

Kudelski Trust Agent

KTA Code Check

Kudelski IoT Security

CC Restricted - Kudelski Confidential

15.08.2024

Copyright notice

This document is the property of KUDELSKI Sàrl and contains confidential information that is subject to the terms and conditions of the Non-Disclosure Agreement signed between the recipient of this document and NAGRAVISION or Kudelski Group.This information may not be disclosed in any form, in whole or in part, without the prior written consent of KUDELSKI Sàrl.

DOCUMENT PROPERTIES

|  |  |
| --- | --- |
| Version | 0.0.1 |
| File name | KTA CODE CHECK.DOCX |
| Publication date | 15.08.2024 |
| Confidentiality Status | CC Restricted - Kudelski Confidential |
| Document owner | Kudelski Group |
| Document recipient |  |
| Document status | Draft |
| Client company name | KSIOT |

Table of Contents

Table of Contents 4

Table of Figures 6

Table of Tables 7

1. Executive Summary 8

2. Methodology 9

2.1. Executing the Code Check 9

3. Coding Rules 11

3.1. Cppcheck Options 11

3.2. Cppcheck Result 11

4. Static Code Analysis 12

4.1. Cppcheck Options 12

4.2. Cppcheck Result 12

5. MISRA Directives 13

Amendment 1 directive 13

5.1. Deviations from "Required" directives 13

6. MISRA C:2012 Compliance 15

6.1. Cppcheck Compliance 15

6.2. Cppcheck Options 18

6.3. Deviations from "Required" rules 19

6.4. Non-compliances with "Advisory" rules 19

6.5. Related deviation permit 21

6.6. Deviation Records 22

6.6.1. MISRA\_DEV\_KTA\_002 23

6.6.2. MISRA\_DEV\_KTA\_003 25

6.6.3. MISRA\_DEV\_KTA\_004 27

6.6.4. MISRA\_DEV\_KTA\_005 29

6.6.5. MISRA\_DEV\_KTA\_006 32

6.6.6. MISRA\_DEV\_KTA\_007 34

6.6.7. MISRA\_DEV\_KTA\_008 36

6.6.8. MISRA\_DEV\_KTA\_009 38

6.6.9. MISRA\_DEV\_KTA\_010 40

6.6.10. MISRA\_DEV\_KTA\_011 42

6.6.11. MISRA\_DEV\_KTA\_012 44

6.6.12. MISRA\_DEV\_KTA\_013 46

6.6.13. MISRA\_DEV\_KTA\_014 48

6.6.14. MISRA\_DEV\_KTA\_015 50

6.6.15. MISRA\_DEV\_KTA\_016 52

6.6.16. MISRA\_DEV\_KTA\_017 54

7. CERT 2016 Guidelines Deviations & Non-Compliances 56

7.1. Cppcheck Options 56

7.2. Non-compliances with Recommendations 56

8. C Compilers 57

8.1. MSVC Compiler 57

8.2. Cygwin GCC 57

Document History 58

Document Recipients 59

Kudelski IoT Contact 60

References 61

Acronyms 62

Appendices 63

Table of Figures

**No table of figures entries found.**

Table of Tables

**No table of figures entries found.**

# Executive Summary

This document contains the code analysis report of the Kudelski Trust Agent (KTA), specified in document [[R1](#R1)].

It describes the static code analysis methodology and describes how the various code checks are performed regarding:

* Coding rules compliance
* Static code analysis
* MISRA C:2012 compliance

# Methodology

The code static analysis is performed with the Cppcheck v2.10 static analysis tool. Cppcheck is an open-source static code analysis tool for C/C++ code (see [R5]).

The Cppcheck tool is run four times with different parameters to:

* Enforce Kudelski coding rules
* Perform static code checks (pointers, overflows…)
* Ensure compliance with MISRA C:2012 rules, MISRA C:2012 Amendment 1 & MISRA C:2012 Amendment 2
* Ensure compliance with CERT rules

The following Cppcheck options are common to all configurations:

--platform=win32A

--std=c99

--enable=warning,style,performance,portability

--template=gcc

--inconclusive --inline-suppr

--suppress=missingIncludeSystem

--suppress=redundantAssignment

--suppress=variableScope

--suppress= unreadVariable

--suppress=misra-c2012-21.6

--suppress=misra-c2012-17.7

--suppress=misra-c2012-17.1

--suppress=misra-c2012-15.4

--suppress=misra-c2012-15.1

--cppcheck-build-dir=./MisraReports

--addon=./DEV/TOOLS/PROJECT/MISRACHECK/misra.json

--platform=native

--suppress=\*:\*/EMU/\*

**Generic suppressions:**

**missingIncludeSystem** is suppressed as we assume includes are available.

**redundantAssignment** is suppressed due to security feature. Returned status is always initialized with an error value.

**variableScope** is suppressed as all local variables are declared in the beginning of the functions.

**unreadVariable**

The code static check is also run using the Sonarqube static analysis tool. The Sonarqube tool comprises of two parts –

* Build wrapper – This tool is used to compile the project as per sonarqube specification and create a json file which contains the information of compiled files for analysis.
* Sonar Scanner – This is a CLI tool to run the static check analysis and upload the details to the sonarqube server (hosted by organization).

## Executing the Code Check

The full code check is started by executing the following command in the /DEV/SOURCE/kta and in the /DEV/SOURCE/salapi folder.:

Each check can be started individually:

* MISRA checks

cppcheck -D <kta\_feature\_flag> -D <pre\_processor\_symbols> -I <header\_files> --<supressions> "list\_of\_c\_files"

Note that certcheck shall be executed only on computer having access to the CppCheck premium license.

# Coding Rules

In order to enforce the coding rules defined in [R2], Cppcheck is run to generate the source code *Abstract Syntax Tree* (AST). The AST is then processed by an in-house Python script (codecheck.py) which verifies that the coding rules are followed.

## Cppcheck Options

Cppcheck is run with the default options listed in §2.

**Note**: the KTA header files are not part of the verification as they were delivered to the customer before the coding rules were enforced. Updating the header files would change the API and impact the customer.

## Cppcheck Result

No error.

# Static Code Analysis

On its own, Cppcheck analyses the source code and looks for undefined behavior that may lead to bugs or vulnerabilities, such as:

* Dead pointers
* Division by zero
* Integer overflows
* Invalid bit shift operands
* Invalid conversions
* Invalid usage of STL
* Memory management
* Null pointer dereferences
* Out of bounds checking
* Uninitialized variables
* Writing const data

## Cppcheck Options

Cppcheck is run with the default options listed in §2.

## Cppcheck Result

No error.

# MISRA Directives

| Directive | Category | Status | Comment |
| --- | --- | --- | --- |
| 1.1 | Required | Compliant | KTA makes no use of standard library, no assembly code, no use of division nor floating point operation |
| 2.1 | Required | Deviation | KTA code is compiled without errors and with as few warning as possible  See C Compilers chapter |
| 3.1 | Required | Compliant | Code is fully traceable to documented requirements |
| 4.1 | Required | Compliant | MISRA Rule 1.3 is verified |
| 4.2 | Required | Compliant | No use of assembly code |
| 4.3 | Required | Compliant | No use of assembly code |
| 4.4 | Required | Compliant | Use of #ifdef instead of commented code |
| 4.5 | Advisory | Disapplied | Identifiers follow the project’s coding rules |
| 4.6 | Advisory | Compliant | Code uses <stdint.h> types. Size\_t is used in interface |
| 4.7 | Required | Deviation | Return codes are always tested |
| 4.8 | Advisory | Compliant | MISRA Rule 17.7 is verified |
| 4.9 | Advisory | Disapplied | Function-like-macros are present only in logging code. This code is not present in final product |
| 4.10 | Required | Compliant | .h files format follow the coding rules |
| 4.11 | Required | Compliant | No library is used except for logging that is not present in final product |
| 4.12 | Required | Compliant | No dynamic memory allocation |
| 4.13 | Advisory | Compliant | No access to resources |

Amendment 1 directive:

| Directive | Category | Status | Comment |
| --- | --- | --- | --- |
| 4.14 | Required | Compliant | Inputs are checked before being used |

## Deviations from "Required" directives

The KTA implementation deviates from the following MISRA C 2012 "Required" directives.

| MISRA Directive # | Synopsis | Category |
| --- | --- | --- |
| 2.1 | All source files shall compile without any compilation errors | project |
| 4.7 | If a function returns error information, then that error information shall be tested | project |

For directive 2.1, please refer to §8 for C compilers options.

Deviation of directive 4.7 is accepted as it implements a security best practice. This practice consists in wiping sensitive buffers in case of error. As it is an error management, this kind of code cannot return an error and shall silently ignore wiping errors.

# MISRA C:2012 Compliance

This chapter identifies the MISRA C 2012 (see [R3]) non-compliances ("Advisory" rules) and deviations ("Required" rules).

The non-compliance/deviation category is:

* "Project" when the rule is disabled for the whole project
* “file” when the rule is disabled on a specific source file
* "Specific" when the rule is disabled on a case-by-case basis in source files

## Cppcheck Compliance

Cppcheck is able to check the following MISRA C:2012 guidelines.

| Guideline | Cppcheck | Category | Status |
| --- | --- | --- | --- |
| 1.1 |  | Required | Compliant |
| 1.2 |  | Advisory | Compliant |
| 1.3 | X | Required | Compliant |
| 2.1 | X | Required | Compliant |
| 2.2 | X | Required | Compliant |
| 2.3 | X | Advisory | Compliant |
| 2.4 | X | Advisory | Compliant |
| 2.5 | X | Advisory | Compliant |
| 2.6 | X | Advisory | Compliant |
| 2.7 | X | Advisory | Compliant |
| 3.1 | X | Required | Compliant |
| 3.2 | X | Required | Compliant |
| 4.1 | X | Required | Compliant |
| 4.2 | X | Advisory | Compliant |
| 5.1 | X | Required | Compliant |
| 5.2 | X | Required | Compliant |
| 5.3 | X | Required | Compliant |
| 5.4 | X | Required | Compliant |
| 5.5 | X | Required | Compliant |
| 5.6 | X | Required | Compliant |
| 5.7 | X | Required | Compliant |
| 5.8 | X | Required | Compliant |
| 5.9 | X | Advisory | Compliant |
| 6.1 | X | Required | Compliant |
| 6.2 | X | Required | Compliant |
| 7.1 | X | Required | Compliant |
| 7.2 | X | Required | Compliant |
| 7.3 | X | Required | Compliant |
| 7.4 | X | Required | Compliant |
| 8.1 | X | Required | Compliant |
| 8.2 | X | Required | Compliant |
| 8.3 | X | Required | Compliant |
| 8.4 | X | Required | Compliant |
| 8.5 | X | Required | Compliant |
| 8.6 | X | Required | Compliant |
| 8.7 | X | Advisory | Compliant |
| 8.8 | X | Required | Compliant |
| 8.9 | X | Advisory | Compliant |
| 8.10 | X | Required | Compliant |
| 8.11 | X | Advisory | Compliant |
| 8.12 | X | Required | Compliant |
| 8.13 | X | Advisory | Compliant |
| 8.14 | X | Required | Compliant |
| 9.1 | X | Mandatory | Compliant |
| 9.2 | X | Required | Compliant |
| 9.3 | X | Required | Compliant |
| 9.4 | X | Required | Compliant |
| 9.5 | X | Required | Compliant |
| 10.1 | X | Required | Compliant |
| 10.2 | X | Required | Compliant |
| 10.3 | X | Required | Compliant |
| 10.4 | X | Required | Compliant |
| 10.5 | X | Advisory | Compliant |
| 10.6 | X | Required | Compliant |
| 10.7 | X | Required | Compliant |
| 10.8 | X | Required | Compliant |
| 11.1 | X | Required | Compliant |
| 11.2 | X | Required | Compliant |
| 11.3 | X | Required | Compliant |
| 11.4 | X | Advisory | Compliant |
| 11.5 | X | Advisory | Compliant |
| 11.6 | X | Required | Compliant |
| 11.7 | X | Required | Compliant |
| 11.8 | X | Required | Compliant |
| 11.9 | X | Required | Compliant |
| 12.1 | X | Advisory | Compliant |
| 12.2 | X | Required | Compliant |
| 12.3 | X | Advisory | Compliant |
| 12.4 | X | Advisory | Compliant |
| 13.1 | X | Required | Compliant |
| 13.2 | X | Required | Compliant |
| 13.3 | X | Advisory | Compliant |
| 13.4 | X | Advisory | Compliant |
| 13.5 | X | Required | Compliant |
| 13.6 | X | Mandatory | Compliant |
| 14.1 | X | Required | Compliant |
| 14.2 | X | Required | Compliant |
| 14.3 | X | Required | Compliant |
| 14.4 | X | Required | Compliant |
| 15.1 | X | Advisory | Deviation |
| 15.2 | X | Required | Compliant |
| 15.3 | X | Required | Compliant |
| 15.4 | X | Advisory | Deviation |
| 15.5 | X | Advisory | Compliant |
| 15.6 | X | Required | Compliant |
| 15.7 | X | Required | Compliant |
| 16.1 | X | Required | Compliant |
| 16.2 | X | Required | Compliant |
| 16.3 | X | Required | Compliant |
| 16.4 | X | Required | Compliant |
| 16.5 | X | Required | Compliant |
| 16.6 | X | Required | Compliant |
| 16.7 | X | Required | Compliant |
| 17.1 | X | Required | Deviation |
| 17.2 | X | Required | Compliant |
| 17.3 |  | Mandatory | Compliant |
| 17.4 | X | Mandatory | Compliant |
| 17.5 | X | Advisory | Compliant |
| 17.6 | X | Mandatory | Compliant |
| 17.7 | X | Required | Deviation |
| 17.8 | X | Advisory | Compliant |
| 18.1 | X | Required | Compliant |
| 18.2 | X | Required | Compliant |
| 18.3 | X | Required | Compliant |
| 18.4 | X | Advisory | Compliant |
| 18.5 | X | Advisory | Compliant |
| 18.6 | X | Required | Compliant |
| 18.7 | X | Required | Compliant |
| 18.8 | X | Required | Compliant |
| 19.1 | X | Mandatory | Compliant |
| 19.2 | X | Advisory | Compliant |
| 20.1 | X | Advisory | Compliant |
| 20.2 | X | Required | Compliant |
| 20.3 | X | Required | Compliant |
| 20.4 | X | Required | Compliant |
| 20.5 | X | Advisory | Compliant |
| 20.6 | X | Required | Compliant |
| 20.7 | X | Required | Compliant |
| 20.8 | X | Required | Compliant |
| 20.9 | X | Required | Compliant |
| 20.10 | X | Advisory | Compliant |
| 20.11 | X | Required | Compliant |
| 20.12 | X | Required | Compliant |
| 20.13 | X | Required | Compliant |
| 20.14 | X | Required | Compliant |
| 21.1 | X | Required | Compliant |
| 21.2 | X | Required | Compliant |
| 21.3 | X | Required | Compliant |
| 21.4 | X | Required | Compliant |
| 21.5 | X | Required | Compliant |
| 21.6 | X | Required | Deviation |
| 21.7 | X | Required | Compliant |
| 21.8 | X | Required | Compliant |
| 21.9 | X | Required | Compliant |
| 21.10 | X | Required | Deviation |
| 21.11 | X | Required | Compliant |
| 21.12 | X | Advisory | Compliant |
| 22.1 | X | Required | Compliant |
| 22.2 | X | Mandatory | Compliant |
| 22.3 | X | Required | Compliant |
| 22.4 | X | Mandatory | Compliant |
| 22.5 | X | Mandatory | Compliant |
| 22.6 | X | Mandatory | Compliant |

**Amendment 2 Guideline:**

| Guideline | Cppcheck | Category | Status |
| --- | --- | --- | --- |
| 21.21 | X | Required | Compliant |

**Amendment 1 Guidelines:**

| Guideline | Cppcheck | Category | Status |
| --- | --- | --- | --- |
| 12.5 | X | Mandatory | Compliant |
| 21.13 | X | Mandatory | Compliant |
| 21.14 | X | Required | Compliant |
| 21.15 | X | Required | Compliant |
| 21.16 | X | Required | Compliant |
| 21.17 | X | Mandatory | Compliant |
| 21.18 | X | Mandatory | Compliant |
| 21.19 | X | Mandatory | Compliant |
| 21.20 | X | Mandatory | Compliant |
| 22.7 | X | Required | Compliant |
| 22.8 | X | Required | Compliant |
| 22.9 | X | Required | Compliant |
| 22.10 | X | Required | Compliant |

Code is compiled with both GCC (version 11.3) and MSVC (v142x86) in order to verify rules 1.1 & 1.2.

Rule 1.1: *The program shall not contain no violations of the standard C syntax and constraints and shall not exceed the implementation’s translation limits*

Rule 1.2: *Language extensions should not be used*

## Cppcheck Options

MISRA C:2012 rules are enforced with Cppcheck using the following options, in addition to those listed in §2:

--suppress=misra-c2012-21.6

--suppress=misra-c2012-17.7

--suppress=misra-c2012-17.1

--suppress=misra-c2012-15.4

--suppress=misra-c2012-15.1

--addon=DEV/TOOLS/PROJECT/MISRACHECK/misra.json

misra.json

{

"script": "misra.py",

"args": [

"--rule-texts=../../../TOOLS/PROJECT/MISRA\_C2012\_Rules.txt"

]

}

## Deviations from "Required" rules

The KTA implementation deviates from the following MISRA C 2012 "Required" rules.

| MISRA Rule # | Synopsis | Category |
| --- | --- | --- |
| 17.1 | The standard header file shall not be used. | Project |
| 17.7 | The value returned by a function having non-void return type shall be used. | Project |
| 21.6 | The Standard Library input/output functions shall not be used. | Project |
| 21.10 | The Standard Library time and date functions shall not be used. | Log |

See the corresponding deviation records in §6.6

## Non-compliances with "Advisory" rules

The KTA implementation does not comply with the following MISRA C 2012 "Advisory" rules.

| MISRA Rule # | Synopsis | Category |
| --- | --- | --- |
| 15.1 | The goto statement should not be used. | Project |
| 15.4 | There should be no more than one break or goto statement used to terminate any iteration statement. | Project |

As no deviation process is required for "advisory" rules, they are just listed here.

Related deviation permit

**R-17.1\_01**:

Rule 17.1 The features of <stdarg.h> shall not be used

Permit / KSIOT / C:2012 / R-17.1\_01

Exceptional use of stdargs is accepted

Reason Performance efficiency (Resource utilization, small code footprint)

Background

As code footprint can be critical in some projects, the feature can be used.

Requirements

- All instance of code using stdarg shall be identified in documentation

- Maximum (minimum) number of parameters shall be explicitly tested before parameters being used.

- Parameters type (or their value ranges) shall be checked before these parameters being used

Deviation Records

MISRA\_DEV\_KTA\_001

|  |  |
| --- | --- |
| **Project** | KTA |
| **Deviation ID** | MISRA\_DEV\_KTA\_001 |
| **MISRA C Reference** | 17.1 – The standard header file shall not be used. |
| **Permit** | N/A |
| **Status** | Analysis |
| **Reason** | Usability (Accessibility) |
| **Tracing tags** | MISRA\_DEV\_KTA\_001 |
| **Source** | KTA Library |
| **Scope** | Project |

#### Summary

The purpose of this rule is to avoid the use of standard header files that might introduce code that is non-compliant with the MISRA guidelines. Standard header files typically define standard library functions and macros which might not be fully compliant with MISRA's rules, leading to unsafe or undefined behavior

This means that instead of using standard headers like <stdio.h>, <stdlib.h>, <string.h>, etc.,

#### Detailed Description

The KTA defines *C\_KTA\_APP\_\_LOG* to print the debug and error logs. The define is using the stdio function printf.

#define C\_KTA\_APP\_\_LOG printf

#### Justification

As KTA uses logging for errors, it uses the stdio functionality to print the logs to the terminal.

#### Conditions under which the deviation is requested

This deviation is applied overall KTA source code.

#### Consequences on non-compliance

#### Actions to control reporting

In *misraCheck.sh*, the following Cppcheck command line parameter is used to suppress the error in the whole project:

MISRA\_OPT+="--suppress=misra-c2012-17.1 "

MISRA\_DEV\_KTA\_002

|  |  |
| --- | --- |
| **Project** | KTA |
| **Deviation ID** | MISRA\_DEV\_KTA\_002 |
| **MISRA C Reference** | 17.7 – The value returned by a function having non-void return type shall be used. |
| **Permit** | N/A |
| **Status** | Analysis |
| **Source** | KTA Library |
| **Reason** | Usability (Accessibility) |
| **Tracing tags** | MISRA\_DEV\_KTA\_002 |
| **Scope** | Project |

#### Summary

This rule is designed to ensure that the return value of a function with a non-void return type is always used. Ignoring the return value of such functions can lead to unintended behavior, bugs, or missed errors, particularly in embedded systems where reliability is critical.

#### Detailed Description

The KTA uses functions like snprintf, memcpy, memset without using the return values.

These functions are non-void functions.

snprintf(aBuffer, C\_MAX\_BUFFER\_SIZE, "%s : [%d]\r\n", xpFmt, xSize);

#### Justification

The KTA uses the function snprintf to compose the logs throughout the project. The usage of this functionality is purely for logging purpose and pose no harm on the production and release variant.

#### Conditions under which the deviation is requested

This deviation is applied on the whole KTA source code.

#### Consequences on non-compliance

#### Actions to control reporting

In *misraCheck.sh*, the following Cppcheck command line parameter is used to suppress the error in the whole project:

MISRA\_OPT+="--suppress=misra-c2012-17.7 "

MISRA\_DEV\_KTA\_003

|  |  |
| --- | --- |
| **Project** | KTA |
| **Deviation ID** | MISRA\_DEV\_KTA\_003 |
| **MISRA C Reference** | 21.6 – The Standard Library input/output functions shall not be used. |
| **Permit** | N/A |
| **Status** | Analysis |
| **Source** | KTA Library |
| **Reason** | Usability (Accessibility) |
| **Tracing tags** | MISRA\_DEV\_KTA\_003 |
| **Scope** | Project |

#### Summary

The project needs to include standard io headers

#### Detailed Description

In this project for example k\_sal\_log.c there is a usage of the function salPrint, which calls the stdio function printf for displaying debug and error logs.

One of the example is mentioned below.

void salPrint(char\* buffer)

{

    printf("%s", buffer);

}

#### Justification

The project uses logging mechanism to display logs for certain cases. These logging mechanism uses printf functionality of the stdio.h library.

#### Conditions under which the deviation is requested

This deviation is applied on the whole KTA source code.

#### Consequences on non-compliance

There are no consequences as there is no risk of loss of precision.

#### Actions to control reporting

In *misraCheck.sh*, the following Cppcheck command line parameter is used to suppress the error in the whole project:

MISRA\_OPT+="--suppress=misra-c2012-21.6 "

MISRA\_DEV\_KTA\_004

|  |  |
| --- | --- |
| **Project** | KTA |
| **Deviation ID** | MISRA\_DEV\_KTA\_004 |
| **MISRA C Reference** | 15.1 – The goto statement should not be used. |
| **Permit** | N/A |
| **Status** | Analysis |
| **Source** | KTA Library |
| **Reason** | Usability (Accessibility) |
| **Tracing tags** | MISRA\_DEV\_KTA\_004 |
| **Scope** | Project |

#### Summary

MISRA Rule 15.1 prohibits the usage of goto and jump statements.

#### Detailed Description

In the project there are usage of goto statements to reduce the number of nesting functions by using multiple if…else statements and also to prevent the usage of multiple return statements in case of null check errors and similar errors in a function.

#### Justification

Removing the goto statements will lead to usage of multiple return or increase the nesting steps in a function, where there is a need to return midway through the function due to some error occurrence.

#### Conditions under which the deviation is requested

This deviation is applied on the whole KTA source code.

#### Consequences on non-compliance

There are no consequences as there is no risk of loss of precision. Also there is no risk to its success or integrity.

#### Actions to control reporting

In *misraCheck.sh*, the following Cppcheck command line parameter is used to suppress the error in the whole project:

MISRA\_OPT+="--suppress=misra-c2012-15.1 "

MISRA\_DEV\_KTA\_005

|  |  |
| --- | --- |
| **Project** | KTA |
| **Deviation ID** | MISRA\_DEV\_KTA\_005 |
| **MISRA C Reference** | 15.4 – There should be no more than one break or goto statement used to terminate any iteration statement. |
| **Permit** | N/A |
| **Status** | Analysis |
| **Source** | KTA Library |
| **Reason** | Usability (Accessibility) |
| **Tracing tags** | MISRA\_DEV\_KTA\_005 |
| **Scope** | Project |

#### Summary

MISRA Rule 15.4 prohibits the usage of multiple goto and jump statements.

#### Detailed Description

In the project there are usage of goto statements to reduce the number of nesting functions by using multiple if…else statements and also to prevent the usage of multiple return statements in case of null check errors and similar errors in a function.

#### Justification

Removing the goto statements will lead to usage of multiple return or increase the nesting steps in a function, where there is a need to return midway through the function due to some error occurrence.

#### Conditions under which the deviation is requested

This deviation is applied on the whole KTA source code.

#### Consequences on non-compliance

There are no consequences as there is no risk of loss of precision. Also there is no risk to its success or integrity.

#### Actions to control reporting

In *misraCheck.sh*, the following Cppcheck command line parameter is used to suppress the error in the whole project:

MISRA\_OPT+="--suppress=misra-c2012-15.4 "

### MISRA\_DEV\_KTA\_006

|  |  |
| --- | --- |
| **Project** | KTA |
| **Deviation ID** | MISRA\_DEV\_KTA\_006 |
| **MISRA C Reference** | 21.10 – The Standard Library time and date functions shall not be used. |
| **Permit** | N/A |
| **Status** | Analysis |
| **Source** | KTA Library |
| **Reason** | Usability (Accessibility) |
| **Tracing tags** | MISRA\_DEV\_KTA\_006 |
| **Scope** | Project |

#### Summary

MISRA Rule 15.4 prohibits the usage of time and date library

#### Detailed Description

In the project there are usage of time and date library for capturing the log time.

#### Justification

Removing the timestamp for logging will reduce the efficiency of the logs

#### Conditions under which the deviation is requested

This deviation is applied on the whole KTA source code.

#### Consequences on non-compliance

There are no consequences as there is no risk of loss of precision. Also there is no risk to its success or integrity.

#### Actions to control reporting

In *misraCheck.sh*, the following Cppcheck command line parameter is used to suppress the error in the whole project:

MISRA\_OPT+="--suppress=misra-c2012-21.10 "

# C Compilers

## MPLABx GNUWin32 C Compiler

KTA Code is compiled with MPLAB v6.20 GNUWin32 C compiler.

## Cygwin GCC

KTA Code has been compiled with GCC version 11.3. The following relevant options have been used:

* -std=c99 (use C99 standard)
* -Wall (enable all warnings)
* -Wextra (enable extra warnings)
* -Werror (treat warning as error)
* -Wno-error=sign-compare

This compilation generates one warning “comparison of promoted bitwise complement of an unsigned value with unsigned”. This is due to a GCC bug (see GCC Bugzilla #38341).

Document History

|  |  |  |
| --- | --- | --- |
| Version  Status | Date | Author / Changes |
| 0.0.1  Draft | 2024-05-29 | KTA Team Leader |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Reviewer | Position | Date | Document Version |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Approver | Position | Date | Document Version |
| KTA Team Leader | Team leader | 2024-05-29 | 0.0.1 |

Document Recipients

|  |  |  |
| --- | --- | --- |
| Name | Position | Contact Info |
|  |  |  |

Kudelski IoT Contact

|  |  |  |
| --- | --- | --- |
| Name | Position | Contact Info |
| www.kudelski-iot.com |  |  |

References

1. MISRA C:2012 Guidelines for the use of the C language in critical systems – March 2013

1. [SEI CERT C Coding Standard](https://wiki.sei.cmu.edu/confluence/display/c/SEI+CERT+C+Coding+Standard) – 2016 Edition
2. Cppcheck static analysis tool - <https://cppcheck.sourceforge.io/>
3. MISRA C:2012 Amendment 1 – April 2016
4. MISRA C:2012 Amendment 2 – February 2020

Acronyms

| Acronym | Stands For |
| --- | --- |
| KTA | Kudelski Trust Agent for KSE5 |

Appendices