Release D - Run

This page describes how to get the release D version of Non-RT RIC up and running locally with three separate Near-RT RIC A1 simulator docker containers providing STD_1.1.3, STD_2.0.0 and OSC_2.1.0 versions of the A1 interface.

All components of the Non-RT RIC run inside docker containers and communicate via a docker network with container port available on localhost. Details of the architecture can be found from Release D page.

Project Requirements

- Docker and docker-compose (latest)

Images

The images used for running the Non-RT RIC can be selected from the table below depending on if the images are built manually (snapshot image) or if staging or release images shall be used.

In general, there is no need to build the images manually so either staging or release images should be used. Instruction on how to build, see Release D - Build.

The run commands throughout this page uses the **staging** images and tags. Replace the staging images/tags in the container run commands in the instructions if release or snapshot images are desired.

Component	Release image and version tag	Staging images and version tag	Manual snapshot (only available if manually built) and version tag
Policy Management Service	nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-policy-agent:2.2.1	nexus3.o-ran-sc.org:10004/o-ran-sc/nonrtric-policy-agent:2.2.1	o-ran-sc/nonrtric-policy-agent:2.2.1-SNAPSHOT
Near-RT RIC A1 Simulator	nexus3.o-ran-sc.org:10002/o-ran-sc/a1-simulator:2.1.0	nexus3.o-ran-sc.org:10004/o-ran-sc/a1-simulator:2.1.0	o-ran-sc/a1-simulator:latest
Enrichment Coordinator Service	nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-enrichment-coordinator-service:1.1.0	nexus3.o-ran-sc.org:10004/o-ran-sc/nonrtric-enrichment-coordinator-service:1.1.0	o-ran-sc/nonrtric-enrichment-coordinator-service:1.1.0-SNAPSHOT
Non-RT RIC Control Panel	nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-controlpanel:2.2.0	nexus3.o-ran-sc.org:10004/o-ran-sc/nonrtric-controlpanel:2.2.0	o-ran-sc/nonrtric-controlpanel:2.2.0-SNAPSHOT
SDNC A1-Controller	nexus3.onap.org:10002/onap/sdnc-image:2.1.2	Use release version	Use release version
Gateway	nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-gateway:1.0.0	nexus3.o-ran-sc.org:10004/o-ran-sc/nonrtric-gateway:1.0.0	o-ran-sc/nonrtric-gateway:1.0.0-SNAPSHOT
App Catalogue Service	nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-r-app-catalogue:1.0.1	nexus3.o-ran-sc.org:10004/o-ran-sc/nonrtric-r-app-catalogue:1.0.1	o-ran-sc/nonrtric-r-app-catalogue:1.0.1-SNAPSHOT

Ports

The following ports will be allocated and exposed to localhost for each component. If other port(s) are desired, then the ports need to be replaced in the container run commands in the instructions further below.

Component	Port exposed to localhost (http/https)

Policy Management Service	8081/8443
Near-RT RIC A1 Simulator	8085/8185, 8086/8186, 8087/8187
Enrichment Coordinator Service	8083/8434
Non-RT RIC Control Panel	8080/8880
SDNC A1-Controller	8282/8443
Gateway	9090
App Catalogue Service	8680/8633

Prerequisites

The containers need to be connected to docker network in order to communicate with each other.

Create a private docker network. If another network name is used, all references to 'nonrtric-docker-net' in the container run commands below need to be replaced.

```
docker network create nonrtric-docker-net
```

Run the Policy Management Service Docker Container

To support local test with three separate Near-RT RIC A1 simulator instances, each running a one of the three available A1 Policy interface versions:

- Create an application_configuration.json file with the configuration below. This will configure the policy management service to use the simulators for the A1 interface
- Note: Any defined ric names must match the given docker container names in near-RT RIC simulator startup, see Run the Near-RT RIC A1 Simulator Docker Containers
- The application supports both REST and DMAAP interface. REST is always enabled but to enable DMAAP (message exchange via message-router) additional config is needed. The examples below uses REST over http.

The policy management service can be configure with or without a A-Controller. Choose the appropriate configuration below.

This file is for running without the SDNC A1-Controller

application_configuration.json

```
{
   "config": {
      "//description": "Application configuration",
      "ric": [
         {
            "name": "ric1",
            "baseUrl": "http://ric1:8085/",
            "managedElementIds": [
               "kista_1",
               "kista_2"
            ]
         },
         {
            "name": "ric2",
            "baseUrl": "http://ric2:8085/",
            "managedElementIds": [
               "kista_3",
               "kista_4"
            ]
         },
         {
            "name": "ric3",
            "baseUrl": "http://ric3:8085/",
            "managedElementIds": [
               "kista_5",
               "kista_6"
            ]
         }
      ]
   }
}
```

This file is for running with the SDNC A1-Controller.

application_configuration.json

```
{
   "config": {
      "//description": "Application configuration",
       "controller": [
             {
             "name": "a1controller",
             "baseUrl": "https://a1controller:8443",
             "userName": "admin",
             "password": "Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U"
         }
      ],
      "ric": [
         {
            "name": "ric1",
            "baseUrl": "http://ric1:8085/",
            "controller": "a1controller",
            "managedElementIds": [
               "kista_1",
               "kista_2"
            ]
         },
         {
            "name": "ric2",
            "baseUrl": "http://ric2:8085/",
            "controller": "a1controller",
            "managedElementIds": [
               "kista_3",
               "kista_4"
            ]
         },
         {
            "name": "ric3",
            "baseUrl": "http://ric3:8085/",
            "controller": "a1controller",
            "managedElementIds": [
               "kista_5",
               "kista_6"
            ]
         }
      ]
   }
}
```

To enable the also the optional DMAAP interface, add the following config (same level as the "ric" entry) to application_configuration.json.

Be sure to update http/host/port below to match the configuration of the used message router.

optional dmaap config in application_configuration.json

```
  ...
  "streams_publishes": {
    "dmaap_publisher": {
      "type": "message-router",
      "dmaap_info": {
        "topic_url": "http://dmaap-mr:3904/events/A1-POLICY-AGENT-WRITE"
      }
    }
  },
  "streams_subscribes": {
    "dmaap_subscriber": {
      "type": "message-router",
      "dmaap_info": {
        "topic_url": "http://dmaap-mr:3904/events/A1-POLICY-AGENT-READ/users/policy-agent?
timeout=15000&limit=100"
      }
    }
  },
  ...
```

Start the container with the following command. Replace "<absolute-path-to-file>" with the the path to the created configuration file in the command. The configuration file is mounted to the container. There will be WARN messages appearing in the log until the simulators are started.

```
docker run --rm -v <absolute-path-to-file>/application_configuration.json:/opt/app/policy-agent/data
/application_configuration.json -p 8081:8081 -p 8433:8433 --network=nonrtric-docker-net --name=policy-agent-
container nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-policy-agent:2.2.1
```

Wait 1 minute to allow the container to start and to read the configuration. Then run the command below another terminal. The output should match the configuration in the file - all three rics (ric1, ric2 and ric3) should be included in the output. Note that each ric has the state "UNAVAILABLE" until the simulators are started.

Note: If the policy management service is started with config for the SDNC A1 Controller (the second config option), do the steps described in section Run the A1 Controller Docker Container below before proceeding.

```
curl localhost:8081/a1-policy/v2/rics
```

Expected output:

```
{"rics":[{"ric_id":"ric1","managed_element_ids":["kista_1","kista_2"],"policytype_ids":[],"state":"
UNAVAILABLE"},{"ric_id":"ric3","managed_element_ids":["kista_5","kista_6"],"policytype_ids":[],"state":"
UNAVAILABLE"},{"ric_id":"ric2","managed_element_ids":["kista_3","kista_4"],"policytype_ids":[],"state":"
UNAVAILABLE"}]}
```

For troubleshooting/verification purposes you can view/access the full swagger API from url: http://localhost:8081/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config

Run the A1 Controller Docker Container (ONAP SDNC)

This step is only applicable if the configuration for the Policy Management Service include the SDNC A1 Controller (second config option), see Run the Policy Management Service Docker Container.

Create the docker compose file

docker-compose.yaml

```
version: '3'

networks:
  default:
    external:
      name: nonrtric-docker-net

services:
  db:
    image: nexus3.o-ran-sc.org:10001/mariadb:10.5
    container_name: sdncdb
    networks:
      - default
    ports:
      - "3306"
    environment:
      - MYSQL_ROOT_PASSWORD=itsASecret
      - MYSQL_ROOT_HOST=%
      - MYSQL_USER=sdnctl
      - MYSQL_PASSWORD=gamma
      - MYSQL_DATABASE=sdnctl
    logging:
      driver:   "json-file"
      options:
        max-size: "30m"
        max-file: "5"

  a1controller:
    image: nexus3.onap.org:10002/onap/sdnc-image:2.1.2
    depends_on :
      - db
    container_name: a1controller
    networks:
      - default
    entrypoint: ["/opt/onap/sdnc/bin/startODL.sh"]
    ports:
      - 8282:8181
      - 8443:8443
    links:
      - db:dbhost
      - db:sdnctldb01
      - db:sdnctldb02
    environment:
      - MYSQL_ROOT_PASSWORD=itsASecret
      - MYSQL_USER=sdnctl
      - MYSQL_PASSWORD=gamma
      - MYSQL_DATABASE=sdnctl
      - SDNC_CONFIG_DIR=/opt/onap/sdnc/data/properties
      - SDNC_BIN=/opt/onap/sdnc/bin
      - ODL_CERT_DIR=/tmp
      - ODL_ADMIN_USERNAME=admin
      - ODL_ADMIN_PASSWORD=Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U
      - ODL_USER=admin
      - ODL_PASSWORD=Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U
      - SDNC_DB_INIT=true
      - A1_TRUSTSTORE_PASSWORD=a1adapter
      - AAI_TRUSTSTORE_PASSWORD=changeit
    logging:
      driver:   "json-file"
      options:
        max-size: "30m"
        max-file: "5"
```

Start the SNDC A1 controller with the following command, using the created docker-compose file.

```
docker-compose up
```

Open this url in a web browser to verify that the SDNC A1 Controller is up and running. It may take a few minutes until the url is available.

http://localhost:8282/apidoc/explorer/index.html#/controller%20A1-ADAPTER-API
Username/password: admin/Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U

The Karaf logs of A1 controller can be followed e.g. by using command

```
docker exec a1controller sh -c "tail -f /opt/opendaylight/data/log/karaf.log"
```

Run the Near-RT RIC A1 Simulator Docker Containers

Start a simulator for each ric defined in in the **application_configuration.json** created in Run the Policy Management Service Docker Container. Each simulator will use one of the currently available A1 interface versions.

Ric1

```
docker run --rm -p 8085:8085 -p 8185:8185 -e A1_VERSION=OSC_2.1.0 -e ALLOW_HTTP=true --network=nonrtric-docker-
net --name=ric1 nexus3.o-ran-sc.org:10002/o-ran-sc/a1-simulator:2.1.0
```

Ric2

```
docker run --rm -p 8086:8085 -p 8186:8185 -e A1_VERSION=STD_1.1.3 -e ALLOW_HTTP=true --network=nonrtric-docker-
net --name=ric2 nexus3.o-ran-sc.org:10002/o-ran-sc/a1-simulator:2.1.0
```

Ric3

```
docker run --rm -p 8087:8085 -p 8187:8185 -e A1_VERSION=STD_2.0.0 -e ALLOW_HTTP=true --network=nonrtric-docker-
net --name=ric3 nexus3.o-ran-sc.org:10002/o-ran-sc/a1-simulator:2.1.0
```

Wait at least one minute to let the policy management service synchronise the rics. Then run the command below another terminal. The output should match the configuration in the file. Note that each ric now has the state "AVAILABLE".

```
curl localhost:8081/a1-policy/v2/rics
```

Expected output - all state should indicated AVAILABLE:

```
{"rics":[{"ric_id":"ric1","managed_element_ids":["kista_1","kista_2"],"policytype_ids":[],"state":"AVAILABLE"},
{"ric_id":"ric3","managed_element_ids":["kista_5","kista_6"],"policytype_ids":[],"state":"AVAILABLE"},
{"ric_id":"ric2","managed_element_ids":["kista_3","kista_4"],"policytype_ids":[""],"state":"AVAILABLE"}]}
```

The simulators using version STD_2.0.0 and OSC_2.1.0 supports policy types. Run the commands below to add one policy types in ric1 and ric3.

Create the file with policy type for ric1

osc_pt1.json

```
{
  "name": "pt1",
  "description": "pt1 policy type",
  "policy_type_id": 1,
  "create_schema": {
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "OSC_PT1_0.1.0",
    "description": "QoS policy type",
    "type": "object",
    "properties": {
      "scope": {
        "type": "object",
        "properties": {
          "ueId": {
            "type": "string"
          },
          "qosId": {
            "type": "string"
          }
        },
        "additionalProperties": false,
        "required": [
          "ueId",
          "qosId"
        ]
      },
      "statement": {
        "type": "object",
        "properties": {
          "priorityLevel": {
            "type": "number"
          }
        },
        "additionalProperties": false,
        "required": [
          "priorityLevel"
        ]
      }
    }
  }
}
```

Put the policy type to ric1

```
curl -X PUT -v "http://localhost:8085/a1-p/policytypes/123" -H "accept: application/json" \
 -H "Content-Type: application/json" --data-binary @osc_pt1.json
```

Create the file with policy type for ric3

std_pt1.json

```
{
    "policySchema": {
      "$schema": "http://json-schema.org/draft-07/schema#",
      "title": "STD_QOS_0_2_0",
      "description": "STD QOS policy type",
      "type": "object",
      "properties": {
        "scope": {
          "type": "object",
          "properties": {
            "ueId": {
              "type": "string"
            },
            "qosId": {
              "type": "string"
            }
          },
          "additionalProperties": false,
          "required": [
            "ueId",
            "qosId"
          ]
        },
        "qosObjectives": {
          "type": "object",
          "properties": {
            "priorityLevel": {
              "type": "number"
            }
          },
          "additionalProperties": false,
          "required": [
            "priorityLevel"
          ]
        }
      }
    },
    "statusSchema": {
      "$schema": "http://json-schema.org/draft-07/schema#",
      "title": "STD_QOS_0.2.0",
      "description": "STD QOS policy type status",
      "type": "object",
      "properties": {
        "enforceStatus": {
          "type": "string"
        },
        "enforceReason": {
          "type": "string"
        },
        "additionalProperties": false,
        "required": [
          "enforceStatus"
        ]
      }
    }
  }
```

Put the policy type to ric3

```
curl -X PUT -v "http://localhost:8087/policytype?id=std_pt1" -H "accept: application/json"  -H "Content-Type:
application/json" --data-binary @std_pt1.json
```

Wait one minute to let the policy management service synchronise the types with the simulators.

List the synchronised types.

```
curl localhost:8081/a1-policy/v2/policy-types
```

Expected output:

```
{"policytype_ids":["","123","std_pt1"]}
```

Run the Enrichment Service Docker Container

Run the following command to start the enrichment coordinator service.

```
docker run --rm -p 8083:8083 -p 8434:8434 --network=nonrtric-docker-net --name=enrichment-service-container
nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-enrichment-coordinator-service:1.1.0
```

Verify the Enrichment Coordinator Service is started and responding (response is an empty array).

```
curl localhost:8083/data-producer/v1/info-types
```

Expected output:

```
[ ]
```

For troubleshooting/verification purposes you can view/access the full swagger API from url: http://localhost:8083/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config

Run the Non-RT RIC Gateway and Control Panel Docker Container

The Gateway exposes the interfaces of the Policy Management Service and the Enrichment Coordinator Service to a single port of the gateway. This single port is then used by the control panel to access both services.

Create the config file for the gateway.

application.yaml

```
server:
  port: 9090
spring:
  cloud:
    gateway:
      httpclient:
        ssl:
          useInsecureTrustManager: true
        wiretap: true
      httpserver:
        wiretap: true
      routes:
      - id: A1-Policy
        uri: https://policy-agent-container:8433
        predicates:
        - Path=/a1-policy/**
      - id: A1-EI
        uri: https://enrichment-service-container:8434
        predicates:
        - Path=/data-producer/**
management:
  endpoint:
    gateway:
      enabled: true
  endpoints:
    web:
      exposure:
        include: "gateway,loggers,logfile,health,info,metrics,threaddump,heapdump"
logging:
  level:
    ROOT: ERROR
    org.springframework: ERROR
    org.springframework.cloud.gateway: INFO
    reactor.netty: INFO
  file:
    name: /var/log/nonrtric-gateway/application.log
```

Run the following command to start the gateway. Replace "<absolute-path-to-file>" with the the path to the created application.yaml.

```
docker run --rm -v <absolute-path-to-config-file>/application.yaml:/opt/app/nonrtric-gateway/config/application.
yaml -p 9090:9090 --network=nonrtric-docker-net --name=nonrtric-gateway  nexus3.o-ran-sc.org:10002/o-ran-sc
/nonrtric-gateway:1.0.0
```

Run the following two commands to check that the services can be reached through the gateway

```
curl localhost:9090/a1-policy/v2/rics
```

Expected output

```
{"rics":[{"ric_id":"ric1","managed_element_ids":["kista_1","kista_2"],"policytype_ids":["123"],"state":"
AVAILABLE"},{"ric_id":"ric3","managed_element_ids":["kista_5","kista_6"],"policytype_ids":["std_pt1"],"state":"
AVAILABLE"},{"ric_id":"ric2","managed_element_ids":["kista_3","kista_4"],"policytype_ids":[""],"state":"
AVAILABLE"}]}
```

Second command:

```
curl localhost:9090/data-producer/v1/info-types
```

Expected output:

```
[ ]
```

Create the config file for the control panel.

nginx.conf

```
events{}

http {
    include /etc/nginx/mime.types;
    resolver 127.0.0.11;
    server {
        listen 8080;
        server_name localhost;
        root /usr/share/nginx/html;
        index index.html;
        location /a1-policy/ {
            set $upstream nonrtric-gateway;
            proxy_pass http://$upstream:9090;
        }
        location /data-producer/ {
            set $upstream nonrtric-gateway;
            proxy_pass http://$upstream:9090;
        }
        location / {
            try_files $uri $uri/ /index.html;
        }
    }
}
```

Run the following command to start the control panel. Replace "<absolute-path-to-file>" with the the path to the created nginx.conf.

```
docker run --rm -v <absolute-path-to-config-file>/nginx.conf:/etc/nginx/nginx.conf -p 8080:8080 --
network=nonrtric-docker-net --name=control-panel  nexus3.o-ran-sc.org:10002/o-ran-sc/nonrtric-controlpanel:2.2.0
```

The webbased UI can be accessed by pointing the web-browser to this URL:

http://localhost:8080/

Run the App Catalogue Service

Start the App Catalogue Service by the following command.

Ric1

```
docker run --rm -p 8680:8680 -p 8633:8633 --network=nonrtric-docker-net --name=rapp-catalogue-service nexus3.o-
ran-sc.org:10002/o-ran-sc/nonrtric-r-app-catalogue:1.0.1
```

Verify that the service is up and running

```
curl localhost:8680/services
```

Expected output:

```
[ ]
```

Run the Helm Manager

See this sub-page: Build/Run Helm Manager