Release F - O-DU Slice Assurance usecase

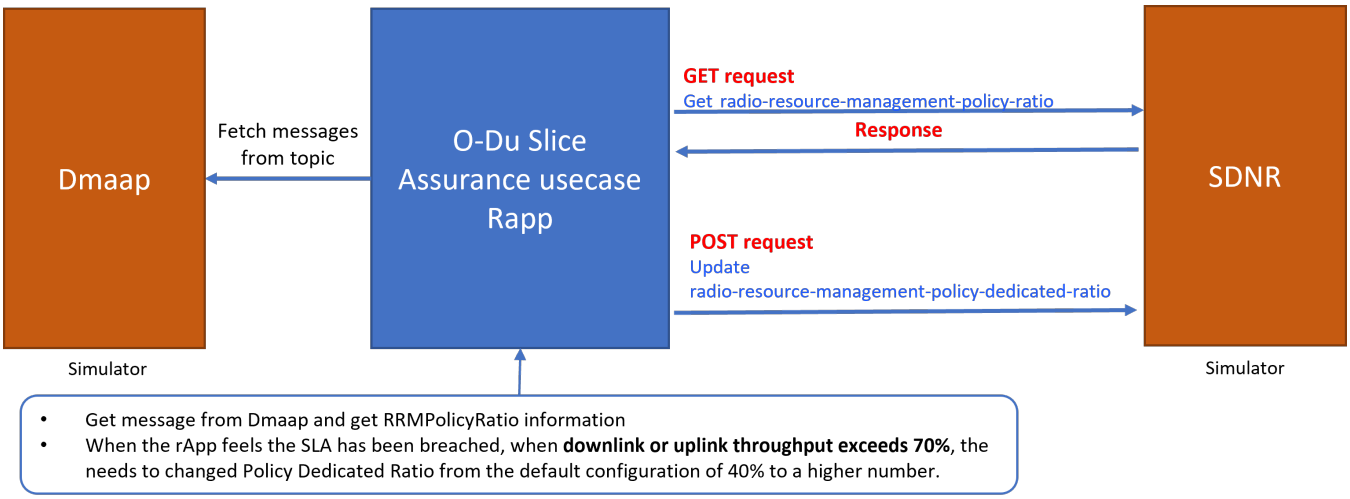
This page describes how to run current implementation for the O-DU Slice Assurance usecase.

- Standalone script version
  - Prerequisites
  - Run using simulators
    - Run Dmaap MR / SDNR stub
    - Run O-DU Slice Assurance Rapp
- ICS version
  - Use case using simulators
    - Run Dmaap MR / SDNR stub
    - Run ICS stub
    - Run O-DU Slice Assurance Rapp
  - Use case using SMO deployment
    - Run docker compose for ics version

Standalone script version

The standalone script version of the usecase is implemented in Golang.

It provides a simulator that stub both Dmaap MR and SDNR (orange boxes in the picture below), so both processes can share its data.



Prerequisites

The following need to be installed to run the script according to these instructions:

1. Go must be installed, see https://go.dev/doc/install.
2. Pull the ran slice assurance usecase repo:

```
git clone "https://gerrit.o-ran-sc.org/r/nonrtric/rapp/ransliceassurance"
```

Run using simulators

Run Dmaap MR / SDNR stub

This stub has been coded on Go, similar to the simulator used in O-RU O-DU Closed loop recovery use case previously described. However, this stub simulates both Dmaap VES messages and also SDNC. By default, the stub listens to port 3905, can be changed with the flag "--sdnr-port".

To run the stub, follow the steps below:

1. Goto "ransliceassurance/smoversion/stub" in the repo.
2. Build the stub, "go build".
3. Start the stub, "./stub"

Run simulator

```
cd ransliceassurance/smoversion/stub
go build
./stub [--sdnr-port <portNo>] [--dmaap-port <portNo>]

Example:
 $  ./sdnr --sdnr-port 3906
Starting DmaapMR stub on port:  3905
Starting SDNR stub on port:  3606
```

Run O-DU Slice Assurance Rapp

The application takes a number of environment variables for configuration, but only MR_HOST and MR_PORT are required, others are optional. More information can be found in README.md file

Run Rapp

```
cd ransliceassurance/smoversion/
go build
```

Environment variables can be defined as part of the command line as follow:

Run rapp with environment variables

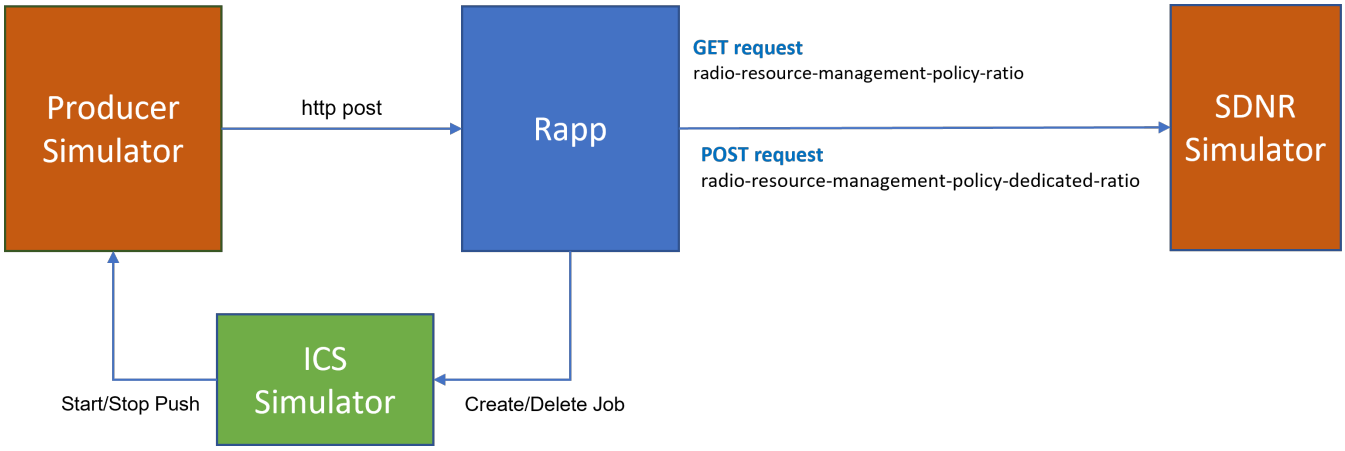
```
Example:
 $  MR_HOST=http://localhost MR_PORT=3905 ./oduclosedloop
```

ICS version

This version of the usecase is implemented as an Information Coordinator Service (ICS) job consumer, and is implemented in Go.

Use case using simulators

There are two simulator that can be used to run this usecase. First, as the standalone version, there is one stub that simulates both received VES messages from Dmaap Adapter with information about performance measurements for the slices in a DU and also simulates SDNR, that sends information about Radio Resource Management Policy Ratio and allows to modify value for RRM Policy Dedicated Ratio from default to higher value, second stub simulates ICS, where job is created with information about the rapp so the Dmaap Adapter can send information to it.



Run Dmaap MR / SDNR stub

To run the stub, follow the steps below:

1. Goto "ransliceassurance/icsversion/stub/prodSdnc" in the repo.
2. Build the stub, "go build".
3. Start the stub, "./prodSdnc"

Run ICS stub

To run the stub, follow the steps below:

1. Goto "ransliceassurance/icsversion/stub/ics" in the repo.
2. Build the stub, "go build".
3. Start the stub, "./ics"

Run O-DU Slice Assurance Rapp

Environment variables can be defined as part of the command line as follow:

Run rapp with environment variables

```
Example:
 $  CONSUMER_HOST=http://localhost CONSUMER_PORT=3905 ./icsversion
```

Use case using SMO deployment

For this case, the following components will be needed:

1. DU simulator, that will send VES message.
2. OAM controller, that will create stream to communicate with VES collector and trigger PM Job event in DU.
3. VES Collector, representing the VES (REST) provider at SMO for all kind of events
4. ONAP Dmaap, representing SMO DMaaP component, includes message-router
5. Dmaap Adapter, that takes information from DMaaP and present it as a coordinated Information Producer
6. ICS Information Coordinator Service, that coordinate/Register Information Types, Producers, Consumers, and Jobs.

For starting the OAM Controller and VES Collector, a minimal SMO deployment is needed. This can be started using the instructions from here and ignoring the ONAP-Policy steps. NonRT-RIC should be replaced for docker-compose file located in

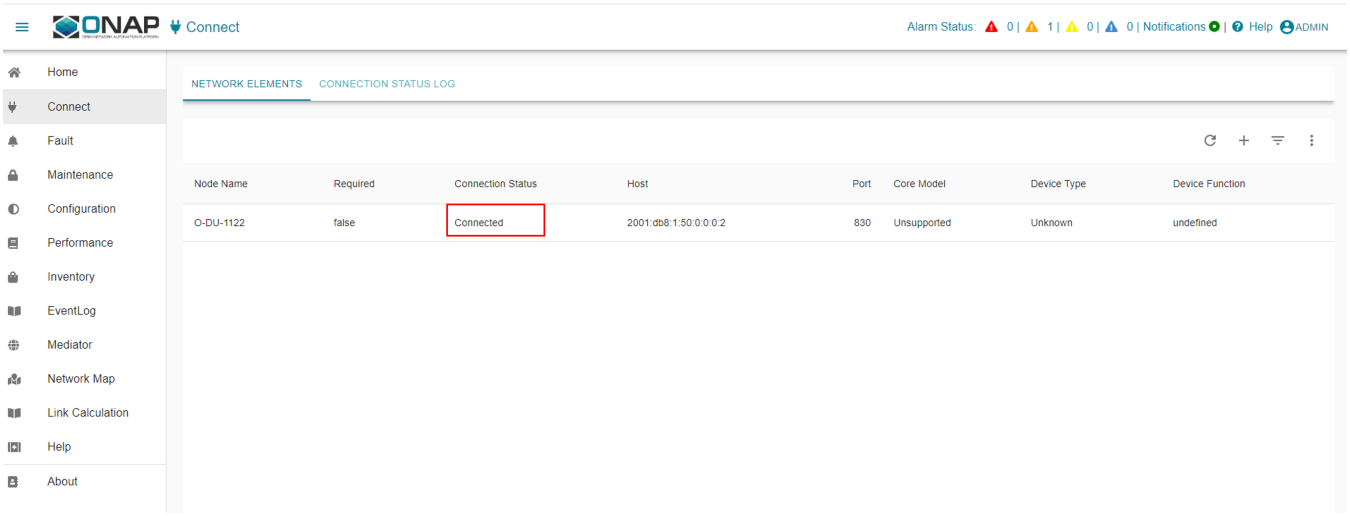
Run rapp with SMO deployment

```
ransliceassurance/docker-compose/icsversion
```

After running instructions from README file in oam repository, at least the following images should be running, note that some components has been remove as they are not needed in this scenario:

Image	Name
nexus3.o-ran-sc.org:10004/o-ran-sc/nts-ng-o-ran-du:1.4.5	ntsim-ng-o-du-1122
nexus3.onap.org:10001/onap/sdnc-image:2.2.3	sdnr
nexus3.onap.org:10001/onap/org.onap.dcaegen2.collectors.ves.vescollector:1.10.1	ves-collector
nexus3.onap.org:10001/onap/sdnc-web-image:2.2.3	sdnc-web
nexus3.onap.org:10001/onap/dmaap/dmaap-mr:1.1.18	onap-dmaap
nexus3.onap.org:10001/onap/dmaap/kafka111:1.0.4	kafka
docker.elastic.co/elasticsearch/elasticsearch-oss:7.9.3	persistence
quay.io/keycloak/keycloak:12.0.4	identity
nexus3.onap.org:10001/onap/dmaap/zookeeper:6.0.3	zookeeper

Using the sdnc-web image, it can be seen that a simulator DU node is "Connected"



After check that the DU node is connected, subscription stream can be created, the following Curl command can be used:

Configure VES details

```
curl -X PUT -H "Content-Type: application/yang-data+json" -H "Accept: application/yang-data+json" -d
'{"subscription-streams":{"id":"stream-1","administrative-state":"unlocked","user-label":"stream1","ves-
endpoint-protocol":"https","ves-endpoint-auth-method":"basic-auth","ves-endpoint-ip":"10.20.11.121","ves-
endpoint-port":"8443","ves-endpoint-username":"sample1","ves-endpoint-password":"sample1"}}' -u admin:
Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U {{baseUrl}}/rests/data/network-topology:network-topology
/topology=topology-netconf/node={{node}}/yang-ext:mount/o-ran-sc-du-hello-world:network-function/subscription-
streams=stream-1
```

Once the subscription stream has been created, a performance-measurement-job should be created as well. In order to do this, the following curl command can be used:

PM Job creation

```
cat payload.json
{"performance-measurement-jobs":{"id":"pm-1","administrative-state":"unlocked","user-label":"pm","job-tag":"my-
job-tag","performance-metrics":["/o-ran-sc-du-hello-world:network-function/o-ran-sc-du-hello-world:distributed-
unit-functions[o-ran-sc-du-hello-world:id='O-DU-1211']/o-ran-sc-du-hello-world:cell[o-ran-sc-du-hello-world:
id='cell-1']/o-ran-sc-du-hello-world:supported-measurements[o-ran-sc-du-hello-world:performance-measurement-
type='user-equipment-average-throughput-uplink']/o-ran-sc-du-hello-world:supported-snssai-subcounter-instances
[o-ran-sc-du-hello-world:slice-differentiator='1'][o-ran-sc-du-hello-world:slice-service-type='1']"],"
granularity-period":30,"stream-target":"stream-1"}}

curl -X PUT -H "Content-Type: application/yang-data+json" -H "Accept: application/yang-data+json" -d @payload.
json -u admin:Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U {{baseUrl}}/rests/data/network-topology:network-
topology/topology=topology-netconf/node={{node}}/yang-ext:mount/o-ran-sc-du-hello-world:network-function
/performance-measurement-jobs=pm-1
```

In this case, a PM job has been created to send periodic information about user-equipment-average-throughput-uplink every 30 sec (according to granularity-period property)

Postman file with above commands can be found here.

After the PM job has been created, O-DU Simulator will start generating JSON VES PM stndDefined message and sending those messages to VES collector. This is an example of the message:

Example VES message


```
{
  "event": {
    "commonEventHeader": {
      "domain": "stndDefined",
      "eventId": "pm-1_1649363550",
      "eventName": "stndDefined_performanceMeasurementStreaming",
      "eventType": "performanceMeasurementStreaming",
      "sequence": 5,
      "priority": "Low",
      "reportingEntityId": "",
      "reportingEntityName": "O-DU-1122",
      "sourceId": "",
      "sourceName": "O-DU-1122",
      "startEpochMicrosec": 1649363550000000,

      "lastEpochMicrosec": 1649363580000000,
      "nfNamingCode": "SIM-O-DU",
      "nfVendorName": "O-RAN-SC SIM Project",
      "stndDefinedNamespace": "o-ran-sc-du-hello-world-pm-streaming-oas3",
      "timeZoneOffset": "+00:00",
      "version": "4.1",
      "vesEventListenerVersion": "7.2.1"
    },
    "stndDefinedFields": {
      "stndDefinedFieldsVersion": "1.0",
      "schemaReference": "https://gerrit.o-ran-sc.org/r/gitweb?p=scp/oam/modeling.git;a=blob_plain;f=data-model
/oas3/experimental/o-ran-sc-du-hello-world-pm-streaming-oas3.yaml#/components/schemas/performance-measurement-
job",
      "data": {
        "id": "pm-1_1649363550",
        "start-time": "2022-04-07T20:32:30.0Z",
        "administrative-state": "unlocked",
        "operational-state": "enabled",
        "user-label": "pm",
        "job-tag": "my-job-tag",
        "granularity-period": 30,
        "measurements": [{"measurement-type-instance-reference": "/o-ran-sc-du-hello-world:network-function
/distributed-unit-functions[id='O-DU-1122']/cell[id='cell-1']/supported-measurements[performance-measurement-
type='(urn:o-ran-sc:yang:o-ran-sc-du-hello-world?revision=2021-11-23)user-equipment-average-throughput-uplink']
/supported-snssai-subcounter-instances[slice-differentiator='1'][slice-service-type='1']","value": 55877,"
unit": "kbit/s"}
]
      }
    }
  }
}
```


Logs for the O-DU simulator can be checked using the following command:

DU logs

```
docker exec -it ntsim-ng-o-du-1122 tail -f /opt/dev/ntsim-ng/log/log.txt
```

Run docker compose for ics version

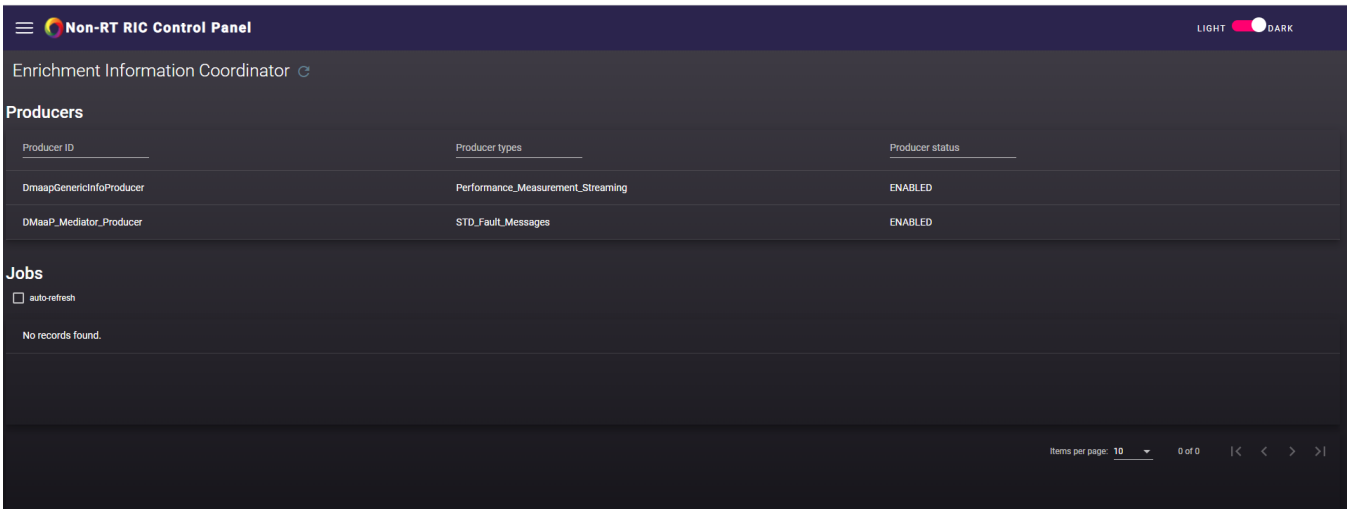
As previously mention, a docker-compose file is available to run NonRt-Ric components and ODU Slice assurance Rapp. Docker compose file can be found in ransliceassurance/docker-compose/icsversion. To run it use the following command:

```
docker-compose up -d
```

After running the command, following docker images should be running:

Image	Name
nexus3.o-ran-sc.org:10004/o-ran-sc/nonrtric-gateway:1.1.0	nonrtric-gateway
nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-dmaap-mediator-producer:1.0.1	dmaap-mediator-service
nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-dmaap-adaptor:1.0.1	dmaap-adaptor-service
nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-controlpanel:2.3.0	nonrtric-control-panel
nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-o-ru-closed-loop-recovery:1.0.1	oru-app
nexus3.o-ran-sc.org:10004/o-ran-sc/nonrtric-information-coordinator-service:1.2.1	ics
nexus3.o-ran-sc.org:10004/o-ran-sc/nonrtric-rapp-ransliceassurance-icsversion:1.0.0	odu-app

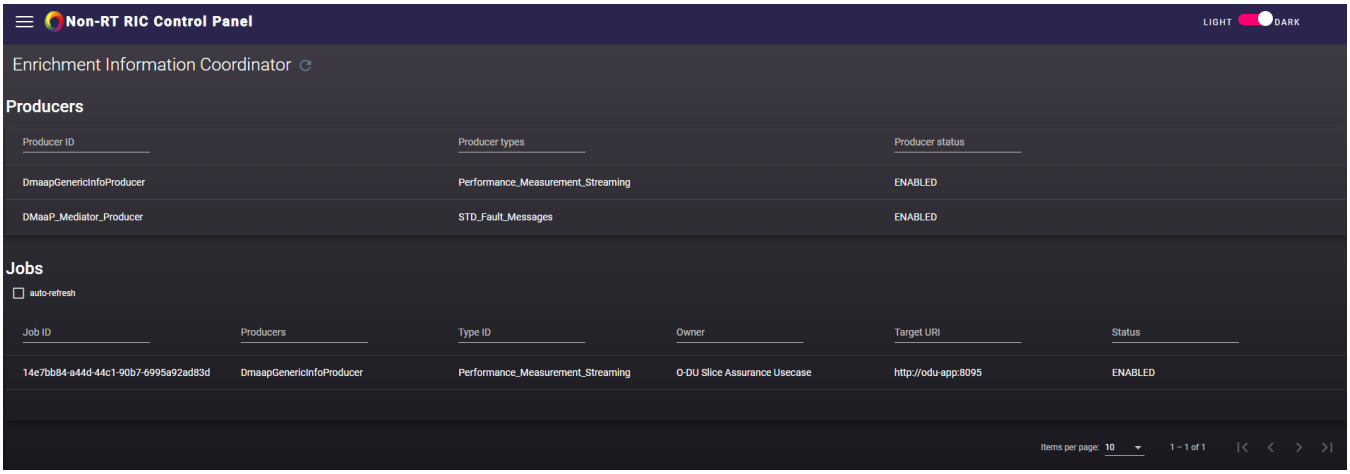
Using the nonrtric-control-panel, information about producers and jobs can be listed:



In order to start Slice assurance rapp, post request has to be send to the rapp:

```
http://localhost:8095/admin/start
```

Now a new job can be seen in the front end with information about Slice assurance rapp:



Logs from the rapp can be seen using:

```
docker logs -f odu-app
```