RTI Connext DDS Professional Quality Declaration (REP-2004)

Version 1.0
May 22, 2020

Document Goal

This document gathers the information relative to the Quality of RTI Connext according to the standard REP-2004. which conveys the quality or maturity of packages in the ROS ecosystem.

This document fulfills the requirements for Quality Level 1 for RTI Connext DDS Professional.

Version Policy

Must have a version policy

The RTI Software Versioning Policy (RTI-DEVP015) document describes the versioning policy for all our products. Generally, RTI Connext product versions consist of 4 fields. The major number is updated when a large new market feature set or re-architecture is introduced. The minor number indicates a major general availability release (GAR). The third number is about a maintenance release roll up, or when we deliver a customer specific release, from the major.minor version. The fourth digit indicates port, patch or limited access customer specific releases.

Must be at a stable version

Every production ready release, be it a general available release, or a customer specific feature, port or patch release, must pass our documented release criteria and are considered a stable release. Furthermore, we take great care in providing wire and API compatibility with previous versions. In the case where there is an incompatibility or interoperability issue, this must be documented, and in many cases we provide a backward compatibility option as well.

A release may include experimental features or products. These are clearly documented (example). Experimental features are used to evaluate potential new features and obtain customer feedback. They are not guaranteed to be consistent or supported and they should not be used in production.

In addition to each major General Access Release of RTI Connext, we also create a generally available maintenance version containing bug fixes that address issues found in the original GAR. For example for our last major GAR 6.0.0, the maintenance release is 6.0.1, for the previous version 5.3.0, the maintenance version is 5.3.1.

Must have a strictly declared public API

RTI Connext APIs are clearly documented, both as shipped with the product, and on the public community portal (https://community.rti.com/documentation). Experimental features are clearly documented as well. For example, this is our C API of our latest stable version 6.0.1

Must have a policy for API stability

Our API stability policy is documented as part of our Development Process Manual (RTI-DEVP001). As a general rule, any incompatibility or interoperability issues or changes must be publicly documented. If wire compatibility is broken between two GAR releases, there should be a backward compatibility option to revert to the old behavior. Wire interoperability or public API compatibility issues must be signed off by the technical leads and the product manager.
Furthermore, non GAR releases (e.g., patches and ports) shall avoid breaking compatibility and interoperability with the GAR from which they are created.

As we mentioned before experimental features are used to evaluate potential new features and obtain customer feedback. They are not guaranteed to be consistent or supported and they should not be used in production and therefore we don't guarantee API stability for them.

Must have a policy for ABI stability

The RTI Development Manual (RTI-DEVP001) clarifies that RTI does not guarantee Application Binary Interface (ABI) compatibility between different versions of Connext DDS. An application compiled using one version of Connext DDS must be recompiled when moving to a different Connext DDS version.

In addition, the public RTI documentation calls this out as well: e.g., see the RTI Connext Migration Guide for Connext 6.

Must have a policy that keeps API and ABI stability within a released ROS distribution

N/A - RTI does not manage the integration of RTI Connext Pro with ROS.

Change Control Process

Must have all code changes occur through a change request (e.g. pull request, merge request, etc.)

Our current process uses Atlassian Bitbucket as a control management system. That allows us to enforce that all code changes to our main branches (development and release branches) are done through a pull request. It's not allowed to do changes on those branches directly.

The RTI Development Manual describes how code reviews and pull requests are to be managed, from selecting the reviewers, to handling emergency situations.

Must have confirmation of contributor origin (e.g. DCO, CLA, etc.)

Our Control Management System (Bitbucket) allows us to identify and track the contributor of every change that is introduced in our code.

Must have peer review policy for all change requests (e.g. require one or more reviewers)

Pull Request to our main development branch requires at least one reviewer. The RTI Development Manual provides clear guidelines on selecting the reviewers. E.g., changes for patches/maintenance releases requires the approval of a Support Manager/Lead.

Must have Continuous Integration (CI) policy for all change requests

We have Continuous Integration for all changes at a different levels:

1.- Developers of Core and Infrastructure Services can validate the changes in their branches prior to being merged to the development or release branches.

2.- Every change introduced in a main branch is compiled and unit tested for Core and Infrastructure services in a Linux architecture.

3.- We also have CI plans that validate daily the build of the changes in the main development branch compiling the code in at least one platform of this kind: Linux, Windows, MacOS, Vxworks and Integrity. We also run our unit test in Linux, Windows, MacOS.

4.- Finally, we compile and run all the unit tests weekly for all the architectures we support.

Must have documentation policy for all change requests

The RTI Development Manual requires each release to be accompanied with a set of documentation describing the release. That includes Release Notes, What's New, Platform Notes, Third Party Software documentation, Getting Started Guide, Migration guide, API Reference and User Manual.

Our development Jira workflow includes a mandatory step for both bugs and new features where the developer needs to provide the documentation that is reviewed by our technical writer team. That documentation is included in the What's New and Release Notes documents explained in the next section.

Documentation

Must have documentation for each "feature" (e.g. for rclcpp: create a node, publish a message, spin, etc.)

RTI provides a full set of documentation with our product releases. This includes a What's New document or section, covering new features part of the product. The Release Notes describe compatibility notes, what's fixed and any known issue. A user manual describes the feature set in great details. The API references cover the various programming language bindings.
For an example, see: https://community.rti.com/documentation

Must have documentation for each item in the public API (e.g. functions, classes, etc.)

Our public API documentation includes a complete description about all public classes and its methods.

Must have a declared license or set of licenses

Our installation includes a Software License Agreement document that is also shown in one of the steps of the installation. The text of the license agreement may be different for different types of installations; developer, evaluation, non-commercial use, etc.

A copy of the Real-Time Innovations, Inc. Software License Agreement can be found at: https://www.rti.com/free-trial/license-agreement

A copy of the Non-Commercial License Agreement can be found at: https://www.rti.com/ncl

We also provide two documents called Third-party Software in RTI Connext DDS – Core Libraries & Utilities and Third-Party Software in RTI Connext Tools, Services & Plugins, that outlines Real-Time Innovations' (RTI) usage of third-party open source software in its core libraries and tools, services, and plugins respectively.

Must state copyrights within the project and attribute all authors

All our product and documentation has a copyright notice declaration. Here we have an example of such a copyright. The Third-Party Software documents (see above) attributes the authors of the third-party open source software we use in the product.

Must have a "quality declaration" document, which declares the quality level and justifies how the package meets each of the requirements

The RTI Development Process Manual (RTI-DEVP001) documents the release acceptance criteria. As part of our release process we review how the product meets these criteria. In more recent releases, we created a more formal artifact, a Release Acceptance Checklist, that is a product agnostic checklist which references the specific acceptance quality criteria that our development process follows. Prior to releasing the software, it shall be signed off by the release manager, product manager(s), and VP of Engineering

Testing

Must have system tests which cover all items in the "feature" documentation

Our development process includes a design document and a test plan of the features released with the product. The results of each test of a release are achieved as artifacts of the release.

An example of the testing plan of a feature could be found here where we define the test plan for XTypes.1.2 interoperability. An example of the result of the execution of those tests in this case using RTI Connext DDS Pro 6.0.1 and RTI Connext Micro 3.02 can be found here.

Must have system, integration, and/or unit tests which cover all of the public API

RTI's development process includes a description of the different kinds of tests we run in our system including point test, unit test, feature test, interoperability test, performance test, memory profiling, scalability, performance. That also includes a traceability plan in case that the tests were not passing.

Code Coverage

Code coverage is something we do selectively at the moment. Individual projects and engineers use "gcov".

Performance

Must have performance tests (exceptions allowed if they don't make sense to have)

RTI has a public website where users can find the results of our performance test for latency, throughput and memory. In addition, the RTI Connext DDS Performance test, used to generate the performance results is publicly available on the community website

Must have a performance regression policy (i.e. blocking either changes or releases on unexpected performance regressions)

Our Release Criteria documentation establishes a process where in each release an acceptance plan of performance is defined that includes latency and throughput criteria. Our Continuous Integration plans on performance allows us to detect and react until possible changes that produce unexpected performance regressions.

Linters and Static Analysis

Must have a code style and enforce it

RTI's development process includes a document called Middleware Coding Standard (RTI-DEV032). This document formally specifies the coding standards that every developer shall follow during the development process. This document is divided into two main sections. The first section specifies the set of rules that target source code formatting. The second section specifies the set of rules that target code construction and best-practices.

Must use static analysis tools where applicable

As part of our development process of RTI Connext DDS Professional we analyze the issues with severity error that the tool cppcheck detects.

Dependencies

Must not have direct runtime "ROS" dependencies which are not at the same level as the package in question ('Level N')

RTI Connext does not have any"ROS" runtime dependencies.

May have optional direct runtime "ROS" dependencies which are not 'Level N', e.g. tracing or debugging features that can be disabled

RTI Connext does not have any"ROS" runtime dependencies.

Must have justification for why each direct runtime "non-ROS" dependency is equivalent to a 'Level N' package in terms of quality

RTI Connext does not have any"ROS" runtime dependencies.

Platform Support

Must support all target platforms for the package's ecosystem.For ROS 2 this means supporting all tier 1 platforms, as defined in REP-2000

RTI Connext is supported in a big variety of platforms including several versions of Linux, Windows, MacOS, VxWorks. RTI Connext DDS Professional includes a Platform Notes document that states the supported platforms in each GAR release.

Security

Must have a declared Vulnerability Disclosure Policy and adhere to a response schedule for addressing security vulnerabilities

RTI has a Software Vulnerability Management Process (RTI-DEVP018) that captures the guidelines to manage vulnerabilities in the RTI Software Products. The document includes information about how to track, communicate, fix, document, and release patches that involve vulnerability fixes.