Introduction and guides

Architecture introduction

Overall architecture (not component-specific)

Installing the near-RT RIC

~~Outdated: Bronze release (Jun 21 2020) Getting Started- Near Realtime RIC Installation (but similar sequence is used in Cherry and Dawn)~~

Generally the latest instructions: https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-ric-dep/en/latest/ Installing Near Realtime RIC in RIC Cluster

There's a demo video which shows how to deploy the near-RT RIC, connect an E2 node (gNB simulator) to it and how to deploy an xApp using the DMS CLI:

- F release: see "Demo on how to deploy" in 2022-05-24 Release F
- E release: demo starting at 00:15:30 in the "link to zoom recording" video in this wiki page: 2021-12-21 Release E
- D release: 2021-11-05 Demo video release Dawn

List of release docker images:

- F release: Near-RT RIC (F release)
- E release Near-RT RIC (E release)
- D release: Near-RT RIC (D release).
- Old list of cherry docker images: Near-RT RIC

Using the E2 simulator from O-RAN-SC

see here: E2 Simulator#Buildingdockerimageandrunningsimulatorinstance

Guides

An example xApp using the Golang xApp framework: link and the xApp_Writer_s_Guide_v2.pdf

Also check the "md" files in the base directory of many of the RIC repositories by name in gerrit.

Also check the currently still somewhat empty information in readthedocs component-by-component.

Xapp frameworks in general: see item with same name "Xapp frameworks in general:" in the "external interfaces" section below.

External interfaces

The RIC platform has the following external interfaces. Note that additional per-component documentation can be found here: link

E2

(2023-06-30) Since the F release we support E2APv2.0. The older protocol versions - E2APv1.1 and E2AP1.0 - are not supported anymore (use old OSC near-RT RIC releases if needed (The E release has E2APv1.1 and is backwards compatible with E2APv1.0); or upgrade). Some features that are worthwhile to point out as supported:

- OID support in function definitions (RIC-640, implemented in E, Added in E2APv1.1)
- configuration transfer capabilities (RIC-638, implemented in F)

Not supported

- no RIC-784 (timer handling in E2 control messages),
- no RIC-785 (error indication handling),
- no RIC-783 (support for control ack to selected xapps)
- no E2 Reset from RIC to RAN (RIC-387) - the other direction from RAN to RIC (RIC-386) is supported.
- In submgr we only support Subscriptions with single actions (RIC-75).
- No support for multiple TNL associations, i.e., no E2 Connection Update procedure (RIC-639)
- RIC subscription delete required is terminated by the submgr (the interface towards xApps for this is coming only via RIC-967).
- No support for the E2 removal procedure (simple SCTP disconnect is currently the way)

Note that the RIC platform is generally independent of E2SMs (E2 service models) and E2APv1.0, 1.1, 2.0 work with the various E2SMs as there are no dependencies.

Xapp Frameworks

- Xapp frameworks in general: See xapp writers guide xApp_Writer_s_Guide_v**2**.pdf attached to this wiki page (Dawn onwards). The earlier xapp_Writer_s_guide.pdf is not to be used anymore. Generally, we are in the process of moving the app writing guide to https://docs.o-ran-sc.org.)
- Xapp framework for Go: as per link under "xApp framework for Go" in Near Realtime RAN Intelligent Controller (RIC)
- Xapp framework for python: as per link under "xApp framework for python" in Near Realtime RAN Intelligent Controller (RIC)
- Xapp framework for C++: as per link under "xApp framework for CXX" in Near Realtime RAN Intelligent Controller (RIC)

SDL API

https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-sdl/en/latest/user-guide.html

O1

- Support for netconf "Hello exchange" (incl. capability exchange)
- Alarms as VES events as per RIC Alarm System
- get E2 stats (on number of packets over E2) via the path **E2T->Prometheus->VESPA(ONAP).** Requires setting the environment variable VESMGR_PLT_CFG_FILE which contains the VESPA mapping for this path: **Prometheus->VESPA**

O2

Support for O2 use case "Deploy xApp in near-RT RIC" in WG6's O-RAN orchestration use cases v2.0. This is done as per the CLI that is in this commit https://gerrit.o-ran-sc.org/r/c/ric-plt/appmgr/+/5816

A1

(2021-05-25) Support for A1-Policy and A1-EI as per A1APv3.0 and A1APv3.1. Note that the A1 mediator in the RIC platform is independent of A1TD (A1 type definitions) and passes them as opaque blobs to xApps. Also A1 mediator only handles numeric policy types and not as in the standard a string.

- Best checked from here: https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-a1/en/latest/user-guide-api.html

Others

RMR: Check the read-the-docs page: https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-lib-rmr/en/latest/rel-notes.html Additionally you might read RIC Message Router (RMR)

Subscription manager: https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-submgr/en/latest/user-guide.html and REST API: link

Xapp onboarding and deployment: check section "RIC applications" in https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#ric-applications on how to use the dms_cli to onboard and deploy xApps. Onboarding is the act of generating a Helm chart for the xApp from the its xApp descriptor. Using the generated helm chart (which is specific to a particular RIC instance) the xApp can be deployed.

Generally check the read-the-docs pages: https://docs.o-ran-sc.org/en/latest/projects.html#near-realtime-ran-intelligent-controller-ric