

Updates in 10.4.1

In this release, we've focused on extending support for environments and enhancing C/C++-test's security testing capabilities.

- [Support for Environments](#)
- [Extended Compliance Packs](#)
- [Flow Analysis Improvements](#)
- [New and Updated Test Configurations](#)
- [Flow Analysis Improvements](#)
- [Standalone License Server](#)
- [New and Updated Code Analysis Rules](#)
- [Other Changes](#)
- [Resolved Bugs and FRs](#)

Support for Environments

New Compilers

We've added support for the following compilers:

Compiler Name	Compiler Acronym
GNU GCC 8.x	gcc_8
GNU GCC 8.x (x86_64)	gcc_8-64
IAR Compiler for RL78 v. 3.10.x	iccr178_3_10

Deprecated Compilers

The following compilers are no longer supported:

Compiler Name	Compiler Acronym
Analog Devices C/C++ Compiler 7.0 for ADSP SHARC	ad21k_7_0
Analog Devices C/C++ Compiler 7.0 for ADSP TigerSHARC	adts_7_0
CodeSourcery Sourcery G++ Lite 2007q3-51	csgccarm_4_2

See [Supported Compilers](#) for details about currently supported compilers.

New IDEs

We've added support for the following IDEs:

- Eclipse 4.8

Extended Compliance Packs


We've extended the Security Compliance Pack to help you enforce compliance with the CERT C and CERT C++ security standards:

CERT C

- We've added the "SEI CERT C Rules" test configuration that helps you enforce the SEI CERT C Coding Standard rules.
- We've extended the "SEI CERT C Coding Guidelines" test configuration" to help you enforce both the SEI CERT C Coding Standard rules and guidelines.

CERT C++

- We've added the "SEI CERT C++ Rules" test configuration that helps you enforce the SEI CERT C++ Coding Standard rules.

 Compliance Packs require dedicated license features to be activated. Contact Parasoft Support for more details on Compliance Packs licensing.

Flow Analysis Improvements

- We've enhanced the presentation of Flow Analysis results in the IDE.
- We've extended Flow Analysis with the options that allow you to specify functions that can check if the resource is open, as well as functions that can be safely called on a closed resource (see [Flow Analysis - Specifying Resources](#)).
- We've improved support for `std::nullptr_t` type in Flow Analysis.

New and Updated Test Configurations

We've added the following built-in test configurations:

- SEI CERT C++ Rules
- SEI CERT C Rules
- Run VxWorks DKM Application with Full Monitoring (File System, WRWB 4.x)
- Run VxWorks DKM Unit Tests (File System, WRWB 4.x)
- Run VxWorks RTP Application with Full Monitoring (File System, WRWB 4.x)
- Run VxWorks RTP Unit Tests (File System, WRWB 4.x)

See [Built-in Test Configurations](#) for the list of test configurations shipped with C/C++test.

Deprecated Test Configurations

The following test configurations are now deprecated:

- CERT C Coding Standard
- CRules
- DISA-STIG Coding Standard
- Ellementel
- ISO 26262 Recommended Rules
- MISRA C 2012 Legacy
- OWASP Top 10 Security Vulnerabilities
- Parasoft's Recommended FDA C++ Phase 1
- Parasoft's Recommended FDA C++ Phase 2
- Parasoft's Recommended FDA C++ Phase 3
- Parasoft's Recommended Rules
- SAMATE Annex A Source Code Weaknesses

The deprecated test configurations are not available by default, but can be applied as team-shared or user-defined test configurations. They are now shipped with C/C++test in the following location: `[INSTALL_DIR]/configs/deprecated`.

Flow Analysis Improvements

- We've improved support for `std::nullptr_t` type in Flow Analysis.
- We've enhanced the presentation of Flow Analysis results in the IDE.
- We've extended Flow Analysis with the options that allow you to specify functions that can check if the resource is open, as well as functions that can be safely called on a closed resource (see [Flow Analysis - Specifying Resources](#)).

Standalone License Server

You can now obtain the Parasoft license from an additional instance of DTP or a standalone License Server. See [Setting the Parasoft License](#) (for desktop) and [Setting the License](#) (for automation).

New and Updated Code Analysis Rules

In this release, we've added new static analysis rules to extend coverage of compliance standards; see [New Rules](#) and [Updated Rules](#) for the lists of new and updated rules.

Other Changes

- We've removed support for Microsoft Team Foundation Server 2008

Resolved Bugs and FRs

Bug/FR ID	Description
CPP-18534	WindRiver Workbench 4.0 IDE support

CPP-33421	Add support for "asm goto" gcc extension (Linux Kernel Module)
CPP-39308	MDK-ARM ARM 6 compiler support
CPP-40407	Violations from SA rules are displayed in incorrect line when they are reported on code from macro
CPP-40551	Extend MISRA2004-11_5 to report on casts of const/volatile objects to reference type
CPP-40553	New rule: CODSTA-MCPP-22 Use explicit ref-qualifiers on auto declarations in range-based for loops
CPP-40623	Clang: fix undefined and mismatching builtins
CPP-40695	VS2017 plugin registration shall not be user-specific.
CPP-40771	New rule CODSTA-201: Do not process structured text data natively
CPP-40772	New rule CODSTA-199: Do not use assertions in production code
CPP-40773	New rule OOP-54: Do not increase the accessibility of overridden or hidden methods
CPP-40774	New rule: PORT-29 Enable serialization compatibility during class evolution
CPP-40775	New rule CODSTA-82_b: Do not use an empty infinite loop
CPP-41516	CODSTA-16 does not trigger violations on sizes of enum or const type
CPP-41517	RW: Missing information about va_list being builtin/predeclared type.
CPP-41520	Reference to Enum type causes compilation problem of auto generated testcase
CPP-41525	Rule MISRA2004-10_4 (CODSTA-198) should not report violations on cast of non complex expressions
CPP-41530	CODSTA-30 false positive on parameter passed as reference
CPP-41541	Fast coverage instrumentation causes compilation errors when asm statement is used
CPP-41553	Eclipse 4.8 IDE support
CPP-41586	Rule PB-27 does not report violation when a wide string is assigned to the pointer to wchar_t type (gcc on linux)
CPP-41605	error: this statement is not allowed inside of a statement expression
CPP-41611	New rule: JSF-37
CPP-41614	Deprecate Parasoft's Recommended Rules test configuration
CPP-41646	GNU GCC 8.x compiler support
CPP-41671	IAR RL78 v3.10 compiler support
CPP-41709	MISRA2012-RULE.21_2_b and MISRA2012-RULE.21_2_c problem with va_list.
CPP-41741	Property 'Entity' for node 'Variables' returns variables used in initializer
CPP-41744	MISRA2012-RULE-20_12 false positive
CPP-41748	Test Case Editor: parameterize test case automatically
CPP-41750	MDK ARM/uVision 5 IDE support
CPP-41831	RULE_OUTPUT_CHANGE Incorrect output messages in NAMING-HN rules
CPP-41840	Rule ID broken in suppression records in C/C++test reports
CPP-41842	RULE_OUTPUT_CHANGE The output message in PB-44 rule should be improved
CPP-41866	Create parameterized test case in Test Case Editor
CPP-41868	Warnings when instrumenting with cpptestcc
CPP-41871	Do not report CLLOCRIIF, CLLOCRIIT, CLLOCRIIM values if there are no logical lines
CPP-41944	HICPP-16_1_5-a rules is missing in dtp server integration package (not available on DTP server)
CPP-41977	Duplicated violations after importing from DTP
CPP-42042	internal error: assertion failed at: "lookup.c", line 2738
CPP-42070	Add support for installing VS2017 plugin for multiple users
CPP-42075	Rule MRM-40 should not report violations when copying is disabled

FA-6689	BD-PB-NP false negative when dynamic_cast is used
FA-6649	BD-PB-CC false positive on bit-AND
FA-6611	BD-RES-LEAKS false positives when resource is casted
FA-6453	Simulation incorrectly assumes pointer dereference operation on "&(ptr->field)" operation.
FA-5769	BD-PB-CHECKRET violation message contains line numbers

New Rules

The following rules have been added:

Rule ID	Header
AUTOSAR-A15_5_2-b	The library functions 'abort()', 'quick_exit()' and '_Exit()' from 'cstdlib' library shall not be used
AUTOSAR-A15_5_3-b	Never allow an exception to be thrown from a destructor, deallocation, and swap
AUTOSAR-A15_5_3-c	Do not throw from within destructor
AUTOSAR-A15_5_3-d	There should be at least one exception handler to catch all otherwise unhandled exceptions
AUTOSAR-A15_5_3-e	An empty throw (throw;) shall only be used in the compound-statement of a catch handler
AUTOSAR-A15_5_3-f	Exceptions shall be raised only after start-up and before termination of the program
AUTOSAR-A15_5_3-g	Each exception explicitly thrown in the code shall have a handler of a compatible type in all call paths that could lead to that point
AUTOSAR-A15_5_3-h	Where a function's declaration includes an exception-specification, the function shall only be capable of throwing exceptions of the indicated type(s)
AUTOSAR-A15_5_3-i	Function called in global or namespace scope shall not throw unhandled exceptions
AUTOSAR-A15_5_3-j	Always catch exceptions
AUTOSAR-A15_5_3-k	Properly define exit handlers
AUTOSAR-A5_1_4-b	Never capture local objects from an outer lambda by reference
AUTOSAR-A5_1_4-c	The lambda that captures local objects by reference should not be assigned to the variable with a greater lifetime
BD-CO-EMPCON	Do not pass empty container iterators to std algorithms as destinations
BD-CO-STRMOD	Use valid references, pointers, and iterators to reference elements of a basic_string
BD-PB-NEWHAN	Properly define new handlers
BD-PB-POLARR	Do not treat arrays polymorphically
BD-PB-PTRCMP	Do not compare two unrelated pointers

BD-PB- PTRVALUE	Do not store an already-owned pointer value in an unrelated smart pointer
BD-PB- SUBSEQMOVE	Do not rely on the value of a moved-from object
BD-PB- TERMHAN	Properly define terminate handlers
BD-PB- UNEXPHAN	Properly define unexpected handlers
BD-PB- VALRANGE	Guarantee that container indices are within the valid range
CERT_C- ARR02-a	Explicitly specify array bounds in array declarations with initializers
CERT_C- DCL10-a	There should be no difference between the number of tags from format string and the number of corresponding argument in 'printf' function invocation
CERT_C- DCL11-a	There should be no mismatch between the '%s' or '%c' tag from format string and its corresponding argument in 'printf' function invocation
CERT_C- DCL11-b	There should be no mismatch between the '%f' tag from format string and its corresponding argument in 'printf' function invocation
CERT_C- DCL11-c	There should be no mismatch between the '%i' or '%d' tag from format string and its corresponding argument in 'printf' function invocation
CERT_C- DCL11-d	There should be no mismatch between the '%u' tag from format string and its corresponding argument in 'printf' function invocation
CERT_C- DCL11-e	There should be no mismatch between the '%p' tag from format string and its corresponding argument in 'printf' function invocation
CERT_C- DCL11-f	There should be no difference between the number of tags from format string and the number of corresponding argument in 'printf' function invocation
CERT_C- ERR01-a	The error indicator errno shall not be used
CERT_C- ERR02-a	The Standard Library input/output functions shall not be used
CERT_C- ERR06-a	Do not use assertions
CERT_C- ERR07-b	The Standard Library input/output functions shall not be used
CERT_C- EXP15-a	Suspicious use of semicolon
CERT_C- FIO22-a	Ensure resources are freed
CERT_C- FIO24-a	Avoid race conditions while accessing files
CERT_C- FIO32-a	Protect against file name injection
CERT_C- INT08-a	Avoid integer overflows
CERT_C- INT15-a	The basic types of char, int, short, long, float and double should not be used, but specific-length equivalents should be typedef'd
CERT_C- MEM00-d	Do not use resources that have been freed
CERT_C- MEM00-e	Ensure resources are freed
CERT_C- MEM02-a	Assignment operator should have operands of compatible types
CERT_C- MEM02-b	Do not assign function return value to a variable of incompatible type

CERT_C-MEM04-a	The validity of values passed to library functions shall be checked
CERT_C-MEM05-a	Do not use recursion
CERT_C-MEM05-b	Ensure the size of the variable length array is in valid range
CERT_C-MEM07-a	The validity of values passed to library functions shall be checked
CERT_C-MS40-a	An inline definition of a function with external linkage shall not contain definitions and uses of static objects
CERT_C-MS41-a	Do not hard code string literals
CERT_C-STR05-a	A string literal shall not be modified
CERT_CPP-CON50-a	Do not destroy another thread's mutex
CERT_CPP-CON51-a	Do not call lock() directly on a mutex
CERT_CPP-CON52-a	Use locks to prevent race conditions when modifying bit fields
CERT_CPP-CON53-a	Do not acquire locks in different order
CERT_CPP-CON54-a	Wrap functions that can spuriously wake up in a loop
CERT_CPP-CON55-a	Do not use the 'notify_one()' function when multiple threads are waiting on the same condition variable
CERT_CPP-CON56-a	Avoid double locking
CERT_CPP-CTR50-a	Guarantee that container indices are within the valid range
CERT_CPP-CTR51-a	Do not modify container while iterating over it
CERT_CPP-CTR52-a	Do not pass empty container iterators to std algorithms as destinations
CERT_CPP-CTR53-a	Do not use an iterator range that isn't really a range
CERT_CPP-CTR53-b	Do not compare iterators from different containers
CERT_CPP-CTR54-a	Do not compare iterators from different containers
CERT_CPP-CTR54-b	Do not compare two unrelated pointers
CERT_CPP-CTR55-a	Do not add or subtract a constant with a value greater than one from an iterator
CERT_CPP-CTR56-a	Don't treat arrays polymorphically
CERT_CPP-CTR56-b	A pointer to an array of derived class objects should not be converted to a base class pointer
CERT_CPP-CTR56-c	Do not treat arrays polymorphically
CERT_CPP-CTR57-a	For associative containers never use comparison function returning true for equal values
CERT_CPP-CTR58-a	Make predicates const pure functions

CERT_CPP-DCL50-a	Functions shall not be defined with a variable number of arguments
CERT_CPP-DCL51-a	Do not #define or #undef identifiers with names which start with underscore
CERT_CPP-DCL51-b	Do not redefine reserved words
CERT_CPP-DCL51-c	Do not #define nor #undef identifier 'defined'
CERT_CPP-DCL51-d	The names of standard library macros, objects and functions shall not be reused
CERT_CPP-DCL51-e	The names of standard library macros, objects and functions shall not be reused (C90)
CERT_CPP-DCL51-f	The names of standard library macros, objects and functions shall not be reused (C99)
CERT_CPP-DCL52-a	Never qualify a reference type with 'const' or 'volatile'
CERT_CPP-DCL53-a	Always declare functions at file scope
CERT_CPP-DCL53-b	Identifier declared in a local or function prototype scope shall not hide an identifier declared in a global or namespace scope
CERT_CPP-DCL54-a	Always provide new and delete together
CERT_CPP-DCL55-a	A pointer to a structure should not be passed to a function that can copy data to the user space
CERT_CPP-DCL56-a	Avoid initialization order problems across translation units by replacing non-local static objects with local static objects
CERT_CPP-DCL57-a	Never allow an exception to be thrown from a destructor, deallocation, and swap
CERT_CPP-DCL57-b	Always catch exceptions
CERT_CPP-DCL58-a	Do not modify the standard namespaces 'std' and 'posix'
CERT_CPP-DCL59-a	There shall be no unnamed namespaces in header files
CERT_CPP-DCL60-a	A class, union or enum name (including qualification, if any) shall be a unique identifier
CERT_CPP-ERR50-a	The execution of a function registered with 'std::atexit()' or 'std::at_quick_exit()' should not exit via an exception
CERT_CPP-ERR50-b	Never allow an exception to be thrown from a destructor, deallocation, and swap
CERT_CPP-ERR50-c	Do not throw from within destructor
CERT_CPP-ERR50-d	There should be at least one exception handler to catch all otherwise unhandled exceptions
CERT_CPP-ERR50-e	An empty throw (throw;) shall only be used in the compound-statement of a catch handler
CERT_CPP-ERR50-f	Exceptions shall be raised only after start-up and before termination of the program
CERT_CPP-ERR50-g	Each exception explicitly thrown in the code shall have a handler of a compatible type in all call paths that could lead to that point
CERT_CPP-ERR50-h	Where a function's declaration includes an exception-specification, the function shall only be capable of throwing exceptions of the indicated type(s)
CERT_CPP-ERR50-i	Function called in global or namespace scope shall not throw unhandled exceptions

CERT_CPP-ERR50-j	Always catch exceptions
CERT_CPP-ERR50-k	Properly define exit handlers
CERT_CPP-ERR50-l	The library functions 'abort()', 'quick_exit()' and '_Exit()' from 'cstdlib' library shall not be used
CERT_CPP-ERR51-a	Always catch exceptions
CERT_CPP-ERR51-b	Each exception explicitly thrown in the code shall have a handler of a compatible type in all call paths that could lead to that point
CERT_CPP-ERR52-a	The setjmp macro and the longjmp function shall not be used
CERT_CPP-ERR52-b	The standard header file setjmp.h shall not be used
CERT_CPP-ERR53-a	Handlers of a function-try-block implementation of a class constructor or destructor shall not reference nonstatic members from this class or its bases
CERT_CPP-ERR54-a	Where multiple handlers are provided in a single try-catch statement or function-try-block for a derived class and some or all of its bases, the handlers shall be ordered most-derived to base class
CERT_CPP-ERR55-a	Where a function's declaration includes an exception-specification, the function shall only be capable of throwing exceptions of the indicated type(s)
CERT_CPP-ERR56-a	Ensure resources are freed
CERT_CPP-ERR57-a	Ensure resources are freed
CERT_CPP-ERR58-a	Exceptions shall be raised only after start-up and before termination of the program
CERT_CPP-ERR59-a	Do not throw an exception across execution boundaries
CERT_CPP-ERR60-a	Exception objects must be nothrow copy constructible
CERT_CPP-ERR60-b	An explicitly declared copy constructor for a class that inherits from 'std::exception' should have a non-throwing exception specification
CERT_CPP-ERR61-a	A class type exception shall always be caught by reference
CERT_CPP-ERR61-b	Throw by value, catch by reference
CERT_CPP-ERR62-a	The library functions atof, atoi and atol from library stdlib.h shall not be used
CERT_CPP-EXP50-a	The value of an expression shall be the same under any order of evaluation that the standard permits
CERT_CPP-EXP50-b	Don't write code that depends on the order of evaluation of function arguments
CERT_CPP-EXP50-c	Don't write code that depends on the order of evaluation of function designator and function arguments
CERT_CPP-EXP50-d	Don't write code that depends on the order of evaluation of expression that involves a function call
CERT_CPP-EXP50-e	Between sequence points an object shall have its stored value modified at most once by the evaluation of an expression
CERT_CPP-EXP50-f	Don't write code that depends on the order of evaluation of function calls
CERT_CPP-EXP51-a	Do not treat arrays polymorphically
CERT_CPP-EXP52-a	The operand of the sizeof operator shall not contain any expression which has side effects

CERT_CPP-EXP52-b	Object designated by a volatile lvalue should not be accessed in the operand of the sizeof operator
CERT_CPP-EXP52-c	The function call that causes the side effect shall not be the operand of the sizeof operator
CERT_CPP-EXP53-a	Avoid use before initialization
CERT_CPP-EXP54-a	Do not use resources that have been freed
CERT_CPP-EXP54-b	The address of an object with automatic storage shall not be returned from a function
CERT_CPP-EXP54-c	The address of an object with automatic storage shall not be assigned to another object that may persist after the first object has ceased to exist
CERT_CPP-EXP55-a	A cast shall not remove any 'const' or 'volatile' qualification from the type of a pointer or reference
CERT_CPP-EXP56-a	Do not call a function with a mismatched language linkage
CERT_CPP-EXP57-a	Do not delete objects with incomplete class at the point of deletion
CERT_CPP-EXP57-b	Conversions shall not be performed between a pointer to an incomplete type and any other type
CERT_CPP-EXP58-a	Use macros for variable arguments correctly
CERT_CPP-EXP60-a	Do not pass a nonstandard-layout type object across execution boundaries
CERT_CPP-EXP61-a	Never return lambdas that capture local objects by reference
CERT_CPP-EXP61-b	Never capture local objects from an outer lambda by reference
CERT_CPP-EXP61-c	The lambda that captures local objects by reference should not be assigned to the variable with a greater lifetime
CERT_CPP-EXP63-a	Do not rely on the value of a moved-from object
CERT_CPP-FIO50-a	Do not alternately input and output from a stream without an intervening flush or positioning call
CERT_CPP-FIO51-a	Ensure resources are freed
CERT_CPP-INT50-a	An expression with enum underlying type shall only have values corresponding to the enumerators of the enumeration
CERT_CPP-MEM50-a	Do not use resources that have been freed
CERT_CPP-MEM51-a	Use the same form in corresponding calls to new/malloc and delete/free
CERT_CPP-MEM51-b	Always provide empty brackets ([]) for delete when deallocating arrays
CERT_CPP-MEM51-c	Both copy constructor and copy assignment operator should be declared for classes with a nontrivial destructor
CERT_CPP-MEM52-a	Check the return value of new
CERT_CPP-MEM52-b	Do not allocate resources in function argument list because the order of evaluation of a function's parameters is undefined
CERT_CPP-MEM53-a	Do not invoke malloc/realloc for objects having constructors
CERT_CPP-MEM55-a	The user defined 'new' operator should throw the 'std::bad_alloc' exception when the allocation fails

CERT_CPP-MEM56-a	Do not store an already-owned pointer value in an unrelated smart pointer
CERT_CPP-MS50-a	Do not use the rand() function for generating pseudorandom numbers
CERT_CPP-MS51-a	Properly seed pseudorandom number generators
CERT_CPP-MS52-a	All exit paths from a function with non-void return type shall have an explicit return statement with an expression
CERT_CPP-MS53-a	Never return from functions that should not return
CERT_CPP-MS54-a	Properly define signal handlers
CERT_CPP-OOP50-a	Avoid calling virtual functions from constructors
CERT_CPP-OOP50-b	Avoid calling virtual functions from destructors
CERT_CPP-OOP50-c	Do not invoke class's virtual functions from any of its constructors
CERT_CPP-OOP50-d	Do not invoke class's virtual functions from its destructor
CERT_CPP-OOP51-a	Avoid slicing function arguments / return value
CERT_CPP-OOP52-a	Define a virtual destructor in classes used as base classes which have virtual functions
CERT_CPP-OOP53-a	List members in an initialization list in the order in which they are declared
CERT_CPP-OOP54-a	Check for assignment to self in operator=
CERT_CPP-OOP55-a	A cast shall not convert a pointer to a function to any other pointer type, including a pointer to function type
CERT_CPP-OOP56-a	Properly define terminate handlers
CERT_CPP-OOP56-b	Properly define unexpected handlers
CERT_CPP-OOP56-c	Properly define new handlers
CERT_CPP-OOP57-a	Do not initialize objects with a non-trivial class type using C standard library functions
CERT_CPP-OOP57-b	Do not compare objects of nonstandard-layout class type with C standard library functions
CERT_CPP-OOP58-a	Copy operations must not mutate the source object
CERT_CPP-STR50-a	Use vector and string instead of arrays
CERT_CPP-STR51-a	Avoid null pointer dereferencing
CERT_CPP-STR52-a	Use valid references, pointers, and iterators to reference elements of a basic_string
CERT_CPP-STR53-a	Guarantee that container indices are within the valid range
CODSTA-197	Do not specify the bound of a character array initialized with a string literal
CODSTA-199	Do not use assertions
CODSTA-200	Explicitly specify array bounds in array declarations with initializers

CODSTA-201	Do not process structured text data natively
CODSTA-202	An inline definition of a function with external linkage shall not contain definitions and uses of static objects
CODSTA-203	Do not hard code string literals
CODSTA-82_b	Do not use empty infinite loops
CODSTA-CPP-93	Do not initialize objects with a non-trivial class type using C standard library functions
CODSTA-CPP-94	Do not compare objects of nonstandard-layout class type with C standard library functions
CODSTA-CPP-95	Do not modify the standard namespaces 'std' and 'posix'
CODSTA-CPP-96	Do not call a function with a mismatched language linkage
CODSTA-CPP-97	Never qualify a reference type with 'const' or 'volatile'
CODSTA-CPP-98	Copy operations must not mutate the source object
CODSTA-MCPP-17_b	Never capture local objects from an outer lambda by reference
CODSTA-MCPP-17_c	The lambda that captures local objects by reference should not be assigned to the variable with a greater lifetime
CODSTA-MCPP-22	Use explicit ref-qualifiers on auto declarations in range-based 'for' loops
EXCEPT-19	Exception objects must be nothrow copy constructible
EXCEPT-20	An explicitly declared copy constructor for a class that inherits from 'std::exception' should have a non-throwing exception specification
JSF-037	A file should directly include only headers containing declarations and definitions needed to a compilation
MISRA2008-15_5_3_b	Never allow an exception to be thrown from a destructor, deallocation, and swap
MISRA2008-15_5_3_c	Do not throw from within destructor
MISRA2008-15_5_3_d	There should be at least one exception handler to catch all otherwise unhandled exceptions
MISRA2008-15_5_3_e	An empty throw (throw;) shall only be used in the compound-statement of a catch handler
MISRA2008-15_5_3_f	Exceptions shall be raised only after start-up and before termination of the program
MISRA2008-15_5_3_g	Each exception explicitly thrown in the code shall have a handler of a compatible type in all call paths that could lead to that point
MISRA2008-15_5_3_h	Where a function's declaration includes an exception-specification, the function shall only be capable of throwing exceptions of the indicated type(s)
MISRA2008-15_5_3_i	Function called in global or namespace scope shall not throw unhandled exceptions
MISRA2008-15_5_3_j	Always catch exceptions
MISRA2008-15_5_3_k	Properly define exit handlers
MRM-53	The user defined 'new' operator should throw the 'std::bad_alloc' exception when the allocation fails
OOP-54	Do not increase the accessibility of overridden or hidden methods
OPT-41	A file should directly include only the headers that contain declarations and definitions required to compile that file
PB-74	Do not add or subtract a constant with a value greater than one from an iterator
PB-75	The library functions 'abort()', 'quick_exit()' and '_Exit()' from 'cstdlib' library shall not be used

PORT-29	A pointer to a structure should not be passed to a function that writes data to a file
PORT-30	Do not throw an exception across execution boundaries
PORT-31	Do not pass a nonstandard-layout type object across execution boundaries
SECURITY-50	Do not use the 'notify_one()' function when multiple threads are waiting on the same condition variable

Updated Rules

We've updated following static analysis rules to improve analysis results:

Rule Category	Rule IDs
AUTOSAR C++14 Coding Guidelines	AUTOSAR-A12_0_1-a, AUTOSAR-A15_1_4-a, AUTOSAR-A15_5_2-a, AUTOSAR-A15_5_3-a, AUTOSAR-A15_5_3-b, AUTOSAR-A2_14_2-a, AUTOSAR-A5_2_2-a, AUTOSAR-A5_2_3-a, AUTOSAR-A7_1_1-a, AUTOSAR-M0_3_1-f, AUTOSAR-M4_5_3-a, AUTOSAR-M5_0_21-a, AUTOSAR-M5_2_8-a
Flow Analysis	BD-CO-ITINVCOMP, BD-CO-ITMOD, BD-PB-NP, BD-PB-STREAMINOUT, BD-PB-VARARGS, BD-PB-VCTOR, BD-PB-VDTOR, BD-RES-FREE, BD-RES-LEAKS, BD-TRS-DLOCK, BD-TRS-DSTRLOCK
SEI CERT C	CERT_C-CON30-a, CERT_C-CON31-a, CERT_C-CON31-b, CERT_C-CON35-a, CERT_C-DCL00-a, CERT_C-ERR33-c, CERT_C-EXP05-a, CERT_C-EXP32-a, CERT_C-EXP34-a, CERT_C-EXP40-a, CERT_C-FIO22-a, CERT_C-FIO39-a, CERT_C-FIO42-a, CERT_C-FIO46-a, CERT_C-INT13-a, CERT_C-INT16-a, CERT_C-INT36-a, CERT_C-MEM00-d, CERT_C-MEM00-e, CERT_C-MEM01-a, CERT_C-MEM12-a, CERT_C-MEM30-a, CERT_C-MEM31-a, CERT_C-MS319-b, CERT_C-MS39-a, CERT_C-POS48-a, CERT_C-POS54-c, CERT_C-STR05-a, CERT_C-STR09-a, CERT_C-STR10-a, CERT_C-STR30-a, CERT_C-WIN30-a
SEI CERT C++	CERT_CPP-CON50-a, CERT_CPP-CON56-a, CERT_CPP-CTR51-a, CERT_CPP-CTR53-b, CERT_CPP-CTR54-a, CERT_CPP-DCL51-e, CERT_CPP-DCL51-f, CERT_CPP-DCL57-a, CERT_CPP-ERR50-a, CERT_CPP-ERR50-b, CERT_CPP-ERR56-a, CERT_CPP-ERR57-a, CERT_CPP-EXP54-a, CERT_CPP-EXP55-a, CERT_CPP-EXP58-a, CERT_CPP-FIO50-a, CERT_CPP-FIO51-a, CERT_CPP-MEM50-a, CERT_CPP-OOP50-c, CERT_CPP-OOP50-d, CERT_CPP-STR51-a
Coding Conventions	CODSTA-16, CODSTA-30, CODSTA-63, CODSTA-65, CODSTA-69
Coding Conventions for C++	CODSTA-CPP-11, CODSTA-CPP-53, CODSTA-CPP-66
Exceptions	EXCEPT-01
High Integrity C++	HICPP-12_4_1-b, HICPP-12_4_1-c, HICPP-18_3_1-a, HICPP-5_2_1-c, HICPP-5_4_1-a, HICPP-5_4_1-c, HICPP-5_6_1-a, HICPP-7_1_2-a, HICPP-8_4_1-b
Joint Strike Fighter	JSF-151.1, JSF-185
MISRA C 2004	MISRA2004-11_5, MISRA2004-20_2_a, MISRA2004-20_2_b
MISRA C++ 2008	MISRA2008-0_3_1_b, MISRA2008-15_5_3, MISRA2008-15_5_3_b, MISRA2008-2_13_5, MISRA2008-4_5_3, MISRA2008-5_0_21, MISRA2008-5_2_4, MISRA2008-5_2_5, MISRA2008-5_2_8, MISRA2008-7_1_1
MISRA C 2012 (Legacy)	MISRA2012-DIR-4_13_a, MISRA2012-DIR-4_13_b, MISRA2012-DIR-4_13_e, MISRA2012-DIR-4_1_b, MISRA2012-RULE-11_8, MISRA2012-RULE-1_3_c, MISRA2012-RULE-21_2_b, MISRA2012-RULE-21_2_c, MISRA2012-RULE-22_1, MISRA2012-RULE-22_2_a, MISRA2012-RULE-22_6, MISRA2012-RULE-7_4
MISRA C 2012	MISRAC2012-DIR_4_1-b, MISRAC2012-DIR_4_13-a, MISRAC2012-DIR_4_13-b, MISRAC2012-DIR_4_13-e, MISRAC2012-RULE_11_8-a, MISRAC2012-RULE_1_3-c, MISRAC2012-RULE_21_2-b, MISRAC2012-RULE_21_2-c, MISRAC2012-RULE_22_1-a, MISRAC2012-RULE_22_2-a, MISRAC2012-RULE_22_6-a, MISRAC2012-RULE_7_4-a
Memory and Resource Management	MRM-40
Possible Bugs	PB-27, PB-38, PB-44

The output messages of the following rules have been updated, and as a result, suppressions associated with these rules on DTP may no longer be available:

- BD-PB-ARRAY

- BD-PB-CHECKRET
- BD-PB-OVERFARRAY
- BD-PB-PTRARR
- BD-PB-ZERO
- BD-TRS-MLOCK
- NAMING-HN-*

You can restore the previous messages and suppressions for the BD category rules; see [Why are suppressions of some rules no longer available on DTP after C/C++test was upgraded to a newer version?](#).