %04d 0
+ OUTPUT_FOLDER=0000_demo_init
+ DISPLAY_NUM=90
+ VARIABLEFILES='-V /share/config/robot_properties.py'
+ kubectl --namespace onap exec dev-robot-6d67844b64-pgbx5 -- /var/opt/ONAP/runTags.sh -V /share
/config/robot_properties.py -d /share/logs/0000_demo_init -i InitDemo --display 90
Starting Xvfb on display :90 with res 1280x1024x24
Executing robot tests at log level TRACE
==============================================================================
Testsuites
==============================================================================
Testsuites.Demo :: Executes the VNF Orchestration Test cases including setu...
==============================================================================
Initialize Customer And Models                                        | PASS |
------------------------------------------------------------------------------
Initialize SO Openstack Identity For V3                               | PASS |
------------------------------------------------------------------------------
Testsuites.Demo :: Executes the VNF Orchestration Test cases inclu... | PASS |
2 critical tests, 2 passed, 0 failed
2 tests total, 2 passed, 0 failed
==============================================================================
Testsuites                                                            | PASS |
2 critical tests, 2 passed, 0 failed
2 tests total, 2 passed, 0 failed
==============================================================================
Output:  /share/logs/0000_demo_init/output.xml
Log:     /share/logs/0000_demo_init/log.html
Report:  /share/logs/0000_demo_init/report.html
```

Testing Steps	Step	Expected Result
	1. Login to ONAP portal as a designer user 2. Navigate to **SDC** application 3. Navigate to **ONBOARD** tab 4. Click on **CREATE NEW VLM** 5. Fill all mandatory parameters 6. Add **License Key Group** and fill all mandatory parameters	

7. Add **Entitlement Pool** and fill all mandatory parameters
8. Add **Feature Group** fill all mandatory parameters and add already created **Entitlement Pool** and **License Key Group**
9. Add **License Agreement** fill mandatory parameters and add already created **Feature Group**
10. Press **Submit** button and next **COMMIT & SUBMIT** button
11. Navigate to **ONBOARD** tab
12. Click on **CREATE NEW VSP**
13. Fill all mandatory parameters:

    a. select Vendor defined in already created VLM
    b. Name will be used in next steps
    c. in **ONBOARDING PROCEDURE** select **Network Package**
14. Click on warning under License Agreement
15. Fill Licensing Version, License Agreement and Feature Groups from already created VLM
16. Click on Overview from left menu and press SELECT FILE button.
17. Select attached pNF.csar file from your PC
18. Press **Submit** button and next **COMMIT & SUBMIT** button
19. Navigate to **Home** tab
20. Click on **IMPORT** button and select **IMPORT VSP**
21. Select previously created **VSP** and press **IMPORT VSP** button
22. In newly opened window yoiu can moduify PNF name.
23. Next click **Create** and **Certify** button
24. Next press **Certify** button . Put comment message in **Certification confirmation** pop-up and press **OK** button
25. Navigate to **SDC** tab / **HOME** menu
26. Click on **ADD +** button and select **Add Service**
27. In **HOME > Create new service > General** page fill all mandatory fields (change service type from default **a'lacarte** to **macro**) and press **Create** button in right top corner
28. Go to **Composition** in left menu
29. In search box find crated PNF using its name

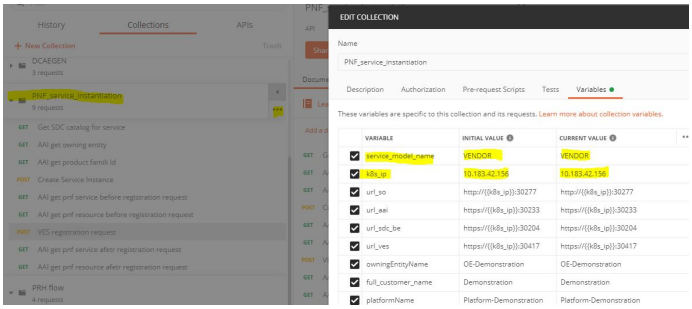
1. User is logged in
2. SDC application is open
3. ONBOARD CATALOG is visible
4. New License Model window is present
5. All mandatory parameters are fulfilled
6. License Key Group is added
7. Entitlement Pool is added
8. Feature Group is added
9. License Agreement is added
10. VLM is submitted sucesfully
11. VLM is visible
12. New Software Product window is present
13. All mandatory parameters are fulfilled
14. Licensing Version, License Agreement and Feature Groups and warning is not present
15. Licensing Version, License Agreement and Feature Groups are defined
16. Select file window is opened
17. File is selected
18. File is sucesfully uploaded and VSP is submited
19. Home tab is opened
20. Import VSP window is presnet with list of dubmited VSPs
21. Previously create VSP is imported as VSP and Create window is opened
22. All modifications are present
23. PNF is successfully **Certified**
24. SDC application is open
25. Page HOME > Create new service > General with fields that must be filled out is open
26. Create/Update saved successfully message is present in left top corner
27. Composition main view is open
28. PNF is visible
29. PNF is added to service
30. Service is Cerified and ready for Distribution
31. Distribute Distribute successfully message is present in left top corner
32. Monitor main view is opened with information about distribution. Thesr should not be any errors.



33. User is re-logged
34. VID application is open
35. In Browse SDC Service Models main view is present entry about newly created service - it can take couple of minutes ~ 15 minutes

	30. Drag ad drop it to main view 31. Next click **Certify** button 32. Next Press **Distribute** button in left top corner 33. Press**Distribution** button in left top corner and verify distribution status, use refresh button 34. Re-login as a a demo user 35. **VID** Navigate to application 36. From left menu select **Browse SDC Service Models**	
Actual Results	In VID is present enabled PNF service.	
Conclusion (Pass/Fail)		
Testing Lab		
Tester Name	Krzysztof Kuzmicki	

Test Case ID	T02
Test Case Name	**PNF registration accepting when AAI entry created in advance**

Description	Verification if PNF resource registration is done properly when correct AAI record (based on *correlationID*) is present before first *InventoryQuery* is done by PRH. Verification if AAI entries: *ipaddress-v4-oam* and *ipaddress-v6-oam* are updated correctly based on *correlationID.* Test case covers following steps from message flow in 5G - PNF Plug and Play: Whole test case can be also executed using postman collection PNF_service_instantiation_v2.postman_collection.json. In order to execute it successfully there is need to set two variables in collection variables: ▪ name of servcie model for pNF ▪ ip onap worker/k8s VM  Also whole test case including **T01** is automated in robot/xtesting robot smoke image - **`~/oom/kubernetes/robot/demo-k8s.sh onap pnf_registarte`**
Release	Frankfurt/Guilin
Preconditions	**1.** Created PNF and Service using Test Case **T01 (Create and distribute service which contains PNF based on imported VSP**)

Testing Steps	Step	Expected Result
	1. Get created service model in Test Case T01 curl --location --request GET 'https://**{worker_ip}**:30204/sdc2/rest/v1/catalog/services/serviceName/**{service model name}**/serviceVersion/1.0' \ --header 'USER_ID: cs0008' \ --header 'X-FromAppId: robot-ete' \ --header 'Content-Type: application/json' \ --header 'Accept: application/json' \ --header 'Authorization: Basic YmVlcDpib29w' **2.** From response get following parameters • **service_model_uuid** resp_json.**uuid**; • **service_model_invariant_uuid** -> resp_json.**invariantUUID** • **nf_resource_name** -> resp_json.componentInstances[0].**name** • **nf_resource_uuid** -> resp_json.componentInstances[0].**customizationUUID** • **componentName** -> resp_json.componentInstances[0].**componentName** • **nf_model_invariant_uuid** -> resp_json.**invariantUUID** • **nf_model_uuid** -> resp_json.**uuid** • **nf_model_name** -> resp_json.**name** **3.** Get owning entity id: curl -k -O --location --request GET 'https://{worker_ip}:30233/aai/v13/business/owning-entities?owning-entity-name=**OE-Demonstration**' \ --header 'Content-Type: application/json' \ --header 'X-FromAppId: dcae-curl' \ --header 'x-transactionId: 9998' \ --header 'Accept: application/json' \ --header 'Authorization: Basic QUFJOkFBSQ==' From response get following parameter: • **owningEntityId** -> resp_json.owning-entity[0].**owning-entity-id**	**1.** Service model is present **2.** Parameters are present **3.** Owning Entity ID is present **4.** Product Family ID is present **5.** SO_request.json is filled accordingly **6.** SO request is send successfully. Service_instance_id is saved from POST response (example of response: {"requestReferences":{"requestId":"10e2577f-3547-4e66-901d-b7b0c6e1d3ab","**instanceId**":"10004a65-20c4-44d2-bd27-53544fe99916","requestSelfLink":"http://10.183.42.156:30277/orchestrationRequests/v7/10e2577f-3547-4e66-901d-b7b0c6e1d3ab"}})

4. Get product family id:
   curl -k -O --location --request GET 'https://{**worker_ip**}:30233/aai/v13/service-design-and-creation/services?service-description=**gNB**' \
   --header 'Content-Type: application/json' \
   --header 'X-FromAppId: dcae-curl' \
   --header 'x-transactionId: 9998' \
   --header 'Accept: application/json' \
   --header 'Authorization: Basic QUFJOkFBSQ=='

   From response get following parameter:
   - **productFamilyId** -> resp_json.service[0].**service-id**
5. Fill SO_request.json with above paremeters and:
   - owningEntityName=OE-Demonstration
   - full_customer_name=Demonstration
   - platformName=Platform-Demonstration
   - lineOfBusinessName=LOB-Demonstration
   - service=gNB
   - nf_instance_name=**<your pnf name - send by pnf>**
6. Send Instantiation request to SO
   curl -k --request POST 'http://{worker_ip}:30277/onap/so/infra/serviceInstantiation/v7/serviceInstances' \
   --header 'Authorization: Basic SW5mcmFQb3J0YWxDbGllbnQ6cGFzc3dvcmQxJA==' \
   --header 'Content-Type: application/json' \
   --header 'Accept: application/json' \
   -d @SO_request.json
7. Login to so-bpmn-infra pod via rke console:
   kubectl exec -it dev-so-bpmn-infra-<....> -n onap /bin/sh
   open debug.log
   vi logs/bpmn/debug.log
8. Verify AAI entry for PNF created by SO service using command:

   curl --location --request GET 'https:///{**worker_ip**}:30233/aai/v17/network/pnfs/pnf/{**nf_instance_name**}' \
   --header 'Content-Type: application/json' \
   --header 'X-FromAppId: dcae-curl' \
   --header 'x-transactionId: 9998' \
   --header 'Accept: application/json' \
   --header 'Authorization: Basic QUFJOkFBSQ==' \
   --header 'Cookie: JSESSIONID=2F951F19C99CDAED4CA5AFB3DCCD5D61'
9. Verify AAI entry for Service Instance created by SO service using command:

   curl --location --request GET 'https://{worker_ip}:30233/aai/v13/business/customers/customer/{**full_customer_name**}/service-subscriptions/service-subscription/{**service**}/service-instances/service-instance/{**instanceId**}' \
   --header 'Content-Type: application/json' \
   --header 'X-FromAppId: dcae-curl' \
   --header 'x-transactionId: 9998' \
   --header 'Accept: application/json' \
   --header 'Authorization: Basic QUFJOkFBSQ==' \
   --header 'Cookie: JSESSIONID=2F951F19C99CDAED4CA5AFB3DCCD5D61'
10. Send PNF Registration request from real PNF or simulate it using following curl comand.
    - Fill registration_request.json:
      - set sourceName=**nf_instance_nam**
    - Send regitration request:

      curl -k --request POST 'https://{worker_ip}:30417/eventListener/v7' \
      --header 'Content-Type: application/json' \
      --header 'Authorization: Basic c2FtcGxlMTpzYW1wbGUx' \
      --header 'Cookie: JSESSIONID=17A9DC67B33C079DE46F4A304143A5C2' \
      -d @reqistration_request.json
11. Verify if SO service has reacted on **PNFReady** message from PRH and has ended
7. In debug log should be presnet following entry:

```
2020-08-20T20:59:35.865Z|87f367b7-5d84-47e9-a955-
6b2143e8424a|o.o.l.filter.spring.
SpringClientPayloadFilter -
=======================response
end==================================================
2020-08-20T20:59:35.875Z|87f367b7-5d84-47e9-a955-
6b2143e8424a|o.o.s.b.i.pnf.dmaap.
PnfEventReadyDmaapClient - registering for pnf ready
dmaap event for pnf correlation id: test_pnf_name
2020-08-20T20:59:35.881Z|87f367b7-5d84-47e9-a955-
6b2143e8424a|org.onap.so.client.RestClient -
RestClientSSL using default SSL context!
2020-08-20T20:59:35.894Z|87f367b7-5d84-47e9-a955-
6b2143e8424a|o.o.logging.filter.base.
PayloadLoggingClientFilter - Sending HTTP POST
(overridden to PATCH) to:https://aai.onap:8443/aai/v19
/network/pnfs/pnf
/test_pnf_name with request headers:{Authorization=
[***REDACTED***], X-RequestID=[87f367b7-5d84-47e9-a955-
6b2143e8424a], X-FromAppId=[MSO], X-ONAP-PartnerName=
[UNKNOWN], X-HTTP-Method-Override=[PATCH], Ac
cept=[application/json], X-InvocationID=[a9b7b7c1-4912-
4e48-8f3a-df7cd3123a54], X-ECOMP-RequestID=[87f367b7-
5d84-47e9-a955-6b2143e8424a], X-TransactionId=[], X-ONAP-
RequestID=[87f367b7-5d84-47e9-a955-6b21
43e8424a], Content-Type=[application/merge-patch+json]}
2020-08-20T20:59:35.895Z|87f367b7-5d84-47e9-a955-
6b2143e8424a|o.o.logging.filter.base.
PayloadLoggingClientFilter - {"pnf-id":"f792d78d-7c2c-
4858-980b-23968923b3f4","orchestration-status":"
Register"}

2020-08-20T20:59:35.982Z|87f367b7-5d84-47e9-a955-
6b2143e8424a|o.o.logging.filter.base.
PayloadLoggingClientFilter - Response from method:POST
(overridden to PATCH) performed on uri:https://aai.onap:
8443/aa
i/v19/network/pnfs/pnf/test_pnf_name has http status
code:200 and response headers:{Content-Length=[0],
content-type=[application/json], Date=[Thu, 20 Aug 2020
20:59:35 GMT], Strict-Transport-Security=[ma
x-age=16000000; includeSubDomains; preload;], vertex-id=
[254096], X-AAI-TXID=[2-aai-resources-200820-20:59:35:
904-22741]}
2020-08-20T20:59:35.983Z|87f367b7-5d84-47e9-a955-
6b2143e8424a|o.o.logging.filter.base.
PayloadLoggingClientFilter - Response was returned with
an empty entity.
2020-08-20T20:59:39.896Z||o.o.s.b.i.pnf.dmaap.
PnfEventReadyDmaapClient - dmaap listener starts
listening pnf ready dmaap topic
2020-08-20T20:59:54.485Z||o.o.s.b.i.pnf.dmaap.
PnfEventReadyDmaapClient - dmaap listener starts
listening pnf ready dmaap topic
2020-08-20T21:00:09.016Z||o.o.s.b.i.pnf.dmaap.
PnfEventReadyDmaapClient - dmaap listener starts
listening pnf ready dmaap topic
```

especially following entries:
correlation id: test_pnf_name same as nf_instance_name from SO request
and
dmaap listener starts listening pnf ready dmaap topic is pnf name

12. Once again Verify AAI entry for PNF created by SO service using command:

    curl --location --request GET 'https:///{**worker_ip**}:30233/aai/v17/network/pnfs/pnf/{**nf_instance_name**}' \
    --header 'Content-Type: application/json' \
    --header 'X-FromAppId: dcae-curl' \
    --header 'x-transactionId: 9998' \
    --header 'Accept: application/json' \
    --header 'Authorization: Basic QUFJOkFBSQ==' \
    --header 'Cookie: JSESSIONID=2F951F19C99CDAED4CA5AFB3DCCD5D61'
13. Once again Verify AAI entry for Service Instance created by SO service using command:

    curl --location --request GET 'https://{worker_ip}:30233/aai/v13/business/customers/customer/{**full_customer_name**}/service-subscriptions/service-subscription/{**service**}/service-instances/service-instance/{**instanceId**}' \
    --header 'Content-Type: application/json' \
    --header 'X-FromAppId: dcae-curl' \
    --header 'x-transactionId: 9998' \
    --header 'Accept: application/json' \
    --header 'Authorization: Basic QUFJOkFBSQ==' \
    --header 'Cookie: JSESSIONID=2F951F19C99CDAED4CA5AFB3DCCD5D61'

8. Request is successful. Request should contain following values:
   - "pnf-name": equals to **nf_instance_name**
   - "orchestration-status": in status **Register**
   - "relationship-list": with information about service id (**instanceId**) to which PNF is connected
   - "model-invariant-id": equals to **nf_model_invariant_uuid**
   - "model-version-id": equals to **nf_model_uuid**
   - "ipaddress-v4-oam": should be missing
   - "ipaddress-v6-oam": should be missing
   - "equip-type": should be missing
   - "equip-vendor": should be missing
   - "equip-model": should be missing
   - "sw-version": should be missing
   - "serial-number": should be missing
   - "nf-role": should be missing
9. Request is successful. Request should contain following values:
   - "service-instance-name": equals to **service_model_name_nf_instance_name**
   - "model-invariant-id": equals to **service_model_invariant_uuid**
   - "model-version-id": equals to **service_model_uuid**
   - "orchestration-status": equals to "Assigned",
   - "relationship-list": with information pnf name (**nf_instance_name**)
10. Regitration request is send sucesfully.
11. SO-BPMN pod in /app/logs/bpmn/debug.log should be present following message:
    2020-08-21T08:01:19.862Z||o.o.s.b.i.pnf.dmaap.PnfEventReadyDmaapClient - unregistering from pnf ready dmaap event for pnf correlation id: **nf_instance_name**
    2020-08-21T08:01:19.862Z||o.o.s.b.i.pnf.dmaap.PnfEventReadyDmaapClient - dmaap listener gets pnf ready event for pnfCorrelationId: **nf_instance_name**
    2020-08-21T08:01:19.930Z|626785f1-baea-427d-aa7a-cdfddc209f5d|org.onap.so.client.RestClient - RestClientSSL using default SSL context!
12. Request is successful. Request should contain following values:
    - "pnf-name": equals to **nf_instance_name**
    - "orchestration-status": in status **Active**
    - "relationship-list": with information about service id (**instanceId**) to which PNF is connected
    - "model-invariant-id": equals to **nf_model_invariant_uuid**
    - "model-version-id": equals to **nf_model_uuid**
    - "ipaddress-v4-oam": should be filled according to registration_request,json
    - "ipaddress-v6-oam": should be filled according to registration_request,json
    - "equip-type": should be filled according to registration_request,json
    - "equip-vendor": should be filled according to registration_request,json
    - "equip-model": should be filled according to registration_request,json
    - "sw-version": should be filled according to registration_request,json
    - "serial-number": should be filled according to registration_request,json
    - "nf-role": should be filled according to registration_request,json
13. Request is successful. Request should contain following values:
    - "service-instance-name": equals to **service_model_name_nf_instance_name**
    - "model-invariant-id": equals to **service_model_invariant_uuid**
    - "model-version-id": equals to **service_model_uuid**
    - "orchestration-status": equals to **Active**
    - "relationship-list": with information pnf name (**nf_instance_name**)

Actual Results	PNF registration is accepted and AAI entries: *ipaddress-v4-oam* and *ipaddress-v6-oam* are updated correctly based on *correlationID. SO service is instantieted.*	
Conclusion (Pass/Fail)		
Testing Lab		

Tester Name	Krzysztof Kuzmicki

Test Case ID	T03	
Test Case Name	**Delete pnf service and pnf resource**	
Description		
Release	Guilin	
Preconditions		
Testing Steps	Step	Expected Result
Actual Results		
Conclusion (Pass /Fail)		
Testing Lab		
Tester Name	Krzysztof Kuzmicki	

Test Case ID	T04	
Test Case Name	**Delete pnf service instance and reassign pnf resource to another service instance**	
Description		
Release	Guilin	
Preconditions		
Testing Steps	Step	Expected Result
Actual Results		
Conclusion (Pass/Fail)		
Testing Lab		
Tester Name	Krzysztof Kuzmicki	

Test Case ID	T05	
Test Case Name	**PNF registration rejected**	
Description	*Verification if PRH drops the PnfRegistration request when no AAI entry exists for the correlationID.* *AAI entries shall not be created by PRH.* *Test case covers following steps from message flow in 5G - PNF Plug and Play:*	
Release	Casablanca	
Preconditions	1. Up and running PnP PNF Simualtor according to https://wiki.onap.org/display/DW/PnP+PNF+Simulator	
Testin	Step	Expected Result

g Steps	1. Verify AAI entry created by SO service using command: curl -X GET -k -H "accept: application/json" -H "Real-Time: true" -H "Content-Type: application/json" -H "X-FromAppId: dcae-curl" -H "x-transactionId: 9998" "https://AAI: AAI@<kubernetes noed ip address>:<aai service port>/aai/v11/network/pnfs/pnf/<correlationID>" 2. Login to virtual machine with simulator 3. In **config.json** file : a. fill value for **sourceName** b. fill **pnfOamIpv4Address**, **pnfOamIpv6Address** with some value 4. Run script `./simulator.sh run simulator` in order to start sending registration request messages 5. Download PRH logs and check that registration requests has been rejected. 6. Once again verify AAI entry created by SO service using command: curl -X GET -k -H "accept: application/json" -H "Real-Time: true" -H "Content-Type: application/json" -H "X-FromAppId: dcae-curl" -H "x-transactionId: 9998" "https://AAI: AAI@<kubernetes noed ip address>:<aai service port>/aai/v11/network/pnfs/pnf/<correlationID>"	1. Command should return HTTP 400 code 2. User is logged in 3. **config.json** file is updated accordingly 4. PnP PNF simulator sends registration request 5. In PRH log should appear following message: "*org.onap.dcaegen2.services.prh.exceptions.AAINotFoundException: Incorrect response code for continuation of tasks workflow"* 6. Command should return HTTP 400 code
Actual Results	PNF registration is rejected and AAI entries *has not be created.*	
Conclusion (Pass /Fail)		
Testing Lab		
Tester Name	Krzysztof Kuzmicki	

Test Case ID	T06
Test Case Name	**PNF registration accepted when AAI entry is created using AAI API (without SO instantiation)**
Description	Verification if PNF resource registration is done properly when correct AAI record (based on *correlationID*) is present - created using AAI API Verification if AAI entries: *ipaddress-v4-oam* and *ipaddress-v6-oam* are updated correctly based on *correlationID.* *Test case covers following steps from message flow in 5G - PNF Plug and Play:*
Release	Casablanca
Preconditions	1. Up and running PnP PNF Simualtor according to https://wiki.onap.org/display/DW/PnP+PNF+Simulator

Testing Steps	Step	Expected Result
	1. Create PNF entry AAI entry using AAI API curl -i -X PUT -k -H "accept: application/json" -H "Real-Time: true" -H "Content-Type: application/json" -H "X-FromAppId: dcae-curl" -H "x-transactionId: 9998" -d '{"pnf-name":"<correlationID>","pnf-name2":"example-pnf-name2-val-78244","pnf-name2-source":"example-pnf-name2-source-val-99275","pnf-id":"example-pnf-id-val-7989","equip-type":"example-equip-type-val-20348","equip-vendor":"example-equip-vendor-val-52182","equip-model":"example-equip-model-val-8370","management-option":"example-management-option-val-72881", "ipaddress-v4-oam": "", "ipaddress-v6-oam": ""}' "http://AAI:AAI@<kubernetes noed ip address>:<aai service port>/aai/v11/network/pnfs/pnf/<correlationID>" 2. Verify AAI entry: curl -X GET -k -H "accept: application/json" -H "Real-Time: true" -H "Content-Type: application/json" -H "X-FromAppId: dcae-curl" -H "x-transactionId: 9998" "https://AAI:AAI@<kubernetes noed ip address>:<aai service port>/aai/v11/network/pnfs/pnf/<correlationID>" 3. Login to each virtual machine with simulator 4. In **config.json** file : a. fill value for **sourceName** key - use **correlationId** value used during service instantiation in prerequisite no. 3 b. fill **pnfOamIpv4Address**, **pnfOamIpv6Address** with some value 5. Run script `./simulator.sh run simulator` in order to start sending registration request messages 6. Once again verify AAI entry created by SO service 4 PNFs using command: curl -X GET -k -H "accept: application/json" -H "Real-Time: true" -H "Content-Type: application/json" -H "X-FromAppId: dcae-curl" -H "x-transactionId: 9998" "https://AAI:AAI@<kubernetes noed ip address>:<aai service port>/aai/v11/network/pnfs/pnf/<correlationID>" 7. Verify available message in Message Router topic curl -i -X GET http://<kubernetes noed ip address>:<message router service port>/events/unauthenticated.PNF_READY/2/1	1. Command should return HTTP 202 code 2. Command should return JSONs with empty value for IPv4 and IPv6 address 3. User is logged in 4. **config.json** file is updated accordingly 5. PnP PNF simulator sends registration request 6. Command should return JSONs with IPv4 and IPv6 address filled accordingly with inputs from simulator's **config.json** 7. Command should return JSONs with IPv4, IPv6 and correlationID filled accordingly with inputs from simulator's **config.json**
Actual Results	*PNF registration is accepted and AAI entries: ipaddress-v4-oam and ipaddress-v6-oam are updated correctly based on correlationID*	
Conclusion (Pass/Fail)		

Testing Lab	
Tester Name	Krzysztof Kuzmicki

PNF PnP Casablanca demo

Theoretical introduction

PnP_PNF_theory.mp4

Live demo

PnP_PNF_live_demo.mp4