



Functional Safety Manual for Klocwork

Document Number: KW2024_4_001

Certified version: Klocwork 2024.4

Author	Revision	Comments	Date
MTookey	0.1	Initial draft created.	May-09-13
AWeekes	0.2	Broadened scope to include IEC 61508 Added TOC and Related Documents reference table.	May-28-13
MTookey	0.3	Updates for TUV SUD feedback	July-2-13
AWeekes	0.4	Updated for 2015 re-certification	June-15-15
SBommaganti	1.0	Updated for Klocwork 2016 (Build 11.x)	March 3-16
SBommaganti	1.1	Updated build numbers for Klocwork 2016 and Tool Classification	April-4-16
SBommaganti	1.2	Updated Klocwork version to 2016.1	June-25-16
SBommaganti	1.3	Updated Klocwork version to 2016.3	November-2-16
SBommaganti	2.0	Updated Klocwork version to 2017	March-02-17
SBommaganti	2.1	Updated Klocwork version to 2017.1 and added EN 50128 standard	July-05-17
SBommaganti	2.2	Updated Klocwork version to 2017.2	Oct-16-17
SBommaganti	2.3	Updated Klocwork version to 2017.3	Nov-2-17
MTofinetti	3.0	Updated Klocwork version to 2018	2018-04-23
MTofinetti	3.1	Updated Klocwork version to 2018.1	2018-06-19
MTofinetti	3.2	Updated Klocwork version to 2018.2	2018-09-27
MTofinetti	3.3	Updated Klocwork version to 2018.3	2018-11-26

MTofinetti	4.0	Updated Klocwork version to 2019	2019-03-22
MTofinetti	4.1	Updated Klocwork version to 2019.1	2019-07-04
MTofinetti	4.2	Updated Klocwork version to 2019.2	2019-07-25
MTofinetti	4.3	Updated Klocwork version to 2019.3; IEC 62304 added	2019-12-18
LRobertson	5.0	Updated Klocwork version to 2020.1	2020-03-12
ABedford	5.1	Updated Klocwork version to 2020.2	2020-06-29
ABedford	5.2	Updated Klocwork version to 2020.3	2020-09-14
ABedford	5.3	Updated Klocwork version to 2020.4	2021-02-24
ADunster	6.0	Updated Klocwork version to 2021.1	2021-04-26
ADunster	6.1	Updated Klocwork version to 2021.2	2021-08-17
ADunster	6.2	Updated Klocwork version to 2021.3	2021-11-30
ADunster	6.3	Updated Klocwork version to 2021.4	2022-01-18
JBritton	6.4	Updated Klocwork version to 2022.2 and rebrand	2022-05-27
JBritton	6.5	Updated Klocwork version to 2022.4	2022-12-17
JBritton	6.6	Updated for 2022.4 SR1	2023-04-21
JBritton	6.7	Updated for 2023.2	2023-07-26
JBritton	6.8	Updated for 2023.4	2023-12-29
JBritton	6.9	Updated for 2024.2 and add EN 50716	2024-07-24
JBritton	7.0	Updated for 2024.4	2024-12-29

Contents

Scope.....	4
How is functional safety achieved in Klocwork?.....	5
How do you determine if your analysis is within the scope of Functional Safety?	7
System requirements.....	10
Usage restrictions and limitations	15
Operational risk.....	16
Upgrading to the latest version of Klocwork	17
Klocwork Checker Qualification Pack.....	19
Using Klocwork Customer Support	20

Referenced Standards

Standards referenced in this document refer to the following versions:

Standard	Version
ISO 26262	ISO 26262:2018
IEC 61508	IEC 61508:2010
IEC 62304	IEC 62304:2006/AMD1:2015
EN 50716	EN 50716:2023

Trademarks

"MISRA", "MISRA C" and "MISRA C++" are registered trademarks of The MISRA Consortium Limited.

Windows is a registered trademark of Microsoft Corporation.

Related Documents

Document ID	Title
KW2024_4_003	Klocwork ISO 26262 / IEC 61508 / IEC 62304 / EN 50716 Certified Checkers
KW2024_4_005	Klocwork Checker Qualification Pack
KW2024_4_006	Klocwork Architecture
KW2024_4_009	Klocwork 2024.3 and 2024.4 Release Notes

Scope

This document describes the basic requirements, best practices, and acceptable workflows for the use of Klocwork 2024.4 certified checkers (i.e. https://help.klocwork.com/current/en-us/concepts/misrac2012amd2c90c99nolinks_all.htm) in safety related software development applications. This document describes:

- general tool classification
- how functional safety is achieved in Klocwork,
- how to determine if your analysis is within the scope of functional safety,
- system requirements,
- typical use cases and general functional safety recommendations,
- usage restrictions and limitations,
- operational risk,
- upgrading to the latest version of Klocwork

How is functional safety achieved in Klocwork?

To ensure security, Klocwork allows the administrator to control who will have access to the Klocwork configuration. User and administrator rights should be clearly documented and understood. In an environment where functional safety is important, the administrator must control which checkers are enabled and disabled, and which issues can and cannot be ignored.

Issue management

Klocwork uses a role-based permissions model to determine which users can access or modify aspects of the analysis. This is extremely important for issue management since you don't want a developer setting the status of an important issue to "Ignore".

To ensure that all issues are triaged and categorized accordingly, make sure that your key developer roles cannot change an issue's status to anything that would remove the issue from the analysis without proper consideration.

For more information about issues, see [Investigating and citing issues in the integration build](#) in the online documentation.

Checker configuration

Klocwork checker configuration is performed using the Klocwork web Portal. It is the Klocwork administrator's responsibility to create, control, and maintain the applicable list of enabled checkers for your project.

In a connected desktop environment, the checker configuration is maintained by the administrator, used by the system build, and deployed to each desktop in the project. Only the system administrator can modify checker configuration that is used by the system build, therefore the system build will always perform the analysis using the complete, approved list of checkers.

For more information about configuring your analysis, see *Configuring checkers for the integration build analysis*.

Ensure all ISO 26262 / IEC 61508 / IEC 62304 / EN 50716 related checkers are enabled

The ISO 26262 / IEC 61508 / IEC 62304 / EN 50716 related checkers are included in the Klocwork 2024.2 Server Installer package. The specific build number for all supported platforms is listed below.

Tool Component	Build number
Klocwork 2024.4	24.4.0.43

Refer to *Installing the MISRA checker packages* in the online documentation for details on how to install and enable the MISRA checkers.

Ensure that your build is accurate

Part of ensuring the functional safety of your system involves using your own common sense and occurs before you even start using Klocwork to perform your analysis.

When performing your initial integration build analysis, always make sure that your native build completes as expected and that your build settings are captured properly by kwinject. See *Integration Build Analysis* later in this document for more information.

Note that the integration build analysis will always take longer to complete than your regular native build. If your integration build analysis build completes in less time than your native build typically does, you should check your build logs to ensure that the build completed as expected. Conversely, if your build took considerably longer to complete, or did not complete at all, there may be other problems. See *Operational Risk*, later in this document for more information.

Generating additional log files

You must take the appropriate steps to ensure that you are performing a clean build. If your build contains parse errors and/or build integration errors related to missing files, headers, or includes, it's critical to deal with them. An analysis that generates these errors will not be accurate.

When analyzing Functional Safety projects, always use the `--strict` command line option with `kwbuildproject` to turn on the generation of detailed log files to produce additional build information for your project.

Always check these logs for errors, and make sure to fix any errors or warnings that may affect the validity of the analysis.

Also make sure that there are no missing include statements in your code. These will also show up in the build log files. See *How do you determine if your analysis is within the scope of Functional Safety?* later in this document for more information on analyzing the log files.

Organizations that are concerned about functional safety and therefore about code coverage should use manual code inspections to review functions that get skipped during the analysis.

How do you determine if your analysis is within the scope of Functional Safety?

The build logs provide a window into the health of your system build. Make sure that your integration build log is clean and free of warnings or errors.

Build messages

During the analysis, brief messages are available on the console from the command line. These messages only provide basic information such as the name of the build stage (for example, Compiling) and the processed files.

Log files

Detailed information pertaining to build health, and thus functional safety are contained in the Klocwork strict-mode log files. To enable generation of these log files, ensure that the `--strict` command with `kwbuildproject` is used.

Once analysis completes, the following log files are generated:

- Build log (build.log)
- Build integration log (build_integration.log)
- Parse errors log (parse_errors.log)
- Database load log (kwloadadb.log)
- Function information log (function_information.log)

See below for more information about each log files and the information it contains.

Build log (build.log)

The build.log contains details on the build and analysis. This log contains all warnings and errors encountered during the build process.

In Klocwork Portal, you can view the build log for a completed analysis by selecting the build and clicking View Log. The build log is displayed within your browser.

Do not use the `--short-log` option with `kwbuildproject` on functional safety projects, since this option reduces the details provided in the build log.

Build integration log (build_integration.log)

This log file contains integration errors that the `kwbuildproject` command detected during the build process.

Please note that errors will only appear in the `build_integration.log` if your compiler tools are candidates for Klocwork enhanced analysis. Refer to support documentation for compilers that support modern C++.

The syntax for each error in the log file is as follows:

- begins with the characters KW_BI: followed by an eight-digit number
- contains the file path to the error, including the location within the file
- ends with a specific message pertinent to the error

Parse errors log (parse_errors.log)

This log file contains information related to the quality of your analysis.

Parse errors may be caused by unsupported language constructs. If your build contains parse errors, it is important to deal with them, since having parse errors degrades the quality of the analysis.

Database load log (kwloadddb.log)

This file contains log information when the analysis data was loaded into the database.

Function information log (function_information.log)

This log file contains a list of functions that were skipped during analysis. This may occur with particularly long or complex functions, or when functions take too long to analyze.

Log file locations

All generated log files are in the <project-directory>/<tables-directory>:

- Build log (build.log)
- Build integration log (build_integration.log)
- Parse errors log (parse_errors.log)
- Database load log (kwloadddb.log)
- Function information log (function_information.log)

The following log files are also uploaded to the Klocwork Server to

<projects_root>/projects/<project_name>/builds/<build_name>/:

- Build log (build.log)
- Parse errors log (parse_errors.log)
- Database load log (kwloadddb.log)

What to look for in the log files

Failure messages can appear anywhere in the build log. The following example shows an error message taken from the build log of a failed analysis:

```
Error: com.klocwork.database.DatabaseException: Database error occurred: FUNCTION
integration_win_bll_baseline_shared.entity_sequence_nextvals does not exist in SQL
statement 'SELECT entity_sequence_nextvals(1819845) '
```

```
Propagation stage failed
```

```
Error occurred during build: Propagation stage failed. Program exited with 9
```

If the analysis build was successful, you'll see a message at the bottom of the log file. For example:

Build successfully created.

System requirements

Refer to the *System Requirements* in the online documentation for a complete listing of the Klocwork system requirements.

Maintain a clean, stable build environment

The system requirements page provides the minimum set of software and hardware requirements necessary to perform your analysis. Note that unless the analysis fails, the results of the analysis will not be impacted by the speed of the computer or the amount of available memory. It is important however to work in a clean, stable environment to ensure that Klocwork and your native build can execute in a repeatable, predictable manner.

Klocwork: typical use cases and general functional safety recommendations

For functional safety, Klocwork supports the following workflows:

- UC1. Integration build analysis

UC1. Integration build analysis

The analysis of your regular software build is called an integration build analysis. The integration build analysis is run at the beginning of the analysis against your regular software build to give you a snapshot of the current health of your software project. After you run an analysis, detected issues and reports are available in Klocwork Review. The integration build analysis is the first step in establishing the connected desktop in your organization.

An integration build analysis involves four simple steps, all performed by the Klocwork administrator:

1. Create a project.
2. Capture your build settings.
3. Analyze the project using the checkers.
4. Load your results into the database for further analysis.

Once the first integration build analysis is complete:

- developers connect their local projects to the integration project
- the results are available in Klocwork Review

Each step is listed in greater detail in the online documentation. See *Running the C and C++ integration build analysis* for more information.

Once you've run an initial full integration build analysis, users can run an incremental analysis for subsequent builds. Incremental analysis is performed against only those files that have changed since the integration build analysis was performed and on any files with dependencies on those changed files. We recommend using incremental analysis with each incremental build of your source files.

Whenever you do a clean build (full rebuild of your source files), you should run a full build analysis. See online documentation for more information on how to run a full build analysis.

Desktop analysis

While desktop analysis is a valuable tool that allows developers to perform on-the-fly analysis in the IDE of their choice; desktop analysis is less relevant to the notion of functional safety and is therefore outside the scope of this document. This is because checker configuration is maintained at the server level in a secure fashion (in environments where functional safety matters) and because the integration build analysis runs on the entire codebase and represents shipping code that is part of the final configuration.

Continuous Integration (CI) analysis

Klocwork's Continuous Integration (CI) capability allows developers to fit static code analysis into their existing CI workflow. The CI build system performs small incremental builds throughout the day, as and when the developers commit new code. This allows developers to detect new Klocwork issues right away, instead of waiting for the nightly builds. However, like the desktop analysis, while this is a useful feature that allows developers to be more productive, it is less relevant to the notion of functional safety and is therefore outside the scope of this document.

Validating performance, completeness and accuracy

At regular intervals, you should continue to check your log files to ensure that your build is free from warnings and errors. This will help to ensure that your analysis is as accurate as possible.

Klocwork checkers do not modify the code base

It's important to note that at no time during the analysis do the Klocwork checkers modify the source code in any way.

Tool classification

IEC 61508

Klocwork 2024.4 Severity 1 and 2 C/C++ checkers and MISRA checkers are classified as a **T2 offline support tool** in accordance with IEC 61508.

IEC 62304

IEC 62304 provides a framework of life cycle processes for the safe design and maintenance of medical device software.

IEC 62304 requires tools to be “suitably validated” (Table C.3). The tool validation according to IEC 61508 is a main aspect of the testing described in this report. Since IEC 62304 does not define how suitable validation is achieved, but refers to IEC 61508 with respect to tools, the validation can be considered suitable also in the sense of IEC 62304.

IEC 62304 AMD1:2015 does not contain changes with regard to tools.

EN 50716

Klocwork 2024.4 Severity 1 and 2 C/C++ checkers and MISRA checkers are classified as a **T2 support tool** in accordance with EN 50716.

ISO 26262

The tool qualification aspects below are based on UC1: Integration Build Analysis which is described above. Desktop analysis is outside the scope of this document.

Tool impact class (TI)

The Klocwork 2024.4 Severity 1 and 2 C/C++ checkers and MISRA checkers fall into the category of TI 2.

Klocwork checkers do not modify the source code base in any way. As such, Klocwork cannot possibly introduce errors into the source code repository.

No software suite could ever claim to find all defects in a given code base. During the development cycle, Klocwork staff perform a high level of due diligence by consistently running all checkers against a large array of internal and open source code examples to provide analysis results that are as accurate as possible. As part of this effort, Klocwork also provides a test pack that users can download and run against their local Klocwork installation to ensure that their checker deployment is functioning as expected. Klocwork has a high level of confidence that a properly functioning implementation will detect all the defects in the test pack. The test pack contains hundreds of code snippets that are used to seed a thorough analysis using the checkers. See *Klocwork Checker Qualification Pack* later in this document for more information.

Your workflow: achieving a satisfactory level of effectiveness for your code reviews

It is worth mentioning again that no software suite could ever claim to find all defects in a given code base. Furthermore, static code analysis by nature can only detect errors that are, by definition,

detectable by static code analysis. This tool cannot detect logic or design errors.

In addition to performing a static analysis, we recommend following the best practices described throughout this document and reiterated below:

- Always perform clean builds
Always make sure that your native build executes as expected with no warnings or errors.
- Reduce the complexity of your code
Studies show a correlation between a program's Cyclomatic Complexity and its maintainability and testability, implying that with files of higher complexity there is a higher probability of errors when fixing, enhancing, or refactoring source code. By decreasing the complexity of your code base, you can reduce the level of risk and number of errors and improve the efficacy of your static analysis.
- Perform peer code reviews
Performing peer code reviews can help to reduce the number of errors in your source code, thus improving the quality of your product. Peer reviews can also improve the skill of your developers.
- Review your log files
Always review your log files after each analysis. See *Integration build log files* above for more information.
- Don't change your project configuration during the life of a project
Changing your configuration during the development process may alter the results of your analysis.

Tool Error Detection Class (TD)

Steps are taken to accurately capture the results of the analysis in detailed build logs. Should the build fail because of errors in the code base, a detailed failure message is outputted, indicating that the user must fix any errors in the code base before completing a proper analysis.

Should the analysis fail, due to external or environmental issues, no erroneous analysis results will be generated. Any incomplete results cannot be loaded for analysis, and therefore can never be seen by the user. However, since no static code analysis tool can claim to find all defects in a given code base, the tool error detection will also depend on the measures of fault avoidance and error detection on the user side.

Depending on the applied measures of error prevention and error detection in the user development process, i.e. the applied techniques and intensity of validation activities, the resulting tool error detection can vary.

Tool Confidence Level (TCL)

To achieve the best possible TCL value, the user's development process, including complete verification and validation, should be conducted according to ISO 26262.

Usage restrictions and limitations

The Klocwork C/C++ compiler does not parse files compiled with /CLR option

Visual Studio allows you to create a C++ project with files that use Microsoft's managed C++ extensions. The Klocwork C/C++ compiler (kwcc) does not parse files compiled with the /CLR option. It issues a warning that the compiler skipped parsing of these files because of the use of managed extensions. This warning is included in build summary statements that count warnings and errors.

The build specification tool kwinject adds entries for all C++ files, but reports the number of files that will be skipped during an analysis (if any), as well as the total number of files added to the build specification.

Changes made to the project framework during the life of a project

Any changes made to the framework of your project (for example: compilers, build configurations, tooling) after the initial integrated build analysis, wholesale or otherwise, introduces a functional safety risk and should be avoided if possible. If you must make changes to your project's framework, make sure to perform a new build integration analysis, taking care to create a new buildspec.

Once the integration build analysis is complete, be sure to verify the build logs and analyze the results of the analysis. If you can, perform regression testing on the new configuration by comparing the results of the build analysis before and after the configuration changes were made. Be sure to compare the results of the analysis using the same codebase revision immediately before and after configuration changes are made.

Impact of customer modifications (such as disabling checkers, or ignoring issues)

If a checker is disabled, all issues reported by this checker will be removed from the next analysis. If an issue is ignored, all recurrences of the same issue will be removed from the analysis the next time the analysis is performed. This is true unless major changes have been introduced into the codebase in the interim, in which case the compiler must run again to process any changes made to the issue list.

As part of the issue management process, the user can analyze each detected issue and assign it a status to indicate how it should be handled. This process is called "citing".

Note: Users require permission to change an issue's status. Permission can be granted to change from or to any status or only for specific statuses. For example, a group of users may only have permission to change issues in "Analyze" status to "Fix".

Operational risk

What happens if performance degrades?

If the performance level of the host computer begins to decrease because of a memory issue, race condition or otherwise, the Klocwork checkers will continue to function until the analysis completes successfully, fails, or is cancelled by the administrator.

A slow build will not impact the analysis results.

In the event of failure, there is no risk to the codebase. This is because Klocwork does not modify the source code in any way. Furthermore, when the build fails, no analysis is performed and no results are created. A build failure message is added to the build logs.

How are memory concerns or other environmental issues handled?

Memory requirements are listed in the [System Requirements](#) section of the online documentation. Note that large projects consume more memory. Customers are encouraged to contact customer support if they think that too much memory is being consumed, or if their builds are failing because of excessive memory consumption.

What happens if analysis is stopped?

The Klocwork analysis is an all-or-nothing operation, meaning that if a software or hardware error causes the build to fail, no usable output is generated. Instead, messages are written to the log files warning the user that the analysis has failed.

Understanding and monitoring the analysis through logs

Logs provide a detailed list of warnings and error messages indicating that an error occurred or that the analysis build failed. As described earlier, you should continue to monitor your log files throughout the life of your project to ensure that your ongoing analysis is as accurate as possible.

See [Viewing integration build log files](#) in the online documentation for more information about viewing the logs files.

Erroneous output

All Klocwork checkers are capable of falsely identifying a code issue (false positive) or missing an existing code issue (false negative). Both situations are expected and the tuning of analysis creates a balance between thoroughness of defect discovery, and the noise created through excessive false positives. While neither false positives nor false negatives are dangerous in themselves, it is possible that a real code defect could be missed by a Klocwork checker. Customers should be aware of this possibility, and apply multiple strategies, including peer code reviews, to ensure that code defect detection is as thorough as possible.

Klocwork provides recommendations for tuning analysis to provide optimum analysis results. Details are available in the documentation at <https://help.klocwork.com/current/en-us/concepts/tuningccanalysis.htm>

Upgrading to the latest version of Klocwork

Best practices

Note that ISO 26262, IEC 61508, IEC 62304 and EN 50716 certifications only apply to the version of Klocwork that is listed at the beginning of this document.

When upgrading to a new version of Klocwork, you should take care to follow the recommended procedure described in the online documentation. See https://help.klocwork.com/current/en-us/concepts/upgradingfromapreviousversion_sca.htm for detailed instructions on upgrading to the latest version of Klocwork.

Before you begin

Make a copy of your `projects_root` directory before performing the upgrade. This way, users can continue to use Klocwork Review, though they should be instructed not to make any changes, such as changing an issue's status.

If you do not use the default server settings, you will need to record your custom settings prior to beginning the upgrade. Otherwise, during the installation these settings will revert to the default settings. If you forget, you can always go in and change the settings for each of your environments after you have completed upgrading.

To avoid losing issues, status changes or comments from the last release in your first Klocwork 2024.2 analysis run, make sure you read "[Upgrading from a previous version](#)".

Before your first version Klocwork 2024.4 integration build analysis

New releases of Klocwork normally have changes to the checker configuration to keep up with current events and respond to customer requests. These changes may mean that your checker configuration from the previous release isn't the same in the new release.

When you're migrating to a new release, the issues, status changes and comments from an analysis run in the previous release are propagated only during the first run in the new release. If any checkers are disabled by default in the new release, you can lose issues or status changes from the old analysis run, so it's important to run your first analysis with the same configuration as before.

To do this, make sure that you have the right checkers enabled to match your old configuration. After you're satisfied with your configuration, perform your first Klocwork 2024.2 integration build analysis on unmodified source code.

Note: If you've already run your first Klocwork 2024.4 analysis and you're missing some issues or status changes, delete that build, reconfigure your checkers, and run a new analysis.

We recommend running your final pre-upgrade integration build analysis and your first Klocwork 2024.4 analysis on identical source code, and then comparing the two builds. This allows you to assess changes in the analysis engine.

Regression testing

Before a new version of Klocwork is released, regression testing is performed on a known set of sample projects to ensure that the analysis results are consistent with those produced by a previous version of the product.

As a customer, it is your responsibility to perform the same regression test using your own codebase. Following the steps above and ensuring that the analysis results are consistent across deployments will help to ensure the functional safety of your installation.

Klocwork Checker Qualification Pack

Klocwork provides a qualification pack for customers wishing to ensure the integrity of their deployment.

The test procedures check the requirements under normal operating conditions. Each procedure involves executing the tool with provided input data to generate a validated pass/fail report for each checker. The tool is deterministic in its execution and will consistently generate the same output results for a given set of input data parameters.

The process of verifying the tool will use a series of test cases, each designed to check one or more operational requirements. In each test case, a set of prepared inputs is fed to the tool and the tool's output is compared against a canonical output file. The canonical file contains the output the tool must produce when presented with the prepared inputs to demonstrate it correctly satisfies its requirements.

For more information about the qualification pack and for instructions that describe how to run the test suite, see the *Klocwork Checker Qualification Pack* (KW2024_4_005).

Using Klocwork Customer Support

Support policies

Klocwork's support and maintenance policies are detailed in the *Klocwork Support and Maintenance Terms and Conditions*, provided to every customer. Unless otherwise agreed to in writing between Perforce ("Klocwork") and the customer, these Support and Maintenance Terms and Conditions shall apply to the provision of support by Klocwork for supported versions of its products.

Contact Support

- by telephone on +1 (612) 361-5646
- email supportklocwork@perforce.com
- via the Perforce website <https://www.perforce.com/support/request-support>

Release notes

Release notes, describing software limitations and known issues, are available in the *Klocwork 2024.4 Release Notes* document KW2024_4_009 and in the online documentation.



This document, as well as the software described in it, is furnished under license and may only be used or copied in accordance with the terms of such license. The information contained herein is the exclusive property of RogueWave Software, Inc. a Perforce company. No part of this documentation may be copied, translated, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of Perforce Software, Inc. If you find any problems in the documentation, please report them to us in writing.

Klocwork is a registered trademark of RogueWave Software, Inc., a Perforce company.

All other trademarks are the property of their respective owners. All help content for Klocwork's MISRA checkers is copyright by the MISRA Consortium Limited