Updates in 10.3.0

The latest release of DTP Engines build on Parasoft's innovative approach to continuously improving software quality processes.

In this release:

Understanding Change in Continuous Delivery Environments

As organizations implement continuous software delivery platforms, conventional reporting mechanisms based on data aggregated over time are no longer viable. Understanding the risk that incremental changes introduce from build to build becomes critical as the release velocity accelerates. Organizations need immediate visibility into changes in coverage, changes in test regressions, and the ability to quickly identify which tests need to be rerun. The updates in this release help you understand these changes and identify any issues that need to be resolved in order to mitigate risk and accelerate delivery.

Reporting Static Analysis Violations by Build

In this release, we've extended the concept of reporting build-based data. DTP Engines can now report new, fixed, and existing static analysis violations by build, enabling organizations to not only address potential defects more efficiently early in the SDLC, but also correlate violations with change to pinpoint specific areas of the code where defects may have been introduced.

Extended Support for Automotive Software Quality

As the focus on quality and safety-critical aspects of automotive software come to the forefront of the industry, Parasoft continues to expand its offering for C and C++ applications.

Complete MISRA C:2012 Coverage for Static Analysis

The static analysis rules shipped with the DTP Engine for C/C++ have been updated to include complete coverage for the MISRA C:2012 standard, including complete coverage for Amendment 1 published in May of 2016.

The marketplace also includes a new MISRA qualification kit, which provides configurations for the DTP Engines and the DTP server for centralized reporting. The kit contains a set of regression tests and supporting documents to help you validate your use of MISRA C:2012, such as when applying static analysis rules for ISO 26262 compliance.

Other Updates and Enhancements

- C/C++test: New rules for MISRA C:2012 (see New and Updated Code Analysis Rules).
- C/C++test: Enhanced support for Modern C++ standards (C++11, C++14, C++17), including dedicated static analysis. rules and test configuration.
- C/C++test: Added support for execution and coverage analysis of the CppUTest unit testing framework.
- Improved rules parameterization in the DTP Test Configuration UI.
- Improved reporting of errors and setup problems within HTML reports and on DTP.

New and Updated Code Analysis Rules

Rule ID	Description
CODSTA-MCPP-01	User-conversion cast operators should be made explicit
CODSTA-MCPP-02	Prefer alias declarations to typedefs
CODSTA-MCPP-03	Prefer Scoped Enums to Unscoped Enums
CODSTA-MCPP-04	Prefer 'nullptr' over 'NULL' or '0'(zero)
CODSTA-MCPP-05	Declare overriding functions with 'override' specifier

CODSTA-MCPP-06_a	Declare copy constructor and copy assignment operators with the 'delete' specifier to prevent copying of class
CODSTA-MCPP-06_b	Declare copy constructor and copy assignment operators with the 'delete' specifier instead of using a base class with private methods to prevent copying of class Coding Conventions
CODSTA-174_a_c90	A program should not exceed the translation limits imposed by The Standard (c90)
CODSTA-174_a_c99	A program should not exceed the translation limits imposed by The Standard (c99)
CODSTA-174_b_c90	A program should not exceed the translation limits imposed by The Standard (c90)
CODSTA-174_b_c99	A program should not exceed the translation limits imposed by The Standard (c99)
CODSTA-175_a	A function should not contain unused type declarations
CODSTA-175_b	A source file should not contain unused type declarations
CODSTA-176_a	A function should not contain unused local tag declarations
CODSTA-176_b	A source file should not contain unused tag declarations
CODSTA-177	A source file should not contain unused macro declarations
CODSTA-178	External identifiers shall be distinct
CODSTA-179_a_c90	Identifiers declared in the file scope and in the same name space shall be distinct (c90)
CODSTA-179_a_c99	Identifiers declared in the file scope and in the same name space shall be distinct (c99)
CODSTA-179_b_c90	Identifiers declared in the same block scope and name space shall be distinct (c90)
CODSTA-179_b_c99	Identifiers declared in the same block scope and name space shall be distinct (c99)
CODSTA-180	Identifiers that define objects or functions with external linkage shall be unique
CODSTA-181	The +, -, += and -= operators should not be applied to an expression of pointer type
CODSTA-182	The 'sizeof' operator shall not have an operand which is a function parameter declared as "array of type"
CODSTA-183	The pointer arguments to the Standard Library functions 'memcmp', 'memmove' and 'memcmp' shall be pointers to qualified or unqualified versions of compatible types
CODSTA-184	The pointer arguments to the Standard Library function 'memcmp' shall point to either a pointer type, an essentially signed type, an essentially unsigned type, an essentially Boolean type or an essentially enum type
CODSTA-185_a	The pointers returned by the Standard Library functions 'localeconv', 'getenv', 'setlocale' or, 'strerror' shall only be used as if they have pointer to const-qualified type
CODSTA-185_b	Strings pointed by members of the structure 'lconv' should not be modified
CODSTA-186	Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly
BD-API-BADPARAM	Do not pass incorrect values to library functions
BD-API-CTYPE	Do not pass incorrect values to ctype.h library functions
BD-API-STRSIZE	The size_t argument passed to any function in string.h shall have an appropriate value
BD-API-VALPARAM	Validate values passed to library functions
BD-PB-EOFCOMP	The macro EOF should be compared with the unmodified return value from the Standard Library function
BD-PB-ERRNO	Properly use errno value
BD-PB-INVRET	Pointers returned by certain Standard Library functions should not be used following a subsequent call to the same or related function
BD-PB-MCCSTR	The Standard Library function memcmp shall not be used to compare null terminated strings

BD-PB-NORETURN	Never return from the function with 'noreturn' attribute
BD-PB-WRRDSTR	The same file shall not be opened for read and write access at the same time on different streams
BD-SECURITY-TDCONSOLE	Avoid printing tainted data on the output console
FORMAT-25_b	Parenthesis shall be used with the "return" statement
MISRA2004-9_2_b	Arrays shall not be partially initialized
MISRA2004-9_2_c	The non-zero initialization of structures requires an explicit initializer for each element
MISRA2012-DIR-4_5	Identifiers in the same name space with overlapping visibility should be typographically unambiguous
MISRA2012-DIR-4_7_a[b]	Consistently check the returned value of non-void functions
MISRA2012-DIR-4_7_b[b]	Always check the returned value of non-void function
MISRA2012-DIR-4_11[b]	Validate values passed to library functions
MISRA2012-DIR-4_13_a[b]	All resources obtained dynamically by means of Standard Library functions shall be explicitly released
MISRA2012-DIR-4_13_b[b]	Do not use resources that have been freed
MISRA2012-DIR-4_13_c[b]	Do not free resources using invalid pointers
MISRA2012-DIR-4_13_d[b]	Do not abandon unreleased locks
MISRA2012-DIR-4_13_e[b]	Avoid double locking
MISRA2012-DIR-4_14_a[b]	Avoid tainted data in array indexes
MISRA2012-DIR-4_14_b[b]	Protect against integer overflow/underflow from tainted data
MISRA2012-DIR-4_14_c[b]	Avoid buffer read overflow from tainted data
MISRA2012-DIR-4_14_d[b]	Avoid buffer write overflow from tainted data
MISRA2012-DIR-4_14_e[b]	Protect against command injection
MISRA2012-DIR-4_14_f[b]	Protect against file name injection
MISRA2012-DIR-4_14_g[b]	Protect against SQL injection
MISRA2012-DIR-4_14_h[b]	Prevent buffer overflows from tainted data
MISRA2012-DIR-4_14_i[b]	Avoid buffer overflow from tainted data due to defining incorrect format limits
MISRA2012-DIR-4_14_j[b]	Protect against environment injection
MISRA2012-DIR-4_14_k[b]	Avoid printing tainted data on the output console

MISRA2012-RULE-1_1_a_c90	A program should not exceed the translation limits imposed by The Standard (c90)
MISRA2012-RULE-1_1_a_c99	A program should not exceed the translation limits imposed by The Standard (c99)
MISRA2012-RULE-1_1_b_c90	A program should not exceed the translation limits imposed by The Standard (c90)
MISRA2012-RULE-1_1_b_c99	A program should not exceed the translation limits imposed by The Standard (c99)
MISRA2012-RULE-1_3_a[b]	Avoid division by zero
MISRA2012-RULE-1_3_b[b]	Avoid use before initialization
MISRA2012-RULE-1_3_c[b]	Do not use resources that have been freed
MISRA2012-RULE-1_3_d	Avoid overflow when reading from a buffer
MISRA2012-RULE-1_3_e[b]	Avoid overflow when writing to a buffer
MISRA2012-RULE-1_3_f	The value of an expression shall be the same under any order of evaluation that the standard permits
MISRA2012-RULE-1_3_g	Don't write code that depends on the order of evaluation of function arguments
MISRA2012-RULE-1_3_h	Don't write code that depends on the order of evaluation of function designator and function arguments
MISRA2012-RULE-1_3_i	Don't write code that depends on the order of evaluation of expression that involves a function call
MISRA2012-RULE-1_3_j	Between sequence points an object shall have its stored value modified at most once by the evaluation of an expression
MISRA2012-RULE-1_3_k	Do not use more than one volatile in one expression
MISRA2012-RULE-1_3_l	Don't write code that depends on the order of evaluation of function calls
MISRA2012-RULE-1_3_m	A function shall not return a pointer or reference to a non-static local object
MISRA2012-RULE-1_3_n	The address of an object with automatic storage shall not be assigned to an object which persists after the object has ceased to exist
MISRA2012-RULE-1_3_o	The left-hand operand of a right-shift operator shall not have a negative value
MISRA2012-RULE-2_3_a	A function should not contain unused type declarations
MISRA2012-RULE-2_3_b	A source file should not contain unused type declarations
MISRA2012-RULE-2_4_a	A function should not contain unused local tag declarations
MISRA2012-RULE-2_4_b	A source file should not contain unused tag declarations
MISRA2012-RULE-2_5	A source file should not contain unused macro declarations
MISRA2012-RULE-5_1	External identifiers shall be distinct
MISRA2012-RULE-5_2_a_c90	Identifiers declared in the file scope and in the same name space shall be distinct (c90)

MISRA2012-RULE-5_2_a_c99	Identifiers declared in the file scope and in the same name space shall be distinct (c99)
MISRA2012-RULE-5_2_b_c90	Identifiers declared in the same block scope and name space shall be distinct (c90)
MISRA2012-RULE-5_2_b_c99	Identifiers declared in the same block scope and name space shall be distinct (c99)
MISRA2012-RULE-5_8	Identifiers that define objects or functions with external linkage shall be unique
MISRA2012-RULE-8_6	An identifier with external linkage shall have exactly one external definition
MISRA2012-RULE-8_7	Functions and objects should not be defined with external linkage if they are referenced in only one translation unit
MISRA2012-RULE-9_4	An element of an object shall not be initialized more than once
MISRA2012-RULE-9_5	Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly
MISRA2012-RULE-12_5	The 'sizeof' operator shall not have an operand which is a function parameter declared as "array of type"
MISRA2012-RULE-18_1_a[b]	Avoid accessing arrays out of bounds
MISRA2012-RULE-18_1_b[b]	Avoid accessing arrays and pointers out of bounds
MISRA2012-RULE-18_4	The +, -, += and -= operators should not be applied to an expression of pointer type
MISRA2012-RULE-20_8	The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1
MISRA2012-RULE-21_13[b]	Any value passed to a function in <ctype.h> shall be representable as an 'unsigned char' or be the value 'EOF'
MISRA2012-RULE-21_14[b]	The Standard Library function 'memcmp' shall not be used to compare null-terminated strings
MISRA2012-RULE-21_15	The pointer arguments to the Standard Library functions 'memcmp', 'memmove' and 'memcmp' shall be pointers to qualified or unqualified versions of compatible types
MISRA2012-RULE-21_16	The pointer arguments to the Standard Library function 'memcmp' shall point to either a pointer type, an essentially signed type, an essentially unsigned type, an essentially Boolean type or an essentially enum type
MISRA2012-RULE-21_17_a[b]	Avoid overflow due to reading a not zero terminated string
MISRA2012-RULE-21_17_b[b]	Avoid overflow when writing to a buffer
MISRA2012-RULE-21_18[b]	The 'size_t' argument passed to any function in <string.h> shall have an appropriate value
MISRA2012-RULE-21_19_a	The pointers returned by the Standard Library functions 'localeconv', 'getenv', 'setlocale' or, 'strerror' shall only be used as if they have pointer to const-qualified type
MISRA2012-RULE-21_19_b	Strings pointed by members of the structure 'lconv' should not be modified
MISRA2012-RULE-21_20[b]	Pointers returned by certain Standard Library functions should not be used following a subsequent call to the same or related function
MISRA2012-RULE-22_3[b]	The same file shall not be opened for read and write access at the same time on different stream
MISRA2012-RULE-22_4[b]	Avoid writing to a stream which has been opened as read only
MISRA2012-RULE-22_7[b]	The macro 'EOF' should be compared with the unmodified return value from the Standard Library function

MISRA2012-RULE-22_8[b]	The value of 'errno' shall be set to zero prior to a call to an errno-setting-function
MISRA2012-RULE-22_9[bc]	The value of 'errno' shall be tested against zero after calling an errno-setting-function
MISRA2012-RULE-22_10[bc]	The value of 'errno' shall only be tested when the last function to be called was an errno-setting-function
MISRA2008-8_5_2_b	Arrays shall not be partially initialized
MISRA2008-8_5_2_c	Structures shall not be partially initialized
NAMING-50	Identifiers in the same name space with overlapping visibility should be typographically unambiguous
PB-69	An element of an object shall not be initialized more than once
PREPROC-19	The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1

Resolved PRs/FRs

PR/FR ID	Description
117075	The Categories used by the "Compliance by Category" Widget don't match the target Test Configurations
119444	Unable to see Test Configuration in DTP Server after it is created
119478	Required feature for Server Edition license is not available with ENT licenses
120275	Jenkins Cobertura Coverage widget does not work with Japanese locale
118760	Rule UC.UP is missing in Jtest 10.x
118762	Rule METRIC.DIF is missing in Jtest 10.x
118761	Rule GLOBAL.ACD is missing in Jtest 10.x
120444	Gradle 3.0 not supported
118860	IntelliJ: sourcelevel in json file is empty
119814	Bulitin Test Configurations are not available in IBM RAD GUI
120140	Discrepancy between documentation and jtestcli.jvm regarding -Xmx value
120595	NAMING.NE does not work correctly in case of inner classes
121001	Wrong example repair code in rule's documentation of HIBERNATE.SLM
116773	BD-PB_NOTINIT stops triggering when unrelated code is added
116892	Add nullptr support in rulewizard
118116	Error: use of __if_exists is not supported in this context
119825	MISRA2008-5.0.6.a False Negative for expression initializers
120019	Differences between the results of the analysis when incremental mode is enabled or disabled
120121	Parsing error: a derived class is not allowed here
120126	Problem when -include/-exclude file is not found
120353	BD.PB.VOVR parameter name is incorrect and should be updated
120455	MISRA2004-9_2 false positive when #define symbol used in array initializer
120123	Incorrect use of website option
120203	Invalid MSBuild target when building WebSite with special characters in name or located in a solution subfolder
120296	Coverage markers are not visible when importing results to a project with path including Japanese characters

117336	Randomly failing Engine analysis
120151	Setup problems reported during diff operation on TFS
120315	Suppressions are ignored when rule id includes severity

a. Available for Eclipse-based IDEs.
b. Requires license for Flow Analysis; contact your Parasoft representative.
c. When using these rules in a custom test configuration the following parameters need to be set:
   For MISRA2012-RULE-22_8:
   ```
   MISRA2012-RULE-22_8-reportOnMissingErrnoCheck=false
   MISRA2012-RULE-22_8-reportOnUnnecessaryErrnoCheck=false
   ```

   For MISRA2012-RULE-22_9:
   ```
   MISRA2012-RULE-22_9-reportWhenErrnoIsNotZero=false
   MISRA2012-RULE-22_9-reportOnUnnecessaryErrnoCheck=false
   ```

   For MISRA2012-RULE-22_10:
   ```
   MISRA2012-RULE-22_10-reportWhenErrnoIsNotZero=false
   MISRA2012-RULE-22_10-reportOnMissingErrnoCheck=false
   ```

   These parameters are already set in the built-in "MISRA C 2012" test configuration