D-Release Integration - Test Environment

Table of Contents

Sub pages

Scope

This wiki describes the test environment for OAM specific test cases focusing on D-Release Closed Loop O-RU recovery use case.

OAM-193 - Getting issue details... STATUS

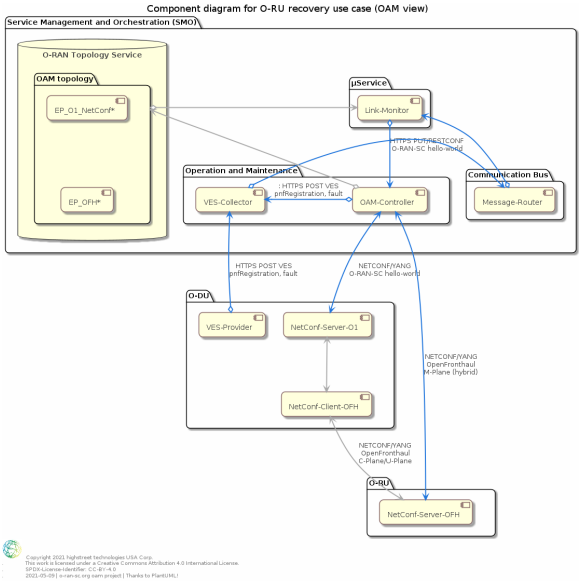
Components

The following Component diagram shows the most relevant components and their associations from O-RAN-SC OAM project point of view.

The different software components can be deployed with Docker-Compose or Kubernetes technologies.

As O-RAN Alliance targets IPv6 please ensure the Docker IPv6 functionality is enabled (see O-RAN-SC Documentation Docker: Enable IPv6) or use Kubernetes version 1.19 or higher.

Diagram



Please see Closed loop use case

Descriptions

O-RU - NETCONF-Server-OFH

For D-Release the O-RU operates in "hybrid" mode. This means that the O-RU exposes O-RAN OpenFronthaul NETCONF/YANG interface directly to the OAM functions of the SMO.

For OAM test environment it is recommended to use the O-RU FH simulation from O-RAN-SC Sim project. The docker imager can be loaded directly from O-RAN-SC Nexus (nexus3.o-ran-sc.org:10004/o-ran-sc/nts-ng-o-ran-ru-fh:1.3.0).

The simulation must be configured in a way that is support NETCONF call-home to the SMO-OAM-Controller and supports NETCONF <create-subscriptions/> RPC for stream "NETCONF".

The simulation must not expose any VES messages.

Depending on the OAM controller capabilities TLS certs of the O-RU must be configured at the OAM-Controller.

The following protocols from O-RU to OAM controller are under test:

- NETCONF-Callhome/SSH/IPv4
- NETCONF-Callhome/TLS/IPv4
- NETCONF-Callhome/SSH/IPv6

- NETCONF-Callhome/TLS/IPv6
- NETCONF/SSH/IPV4 for all NETCONF operations (but mainly GET, GET-CONFIG, EDIT-CONFIG)
- NETCONF/TLS/IPv4 for all NETCONF operations (but mainly GET, GET-CONFIG, EDIT-CONFIG)
- NETCONF/SSH/IPV6 for all NETCONF operations (but mainly GET, GET-CONFIG, EDIT-CONFIG)
- NETCONF/TLS/IPv6 for all NETCONF operations (but mainly GET, GET-CONFIG, EDIT-CONFIG)

O-DU - NETCONF-Server-O1

For D-Release the O-DU operates in "hybrid" mode. This means that the O-DU does not expose any management plane functions of connected O-RUs. To reduce the development effort on O-DU it was decided in RSAC meetings that the O-DU at minimum exposes the o-ran-sc-hello-world.yang (see

OAM-169 - Getting issue details... STATUS

and

ODUHIGH-322 - Getting issue details... STATUS

).

For OAM test environment it is recommended to use the O-DU O1 simulation from O-RAN-SC Sim project. The docker imager can be loaded directly from O-RAN-SC Nexus (nexus3.o-ran-sc.org:10004/o-ran-sc/nts-ng-o-ran-du:1.3.0).

The simulation must be configured in a way that is support NETCONF for configuration management only and VES for heartbeat, fault and pnfRegistration.

Depending on the OAM controller capabilities TLS certs of the O-DU must be configured at the OAM-Controller.

The following protocols from O-RU to OAM controller are under test:

- VES:pnfRegistration https/IPv4
- VES:heartbeat https/IPv4
- VES:fault https/IPv4
- VES:pnfRegistration https/IPv6
- VES:heartbeat https/IPv6
- VES:fault https/IPv6
- NETCONF/SSH/IPV4 for all NETCONF operations (but mainly GET, GET-CONFIG, EDIT-CONFIG)
- NETCONF/TLS/IPv4 for all NETCONF operations (but mainly GET, GET-CONFIG, EDIT-CONFIG)
- NETCONF/SSH/IPV6 for all NETCONF operations (but mainly GET, GET-CONFIG, EDIT-CONFIG)
- NETCONF/TLS/IPv6 for all NETCONF operations (but mainly GET, GET-CONFIG, EDIT-CONFIG)

OAM Controller

For D-Release the OAM controller provides a NETCONF client for SSH and/or TLS, a RESCONF server, and providers and listeners to the message router.

For OAM test environment it is recommended to use the ONAP CCSDK/SDNC/SDN-R in combination with KeyCloak as centralized user management. The docker imager can be loaded directly from ONAP Nexus (nexus3.onap.org:10001/onap/sdnc-image:2.1.5) and from KeyCloak (quay.io/keycloak/keycloak: 12.0.4)

The new functionalities for D-Release are:

- to convert NETCONF call-home into VES:pnfRegistration (JIRA | CCSDK-3160)
- to convert NETCONF notification by O-RU/alarm-notify into VES:fault (JIRA | CCSDK-3158)
- Gui-Cut-Through for O-RUs

The following test cases should be performed using a centralized user management based on KeyCloak.

- O-RU/NETCONF-Callhome/SSH/IPv4 VES:pnfRegistration to message router
- O-RU/NETCONF-Callhome/TLS/IPv4 VES:pnfRegistration to message router
- O-RU/NETCONF-Callhome/SSH/IPv6 VES:pnfRegistration to message router
- O-RU/NETCONF-Callhome/TLS/IPv6 VES:pnfRegistration to message router
- O-RU/NETCONF-Notification/SSH/IPv4/alarm-notifiy VES:fault to message router
- O-RU/NETCONF-Notification/TLS/IPv4/alarm-notifiy VES:fault to message router
- O-RU/NETCONF-Notification/SSH/IPv6/alarm-notifiy VES:fault to message router
- O-RU/NETCONF-Notification/TSL/IPv6/alarm-notifiy VES:fault to message router
- VES:pnfRegistration from message router O-DU/NETCONF/SSH/IPv4/<hello/>
- VES:pnfRegistration from message router O-DU/NETCONF/TLS/IPv4/<hello/>
- VES:pnfRegistration from message router O-DU/NETCONF/SSH/IPv6/<hello/>
- VES:pnfRegistration from message router O-DU/NETCONF/TSL/IPv6/<hello/>

- RESTCONF/HTTPS-GET/RESTCONF/IPv4 O-DU/NETCONF/SSH/IPv4/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-GET/RESTCONF/IPv4 O-DU/NETCONF/TLS/IPv4/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-GET/RESTCONF/IPv4 O-DU/NETCONF/SSH/IPv6/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-GET/RESTCONF/IPv4 O-DU/NETCONF/TLS/IPv6/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-GET/RESTCONF/IPv6 O-DU/NETCONF/SSH/IPv4/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-GET/RESTCONF/IPv6 O-DU/NETCONF/TLS/IPv4/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-GET/RESTCONF/IPv6 O-DU/NETCONF/SSH/IPv6/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-GET/RESTCONF/IPv6 O-DU/NETCONF/TLS/IPv6/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-PUT/RESTCONF/IPv4 O-DU/NETCONF/SSH/IPv4/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-PUT/RESTCONF/IPv4 O-DU/NETCONF/TLS/IPv4/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-PUT/RESTCONF/IPv4 O-DU/NETCONF/SSH/IPv6/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-PUT/RESTCONF/IPv4 O-DU/NETCONF/TLS/IPv6/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-PUT/RESTCONF/IPv6 O-DU/NETCONF/SSH/IPv4/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-PUT/RESTCONF/IPv6 O-DU/NETCONF/TLS/IPv4/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-PUT/RESTCONF/IPv6 O-DU/NETCONF/SSH/IPv6/<get-config/>/hello-world.yang
- RESTCONF/HTTPS-PUT/RESTCONF/IPv6 O-DU/NETCONF/TLS/IPv6/<get-config/>/hello-world.yang
- Gui-Cut-Through to O-RUs

VES Collector

There are no new functions required for the VES Collector. The regression tests are covered by test cases mentioned above.

For OAM test environment it is recommended to use the ONAP DCAE VES-Collector. The docker imager can be loaded directly from ONAP Nexus (nexus3.onap.org:10001/onap/org.onap.dcaegen2.collectors.ves.vescollector:1.8.0).

Message Router

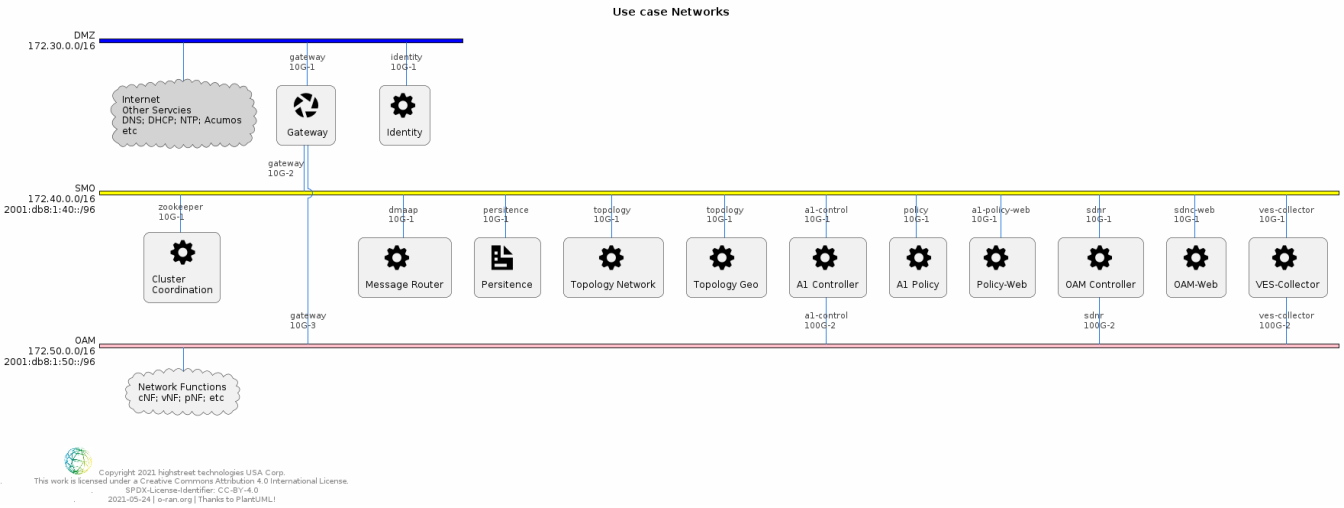
There are no new functions required for the Message Router. The regression tests are covered by test cases mentioned above.

For OAM test environment it is recommended to use the ONAP DMaaP. The docker imager can be loaded directly from ONAP Nexus (nexus3.onap.org:10001/onap/dmaap/dmaap-mr:1.1.18).

For manual checks of activities on the Message Router a REST client is recommended (e.g. PostMan, vsCode with REST-client extension, Browser-RestClient-extensions, ...)

Test Networks

it is recommended to run the different components in different networks for security and responsibility reasons.



DMZ

DMZ (demilitarized zone) contains and exposes external services to the internet (IPv4 only).

The network-id is 30 - the short name is dmz.

- IPv4
  - Subnet: 172.30.0.0/16
  - Gateway: 172.30.0.1

SMO

The primary network to exchange data between the physical nodes and virtual machines related to the SMO components.

The network-id is 40 - the short name is smo.

- IPv4
  - Subnet: 172.40.0.0/16
  - Gateway: 172.40.0.1
- IPv6
  - Subnet: 2001:db8:1:40::/96
  - Gateway: 2001:db8:1:40::1

OAM

The primary network for interactions between SMO and the O-RAN network (O1, OpenFronthaul-Managment-Plane).

The network-id is 50 - the short name is net.

- IPv4
  - Subnet: 172.50.0.0/16
  - Gateway: 172.50.0.1
- IPv6
  - Subnet: 2001:db8:1:50::/96
  - Gateway: 2001:db8:1:50::1

Deployment

For one-click deployment the teas chosen "docker-compose" as it is more flexible and can be easily converted to kubernetes/helm.

Prerequisites

```
$ cat /etc/os-release | grep PRETTY_NAME
PRETTY_NAME="Ubuntu 20.04.2 LTS"

$ docker --version
Docker version 20.10.2, build 20.10.2-0ubuntu1~20.04.2

$ docker-compose version
docker-compose version 1.29.1, build c34c88b2
docker-py version: 5.0.0
CPython version: 3.7.10
OpenSSL version: OpenSSL 1.1.0l 10 Sep 2019

$ git --version
git version 2.25.1

$ git clone "https://gerrit.o-ran-sc.org/r/oam"

$ cd oam/solution/integration   /TODO update once merged
```

Please modify the /etc/hosts of your system.

- <your-system>: is the hostname of the system, where the browser is started
- <deloyment-system-ipv4>: is the IP address of the system where the solution will be deployed

For development purposes <your-system> and <deployment-system> may reference the same system.

```
$ cat /etc/hosts
127.0.0.1               localhost
127.0.1.1               <your-system>
<deloyment-system-ipv4> sdnc-web <your-system>
<deloyment-system-ipv4> identity <your-system>
```

Expected Folder Structure

The folder structure below helps addressing start commands and decouples the different involved project.

The folder '**network**' includes docker compose file and its configuration from O-RAN-SC SIM O1-interface project.

The folder '**smo/non-rt-ric**' includes docker compose file and its configuration from O-RAN-SC NON-RT-RIC project.

```
network
   .env
   config.py
   docker-compose.yml

   ntsim-ng-o-du
   ntsim-ng-o-ru
smo
    common
       .env
       docker-compose.yml

       dmaap
       docker
       identity
       kafka
       zookeeper
    non-rt-ric
       .env
       docker-compose.yml

       <config-folders>
    oam
        docker-compose.yml

        sdnc-web
        sdnr
        ves-collector
```

Usage

Customizing the solution

Please check and adjust your environment variables, if necessary.

```
nano smo/common/.env
nano smo/non-rt-ric/.env
nano smo/oam/.env
nano network/.env
```

Startup solution

Please note that it is necessary to configure first the identity service,
before starting further docker images.

The several docker-compose yml files must be started in the right order as listed below:

```
docker-compose -f smo/common/docker-compose.yml up -d
python smo/common/identity/config.py
```

The python script configure the users within the identity service (keycloak).

A system user (%USER) is also created with administration rights.

```
docker-compose -f smo/non-rt-ric/docker-compose.yml up -d
docker-compose -f smo/oam/docker-compose.yml up -d
```

Please wait about 2min until all the service are up and running.

If you see the login page (curl http://localhost:8181/odlux/index.
html) you are good to go and can start the (simulated) network.

```
docker-compose -f network/docker-compose.yml up -d
python network/config.py
```

The python script configures the simulated O-DU and O-RU according to O-RAN hybrid architecture.

O-DU - NETCONF Call HOME and NETCONF notifications
O-RU - ves:pnfRegistration and ves:fault, ves:heartbeat

Log files and karaf console

OpenDaylight karaf.logs

```
docker exec -it sdnr  tail -f /opt/opendaylight/data/log/karaf.log
```

karaf console access (karaf:karaf)

```
ssh karaf@localhost -p 8101
```

ves-collector logs

```
docker logs -f ves-collector
```

Verification Solution

Login into SDN-R's UI called ODLUX

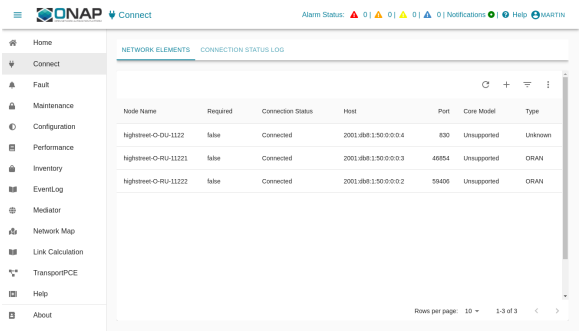
https://sdnc-web:8453

User: admin // see .env file

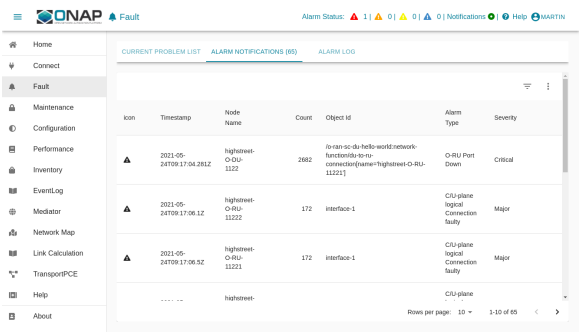
Password: Kp8bJ4SXszM0WXlhak3eHlcse2gAw84vaoGGmJvUy2U

In case of trouble, please update the commands with your customized '.env' file.

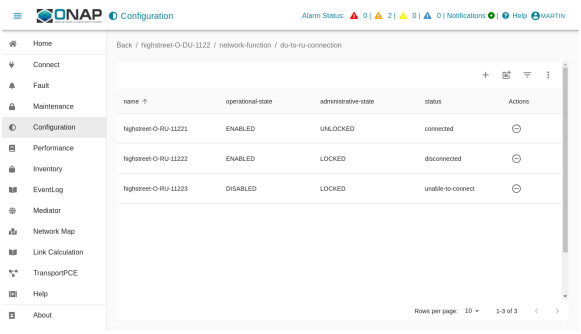
The following picture show the connection status connected to the OAM Controller.



The following figure shows NETCONF/YANG/O-RAN-Fronthaul/alarm-notify from O-RUs and VES:fault notifications. The WebUI (ODLUX) is automatically update based on websocket send northbound to a browser.



The following figure show the configuration of the (simulated) O-DU implementing the o-ran-sc-hello-world.yang interface.



Terminate solution

To stop all container please respect the following order

```
docker-compose -f network/docker-compose.yml down
docker-compose -f smo/oam/docker-compose.yml down
docker-compose -f smo/non-rt-ric/docker-compose.yml down
docker-compose -f smo/common/docker-compose.yml down
```

Cleanup

!!! be careful if other stopped containers are on the same system.

```
docker system prune -a -f
```

Troubleshooting

In most cases the .env setting do not fit to the environment and need to be adjusted.

Please make sure that the network settings to not overlap with other networks.

The commands ...

```
docker ps -a
docker network ls
```

... are your friends.