O-Ran Software Community  
 “A” Release Requirements

Contents

[Introduction 3](#_Toc9283172)

[1. O-RAN SC Release “A” 3](#_Toc9283173)

[A. Release “A” schedule dates 3](#_Toc9283174)

[B. Target Sprint Timeline 3](#_Toc9283175)

[2. Release “A” Objective 4](#_Toc9283176)

[3. Use Case Definitions 5](#_Toc9283177)

[A. Use Case Descriptions 5](#_Toc9283178)

[B. Platform use cases: 5](#_Toc9283179)

[C. Use Case Testing 5](#_Toc9283181)

[4. O-RAN Software Project Objectives 6](#_Toc9283182)

[A. Non-RT RIC: Orchestration and Automation with ONAP 6](#_Toc9283183)

[B. xAPPs 7](#_Toc9283184)

[D. O-CU 8](#_Toc9283185)

[E. O-DU 8](#_Toc9283186)

[F. O-RU 8](#_Toc9283187)

[5. External and Common Software Projects 8](#_Toc9283188)

[A. Virtualization and acceleration Layer 8](#_Toc9283189)

[B. Operations, Administrations, and Maintenance (OA&M) 8](#_Toc9283190)

[C. Logging and Tracing 8](#_Toc9283191)

[D. Common Software Library and Tools. 9](#_Toc9283192)

[6. Integration and Test “A” Objectives 9](#_Toc9283193)

[A. End to End O-RAN Use Case Testing 9](#_Toc9283194)

[B. Software testing should be covered by automated software testing using robot framework or other tools. https://robotframework.org/ 9](#_Toc9283195)

[7. Simulator Objectives 9](#_Toc9283196)

[A. Simulators for O-RAN components, interfaces, and messages. 9](#_Toc9283197)

[B. Simulators for external components that interoperate with O-RAN 10](#_Toc9283198)

[C. Simulators for User Equipment (UE) 10](#_Toc9283199)

[8. Addendums 10](#_Toc9283200)

[A. Contributors 10](#_Toc9283201)

[B. Definitions 10](#_Toc9283202)

[C. Near-RT RIC component details 11](#_Toc9283203)

# Introduction

O-RAN Software Community (O-RAN SC ) is a Linux Foundation Project in collaboration with the O-RAN Alliance to create an open source software solution that implements a Radio Access network implementation that aligned with O-RAN Alliance specifications.

Each release for the O-RAN SC will need to define the scope of the software for each release. Each release planning will need to set goals that advance the software but works within the limitations of the available O-ARN alliance specification, testable functions, and software development resources. Open source projects work to accomplish a solution over multiple releases, however each release should have clear objectives that advance the software projects and end to end stack objectives.

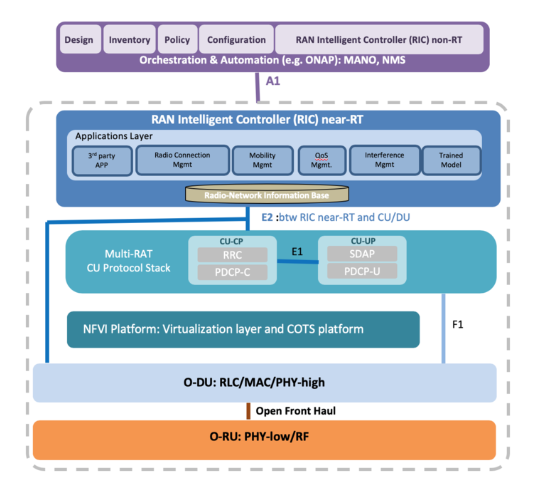


Figure 1 - O-RAN architecture

## O-RAN SC Release “A”

### Release “A” schedule dates

M0 (6/3) to M1 (6/29). This M0 involves identifying Epics, breaking down Epics into User Stories and Tasks, map Epics and US/Task to projects and dependencies, and review and finalize tasks and sub tasks in preparation for the software developers and testers to plan development sprints.

### Target Sprint Timeline

This sprint calendar is show below in Figure 1

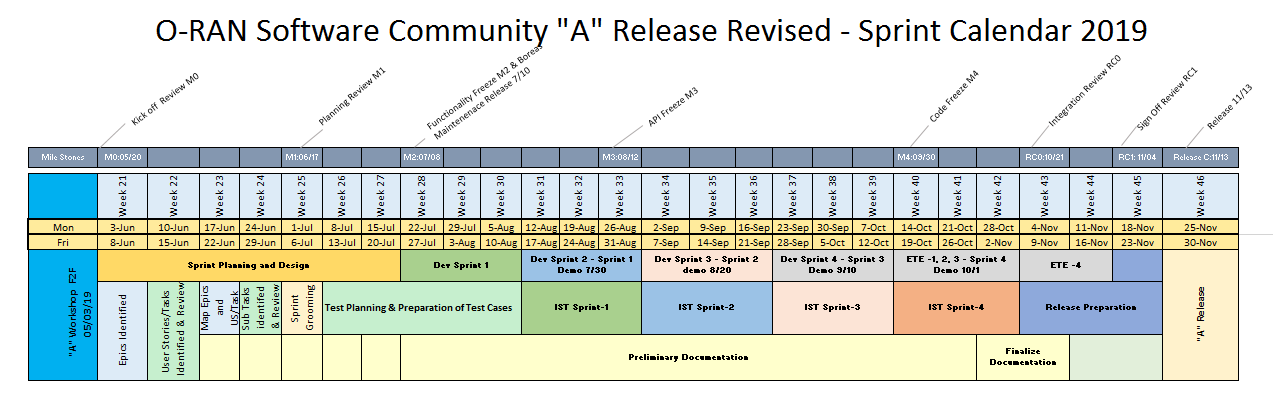


Figure 2 - Sprint calendar

## Release “A” Objective

* Demonstrate a working reference implementation of a full end-to-end O-RAN stack. The release will have some items identified as stretch goals.
  + Non-RT RIC (e.g., ONAP) implementing a basic A1 interface
  + Near-RT RIC framework implementing (see Fig. 1)
    - Application/microservices framework to host multiple xApps
    - A message bus to publish and subscribe messages inside RIC
    - A1 mediator to relay basic policies from ONAP to RIC
    - E2 Termination agent to interface with O-CU
    - R-NIB DBaaS to store RAN state information
    - UE manager to transact E2 data into the R-NIB
    - Dashboard to display FM and PM counters from RIC
  + A virtualization layer based on COTS platform (e.g., Akraino)
  + O-CU implementing a basic E2 interface agent
  + O-DU implementing
    - L1 functional blocks
    - L2 functional blocks
  + Open Front Haul implementation
  + O-RU

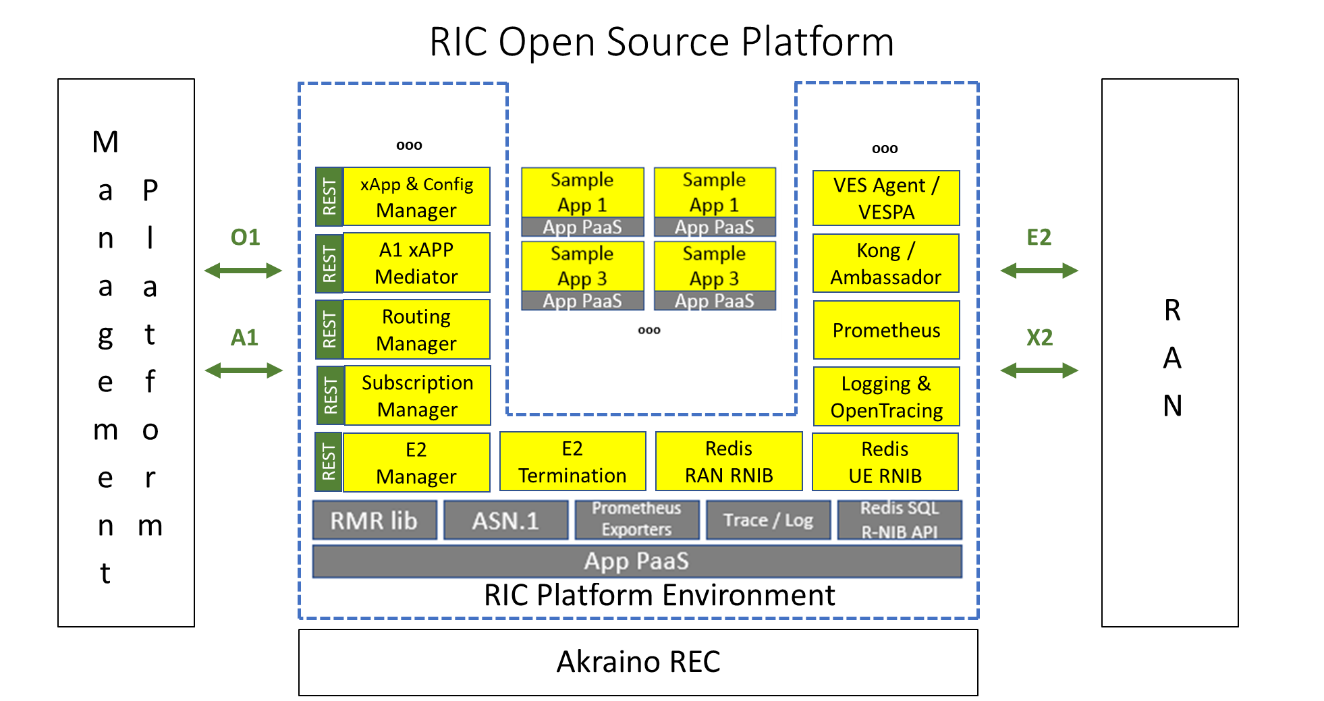


Figure 1 - near-RT RIC open source platform

Figure 1 shows an example of the near-RT RIC architecture platform deployed on a virtualized radio edge cloud (REC) infrastructure. The RIC allows xApps to be deployed and managed. The RIC also connects southbound to the RAN via E2 and northbound via A1.

## Use Case Definitions

The integration and test effort will focus on testing the requirements documented in each release. This will focus on end to end use case testing.

### Use Case Descriptions

The use cases are categorized into two categories, namely platform and service.

### Platform use cases:

#### Non-RT RIC (ONAP)

Verify ONAP Deploy (VES Collector, DMaap)

Verify Portal Deploy (Dashboard applications)

Verify Policy insertion to near-RT RIC via A1

#### Near-RT RIC

Verify RIC Deploy

     Verify Heartbeat

     Verify Fault Reporting

     Verify PM Streaming

     Verify Bulk PM

     Verify Trace Reporting

     Verify xAPP Deploy

     Verify xAPP Start

     Verify xAPP Stop

     Verify xAPP unDeploy

Verify E2 Setup Request

Verify E2 Setup Release

     Verify E2 Subscribe Report

     Verify E2 Insert (Stretch)

     Verify E2 Control (Stretch)

     Verify E2 Subscribe Delete

Service use cases: [TBD]

Verify a simple end-to-end service demonstrating

Report subscription and KPI computation; Dashboard visualization;

Load balancing or policy based HO for select UE, etc. …

### Use Case Testing

## O-RAN Software Project Objectives

The O-RAN software community will sponsor projects and supply repos and build tools for projects supporting O-RAN Alliance aligned Radio Access Network (RAN) solution.

Each project will require a set of requirements that can drive Epics. The Epics are provide the goals for the software developers and testers. User stories are used to drive the software development team’s dev cycles. The epics and user stories all role up the high level use cases and are validated by the integration and test team.

### Non-RT RIC: Orchestration and Automation with ONAP

A project should focus on use cases supported in ONAP. The work should be sponsored in O-RAN and initial code contributions in O-RAN SC. Interoperability with ONAP should be aligned between the standards and will follow initial contributions in this project.

Non-RT RIC Functional Requirements

* ORAN contributions to support A1 and O1 interface via SDN-R; The non-RT RIC will primarily include A&AI, SDN-R and DCAE
* Non-RT RIC shall support data retrieval and analysis; the data may include performance, configuration or other data related to the application
* Non-RT RIC shall support performance monitoring and evaluation

A1 Interface Functional Requirements

* A1 interface shall support communication of policies/intents from Non-RT RIC to Near-RT RIC
* A1 interface shall support communication of enrichment information from Non-RT RIC to Near-RT RIC.
* A1 interface shall support the policy/intents feedback from Near-RT RIC to non-RT RIC

Non-RT RIC Non-Functional Requirements

* Non-RT RIC shall not update the same policy or configuration parameter for a given near-RT RIC or RAN function more often than once per second
* Non-RT RIC shall be able to update policies in several near-RT RICs (stretch) Non-RT RIC:
  + Deploys xApps in near-RT RIC;
  + Provides a dashboard to view time series of KPIs computed from measurement data obtained via E2

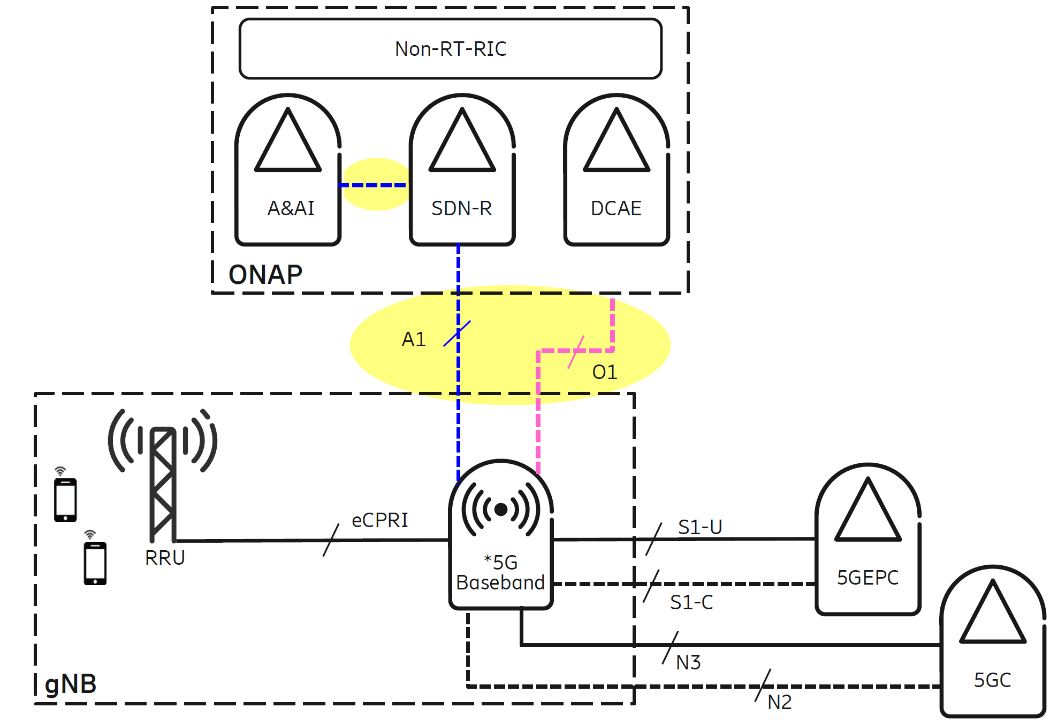


Figure 3 - A1 interface example

### xAPPs

xApps should be targeted for Release “A”. This includes open source sample xAPPs and platform applications that can we used for integration, testing, and demonstrations.

#### Sample xAPP – Admission Control xApp to allow basic admission control functionality of UEs based on a simple policy (e.g., time of day)

#### Sample xApp – Reporting xApp that computes KPIs from RNIB data (e.g., UE throughput, cell load) and provides a dashboard view in non-RT RIC

#### Sample platform application – UE Manager platform application to read E2 messages and extract specific information elements (IEs) into the near-RT RIC database (RNIB). The RNIB reflects the most recent RAN information state.

1. The Near RT-RIC should have a focus for this release. An initial framework should be put in place with support for xAPP development and operation.

#### The CU should target an initial software deliverables with limited functionality. This would be included as part of the integration and test plans.

* Near-RT RIC shall support interpreting policies and intents
* Near-RT RIC shall support deployment and execution of xApps; Life cycle management (LCM) of Xapps (e.g., start/stop/deploy/undeploy)
* Near-RT RIC platform should provide an up to date, consistent, and query-able view of UE and RAN config, context, state, and performance data to xApps running in the RIC; The RIC has a Radio Network Information Base (RNIB) which holds an accurate, updated information of the RAN (including GNB, ENB etc) state. The RNIB is a part of the overall RIC database / persistent storage which serves the needs of the various RIC and xAPP functions. We assume that whenever there is an update in the RAN state, the RIC receives REPORT messages from the RAN over E2, and this information is processed by an E2 Manager xAPP which updates the RNIB
* Near-RT RIC platform should support xApp execution with input data, ML model inference, and policy execution, to generate the corresponding RRM enforcement decisions.
* Near-RT RIC platform should be able to meet the necessary latency requirements for xApp execution

### O-CU

The O-RAN CU is targeted for Release “A”.

#### The O-CU should recruit seed code contributions from contributors.

#### The O-CU should target an initial software deliverable with limited functionality.

#### O-CU shall implement a basic E2 interface. This would be included as part of the integration and test plans.

### O-DU

The O-RAN DU is targeted for Release “A”.

#### The DU should recruit seed code contributions from contributors.

#### The DU should target a proof of concept leveraging seed code contributions.

### O-RU

The O-RAN RU is not the focus of Release “A”.

#### The RU should focus plans for future releases.

## External and Common Software Projects

This chapter provides dependencies on other open source projects (Akraino, Acumos, ONAP, etc.) that need to be leveraged to realize an end-to-end ORAN deployment.

### Virtualization and acceleration Layer

### Operations, Administrations, and Maintenance (OA&M)

#### Portal and Dashboard Analytics Software RIC and xAPP Life Cycle Management

#### O1 interface implementation (FCAPS)

FM -

CM -

PM –

Dashboard -

### Logging and Tracing

### Common Software Library and Tools.

## Integration and Test “A” Objectives

The integration and test effort will focus on testing the requirements documented in each release. This will focus on end to end test and use case testing.

### End to End O-RAN Use Case Testing

### Software testing should be covered by automated software testing using robot framework or other tools. <https://robotframework.org/>

## Simulator Objectives

Simulators are very important to O-RAN Software Community. Developers and testers need simulators to allow the teams to work on parts of the software without having to have all part available. This is very important when RF interfaces and User Equipment (UE) interoperation are required.

In early releases simulators are very important to allow partial implementations of O-RAN interfaces and modules. The needs are to allow software to be developed with a reasonable level of isolation or until other software can be completed.

### Simulators for O-RAN components, interfaces, and messages.

#### E2 interface similar is needed for near RT-RIC and xAPPs. The E2 interface connects the RIC to the RAN Control Plane.

The roles of the E2 interface are:

* To manage the SCTP connections to the RAN E2 interface
* To manage the connection lifecycle of the RAN (CRUD) using the XSetup xApp
* To receive/send E2 messages from and to the RAN devices over SCTP
* To strip the E2 protocol envelope from received messages and extract the underlying RAN E2 Service Model containers for the xApps
* To send the RAN E2 Service Model containers to the xApps that we registered to receive them using the UTA library
* To receive RAN E2 Service Model containers from the Xapps using the UTA library wrap them with E2 wrapper and send them to the relevant E2 interface
* To take care of re-transmissions, failures, timeouts, etc

The set of E2 operations that need to be supported by the gNB simulator are

| Initiated by | Elementary Procedure | Initiating Message | Successful Outcome | Unsuccessful Outcome |
| --- | --- | --- | --- | --- |
| Response message | Response message |
| RIC | RIC Subscription (r1) | RIC SUBSCRIPTION REQUEST (r1) | RIC SUBSCRIPTION RESPONSE (r1) | RIC SUBSCRIPTION FAILURE(>r2) |  |
| RIC | RIC Subscription delete (r1) | RIC SUBSCRIPTION DELETE REQUEST (r1) | RIC SUBSCRIPTION DELETE RESPONSE (r1) | RIC SUBSCRIPTION DELETE FAILURE(>r2) |  |
| RAN 8.2.5 | RIC Service Update (>r2) | RIC SERVICE UPDATE (>r2) | RIC SERVICE UPDATE ACKNOWLEDGE (>r2) | RIC SERVICE UPDATE FAILURE (>r2) |  |
| RIC or RAN | E2 Reset (r1) |  |  |  |  |
| RIC | E2 Setup (r1) | [EN-DC]/- X2 SETUP REQUEST | [EN-DC]/- X2 SETUP RESPONSE | [EN-DC]/- X2 SETUP FAILURE |  |
| RIC | Resource Status Reporting Initiation (r2) | RESOURCE STATUS REQUEST | RESOURCE STATUS RESPONSE | RESOURCE STATUS FAILURE |  |
| RAN 8.3.7 | RAN Configuration update (>r2) | [EN-DC]/[eNB] CONFIGURATION UPDATE | [EN-DC]/[eNB] CONFIGURATION UPDATE ACKNOWLEDGE | [EN-DC]/[eNB] CONFIGURATION UPDATE FAILURE |  |

| Initiated by | Elementary Procedure | Initiating Message | Which xApp? |
| --- | --- | --- | --- |
| RAN | RIC Indication | RIC INDICATION (r1) |  |
| RIC | RIC Control | RIC CONTROL INDICATION (r1) |  |
| RAN | Resource Status Reporting | RESOURCE STATUS UPDATE (r1) |  |
| RAN or RIC | Error Indication | ERROR INDICATION (r1) |  |

Legend: r1 is targeted for A release, r2 is B release

### Simulators for external components that interoperate with O-RAN

#### Not Required

### Simulators for User Equipment (UE)

#### Not Required

## Addendums

There will be different sections to support the document and its objectives.

### Contributors

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Company | Contact Info |  |
| John Murray | AT&T | jfm@research.att.com | 5/5 |
| Rittwik Jana / David Kinsey | AT&T | [rjana@research.att.com](mailto:rjana@research.att.com); dk8126@att.com | 5/20 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Definitions

|  |  |
| --- | --- |
| Term | Description |
| Not Required | This release does not require any addition work on this topic for this release. The current software may be adequate or additional requirements may appear in the future. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### Near-RT RIC component details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Is docker container/binary?** | **Component** |  |  | **Description** |
| **Near-RT RIC Platform** |  | xApp Manager |  |  | Manages xApps |
|  | Configuration Manager |  |  |  |
|  | Routing Manager |  |  |  |
| yes | E2 Termination |  |  | Setup E2 connection between RIC and gNB |
| yes | E2 Manager |  |  | Manages E2 connections |
| yes | DBaaS (Redis instantiation) |  |  | R-NIB |
| no, library | ASN.1 encoding/decoding library |  |  | Message library |
| no, library | AppPaaS - SDL (Shared data layer) |  |  | R-NIB database abstraction layer |
| no, library | AppPaas - Logging |  |  |  |
| no, library | AppPaaS - message library and routing table |  |  |  |
| no, library | OpenTracing |  |  |  |
|  | A1 Mediator |  |  | A1 interface impl. |
|  | VES agent |  |  | Collects PM data |
|  | UE manager |  |  |  |
|  | Subscription manager |  |  |  |
|  | DBaSS |  |  |  |
| **Aux** |  | VES collector |  |  |  |
|  | DMaaP |  |  |  |
| yes, in ONAP | RIC Dashboard |  |  |  |
|  | Nexus Repo for deployment |  |  |  |
| **xApps** | yes | Admission Control xApp |  |  |  |
| yes | Measurement reporting xApp |  |  |  |