

# Updates in 10.3.3

In this release, we've focused on support for new compilers and extended our support for coding standards, including MISRA 2008.

## Support for New Compilers

- Microsoft Visual C++ 14.1x/ 2017
- WindRiver GCC 4.8.x (Static Analysis only)
- IAR Compiler for ARM 8.11.x

## Enhanced Unit Testing

We've added support for using enumerator values in data sources; see [Handling Enum Values](#) and [Execution Tab Settings - Defining How Tests are Executed](#).

## Enhancements to Static Analysis

In this release, we've added new static analysis rules to cover the MISRA C++ 2008; see [New Code Analysis Rules](#).

The severity levels for MISRA C++ 2008 rules have been updated:

- Required Severity 2
- Advisory Severity 4
- Document Severity 5

Flow Analysis has been extended with support for C++11 Thread API (for BD-TRS rules) and new configuration options that allow you to:

- disable terminators / functions with the `noreturn` attribute
- parameterize the BD-PB-VOVR rule with the "Report when there is at least one path where the value of the variable is not used" option; see the rule documentation for details.
- disable reporting violations whose paths pass via inline assembly code
- enable or disable all tainted data sources in the rule parameters

The RuleWizard Module has been extended with the following nodes and properties:

- `IsFinal` - returns true if function or class was declared with 'final' or 'sealed' specifier
- `IsExplicitFinal` - returns true if function or class was declared with 'final' specifier
- `IsSealed` - returns true if function or class was declared with 'sealed' specifier

See RuleWizard 10.3.3 User's Guide for more details.

## New Code Analysis Rules

| Rule ID            | Header   |
|--------------------|--|
| BD-PB-PTRARR       | A pointer operand and any pointer resulting from pointer arithmetic using that operand shall both address elements of the same array   |
| BD-TRS-CMF         | Make const member functions thread-safe  |
| CODSTA-CPP-90      | Using-directives and using-declarations (excluding class scope or function scope using-declarations) shall not be used in header files   |
| CODSTA-CPP-91      | The overloaded binary operator should be implemented in terms of its corresponding compound assignment operator  |
| CODSTA-CPP-92      | All accessible entity names within a multiple inheritance hierarchy should be unique   |
| GLOBAL-COMPATDECLS | All declarations of an object or function shall have compatible types  |
| GLOBAL-EXCSPECDECL | If a function is declared with an exception-specification, then all declarations of the same function (in other translation units) shall be declared with the same set of type-ids |
| GLOBAL-ONEDEFRULE  | The One Definition Rule shall not be violated  |
| GLOBAL-ONEFILEDECL | A type, object or function that is used in multiple translation units shall be declared in one and only one file   |
| GLOBAL-ONEUSEVAR   | A project shall not contain non-volatile POD variables having only one use   |

|                        |   |
|------------------------|---|
| GLOBAL-TEMPLNOINST     | All class templates, function templates, class template member functions and class template static members shall be instantiated at least one |
| GLOBAL-UNIQUETYPEDEF   | A typedef name (including qualification, if any) shall be a unique identifier   |
| GLOBAL-UNUSEDTYPE      | A project shall not contain unused type declarations  |
| GLOBAL-UNUSEDVIRTPARAM | There shall be no unused parameters (named or unnamed) in the set of parameters for a virtual function and all the functions that override it |
| GLOBAL-VIRTBASECLASS   | A base class shall only be declared virtual if it is used in a diamond hierarchy  |
| MISRA2008-0_1_12       | There shall be no unused parameters (named or unnamed) in the set of parameters for a virtual function and all the functions that override it |
| MISRA2008-0_1_4        | A project shall not contain non-volatile POD variables having only one use  |
| MISRA2008-0_1_5        | A project shall not contain unused type declarations  |
| MISRA2008-0_1_6        | Avoid unused values   |
| MISRA2008-0_1_9:       | All non-null statements shall either have at least one side-effect however executed or cause control flow to change                           |
| MISRA2008-0_3_1_a      | Avoid accessing arrays out of bounds  |
| MISRA2008-0_3_1_b      | Avoid null pointer dereferencing  |
| MISRA2008-0_3_1_c      | Avoid division by zero  |
| MISRA2008-0_3_1_d      | Avoid buffer overflow due to defining incorrect format limits   |
| MISRA2008-0_3_1_e      | Avoid overflow due to reading a not zero terminated string  |
| MISRA2008-0_3_1_f      | Do not check for null after dereferencing   |
| MISRA2008-0_3_1_g      | Avoid overflow when reading from a buffer   |
| MISRA2008-0_3_1_h      | Avoid overflow when writing to a buffer   |
| MISRA2008-0_3_1_i      | Pointer arithmetic shall only be applied to pointers that address an array or array element   |
| MISRA2008-0_3_1_j      | ,=,= shall not be applied to objects of pointer type, except where they point to the same array   |
| MISRA2008-2_10_3       | A typedef name (including qualification, if any) shall be a unique identifier   |
| MISRA2008-3_2_1        | All declarations of an object or function shall have compatible types   |
| MISRA2008-3_2_2        | The One Definition Rule shall not be violated   |
| MISRA2008-3_2_3        | A type, object or function that is used in multiple translation units shall be declared in one and only one file                              |
| MISRA2008-3_2_4        | An identifier with external linkage shall have exactly one external definition  |
| MISRA2008-5_0_16_a     | Avoid accessing arrays out of bounds  |
| MISRA2008-5_0_16_b     | A pointer operand and any pointer resulting from pointer arithmetic using that operand shall both address elements of the same array          |
| MISRA2008-5_17_1       | The overloaded binary operator should be implemented in terms of its corresponding compound assignment operator                               |
| MISRA2008-5_19_1_a     | Integer overflow or underflow in constant expression in '+', '-', '*' operator  |
| MISRA2008-5_19_1_b     | Integer overflow or underflow in constant expression in " operator  |
| MISRA2008-7_2_1        | An expression with enum underlying type shall only have values corresponding to the enumerators of the enumeration                            |
| MISRA2008-7_3_6        | Using-directives and using-declarations (excluding class scope or function scope using-declarations) shall not be used in header files        |
| MISRA2008-10_1_2       | A base class shall only be declared virtual if it is used in a diamond hierarchy  |
| MISRA2008-10_2_1       | All accessible entity names within a multiple inheritance hierarchy should be unique  |
| MISRA2008-14_5_1       | Do not declare non-member generic functions in associated namespaces  |
| MISRA2008-14_6_2       | The function shall resolve to a function declared previously in the translation unit  |

|                       |  |
|-----------------------|--|
| MISRA2008-14_7_1      | All class templates, function templates, class template member functions and class template static members shall be instantiated at least one.                                     |
| MISRA2008-15_4_1      | If a function is declared with an exception-specification, then all declarations of the same function (in other translation units) shall be declared with the same set of type-ids |
| MISRA2012-RULE-18_1_c | A pointer operand and any pointer resulting from pointer arithmetic using that operand shall both address elements of the same array   |
| PB-70                 | An expression with enum underlying type shall only have values corresponding to the enumerators of the enumeration   |
| TEMPL-13              | Do not declare non-member generic functions in associated namespaces   |
| TEMPL-14              | The function shall resolve to a function declared previously in the translation unit   |

## Updated Code Analysis Rules

- BD-PB-OVERFWR, BD-PB-OVERFRD, BD-PB-VOVR, BD-SECURITY-ARRAY, BD-TRS-LOCK, BD-TRS-TSHL
- CODSTA-103, CODSTA-163\_b
- COMMENT-13
- INIT-06
- JSF-071\_b, JSF-111, JSF-171, JSF-187, JSF-204\_b
- MISRA2004-14\_2, MISRA2004-15\_1, MISRA2004-17\_2, MISRA2004-17\_3, MISRA2004-17\_6\_a
- MISRA2008-0\_1\_6, MISRA2008-0\_1\_9, MISRA2008-0\_3\_1\_g, MISRA2008-0\_3\_1\_h, MISRA2008-0\_3\_1\_i, MISRA2008-0\_3\_1\_j, MISRA2008-5\_0\_17, MISRA2008-5\_0\_18, MISRA2008-5\_19\_1\_a, MISRA2008-5\_19\_1\_b, MISRA2008-6\_4\_3\_a, MISRA2008-6\_4\_4, MISRA2008-7\_5\_1, MISRA2008-7\_5\_2\_a
- MISRA2012-DIR-4\_1\_g, MISRA2012-DIR-4\_1\_h, MISRA2012-DIR-4\_1\_i, MISRA2012-DIR-4\_1\_j, MISRA2012-DIR-4\_13\_d, MISRA2012-DIR-4\_14\_a, MISRA2012-RULE-1\_3\_d, MISRA2012-RULE-1\_3\_e, MISRA2012-RULE-1\_3\_m, MISRA2012-RULE-2\_2\_a, MISRA2012-RULE-3\_2, MISRA2012-RULE-10\_3\_b, MISRA2012-RULE-12\_4\_a, MISRA2012-RULE-12\_4\_b, MISRA2012-RULE-16\_1\_b, MISRA2012-RULE-16\_2, MISRA2012-RULE-18\_2, MISRA2012-RULE-18\_3, MISRA2012-RULE-18\_6\_a, MISRA2012-RULE-21\_17\_b
- PB-11, PB-66\_a, PB-66\_b

## Resolved Bugs and FRs

| Bug/FR ID | Description   |
|-----------|---|
| CPP-18579 | Rule MISRA2012-RULE-10_3_b (CODSTA-163_b) reports false positives   |
| CPP-36999 | base_from_member.hpp", line 136: error: expected a ")" ::new ((void*) 0) MemberType( static_cast<T&&>(x)... ) |
| CPP-38187 | The rule MISRA2004-17_6_a reports false positive violation  |
| CPP-38241 | PB-11 incorrect behaviour   |
| CPP-38336 | MISRA2004-17_6_a-3 reporting a false positive   |
| CPP-38342 | Rule MISRA2004-15_1 throwing a false positive with 'default   |
| CPP-38589 | CDD rules regression in C++test 10.3.2 (Japanese only)  |
| CPP-38602 | Add C++11 "final" specifier for function to RuleWizard dictionary   |
| CPP-38661 | Seeing log4cplus:ERROR when running the cpptestcli  |
| CPP-39168 | Cannot generate BDF using msbuild on Windows 7  |
| CPP-39210 | Report generation failed for exemplary ut_ds_details_to_csv.xml (Visual Studio)                               |
| CPP-39211 | User rule may not be executed if there's a sub-rule (text rule) in the same directory                         |
| CPP-39356 | Parse errors related to C++11 with Keil for ARM v5.x  |
| CPP-39405 | INIT-06 false positive for default move constructor   |
| CPP-39406 | [DESKTOP] TC editor shows inappropriate selection for the radio button rule parameter                         |
| CPP-39407 | double argument to static_assert  |
| CPP-39409 | MISRA2012-RULE-7_1 false positive   |
| CPP-39410 | MISRA2012-RULE-16_4_b false positive  |
| CPP-39415 | RVCT compilers accept the big-letter '--C99' flag   |
| CPP-39495 | Support for IAR EWARM 8.x   |