CII status: near-RT RIC

Official status:

https://bestpractices.coreinfrastructure.org/en/projects/4605

How to read

All items marked in yellow are optional items that we do not fully meet

All items marked in orange are mandatory items that we fully meet, but where it is sensible to improve

All items that we do not meet have a statement on their priority: (fix-priority very-low|low|medium|high|very-high)

Basics (12 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

	near-RT RIC (end of Cherry)		
Criteria	**Result / Proof point / Notes**		
Identification			
What is the human-readable name of the project?	yes	**O-RAN SC's Near-RT RIC** RT = realtime RIC = RAN intelligent controller RAN = Radio Access Network O-RAN = Open RAN SC = software community	
What is a brief description of the project?	yes	The **near-RT RIC Platform** is a software based nearrealtime microservicebased platform for hosting micro-service-based applications - the xApps - that run on the near-RT RIC platform. xApps are not part of the RIC platform and developed in projects that are separate from the near-RT RIC platform project. The near-RT RIC platform is providing xApps the infrastructure for controlling a distributed collection of RAN base stations (eNB, gNB, CU, DU) in a region via the O-RAN alliance's E2 protocol ("southbound"). (quote from Scope of the near-RT RIC platform and its components (summary))	
What is the URL for the project (as a whole)?	yes	Near Realtime RAN Intelligent Controller (RIC)	
What is the URL for the version control repository (it may be the same as the project URL)?	yes	Multiple repositories in Linux Foundation Gerrit: https://gerrit.o-ran-sc.org/r/admin/repos/ List of repos: Scope of the near-RT RIC platform and its components (summary)	
What programming language(s) are used to implement the project?	yes	C++, Golang, Python	
What is the Common Platform Enumeration (CPE) name for the project (if it has one)?		no ID	
Basic project website content			
The project website MUST succinctly describe what the software does (what problem does it solve?	yes	Scope of the near-RT RIC platform and its components (summary))	

The project website MUST provide information on how to: obtain, provide feedback (as bug reports or enhancements), and contribute to the software.	yes	obtain: from gerrit repos or from the OSC releases: Releases bugs: Tools (mailing list, JIRA, Gerrit) enhancements: Same JIRAS tool as for feature planning. contribute: See OSC guidelines: Project Developer Wiki	
The information on how to contribute MUST explain the contribution process (e.g., are pull requests used?) (URL required)	yes	contribute: See OSC guidelines: Project Developer Wiki	
The information on how to contribute SHOULD include the requirements for acceptable contributions (e.g., a reference to any required coding standard). (URL required)	no (fix-priority very-low)	not available.	Governance 2021-02-19 Follow standard coding styles for components. Its optional
FLOSS license			
What license(s) is the project released under?	yes	Apache 2.0	
The software produced by the project MUST be released as FLOSS.	yes	Apache 2.0	
It is SUGGESTED that any required license(s) for the software produced by the project be approved by the Open Source Initiative (OSI).	yes		
The project MUST post the license(s) of its results in a standard location in their source repository.	yes	root dir of all repos included in the project	
Documentation			
The project MUST provide basic documentation for the software produced by the project.	yes	https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-ric-dep/en/latest/ and other documentation under: https://docs.o-ran-sc.org/en/latest/projects.html	
The project MUST provide reference documentation that describes the external interface (both input and output) of the software produced by the project.	yes	2021-05-25: See section "external interface" in Introduction and guides	Governance 2021-02-17 A page is to be created Introduction and guides & the page will be regularly updated for below: • xapp framework (platform lib) interface (c/python/go interface) • o1,o2, e2,a1 interfaces , description of implemented /not-implemented work • How do deploy RIC platform

Other			
The project sites (website, repository, and download URLs) MUST support HTTPS using TLS.	yes		
The project MUST have one or more mechanisms for discussion (including proposed changes and issues) that are searchable, allow messages and topics to be addressed by URL, enable new people to participate in some of the discussions, and do not require client-side installation of proprietary software.	yes	Tools (mailing list, JIRA, Gerrit)	
The project SHOULD provide documentation in English and be able to accept bug reports and comments about code in English.	yes		

Change Control (9 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

	Project A		
Criteria	**Result / Proof point**		
Public version-controlled source repository			
The project MUST have a version-controlled source repository that is publicly readable and has a URL.	yes	Scope of the near-RT RIC platform and its components (summary)	
The project's source repository MUST track what changes were made, who made the changes, and when the changes were made.	yes	Scope of the near-RT RIC platform and its components (summary)	
To enable collaborative review, the project's source repository MUST include interim versions for review between releases; it MUST NOT include only final releases.	yes	Scope of the near-RT RIC platform and its components (summary)	
It is SUGGESTED that common distributed version control software be used (e.g., git) for the project's source repository.	yes	Scope of the near-RT RIC platform and its components (summary)	
Unique version numbering			
The project results MUST have a unique version identifier for each release intended to be used by users	yes		
It is SUGGESTED that the Semantic Versioning (SemVer) format be used for releases.	yes		
It is SUGGESTED that projects identify each release within their version control system. For example, it is SUGGESTED that those using git identify each release using git tags.	yes	named branches	
Release notes			
[release_notes] The project MUST provide, in each release, release notes that are a human-readable summary of major changes in that release to help users determine if they should upgrade and what the upgrade impact will be. The release notes MUST NOT be the raw output of a version control log (e.g., the "git log" command results are not release notes). Projects whose results are not intended for reuse in multiple locations (such as the software for a single website or service) AND employ continuous delivery MAY select "N/A". (URL required)	yes	We as per RC-1 in release check list by repo, but not all repos have release notes. Good example: https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-lib-rmr/en/latest/rel-notes.html	Governance 2021-02-19 Create a release checklist comprising of this & few other from this page. Did every component update their rst release notes & did PTL summarized those on one wiki page ? RC-1 in Release criteria checklist template

[release_notes_vulns] The release notes MUST identify every publicly known vulnerability with a CVE assignment or similar that is fixed in each new release, unless users typically cannot practically update the software themselves. If there are no release notes or there have been no publicly known vulnerabilities, choose "not applicable" (N/A).	yes	As per RC-2 in release check list	Governance/Technical 2021-02-19 For own source-code bugs this can be handled manually as part of release checklist (If JIRA based security bug has been created) But for containers we should find a technical solution (automated) involving some tool e.g. docker image scanning tool (LFN provided preferred) RC-2 in Release criteria checklist template

Reporting (8 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

	near-RT RIC (end of Cherry)		
Criteria	**Result / Proof point**		
Bug-reporting process			
The project MUST provide a process for users to submit bug reports (e.g., using an issue tracker or a mailing list). (URL required)	yes	Tools (mailing list, JIRA, Gerrit)	
The project SHOULD use an issue tracker for tracking individual issues.	yes	Tools (mailing list, JIRA, Gerrit)	
The project MUST acknowledge a majority of bug reports submitted in the last 2-12 months (inclusive); the response need not include a fix.	yes	TODO	
The project SHOULD respond to a majority (>50%) of enhancement requests in the last 2-12 months (inclusive).	yes	TODO	
[report_archive] The project MUST have a publicly available archive for reports and responses for later searching. (URL required)	yes	as per RC-3	Governance 2021-02-19 Depends on previous two criterion As part of release checklist store the snapshot copy of the reports of previous two criterion into wiki page. RC-3 in Release criteria checklist template
Vulnerability report process			
The project MUST publish the process for reporting vulnerabilities on the project site. (URL required)	yes	as per section "security bugs" in Tools (mailing list, JIRA, Gerrit)	Governance 2021-02-19 Tools (mailing list, JIRA, Gerrit) Jira issues will need to be labelled for security bugs.
If private vulnerability reports are supported, the project MUST include how to send the information in a way that is kept private. (URL required) Examples include a private defect report submitted on the web using HTTPS (TLS) or an email encrypted using OpenPGP. If vulnerability reports are always public (so there are never private vulnerability reports), choose "not applicable" (N/A).	NA		Governance 2021-02-19 NA (We don't support private vulnerability)
[vulnerability_report_response] The project's initial response time for any vulnerability report received in the last 6 months MUST be less than or equal to 14 days. If there have been no vulnerabilities reported in the last 6 months, choose "not applicable" (N/A).	not applicable	Note we have RC-3 to make sure we have a report on it if we have security vulnerabilities	Governance 2021-02-19 JIRA Report & Release checklist as criteria RC-3 in Release criteria checklist template

Quality (13 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

	near-RT RIC (end of Cherry)		
Criteria	**Result / Proof point / Notes**		
Working build system			
If the software produced by the project requires building for use, the project MUST provide a working build system that can automatically rebuild the software from source code.	yes	LF Jenkins	
It is SUGGESTED that common tools be used for building the software.	yes	LF jenkins	
The project SHOULD be buildable using only FLOSS tools.	yes		
Automated test suite			
The project MUST use at least one automated test suite that is publicly released as FLOSS (this test suite may be maintained as a separate FLOSS project).	yes	robot test cases in https://gerrit.o-ran-sc.org/r/gitweb?p=it/test.git;a=tree;f=ric_robot_suite	
A test suite SHOULD be invocable in a standard way for that language. For example, "make check", "mvn test", or "rake test" (Ruby).	no (fix-priority low)		Governance 2021-02-17
It is SUGGESTED that the test suite cover most (or ideally all) the code branches, input fields, and functionality.	partial (fix-priority medium)	improvements possible	Governance 2021-02-17 Explore code coverage report if it covers input field/functionality aspects. • Cover all component & reach a code coverage level threshold as 50% • Confluence link to code coverage report for all component • Create development item for below 50% coverage • Add a regular item in the meeting (once per month) to check code coverage
It is SUGGESTED that the project implement continuous integration (where new or changed code is frequently integrated into a central code repository and automated tests are run on the result).	yes		
New functionality testing			
The project MUST have a general policy (formal or not) that as major new functionality is added to the software produced by the project, tests of that functionality should be added to an automated test suite. As long as a policy is in place, even by word of mouth, that says developers should add tests to the automated test suite for major new functionality, select "Met.	yes	See RC-4 for testing policy in new commits	Governance 2021-02-17 • To be brought up in community call for wider discussion. • Draft: Check that all large or XL commits of the last two weeks have also new unit tests. This is our policy.
The project MUST have evidence that the test_policy for adding tests has been adhered to in the most recent major changes to the software produced by the project. Major functionality would typically be mentioned in the release notes. Perfection is not required, merely evidence that tests are typically being added in practice to the automated test suite when new major functionality is added to the software produced by the project.	yes	See RC-4	Governance 2021-02-17 Check that all large or XL commits of the last two weeks have also new unit tests. This is our policy.

It is SUGGESTED that this policy on adding tests (see test_policy) be *documented* in the instructions for change proposals. However, even an informal rule is acceptable as long as the tests are being added in practice.	no (fix-priority medium)		Governance 2021-02-17 • Wiki on how to do testing in RIC • Robot framework is the primary test framework for functional testing • Unit test recommendation for component
Warning flags			
The project MUST enable one or more compiler warning flags, a "safe" language mode, or use a separate "linter" tool to look for code quality errors or common simple mistakes, if there is at least one FLOSS tool that can implement this criterion in the selected language.	Yes	We use a separate "linter tool" : Sonar. Report: https://sonarcloud.io/organizations/o-ran-sc/projects?search=ric&sort=coverage	Technical 2021-02-17 • Every component should be on Sonar • Have a list for all component on a wiki pointing to Sonar reports
The project MUST address warnings.	Yes	As per RC-5	Governance 2021-02-17 Every blocker (under sonar code smell report) should be addressed by committer of component
It is SUGGESTED that projects be maximally strict with warnings in the software produced by the project, where practical. Some warnings cannot be effectively enabled on some projects. What is needed is evidence that the project is striving to enable warning flags where it can, so that errors are detected early.	partial (fix-priority low)		Governance 2021-02-17 No to be picked as of now.

Security (16 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

	near-RT RIC (end of Cherry)		
Criteria	**Result / Proof point / Notes**		
Secure development knowledge			
The project MUST have at least one primary developer who knows how to design secure software. (See 'details' for the exact requirements.)	yes	the PTL and many members are trained for this	
At least one of the project's primary developers MUST know of common kinds of errors that lead to vulnerabilities in this kind of software, as well as at least one method to counter or mitigate each of them.	yes	the PTL and many members are trained for this	
Use basic good cryptographic practices			
The software produced by the project MUST use, by default, only cryptographic protocols and algorithms that are publicly published and reviewed by experts (if cryptographic protocols and algorithms are used).These cryptographic criteria do not always apply because some software has no need to directly use cryptographic capabilities.	yes	no TLS yet, but once it comes in Dawn we need to assure this.	
If the software produced by the project is an application or library, and its primary purpose is not to implement cryptography, then it SHOULD only call on software specifically designed to implement cryptographic functions; it SHOULD NOT re-implement its own.	yes		
All functionality in the software produced by the project that depends on cryptography MUST be implementable using FLOSS. See the Open Standards Requirement for Software by the Open Source Initiative.	yes		
The security mechanisms within the software produced by the project MUST use default keylengths that at least meet the NIST minimum requirements through the year 2030 (as stated in 2012). It MUST be possible to configure the software so that smaller keylengths are completely disabled.These minimum bitlengths are: symmetric key 112, factoring modulus 2048, discrete logarithm key 224, discrete logarithmic group 2048, elliptic curve 224, and hash 224 (password hashing is not covered by this bitlength, more information on password hashing can be found in the crypto_password_storage criterion). See https://www.keylength.com for a comparison of keylength recommendations from various organizations. The software MAY allow smaller keylengths in some configurations (ideally it would not, since this allows downgrade attacks, but shorter keylengths are sometimes necessary for interoperability).	yes	no TLS yet, but once it comes in Dawn we need to assure this.	

The default security mechanisms within the software produced by the project MUST NOT depend on broken cryptographic algorithms (e.g., MD4, MD5, single DES, RC4, Dual_EC_DRBG), or use cipher modes that are inappropriate to the context, unless they are necessary to implement an interoperable protocol (where the protocol implemented is the most recent version of that standard broadly supported by the network ecosystem, that ecosystem requires the use of such an algorithm or mode, and that ecosystem does not offer any more secure alternative). The documentation MUST describe any relevant security risks and any known mitigations if these broken algorithms or modes are necessary for an interoperable protocol.	yes	no TLS yet, but once it comes in Dawn we need to assure this.	
The default security mechanisms within the software produced by the project SHOULD NOT depend on cryptographic algorithms or modes with known serious weaknesses (e.g., the SHA-1 cryptographic hash algorithm or the CBC mode in SSH).	yes	no TLS yet, but once it comes in Dawn we need to assure this.	
The security mechanisms within the software produced by the project SHOULD implement perfect forward secrecy for key agreement protocols so a session key derived from a set of long-term keys cannot be compromised if one of the long-term keys is compromised in the future.	yes	no TLS yet, but once it comes in Dawn we need to assure this.	
If the software produced by the project causes the storing of passwords for authentication of external users, the passwords MUST be stored as iterated hashes with a per-user salt by using a key stretching (iterated) algorithm (e.g., Argon2id, Bcrypt, Scrypt, or PBKDF2). See also OWASP Password Storage Cheat Sheet).	N/A	right now we consider this interface as non-authenticated (hardcoded username /password)	Technical 2021-02-17

			. R E S T Cl ie nt / H u m a n s c a n b e c o n si d er e d a s e xt er n al u s er s. . O 1 fe at ur e u s e s n et c o nf . R I C -7 4 9 h a s b e e n cr e at e d fo r th is
The security mechanisms within the software produced by the project MUST generate all cryptographic keys and nonces using a cryptographically secure random number generator, and MUST NOT do so using generators that are cryptographically insecure.	yes	no TLS yet, but once it comes in Dawn we need to assure this.	

Secured delivery against man-in-the-middle (MITM) attacks			
The project MUST use a delivery mechanism that counters MITM attacks. Using https or ssh+scp is acceptable.	yes		
A cryptographic hash (e.g., a sha1sum) MUST NOT be retrieved over http and used without checking for a cryptographic signature.	yes		
Publicly known vulnerabilities fixed			
[vulnerabilities_fixed_60_days] There MUST be no unpatched vulnerabilities of medium or higher severity that have been publicly known for more than 60 days.	Yes	the build process uses latest versions of e.g. Ubuntu	Governance, Technical 2021-02-17 LFN supported tool to be explored for container scanning

			JIRA based reporting for vulnerabilities Static code analysis for usage of other open source modules Container scanning

Projects SHOULD fix all critical vulnerabilities rapidly after they are reported.	no (fix-priority low)		Governance 2021-02-17 Depends of availability of report from tools (scanning & JIRA) We may not pick as of now
Other security issues			

The public repositories MUST NOT leak a valid private credential (e.g., a working password or private key) that is intended to limit public access.	yes	2021-02-17: Not sure why the criteria earlier read 'A project MAY leak "sample" credentials for testing and unimportant databases, as long as they are not intended to limit public access.'	Governance 2021-02-17 To be revisited

Analysis (8 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

	near-RT RIC (end of Cherry)		
Criteria	**Result / Proof point / Notes**		
Static code analysis			
At least one static code analysis tool (beyond compiler warnings and "safe" language modes) MUST be applied to any proposed major production release of the software before its release, if there is at least one FLOSS tool that implements this criterion in the selected language.	yes	Sonar	
It is SUGGESTED that at least one of the static analysis tools used for the static_analysis criterion include rules or approaches to look for common vulnerabilities in the analyzed language or environment.	partial (fix-priority medium)	TODO-check container build system	Governance 2021-02-17 . Fix all the vulnerabilities from Sonar scan report . Sonar cloud has extra category for bugs that seem to be security relevant
All medium and higher severity exploitable vulnerabilities discovered with static code analysis MUST be fixed in a timely way after they are confirmed.	Yes	Addressed by RC-5	Governance 2021-02-17 Fix all the vulnerabilities from Sonar scan report
It is SUGGESTED that static source code analysis occur on every commit or at least daily.	partial (fix-priority high)		Technical 2021-02-17 Consider components not using Sonar
Dynamic code analysis			
It is SUGGESTED that at least one dynamic analysis tool be applied to any proposed major production release of the software before its release.	no	code coverage tool 2021-02-17: changed from no to yes after better understanding what tools are meant.	
It is SUGGESTED that if the software produced by the project includes software written using a memory-unsafe language (e.g., C or C++), then at least one dynamic tool (e.g., a fuzzer or web application scanner) be routinely used in combination with a mechanism to detect memory safety problems such as buffer overwrites. If the project does not produce software written in a memory-unsafe language, choose "not applicable" (N/A).	no (fix-priority low)		Governance 2021-02-17 Not to be picked of now

It is SUGGESTED that the software produced by the project include many run-time assertions that are checked during dynamic analysis.	no (fix-priority low)		Governance 2021-02-17 Not to be picked as of now
All medium and higher severity exploitable vulnerabilities discovered with dynamic code analysis MUST be fixed in a timely way after they are confirmed.	N/A (fix-priority low)		Governance, Technical 2021-02-17 • Check with LFN support for any dynamic code analysis tool (Keep Thoralf in cc) • Potential candidates are O1 (ssh + netconf) Or A1 with http