OSC NONRTRIC – Demo – Cherry Release
A1 Policies & A1 Enrichment Information

Last updated 2020-12-09

OSC NONRTRIC

Non-RealTime RAN Intelligent Controller (*NONRTRIC*) project in OSC

The NONRTRIC project provides:

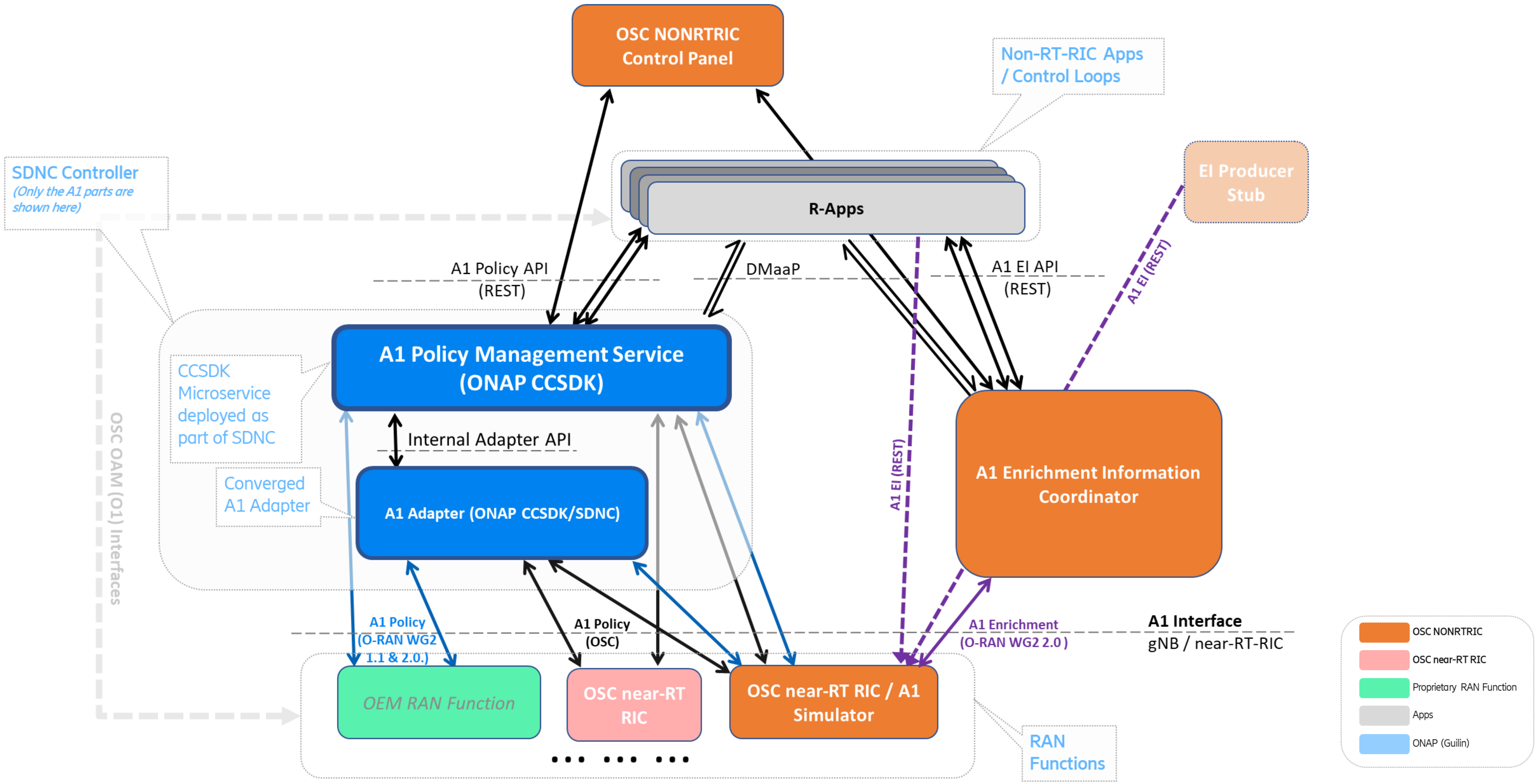
- Concepts
- Architecture
- Reference implementations

… for Non-RealTime RIC functionality as defined and described by the O-RAN Alliance architecture.

All implementations will be demonstrated in open community labs to prove functionalities and to give feedback to the O-RAN working groups

- Issue tracker / Backlog: JIRA: https://jira.o-ran-sc.org/projects/NONRTRIC/issues
- Code Repositories: Gerrit:
  - https://gerrit.o-ran-sc.org/r/admin/repos/nonrtric
  - https://gerrit.o-ran-sc.org/r/admin/repos/portal/nonrtric-controlpanel
  - https://gerrit.o-ran-sc.org/r/admin/repos/sim/a1-interface
- Documentation & Discussion:
  - Wiki: https://wiki.o-ran-sc.org/display/RICNR
  - Guides:
    - https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric/en/latest/index.html
    - https://docs.o-ran-sc.org/projects/o-ran-sc-sim-a1-interface/en/latest/index.html
    - https://docs.o-ran-sc.org/projects/o-ran-sc-portal-nonrtric-controlpanel/en/latest/index.html
- Team Status & Planning Meetings: https://wiki.o-ran-sc.org/display/RICNR/Meetings
- PTL: John Keeney (Ericsson Software Technology)

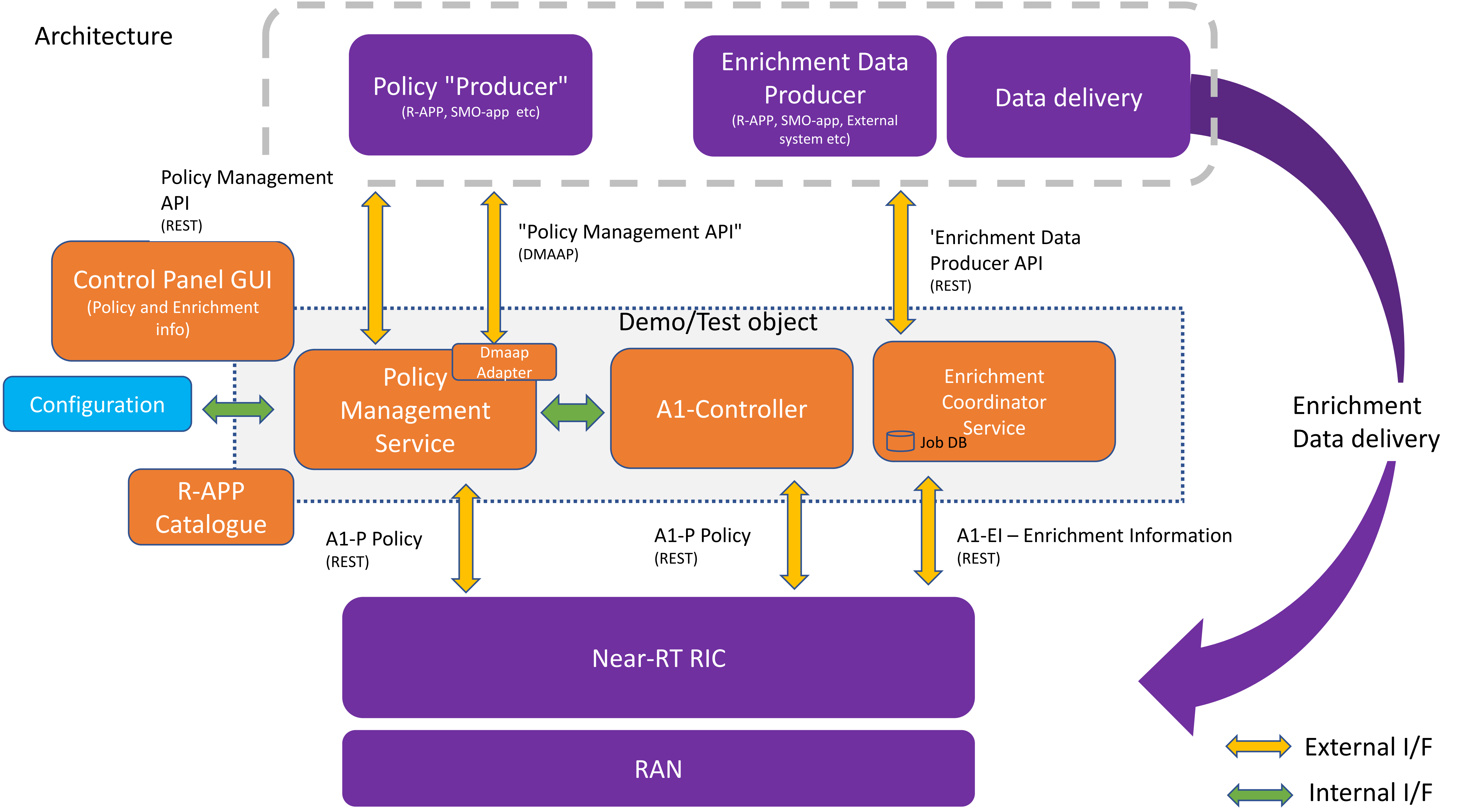
A1 Policies & Enrichment Information in ONAP Guilin & OSC Cherry:


OSC NONRTRIC Control Panel
Non-RT-RIC Apps / Control Loops
SDNC Controller
(Only the A1 parts are shown here)
R-Apps
EI Producer Stub
A1 Policy API (REST)
DMaaP
A1 EI API (REST)
A1 EI (REST)
A1 EI (REST)
CCSDK Microservice deployed as part of SDNC
A1 Policy Management Service (ONAP CCSDK)
Internal Adapter API
Converged A1 Adapter
A1 Adapter (ONAP CCSDK/SDNC)
A1 Enrichment Information Coordinator
OSC OAM (O1) Interfaces
A1 Policy (O-RAN WG2 1.1 & 2.0.)
A1 Policy (OSC)
A1 Enrichment (O-RAN WG2 2.0)
A1 Interface
gNB / near-RT-RIC
OEM RAN Function
OSC near-RT RIC
OSC near-RT RIC / A1 Simulator
• • • • • • • • •
RAN Functions
🟧 OSC NONRTRIC
🟥 OSC near-RT RIC
🟩 Proprietary RAN Function
⬜ Apps
🟦 ONAP (Guilin)

Demo description

- Management of Policies in the Near-RT RIC
  - A policy intend to make transient changes to the behavior of the RAN.
  - For example:
    - A certain QOS should have a higher priority
    - A certain UE should be allowed to allocate more resources
- Management of Enrichment information Jobs requested by the Near-RT RIC
  - An enrichment information job may delivery any type of extra information to potentially improve the performance in the RAN.
  - For example:
    - Network/area wide information, traffic/load pattern
    - Weather information/forecast
- Registration of R-APPs in the R-APP Catalogue
  - For query of registered R-APPs
  - Status – initial prototype

Architecture



Policy "Producer"
(R-APP, SMO-app etc)

Enrichment Data Producer
(R-APP, SMO-app, External system etc)

Data delivery

Policy Management API
(REST)

"Policy Management API"
(DMAAP)

'Enrichment Data Producer API
(REST)

Control Panel GUI
(Policy and Enrichment info)

Configuration

R-APP Catalogue

Demo/Test object

Policy Management Service

Dmaap Adapter

A1-Controller

Enrichment Coordinator Service

Job DB

Enrichment Data delivery

A1-P Policy
(REST)

A1-P Policy
(REST)

A1-EI – Enrichment Information
(REST)

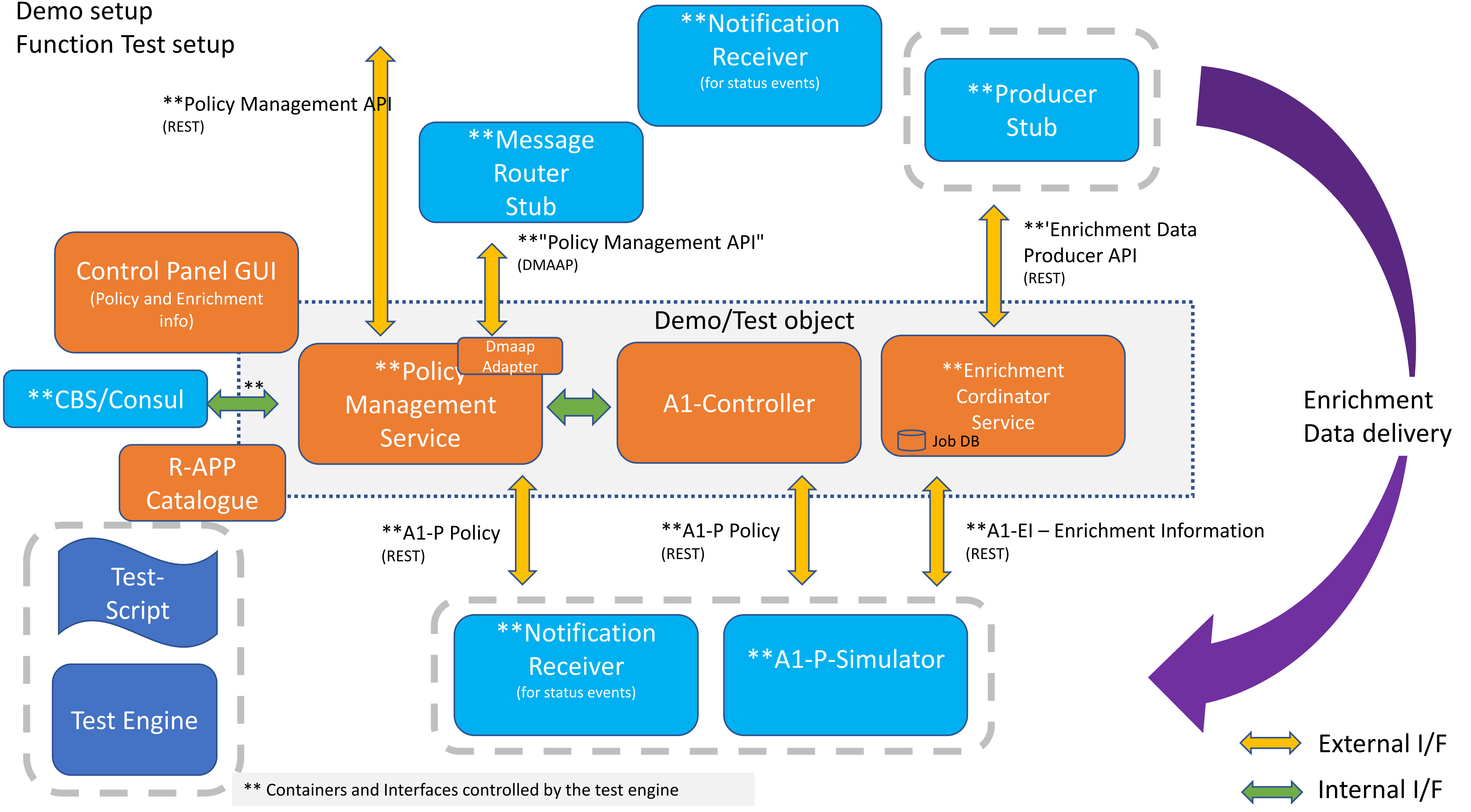
Near-RT RIC

RAN

External I/F

Internal I/F

Demo setup
Function Test setup



**Notification Receiver
(for status events)

**Producer Stub

**Policy Management API
(REST)

**Message Router Stub

**"Policy Management API"
(DMAAP)

**'Enrichment Data Producer API
(REST)

Control Panel GUI
(Policy and Enrichment info)

Demo/Test object

**CBS/Consul

**

**Policy Management Service

Dmaap Adapter

A1-Controller

**Enrichment Cordinator Service

Job DB

Enrichment Data delivery

R-APP Catalogue

**A1-P Policy
(REST)

**A1-P Policy
(REST)

**A1-EI – Enrichment Information
(REST)

Test-Script

Test Engine

**Notification Receiver
(for status events)

**A1-P-Simulator

External I/F

Internal I/F

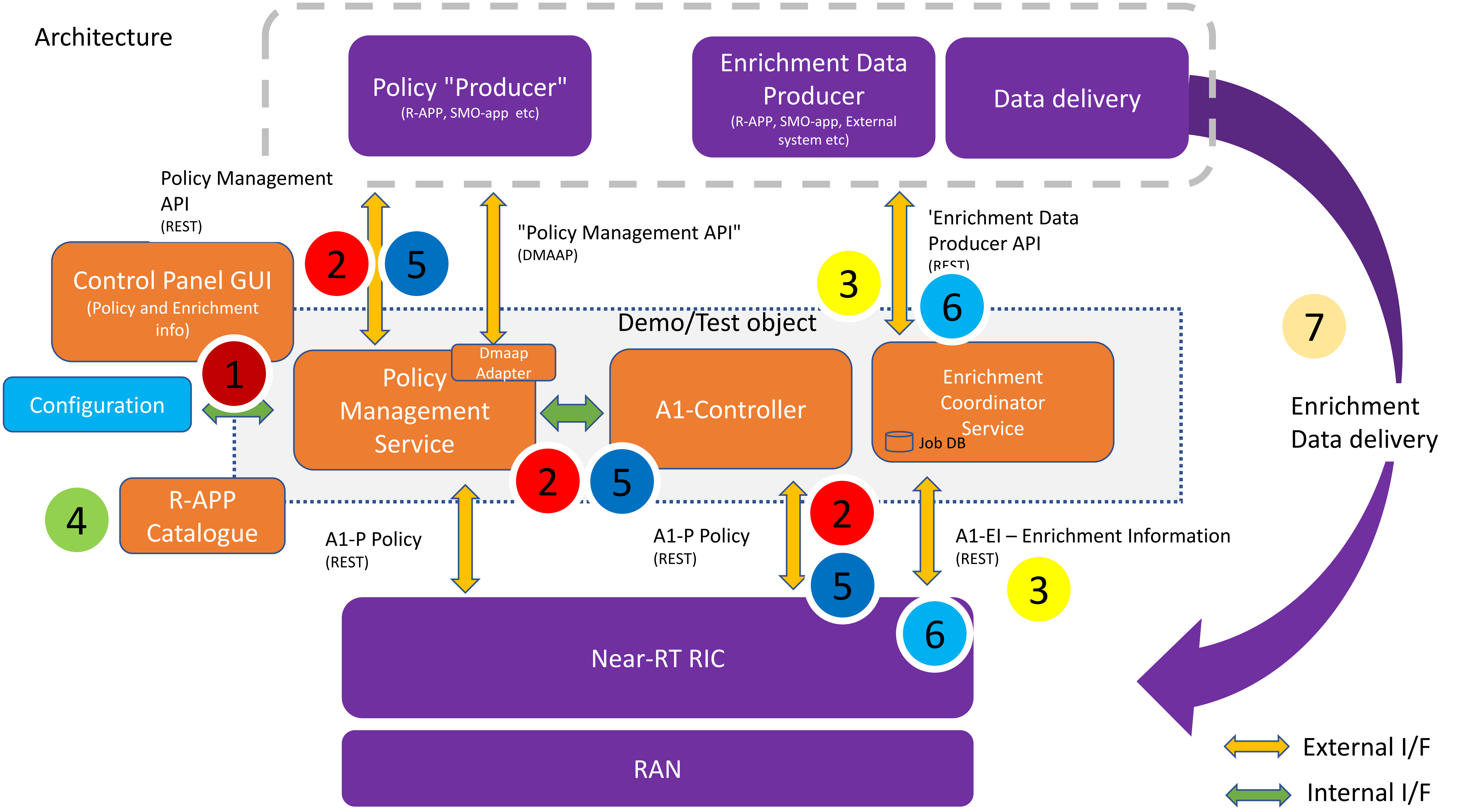
** Containers and Interfaces controlled by the test engine

Demo story

Deploy policies + Enrichment data job request

1. The Policy Management Service reads the configured Near-RT RIC
2. The Policy Mangement Service reads the exposed Policy Types from the configured Near-RT RICs. Those policy types becomes available to apps producing policy types
3. The Enrichment data producers register their supported Enrichment data types towards the Enrichment Coordinator Service and expose those types over A1-EI towads the Near-RT RICs
4. The Data Producer apps registers in the R-APP Catalogue
5. An emergency response app (produces policies) in the SMO determine that some UEs need to be prioritized in the RAN.
    1. The app creates policies, of the types discovered by the Non-RT RIC, for the UEs.
    2. The policies are deployed to the Near-RT RIC in order to impact the RAN in the desired way
6. Near-RT RIC determines that enrichment data can improve the ability to prioritize the UEs according to the deployed policies
    1. The Near-RT RIC determines that enrichment data can improve the ability to prioritize the UEs according to the deployed policies
    2. The Near-RT RIC initiates Enrichment Data Job towards the Non-RIC (Enrichment Service)
7. Enrichment data is delivered from the Data Producer to the Near-RT RIC

Architecture



Policy "Producer"
(R-APP, SMO-app etc)

Enrichment Data Producer
(R-APP, SMO-app, External system etc)

Data delivery

Policy Management API
(REST)

"Policy Management API"
(DMAAP)

'Enrichment Data Producer API
(REST)

Control Panel GUI
(Policy and Enrichment info)

Configuration

R-APP Catalogue

Demo/Test object

Policy Management Service

Dmaap Adapter

A1-Controller

Enrichment Coordinator Service

Job DB

A1-P Policy
(REST)

A1-P Policy
(REST)

A1-EI – Enrichment Information
(REST)

Near-RT RIC

RAN

Enrichment Data delivery

External I/F

Internal I/F

Demo - run yourself

- A similiar demo test script and all function test scripts are available in the nonrtric repo:
  nonrtric/test/auto-test/
- Requires: docker, docker-compose and python 3
- Run (bash):
  ./PM_EI_DEMO.sh remote --env-file ../common/test_env-oran-cherry.sh