PRQA Programming Research

QA·C 8.1.2 SOURCE CODE ANALYZER

RELEASE NOTES

September 2013

This document lists the QA·C Source Code Analyzer 8.1.2 application release notes.

IMPORTANT NOTICE

DISCLAIMER OF WARRANTY

The staff of Programming Research Ltd have taken due care in preparing this document which is believed to be accurate at the time of printing. However, no liability can be accepted for errors or omissions nor should this document be considered as an expressed or implied warranty that the products described perform as specified within.

COPYRIGHT NOTICE

TRADEMARKS

PRQA, the PRQA logo, and QA·C are registered trademarks of Programming Research Ltd. Windows is a registered trademark of Microsoft Corporation.

CONTACTING PROGRAMMING RESEARCH LTD

For technical support, contact your nearest Programming Research Ltd authorized distributor or you can contact Programming Research's head office:

by telephone on +44 (0) 1 932 888 080
by fax on +44 (0) 1 932 888 081
or by e-mail on support@programmingresearch.com
www.programmingresearch.com

Table of Contents

1. INTRODUCTION

The following section outlines the principal improvements in this maintenance release of QA·C 8.1.2.

This document only identifies upgrades from release 8.1.1 to release 8.1.2.

For the upgrades from release 8.0 to release 8.1 please refer to the "QA·C 8.1 Source Code Analyzer Release Notes" document.

For the upgrades from release 8.1 to release 8.1.1 please refer to the "QA·C 8.1.1 Source Code Analyzer Release Notes" document.

1.1 Native 64 bit Support

QA·C 8.1.2 includes full support of native 64 bit hardware environments. As well as allowing for a larger address space, it is particularly important for Linux where 32 bit libraries are no longer shipped with 64 bit versions of the OS.

1.1.1 Known Issue

When running QA·C on Windows 7 64bit, the online help will cause a failure of the QA·C GUI unless you have version 10 or above of Internet Explorer installed.

1.2 Floating Point Threshold Values

Previous releases of QA·C did not support floating point values to the -THRESHold configuration option. Additionally, although the documentation implied that the threshold for the STCDN metric could be specified as a percentage, this was not supported.

QA·C 8.1.2 allows for floating point values to be used as arguments to the -THRESHold option.

For example: -thresh="STCDN < 0.25" will result in message 4700 being generated should the Comment Density for the file be less than 25%.

1.3 Dataflow

A number of fixes and extensions to dataflow analysis have been implemented. Due to the interconnected nature of dataflow analysis the ripple effects of these upgrades can be wide ranging over the source code and potentially affect nearly all dataflow messages. The table in section 5 provides the full list of dataflow messages.

2. NEW MESSAGES

There are no new messages.

3. REMOVED MESSAGES

No messages have been removed.

4. MESSAGES WITH MODIFIED BEHAVIOR

The following table summarises the messages whose behaviour has been modified as a result of a change in specification.

Msg	Message Text	CR
0178	*Precision of format conversion specifier exceeds 509 characters.*	10909
0778	*Identifier matches other identifier(s) (e.g. '%s') in first 31 characters - program does not conform strictly to ISO:C90.*	11455 13438
0779	*Identifier does not differ from other identifier(s) (e.g. '%s') within the specified number of significant characters.*	11455 13438
1006	*This in-line assembler construct is a language extension. The code has been ignored.*	13261 13574
1300	*'%s' is a keyword in C++.*	11240
2700-3000	[nearly all dataflow analysis messages can be affected by these CRs, see section 5]	12297 14222 14278 14504 15502 20205 20207 20264 20265 20279
2961	*Definite: Using value of uninitialized automatic object '%s'.*	13854 14372
2971	*Definite: Passing address of uninitialized object '%s' to a function parameter declared as a pointer to const.*	14307 14372
2983	*This assignment is redundant. The value of this object is never subsequently used.*	14221
2984	*This operation is redundant. The value of the result is always '%1s'.*	20284
2985	*This operation is redundant. The value of the result is always that of the left-hand operand.*	20284
3006	*This function contains a mixture of in-line assembler statements and C statements.*	13261 13574
3389	*Extra parentheses recommended to clarify the ordering of a % operator and another arithmetic operator (* / % + -).*	13037
3391	*Extra parentheses recommended. A conditional operation is the operand of another conditional operator.*	13037

3392	*Extra parentheses recommended. A shift, relational or equality operation is the operand of a second identical operator.*	13037
3393	*Extra parentheses recommended. An arithmetic operation (* / + -) is the operand of a different operator with the same precedence.*	13037
3394	*Extra parentheses recommended. A shift, relational or equality operation is the operand of a different operator with the same precedence.*	13037
3395	*Extra parentheses recommended. A * or / operation is the operand of a + or - operator.*	13037
3396	*Extra parentheses recommended. A binary operation is the operand of a conditional operator.*	13037
3397	*3397: Extra parentheses recommended. A binary operation is the operand of a binary operator with different precedence.*	13037
3398	*Extra parentheses recommended. A function call, array subscript, or member operation is the operand of a logical && or ||.*	13037
3399	*Extra parentheses recommended. A unary operation is the operand of a logical && or ||.*	13037
3442	*Operator other than & (address-of) or = (assignment) applied to a volatile object.*	13714
3664	*Using a dot operator to access an individual bit is a language extension.*	12297
3892	*The result of this cast is implicitly converted to another type.*	13615
4700	*Metric value out of threshold range: %s.*	11239 12306 14432

5. DATAFLOW MESSAGES

The following table lists all the dataflow analysis messages, whose behaviour can be indirectly affected by CRs 12297, 14222, 14278, 14504, 15502, 20205, 20207, 20264, 20265 and 20279:

Msg	Message Text
2740	*This loop controlling expression is a constant expression and its value is 'true'.*
2741	*This 'if' controlling expression is a constant expression and its value is 'true'.*
2742	*This 'if' controlling expression is a constant expression and its value is 'false'.*
2743	*This 'do - while' loop controlling expression is a constant expression and its value is 'false'. The loop will only be executed once.*
2744	*This 'while' or 'for' loop controlling expression is a constant expression and its value is 'false'. The loop will not be entered.*
2750	*Internal dataflow problem. Dataflow analysis continues with the next function. Please inform Programming Research.*
2751	*This function is too complex. Dataflow analysis continues with the next function.*
2752	*This '%1s' results in the function being too complex. Dataflow analysis continues with the next function.*
2753	*As a result of error message '%s', dataflow analysis of the remainder of this function is not possible.*
2754	*As a result of error message '%s', dataflow analysis of the remainder of this translation unit is not possible.*
2755	*Analysis time of function '%1s' has exceeded the configured maximum: '%2sms'. Dataflow analysis continues with the next function.*
2756	*Could not expand function call to '%1s' with maximum '-po df::inter' value.*
2757	*Could not analyse function '%1s'. Try a smaller '-po df::inter' value?*
2771	*Definite: These pointers address different objects.*
2772	*Apparent: These pointers address different objects.*
2773	*Suspicious: These pointers address different objects.*
2776	*Definite: Copy between overlapping objects.*
2777	*Apparent: Copy between overlapping objects.*
2778	*Suspicious: Copy between overlapping objects.*
2790	*Constant: Right hand operand of shift operator is negative or too large.*
2791	*Definite: Right hand operand of shift operator is negative or too large.*
2792	*Apparent: Right hand operand of shift operator is negative or too large.*
2793	*Suspicious: Right hand operand of shift operator is negative or too large.*
2800	*Constant: Overflow in signed arithmetic operation.*
2801	*Definite: Overflow in signed arithmetic operation.*
2802	*Apparent: Overflow in signed arithmetic operation.*
2803	*Suspicious: Overflow in signed arithmetic operation.*
2810	*Constant: Dereference of NULL pointer.*
2811	*Definite: Dereference of NULL pointer.*
2812	*Apparent: Dereference of NULL pointer.*
2813	*Suspicious: Dereference of NULL pointer.*
2814	*Possible: Dereference of NULL pointer.*
2820	*Constant: Arithmetic operation on NULL pointer.*
2821	*Definite: Arithmetic operation on NULL pointer.*
2822	*Apparent: Arithmetic operation on NULL pointer.*
2823	*Suspicious: Arithmetic operation on NULL pointer.*
2824	*Possible: Arithmetic operation on NULL pointer.*
2830	*Constant: Division by zero.*

2831	*Definite: Division by zero.*
2832	*Apparent: Division by zero.*
2833	*Suspicious: Division by zero.*
2834	*Possible: Division by zero.*
2840	*Constant: Dereference of an invalid pointer value.*
2841	*Definite: Dereference of an invalid pointer value.*
2842	*Apparent: Dereference of an invalid pointer value.*
2843	*Suspicious: Dereference of an invalid pointer value.*
2845	*Constant: Maximum number of characters to be written is larger than the target buffer size.*
2846	*Definite: Maximum number of characters to be written is larger than the target buffer size.*
2847	*Apparent: Maximum number of characters to be written is larger than the target buffer size.*
2848	*Suspicious: Maximum number of characters to be written is larger than the target buffer size.*
2850	*Constant: Implicit conversion to a signed integer type of insufficient size.*
2851	*Definite: Implicit conversion to a signed integer type of insufficient size.*
2852	*Apparent: Implicit conversion to a signed integer type of insufficient size.*
2853	*Suspicious: Implicit conversion to a signed integer type of insufficient size.*
2855	*Constant: Casting to a signed integer type of insufficient size.*
2856	*Definite: Casting to a signed integer type of insufficient size.*
2857	*Apparent: Casting to a signed integer type of insufficient size.*
2858	*Suspicious: Casting to a signed integer type of insufficient size.*
2860	*Constant: Implementation-defined value resulting from left shift operation on expression of signed type.*
2861	*Definite: Implementation-defined value resulting from left shift operation on expression of signed type.*
2862	*Apparent: Implementation-defined value resulting from left shift operation on expression of signed type.*
2863	*Suspicious: Implementation-defined value resulting from left shift operation on expression of signed type.*
2870	*Infinite loop construct with constant control expression.*
2871	*Infinite loop identified.*
2872	*This loop, if entered, will never terminate.*
2877	*This loop will never be executed more than once.*
2880	*This code is unreachable.*
2881	*The code in this 'default' clause is unreachable.*
2882	*This 'switch' statement will bypass the initialization of local variables.*
2883	*This 'goto' statement will always bypass the initialization of local variables.*
2887	*Function 'main' ends with an implicit 'return' statement.*
2888	*This function has been declared with a non-void 'return' type but ends with an implicit 'return ;' statement.*
2889	*This function has more than one 'return' path.*
2890	*Constant: Negative value implicitly converted to an unsigned type.*
2891	*Definite: Negative value implicitly converted to an unsigned type.*
2892	*Apparent: Negative value implicitly converted to an unsigned type.*
2893	*Suspicious: Negative value implicitly converted to an unsigned type.*
2895	*Constant: Negative value cast to an unsigned type.*
2896	*Definite: Negative value cast to an unsigned type.*
2897	*Apparent: Negative value cast to an unsigned type.*
2898	*Suspicious: Negative value cast to an unsigned type.*
2900	*Constant: Positive integer value truncated by implicit conversion to a smaller unsigned type.*
2901	*Definite: Positive integer value truncated by implicit conversion to a smaller unsigned type.*
2902	*Apparent: Positive integer value truncated by implicit conversion to a smaller unsigned type.*

2903	*Suspicious: Positive integer value truncated by implicit conversion to a smaller unsigned type.*
2905	*Constant: Positive integer value truncated by cast to a smaller unsigned type.*
2906	*Definite: Positive integer value truncated by cast to a smaller unsigned type.*
2907	*Apparent: Positive integer value truncated by cast to a smaller unsigned type.*
2908	*Suspicious: Positive integer value truncated by cast to a smaller unsigned type.*
2910	*Constant: Wraparound in unsigned arithmetic operation.*
2911	*Definite: Wraparound in unsigned arithmetic operation.*
2912	*Apparent: Wraparound in unsigned arithmetic operation.*
2913	*Suspicious: Wraparound in unsigned arithmetic operation.*
2920	*Constant: Left shift operation on expression of unsigned type results in loss of high order bits.*
2921	*Definite: Left shift operation on expression of unsigned type results in loss of high order bits.*
2922	*Apparent: Left shift operation on expression of unsigned type results in loss of high order bits.*
2923	*Suspicious: Left shift operation on expression of unsigned type results in loss of high order bits.*
2930	*Constant: Computing an invalid pointer value.*
2931	*Definite: Computing an invalid pointer value.*
2932	*Apparent: Computing an invalid pointer value.*
2933	*Suspicious: Computing an invalid pointer value.*
2940	*Constant: Result of implicit conversion is only representable in a two's complement implementation.*
2941	*Definite: Result of implicit conversion is only representable in a two's complement implementation.*
2942	*Apparent: Result of implicit conversion is only representable in a two's complement implementation.*
2943	*Suspicious: Result of implicit conversion is only representable in a two's complement implementation.*
2945	*Constant: Result of cast is only representable in a two's complement implementation.*
2946	*Definite: Result of cast is only representable in a two's complement implementation.*
2947	*Apparent: Result of cast is only representable in a two's complement implementation.*
2948	*Suspicious: Result of cast is only representable in a two's complement implementation.*
2950	*Constant: Negative value used in array subscript or pointer arithmetic operation.*
2951	*Definite: Negative value used in array subscript or pointer arithmetic operation.*
2952	*Apparent: Negative value used in array subscript or pointer arithmetic operation.*
2953	*Suspicious: Negative value used in array subscript or pointer arithmetic operation.*
2961	*Definite: Using value of uninitialized automatic object '%s'.*
2962	*Apparent: Using value of uninitialized automatic object '%s'.*
2963	*Suspicious: Using value of uninitialized automatic object '%s'.*
2971	*Definite: Passing address of uninitialized object '%s' to a function parameter declared as a pointer to const.*
2972	*Apparent: Passing address of uninitialized object '%s' to a function parameter declared as a pointer to const.*
2973	*Suspicious: Passing address of uninitialized object '%s' to a function parameter declared as a pointer to const.*
2980	*The value of this function parameter is never used before being modified.*
2981	*This initialization is redundant. The value of this object is never used before being modified.*
2982	*This assignment is redundant. The value of this object is never used before being modified.*
2983	*This assignment is redundant. The value of this object is never subsequently used.*
2984	*This operation is redundant. The value of the result is always '%1s'.*

2985	*This operation is redundant. The value of the result is always that of the left-hand operand.*
2986	*This operation is redundant. The value of the result is always that of the right-hand operand.*
2990	*The value of this loop controlling expression is always 'true'.*
2991	*The value of this 'if' controlling expression is always 'true'.*
2992	*The value of this 'if' controlling expression is always 'false'.*
2993	*The value of this 'do - while' loop controlling expression is always 'false'. The loop will only be executed once.*
2994	*The value of this 'while' or 'for' loop controlling expression is always 'false'. The loop will not be entered.*
2995	*The result of this logical operation is always 'true'.*
2996	*The result of this logical operation is always 'false'.*

6. CR SUMMARY

The following table summarises the change requests (CRs) that have been implemented in QA·C 8.1.2.

Note: CRs are categorised into 3 types (column "T"):

C – A significant change has been implemented to existing behaviour.
F – A fix of a bug or problem feature.
N – New functionality has been introduced.

CR	T	Resolution
10909	F	A problem is fixed so that message 178 can also be generated in the case where the format string only specifies the precision and not the field width. *0178: Precision of format conversion specifier exceeds 509 characters.*
11239 12306 14432	F	It is now possible to specify floating point values when configuring messages for metric thresholds. It was previously documented that thresholds for the STCDN metric could be specified as a percentage. However, that was not the case. *4700: Metric value out of threshold range: %s.*
11240	C	Message 1300 is generated for all C++11 keywords that are not also defined in the C90 or C99 standard. In particular, the following additional keywords are now reported: alignas, alignof, char16_t, char32_t, constexpr, const_cast, decltype, dynamic_cast, explicit, export, mutable, namespace, noexcept, nullptr, reinterpret_cast, static_assert, static_cast, thread_local, typeid, typename, using. *1300: '%s' is a keyword in C++.*
11455	F	Conflicts among identifiers in the ordinary identifier namespace are detected not just among internal identifiers, but also between internal and external identifiers. *0778: Identifier matches other identifier(s) (e.g. '%s') in first 31 characters - program does not conform strictly to ISO:C90.* *0779: Identifier does not differ from other identifier(s) (e.g. '%s') within the specified number of significant characters.*
11854	F	The calculation of the size of structures containing bit fields is upgraded to reflect more closely the specification provided by: "*System V Application Binary Interface Intel386 (TM) - Architecture Processor Supplement, Fourth Edition*". Note: the calculation is implementation defined and the above specification covers most Unix based systems.
12297	F	Dataflow analysis is extended to cover the bit access notation, which is a language extension in the Renesas C compiler. *2700-3000:* [see section 5]. *3664: Using a dot operator to access an individual bit is a language extension.*

CR	T	Resolution
13037	F	Messages 3389, 3391, 3392, 3393, 3394, 3395, 3396, 3397, 3398 and 3399 are now located on the lower-precedence operator to which they refer, instead of at the beginning of the expression. *3389: Extra parentheses recommended to clarify the ordering of a % operator and another arithmetic operator (* / % + -).* *3391: Extra parentheses recommended. A conditional operation is the operand of another conditional operator.* *3392: Extra parentheses recommended. A shift, relational or equality operation is the operand of a second identical operator.* *3393: Extra parentheses recommended. An arithmetic operation (* / + -) is the operand of a different operator with the same precedence.* *3394: Extra parentheses recommended. A shift, relational or equality operation is the operand of a different operator with the same precedence.* *3395: Extra parentheses recommended. A * or / operation is the operand of a + or - operator.* *3396: Extra parentheses recommended. A binary operation is the operand of a conditional operator.* *3397: Extra parentheses recommended. A binary operation is the operand of a binary operator with different precedence.* *3398: Extra parentheses recommended. A function call, array subscript, or member operation is the operand of a logical && or ||.* *3399: Extra parentheses recommended. A unary operation is the operand of a logical && or ||.*
13261 13574	F	The parsing of GCC inline assembler syntax is extended so that the use of the volatile and goto keywords, either inside or outside macros, is properly handled. *1006: This in-line assembler construct is a language extension. The code has been ignored.* *3006: This function contains a mixture of in-line assembler statements and C statements.*
13438	C	Conflicts among identifiers are detected in all applicable namespaces, i.e. not just in the ordinary identifier namespace but also in the Tags, Label and Struct/Union Member namespaces. *778: Identifier matches other identifier(s) (e.g. '%s') in first 31 characters - program does not conform strictly to ISO:C90.* *779: Identifier does not differ from other identifier(s) (e.g. '%s') within the specified number of significant characters.*
13615	F	Message 3892 is no longer generated when an expression is cast to a type and then assigned to a bit-field of the same type. *3892: The result of this cast is implicitly converted to another type.*
13714	F	Message 3442 is no longer incorrectly generated when a volatile object is assigned to another object or used as an expression statement, or a pointer to a volatile object is de-referenced. *3442: Operator other than & (address-of) or = (assignment) applied to a volatile object.*
13755	F	Metrics STOPT, STTOT, STZIP, STHAL, STSHN, STDIF, STEFF, STVOL, STDEV, STBUG and STM21 are no longer corrupted when the length of the path of the file to be analysed exceeds 128 characters.
13854	F	Message 2961 is generated also in the case where the value of an uninitialized automatic object is used right after entering to a loop. *2961: Definite: Using value of uninitialized automatic object '%s'.*

CR	T	Resolution
14221	F	Message 2983 is generated also in the case where an assignment takes place on the right hand side of a logical operator and the resulting value is not used. *2983: This assignment is redundant. The value of this object is never subsequently used.*
14222 20264 20265 20279	F	The modelling of ternary and logical operators during dataflow analysis is improved so that the number of false positive "apparent" messages is reduced, and values for logical && and logical || and conditional operators in sub-messages are fixed *2700-3000:* [see section 5].
14278	F	On entering function main the initialization of global objects is taken into account for the purpose of dataflow analysis *2700-3000:* [see section 5].
14307	F	A problem is fixed so that message 2971 is generated also in the case where inter-function dataflow analysis is enabled and the address of an uninitialized object is passed to a pointer to const parameter to a function call that is inlined. *2971: Definite: Passing address of uninitialized object '%s' to a function parameter declared as a pointer to const.*
14372	F	A problem is fixed so that messages 2961 and 2971 are generated also in the case where uninitialized variables with floating point types are used. *2961: Definite: Using value of uninitialized automatic object '%s'.* *2971: Definite: Passing address of uninitialized object '%s' to a function parameter declared as a pointer to const.*
14504	F	A problem is fixed so that certain sub expressions that are part of the control expression of for or while statements are now also checked for iterations other than the first. *2700-3000:* [see section 5].
15502	F	A problem is fixed with floating point casts in constant expressions, which caused the generation of incorrect dataflow analysis messages under certain circumstances. *2700-3000:* [see section 5].
15602	C	The following new environments are supported: • Windows 7: 64 bit • Linux RHEL5: 64 bit • Solaris 10: 64 bit The following environments remain supported: • Windows 7: 32 bit • Linux RHEL5: 32 bit The following environments are no longer supported: • Windows XP: 32 bit • Solaris 8: 32 bit The functionality provided in the new 64 bit Windows and Linux environments is identical to the functionality provided in the corresponding 32 bit environments.
15642	F	A problem is fixed with the parsing of the 'asm' (assembly) language extension, which caused incorrect STST1, STST2 and STST3 metrics calculation under certain circumstances.
20046	F	In the Message Browser the rendering of the HTML pages that provide help for interpreting the QA·C messages is improved
20048	F	Issue in the GUI where the baseline is not applied when a project has been closed and reopened.
20058	F	A problem is fixed which caused a lock up during the analysis of a syntactically incorrect recursive structure definition.
20133	F	It is now possible to select official Visual Studio Names from the list of editors in the message browser. Visual Studio 2010 is now included in the list.

CR	T	Resolution
20176	F	Metric identifiers that are specified in the '-threshold' configuration option are recognised not only when they are upper case but also when they are lower case or mixed case.
20205	F	A problem is fixed with dataflow analysis, which caused incorrect messages to be generated when using global objects as parameters in inlined functions. *2700-3000:* [see section 5].
20207	F	A problem is fixed with dataflow analysis, which manifested when inlining calls to functions with volatile qualified return types. *2700-3000:* [see section 5].
20284	F	Messages 2984 and 2985 are generated also in cases where volatile pointer variables are involved. *2984: This operation is redundant. The value of the result is always '%1s'.* *2985: This operation is redundant. The value of the result is always that of the left-hand operand.*
20342	F	A problem is fixed with dataflow analysis, which manifested when inlining non returning functions on the left hand side of a comma expression.
20358	F	QA·C's average analysis speed when the -q (-quiet) option is applied to a large number of paths is significantly increased.