PRQA Plugin

Plugin Information
View PRQA on the plugin site for more information.

Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version:

- Credentials stored in plain text

Official Maintainer

Currently developed by

Originally developed by

A plug-in for doing static code analysis using PRQA.
This plug-in was developed and maintained by Praqma up until release 1.2.2. Programming Research Ltd is currently maintaining the plug-in. Please direct service and feature requests directly to Programming Research**.**

Programming Research Static Analysis Plug-in

This plug-in enables the PRQA static analysis tools QA·C and QA·C++ to be integrated easily with Jenkins.

This plug-in is maintained by Programming Research. It was originally developed and maintained by Praqma as far as release 1.2.2, and is currently being developed by Present Technologies. All service and feature requests should be sent directly to Programming Research.

Plug-in version 3.0.1 supports PRQA Framework version 2.2.0 and later. PRQA Jenkins Plug-in support Jenkins version up to 2.107.2.

Features

This plug-in allows source code to be analyzed with QA·C or QA·C++, PRQA's static analysis tools. The plug-in performs the following key tasks in the post-build stage automatically:

- Analyzes a PRQA project – optionally including dataflow and cross module analysis.
- Generates a compliance report

- Compares the total number of messages in the project against a configurable threshold and sets the build status to ‘unstable’ if the threshold is exceeded. This can be used as a gateway to preventing subsequent tasks from running.
- Optionally uploads the analysis results to QA·Verify, the PRQA web-based Quality Management System

This plug-in is able to analyze configured PRQA projects.

The projects are configured using PRQA Framework. They can be configured using PRQA tools to support the old analysis interface.

The plug-in allows the automation of the creation and analysis of PRQA-enabled projects. This enables the analysis to be performed as a part of a continuous integration strategy.

The plug-in displays a graphical history depicting the number of messages, and overall compliance levels in the project.

Download

Download the latest version manually HERE or alternatively, use the Jenkins CI Update Center. Earlier versions of the plugin are available in archive.

Prerequisites for "PRQA Framework"

Supported versions of PRQA Framework

PRQA Jenkins plug-in currently supports PRQA Framework 2.2.0 and later. If you encounter any issues drop us a note.

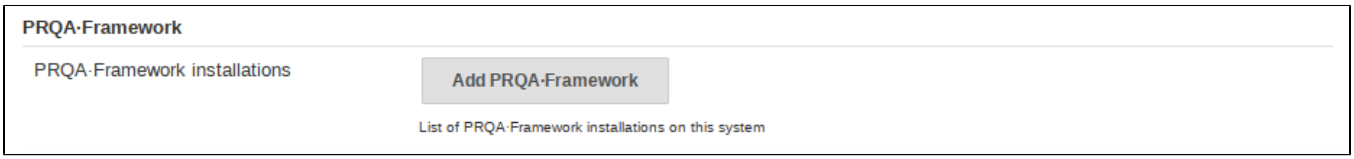
Installed Applications

Prior to installing and using the PRQA Jenkins plug-in, you need to ensure that the following PRQA applications are installed:

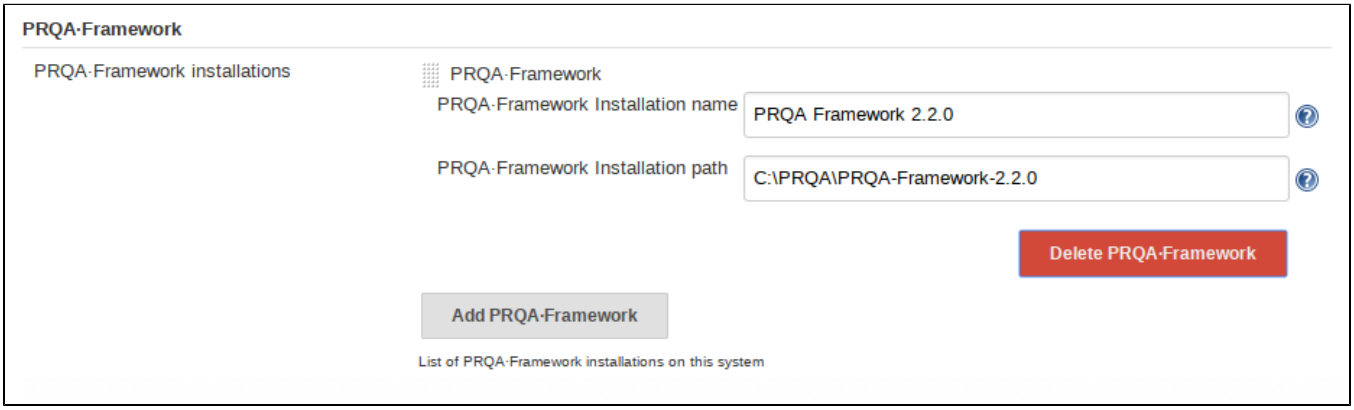
- PRQA Framework
- Reprise license manager
- QA·Verify Server (Optional)

PRQA Framework Configuration

The plug-in should be configured to use the installed version of PRQA Framework. This is done by navigating to the 'Manage Jenkins' > 'Global Tool Configuration', and adding the information about PRQA Framework installation.



To add details of PRQA Framework, press the Add PRQA Framework button.



This is an example of a PRQA Framework configuration, where the default installation directories have been used. The plug-in will automatically expand locations and add the necessary bin folders to the path. Please enter the correct details for the PRQA Framework installation.

As there might have multiple versions of PRQA Framework installed on the machine, it is best practice to include the PRQA Framework version number (or some other identification in the PRQA Framework Installation Name). The plug-in allows multiple versions of the PRQA Framework to be installed.

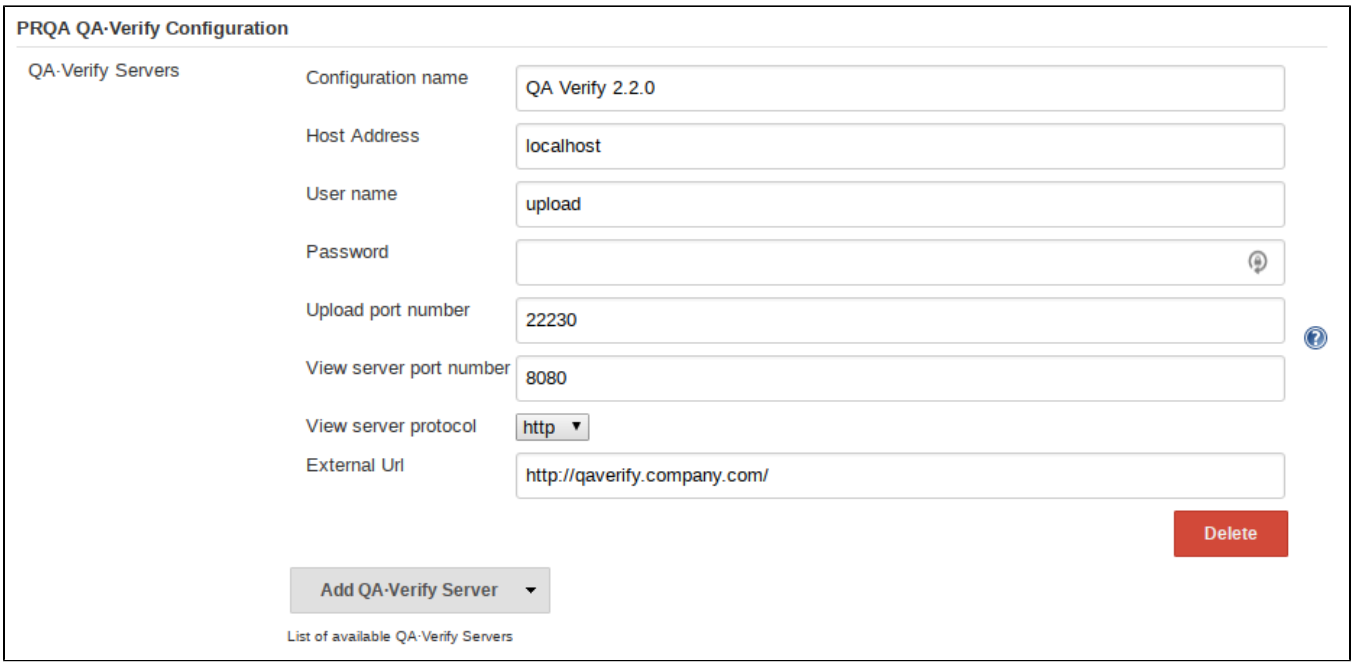
Support for PRQA tools prior to PRQA Framework 2.2.0

PRQA Jenkins 3.0.0 (or above) no longer supports PRQA tools prior to PRQA Framework 2.2.0. If you require support for previous version of PRQA tools please consider using an older version of the plug-in from archive. Download links are available above.

QA·Verify Configuration

PRQA Jenkins plug-in can be configured to use a QA·Verify server instance if required. This is done through the options available at 'Manage Jenkins' > 'Configure System'.

To add details of a QA·Verify server instance, click the 'Add' button.



This is an example of a QA·Verify configuration. Please add the correct credentials for the QA·Verify Server.

PRQA Static Analysis Configuration

Creation of PRQA Framework Project

PRQA Jenkins plug-in requires that the PRQA Project is already set up - namely that the PRQA Project file (prqaproject.xml) exists and is populated with all the necessary information to perform the analysis.

If the PRQA project is not part of the source project, it can be created as a build step prior to the PRQA static analysis. This step determines the location of the project file.

To do this, select 'Add build step'. If the build machine is a Windows machine then select the 'Execute Windows' batch command option. Otherwise, select the 'Execute shell' option.

It is possible that one of these steps is already set up to run the make or some other build script. This step can be used instead of adding a new step.

Enter into the command box the command line entry options to create and configure the PRQA project.

QA·CLI provides a command line interface to PRQA Framework and PRQA Tools. QACLI allows these to be integrated with builds on build servers. It features a command line interface in the format:

> qacli <subcommand> [options...]

where related features are grouped under a subcommand.

The address of the PRQA license server should be set where the system can retrieve a license. The following command can be used to accomplish this:

> qacli admin --set-license-server <port (hostname)ip>

where the argument refers to an application layer address, for example, 5055@192.168.1.10 or 5055@myhost.

A PRQA project file contains the names of all configuration files that contain related settings. Creating a new PRQA project file is a case of specifying the directory where the project should be created, and specifying the configuration files that it should contain. A PRQA project consists of:

- at least one Compiler Compatibility Template (CCT)
- one Rule Configuration File (RCF)
- one Analysis Configuration File (ACF)

The PRQA project can also contain:

- a Naming Convention File (NCF)
- a Version Control Compatibility File (VCF)
- a QA·Verify Connection File (QCF)

The following command creates a new PRQA project file:

> qacli admin --qaf-project-config --qaf-project <directory> --cct <path> --rcf <path> --acf <path>

For example:

> qacli admin --qaf-project-config --P . --C MS_VC+_CL_16_x86_C.cct --C MS_VC+_CL_16_x86_C++.cct --A default.acf -R default-1.0.rcf

After creating the PRQA project file, the next task is to populate it with the source files and analysis data (i.e. the include paths and macro definitions used to compile each source file) from the build. This data can be extracted using build process monitoring. PRQA will monitor the build process, extract the relevant data and add it to the specified project.

The following command can be used to accomplish this:

> qacli admin -P <project-path> --build-command <command>

where project-path is a path to a valid PRQA project file ,and build-command is a command that will set the build in motion and compile the files.

If the build command contains more than one word, it should be enclosed within quotes. If the build command is complex (e.g. has quotes within it), the command should be placed in a script. The path of this script should be used as the argument.

The above command can be reused to replace the old data with the data gathered from the build. This is useful for keeping PRQA projects synchronized with the build projects.

For example:

> qacli admin -P . --build make

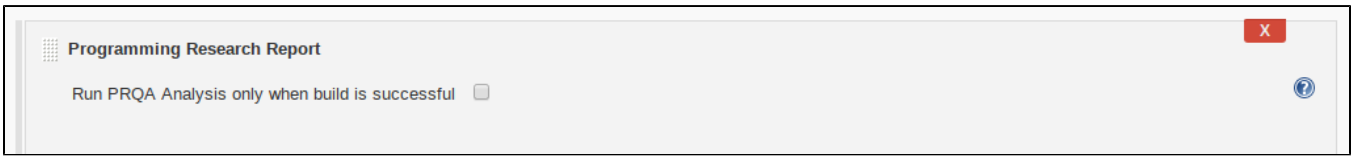
Configuration of the PRQA Static Analysis

The PRQA Jenkins plug-in enables the configuration of the PRQA static analysis as part of Jenkins jobs. In the job configuration, simply add a post-build action, by selecting 'Configure' on the job homepage.

Click on the Add post-build action button and select 'Programming Research Report'.

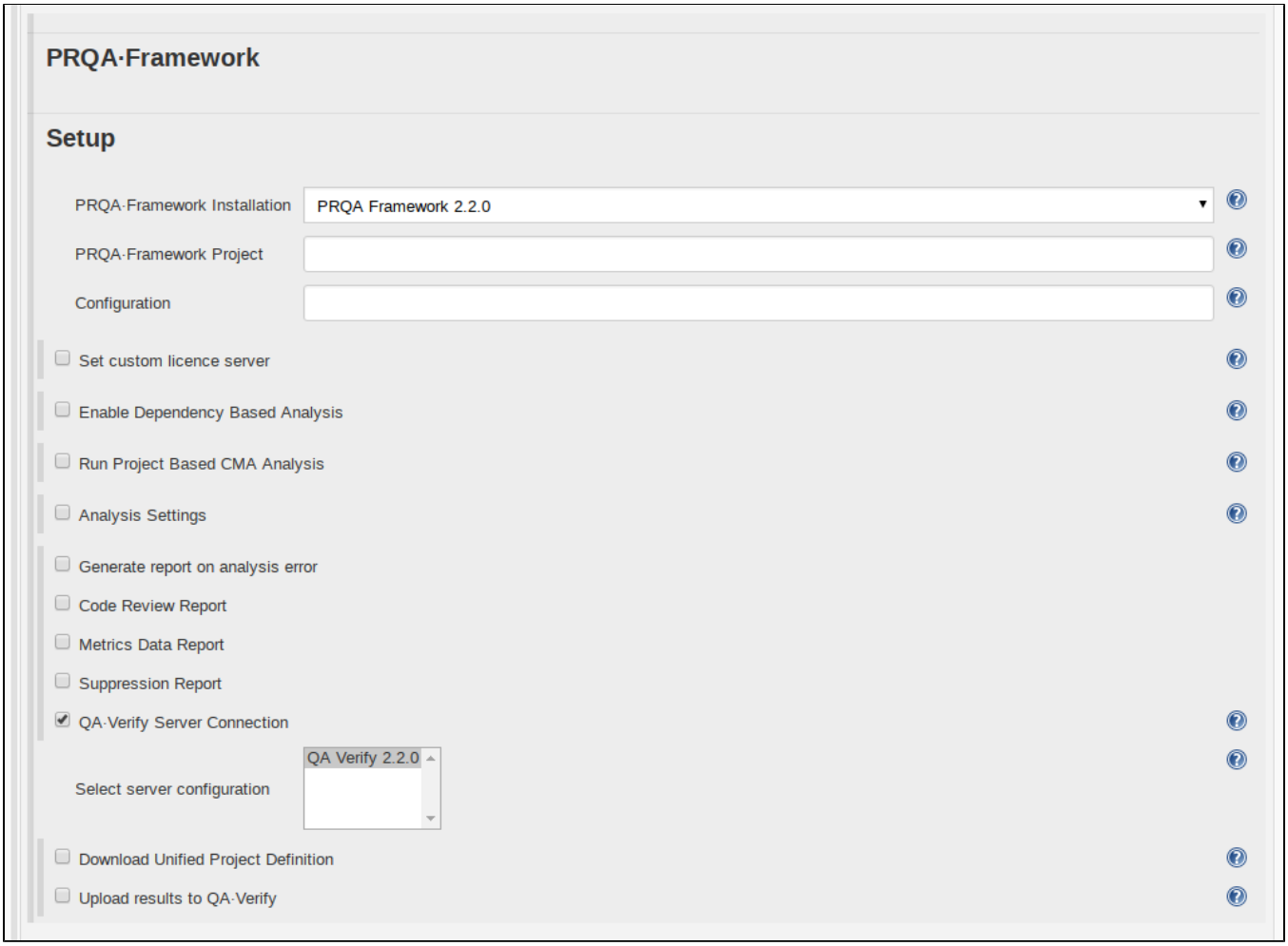
Conditional execution of PRQA static analysis

By default, the PRQA analysis is always performed, regardless of whether project build steps pass or fail. Please select 'Run PRQA Analysis only when build is successful' to perform the analysis in the event that build steps are successful.



Configuration of PRQA Framework

The details of the static analysis with PRQA Framework can be configured using the following PRQA Jenkins plug-in options:



PRQA Framework Installation:

Select one of the PRQA Framework installations that was set up in the Global Tools Configuration section.

PRQA Framework Project:

Enter the path to the directory containing the PRQA project file, prqaproject.xml, for the project to be analyzed.

If the project file is in the top level directory of this module in the repository, then this can be left blank.

It is also possible to create a PRQA project file as a prior build step: that step will determine the location of the project file.

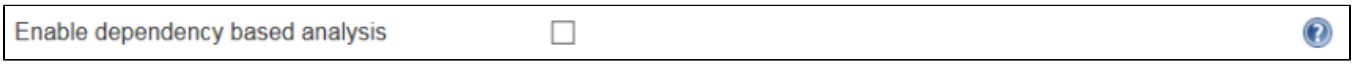
If the project file is inside the Jenkins workspace, then you can just supply the relative path to the directory from the project root directory.

The plug-in will automatically run static analysis on this project and produce the report.

License Server:

To use a custom license server, select 'Set Custom license server' and enter the license server address.

Dependencies based analysis:



The plug-in will by default analyze all the source files. The amount of time taken to analyze the project can be reduced by only analyzing the source files that need analysis.

Select 'Enable dependency-based analysis' if this condensed analysis is required. Only source files that have changed, or have no previous analysis output, will be analyzed. Files that have up to date analysis output will not be analyzed.

Project Based Cross-Module Analysis:

Project Based Cross-Module Analysis (CMA) is performed once all of the source files have been analyzed. CMA must be enabled to fully enforce a coding standard.

Note that CMA may take a long time to complete. If enabled, it will always be run regardless of the dependency-based analysis setting.

If Project Based Cross-Module Analysis is required, select 'Run Project Based CMA Analysis'.

The plug-in will by default analyze all the source files. The amount of time taken to analyze the project can be decreased by only analyzing the source files that need analysis.

Select 'Enable dependency based analysis' if this condensed analysis is required. Only source files that have changed' or have no previous analysis output, will be analyzed. Files that have up to date analysis output will not be analyzed.

Reports:

The generation of reports in the event that errors are found during the static analysis, can be disabled by selecting 'Generate report on analysis error'.

PRQA Jenkins plug-in support the configuration of multiple reports: code review, metrics and suppressions report.

Jenkins startup options

To ensure that reports are shown correctly, Jenkins' Content Security Policy configuration needs to be overridden using the hudson.model.DirectoryBrowserSupport.CSP property. The option must be added to Jenkins startup:

```
java -Dhudson.model.DirectoryBrowserSupport.CSP="sandbox; default-src 'none'; img-src 'self'; style-src 'self' 'unsafe-inline';" -jar jenkins.war
```

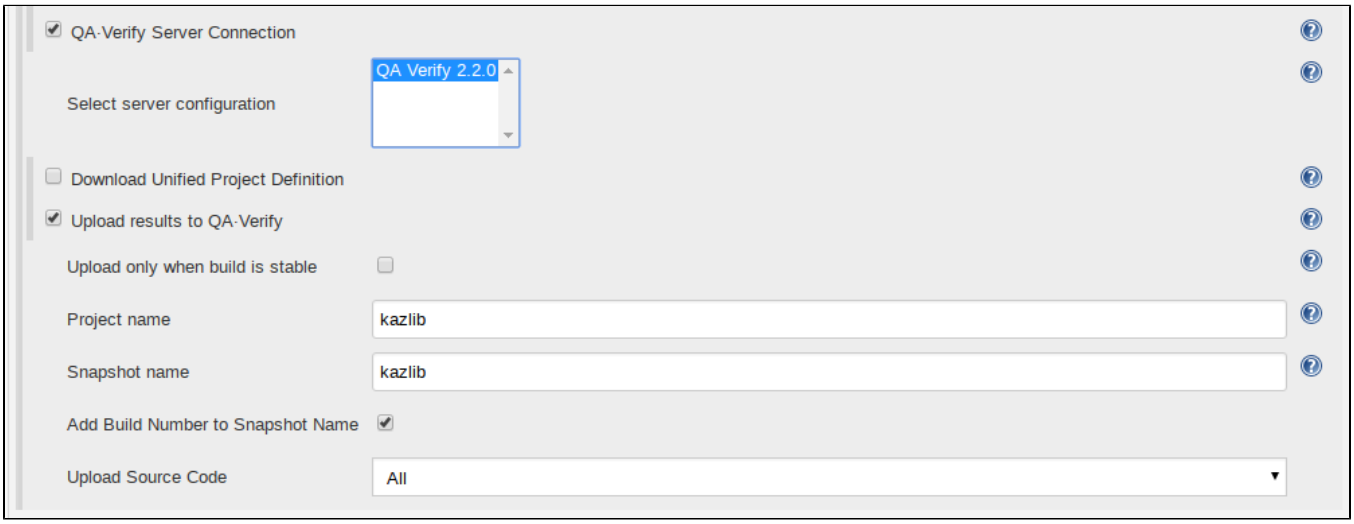
Upload results to QA·Verify

One of the most powerful features is the ability to upload the results of the analysis to QA·Verify. This enables Developers to see the in-depth results of the analysis including annotated source code via a web browser showing the exact location of the coding standard violations. The non-compliances can then be fixed before re-committing the code.

It also allows deviations from the coding standard to be added, managed and reported providing the proof required to comply with coding standards such as MISRA C.

QA·Verify also allows other stakeholders such as Managers and even Customers to view the quality information. They are able to see how many defects there are in the code – and whether this is increasing or decreasing. This visibility of project level information equips Managers with the data needed to make decisions or where to deploy resources if required.

If the results need to be uploaded to QA·Verify, select 'Upload results to QA·Verify'. This will enable uploading the project analysis as a snapshot to QA·Verify.



Select a server configuration to be used when uploading to QA·Verify that was set up in the global Jenkins configuration.

To automatically upload the static analysis results into QA·Verify, select 'Upload results to QA·Verify'. This will enable uploading the project analysis as a snapshot to QA·Verify.

Enter the name of the QA·Verify project to be uploaded. Once a project has been uploaded to QA·Verify, the project name must stay the same, otherwise the snapshot will not be attached to the correct project. It is recommended that you first create the project in QA·Verify, so that you can verify that the project configuration is correct before uploading snapshots.

Code upload settings control how the source code is uploaded to QA·Verify.

- *All code uploaded* will unconditionally upload all source and header files to QA·Verify so they will not need to access the repository. The VCS can still be used during upload to obtaain the file versions and authors - specify a VCS configuration file below.
- *No code uploaded* will not upload any code. QA·Verify will retrieve all required code from the repository. A VCS configuration file must be specified below.
- *Only code not in VCS* will only upload files that are not in the Version Control System (e.g. generated files). Files that are in the repository will not be uploaded.

Specify the full path to the top level of the source directory structure as the source origin. The source origin will default to the current workspace if nothing is specified

This value is only used when source code is uploaded to QA·Verify (i.e. using *All code uploaded* or *Only code not in VCS* settings) and is used to remove the leading parts of pathnames in the Web Message Browser.

Specify Thresholds for PRQA Static Analysis

The number of messages generated by the PRQA static analysis can be compared against thresholds for build stability. If a threshold is exceeded, the build status is set to unstable. This can be used as a gateway to prevent subsequent tasks from running.

The messages are organized into message levels according to the severity of the message, with Level 0 being the lowest.

The thresholds can be set to operate at a message level, allowing the severity of the messages to be taken into account. Code that contains low-severity maintenance and style issues may pass but code that contains critical issues would fail.

There are three kinds of threshold: Messages, File Compliance and Project Compliance.

Message Threshold Level

This sets the message level from which the message total is calculated. Only messages from this level and above are summed to give the message total that is used for the Messages Threshold comparison. Note that File and Project Compliance thresholds are not affected by this setting.

Message Compliance Threshold

Either enter the 'Maximum messages target' or select 'Continuous improvement'.

The 'Maximum messages target' is the total number of messages that are allowed on all levels for all the files in the project. Messages in header files are de-duplicated (i.e. they are only counted once and not for each source file with which the header is included).

The 'Message Compliance value' is the total number of messages for all the files in the project that are at the 'Message Threshold Level' or above.

If the 'Message Compliance value' for the build is higher than the 'Maximum messages target', the build is marked as unstable.

'Continuous improvement' states that the 'Message Compliance value' for the build will be less than or equal to the 'Message Compliance value' found in the previous build. If the 'Message Compliance value' for the build is higher, the build is marked as unstable.

Project Compliance Threshold

Either enter the 'Project compliance target' or select 'Continuous improvement'.

The 'Project Compliance target' is the minimum percentage of message groups (a coding standard ‘rule’) that have no messages in the project.

The 'Project Compliance level' is the percentage of message groups that have no messages in the project.

If the 'Project Compliance level' for the build is lower than the 'Project Compliance target', the build is marked as unstable.

'Continuous improvement' states that the 'Project Compliance level' for this build will be greater than or equal to the 'Project Compliance level' for the previous build. If the 'Project Compliance level' for the build is lower, the build is marked as unstable.

File Compliance Threshold

Either enter the 'File Compliance target' or select 'Continuous improvement'.

The 'File Compliance target' is the minimum mean across all files of the percentage of message groups (a coding standard ‘rule’) that have no messages in each file.

The 'File Compliance level' is the mean across all files of the percentage of message groups that have no messages in each file.

If the 'File Compliance level' for the build is lower than the 'File Compliance target', the build is marked as unstable.

'Continuous improvement' states that the 'File Compliance level' for this build will be greater than or equal to the 'File Compliance level' for the previous build. If the 'File Compliance level' is lower, the build is marked as unstable.

A high 'File Compliance level' and low 'Project Compliance level' indicate that each file violates only a small number of rules but that each file violates a different rule.

If the 'File' and 'Project Compliance levels' are close, then each file is violating more or less the same rules.

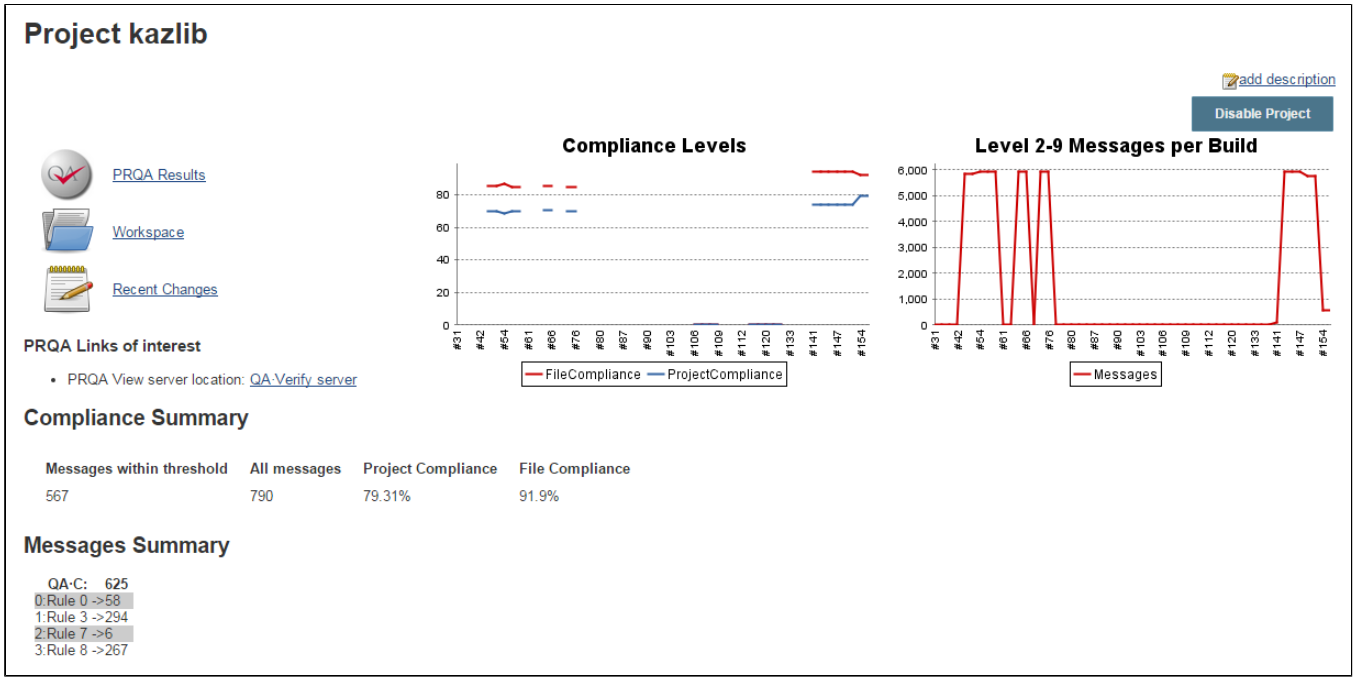
If any of these thresholds are not met, then the build will be marked as unstable. Note that only configuration errors or analysis errors will mark the build as failed.

The PRQA Jenkins plug-in will be set by default to 'No thresholds'.

PRQA Static Analysis results

The PRQA Jenkins plug-in automatically creates graphs showing the current number of messages and the current compliance levels in the Job project page. It also shows:

- Historical depiction of the overall compliance history.
- Link to the QA·Verify server that was used in the upload.
- Relevant build artifacts in the artifacts list, for easy access to generated reports and logs.



Release notes

Changelog

Version 3.0.1 - 2018-04-17

- Updated plug-in to fix JENKINS-48939

Version 3.0.0 - 2017-11-11

- Added support for Jenkins 2.x
- Added ability to upload to multiple QA·Verify servers.
- Added ability to customize QA·Verify snapshot name based on environment variables.
- Improved static analysis progress information during Job execution.
- Improved error handling of inconsistencies in the PRQA Framework configuration.

Version 2.0.12

- Fixed the wiki page and broken links for download

Version 2.0.11

- QA·Framework 1.0.5 support
- Fixed: Graphs not getting generated (JENKINS-22763)

Version 2.0.11

- QA·Framework 1.0.5 support
- Fixed: Graphs not getting generated (JENKINS-22763)

Version 2.0.9

- QA·Framework 1.0.3 support

Version 1.2.2

- Fixed: Graph should use messages within the current selected threshold (JENKINS-20045)

Version 1.2.1

- Fixed: Graph incorrectly scales to max number of messages (JENKINS-18018)
- Improved the behaviour of the plug-in. New way to add thresholds.

Version 1.2

- Implemented File list as a report source. (JENKINS-17283)
- Tool installations. You can now configure multiple versions of PRQA tools for use. (JENKINS-17282)
- Better threshold handling. (JENKINS-15905)

Version 1.1.3

- Fixed a small issue with missing CMA in upload part.

Version 1.1.2

- Fixed an error introduced in 1.1.1 which caused analysis without CMA option to fail.

Version 1.1.1

- Implemented improved behavior for analysis (JENKINS-15699)

Version 1.1.0

- Enabled QA Verify upload feature (JENKINS-15279)
- Multiple report generation feature added (JENKINS-15280)
- Enhanced build artifacts (JENKINS-15282)
- Enabled various advanced analysis options(JENKINS-15281)

Version 1.0.5

- Fixed an issue with incorrect permissions for logging in JENKINS-14802

Version 1.0.4

- Fixed layout issues with the new publishers as a drop-down list feature in JENKINS-13786

Version 1.0.3

- Fixed Null Pointer Exception reported in JENKINS-13803

Version 1.0.2

- Fixed a critical issue that would cause Jenkins clients to exit abruptly and terminate the underlying JVM: JENKINS-13761
- Fixed various spelling issues in GUI: JENKINS-13777
- Updated Programming Research Icon: JENKINS-13778

Version 1.0.1

- Fixed various cosmetic issues and corrected a few spelling mistakes.
- Fixed logic error in 'Max. Messages' comparison.
- Changed it so that threshold values are now stored on a per-build basis.
- Changed fields to default to 0.
- Removed 'Java' from the list of supported products.
- Made project link graphs slightly larger than before.

Version 1.0

- First official Jenkins release.
- Shows threshold graphs.
- Fixed bug where quick consecutive builds would lock workspace files.