QA·C 8.1 SOURCE CODE ANALYZER

RELEASE NOTES

July 2012

This document lists the QA·C Source Code Analyzer 8.1 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 8.1 release of QA·C features a revision of the arithmetic type conversion analysis to provide more intuitive, easier to understand messages. There are a number of incremental improvements in parsing extended keywords as well as new analysis messages and numerous bug fixes. In addition the dataflow engine has been enhanced to understand pointer aliasing and perform inter-function within a TU analysis.

2. KEY ADDITIONAL FUNCTIONALITY

- Parsing support for
  - C99 type _Bool
  - C99 “inline” functions
  - GCC #include_next syntax
  - Source files in UTF-8 encoding
  - Renesas compiler inline assembler syntax
  - Renesas compiler default parameter value syntax
- Primary Analysis
  - Radical revisions to message system to improve analysis of expressions of arithmetic type
  - Improved enforcement of MISRA-C:2004 rules which use ‘underlying type’ concept
  - Other additional new analysis messages
  - Sundry bug-fixes
- Dataflow Analysis
  - Pointer alias analysis – tracking the state of an object which is accessed via a pointer.
  - Inter-function dataflow analysis within a single translation unit.
  - Some additional analysis improvements.
- GUI Modifications

A set of dataflow configuration files is provided which support simpler control of dataflow analysis. In the Windows GUI, a dataflow level can be selected using the new Dataflow Level selector or via the Dataflow Settings dialog from the Configuration menu. Alternatively, the user can tailor the individual dataflow settings from this Dataflow Settings dialog by creating a user-defined configuration file. The Unix GUI provides access to the individual dataflow settings via a Dataflow Settings dialog which is accessible from the main toolbar.

3. UNDERLYING TYPE / ESSENTIAL TYPE

Version 8.1 of QA·C introduces an extensive range of new messages associated with type usage and type conversion. These messages are intended ultimately to supersede many pre-existing messages which are now classified as obsolete. (Messages classified as obsolete are generally removed after 2 full releases of the product).

The new type messages refer extensively to the concept of "*essential type*" – a term which is closely based on the principles of "*underlying type*" first introduced in MISRA-C:2004. *Essential type* is simply a way of capturing the essential nature of an expression in a way that avoids some of the unfortunate anomalies which exist in the C language. In many instances, the essential type of an expression is defined to be identical to its *standard type* – the type defined in the C language.

By adopting "essential type" concepts, a static analysis tool is able to provide:

- A message system which is more intuitive
- A more comprehensive enforcement of MISRA-C coding guidelines.
- A reduction in the incidence of noisy messages
- A full discussion of essential type principles and the new QA·C messages is provided in the *QAC-8.1-EssentialTypeUsersGuide.pdf* document supplied with this release.

4. CHANGES TO DATAFLOW METRICS GENERATION

Metrics that are generated as part of the dataflow analysis have had their behaviour changed. When dataflow is disabled, either explicitly using -ed- or implicitly via a dataflow termination message (see Appendix B of the Dataflow Reference Guide), then in QA·C 8.1 no dataflow metrics will be written to the .met file. In QA·C 8.0 when dataflow was disabled the metrics appeared as 0 in the .met file. Furthermore, when QA·C 8.1 does generate the metrics they will be generated in a separate metric group from that of the metrics generated as part of the initial parser analysis.

The affected metrics are:

- STAV1 Average size of function statements (variant 1)
- STAV2 Average size of function statements (variant 2)
- STAV3 Average size of function statements (variant 3)
- STST1 Number of statements in function (variant 1)
- STST2 Number of statements in function (variant 2)
- STST3 Number of statements in function (variant 3)
- STRET Number of function return points
- STUNR Number of unreachable statements

5. NEW MESSAGES

The following table lists messages which are new in QA·C 8.1.

Msg	Message Text	CR
0318	*Redundant type qualifier used in cast.*	11300
0360	*An expression of pointer type is being converted to type _Bool on assignment.*	14059
0361	*An expression of pointer type is being cast to type _Bool.*	14059
0362	*An expression of essentially Boolean type is being cast to a pointer.*	14059
0570	*This switch case label of 'essential type' '%1s', is not consistent with a controlling expression of essential type '%2s'.*	14325
0571	*This switch case label of 'essential type' '%1s' is not consistent with a controlling expression which has an essential type of higher rank (%2s).*	14325
0572	*This switch case label of 'essential type' '%1s' is not consistent with a controlling expression which has an essential type of lower rank (%2s).*	14325
1045	*[E] Use of the #include_next preprocessing directive is a language extension.*	13407
1046	*[E] Function is being declared with default argument syntax. This is a language extension.*	13468
1047	*[C] Function is being declared with default argument syntax after a previous call to the function. This is not allowed.*	13468
1048	*[C] Default argument values are missing for some parameters in this function declaration. This is not allowed.*	13468
1055	*[C99] The keyword 'inline' has been used.*	10977
1056	*[C99] The keyword '_Bool' has been used.*	14059
1290	*An integer constant of 'essentially signed' type is being converted to unsigned type on assignment.*	11047
1291	*An integer constant of 'essentially unsigned' type is being converted to signed type on assignment.*	11047
1292	*An integer constant of 'essentially signed' type is being converted to type char on assignment.*	11047
1293	*An integer constant of 'essentially unsigned' type is being converted to type char on assignment.*	11047
1294	*An integer constant of 'essentially signed' type is being converted to type _Bool on assignment.*	11047
1295	*An integer constant of 'essentially unsigned' type is being converted to type _Bool on assignment.*	11047
1296	*An integer constant of 'essentially signed' type is being converted to enum type on assignment.*	11047
1297	*An integer constant of 'essentially unsigned' type is being converted to enum type on assignment.*	11047
1298	*An integer constant of 'essentially signed' type is being converted to floating type on assignment.*	11047
1299	*An integer constant of 'essentially unsigned' type is being converted to floating type on assignment.*	11047
1337	*Function defined with a variable number of parameters.*	13590
1800	*The %1s operand (essential type: '%2s') will be implicitly converted to a floating type, '%3s', in this arithmetic operation.*	14284
1802	*The %1s operand (essential type: '%2s') will be implicitly converted to a floating type, '%3s', in this relational operation.*	14284
1803	*The %1s operand (essential type: '%2s') will be implicitly converted to a floating type, '%3s', in this equality operation.*	14284

Msg	Message Text	CR
1804	*The %1s operand (essential type: '%2s') will be implicitly converted to a floating type, '%3s', in this conditional operation.*	14284
1810	*An operand of 'essentially character' type is being added to another operand of 'essentially character' type.*	14286
1811	*An operand of 'essentially character' type is being subtracted from an operand of 'essentially signed' type.*	14286
1812	*An operand of 'essentially character' type is being subtracted from an operand of 'essentially unsigned' type.*	14286
1813	*An operand of 'essentially character' type is being balanced with an operand of 'essentially floating' type in this arithmetic operation.*	14286
1820	*The %1s operand is non-constant and 'essentially signed' (%2s) but will be implicitly converted to an unsigned type (%3s) in this arithmetic operation.*	14281 14285
1821	*The %1s operand is non-constant and 'essentially signed' (%2s) but will be implicitly converted to an unsigned type (%3s) in this bitwise operation.*	14285
1822	*The %1s operand is non-constant and 'essentially signed' (%2s) but will be implicitly converted to an unsigned type (%3s) in this relational operation.*	14285
1823	*The %1s operand is non-constant and 'essentially signed' (%2s) but will be implicitly converted to an unsigned type (%3s) in this equality operation.*	14285
1824	*The %1s operand is non-constant and 'essentially signed' (%2s) but will be implicitly converted to an unsigned type (%3s) in this conditional operation.*	14285
1830	*The %1s operand is constant, 'essentially signed' (%2s) and negative but will be implicitly converted to an unsigned type (%3s) in this arithmetic operation.*	14285
1831	*The %1s operand is constant, 'essentially signed' (%2s) and negative but will be implicitly converted to an unsigned type (%3s) in this bitwise operation.*	14285
1832	*The %1s operand is constant, 'essentially signed' (%2s) and negative but will be implicitly converted to an unsigned type (%3s) in this relational operation.*	14285
1833	*The %1s operand is constant, 'essentially signed' (%2s) and negative but will be implicitly converted to an unsigned type (%3s) in this equality operation.*	14285
1834	*The %1s operand is constant, 'essentially signed' (%2s) and negative but will be implicitly converted to an unsigned type (%3s) in this conditional operation.*	14285
1840	*The %1s operand is constant, 'essentially signed' (%2s) and non-negative but will be implicitly converted to an unsigned type (%3s) in this arithmetic operation.*	14285
1841	*The %1s operand is constant, 'essentially signed' (%2s) and non-negative but will be implicitly converted to an unsigned type (%3s) in this bitwise operation.*	14285
1842	*The %1s operand is constant, 'essentially signed' (%2s) and non-negative but will be implicitly converted to an unsigned type (%3s) in this relational operation.*	14285
1843	*The %1s operand is constant, 'essentially signed' (%2s) and non-negative but will be implicitly converted to an unsigned type (%3s) in this equality operation.*	14285
1844	*The %1s operand is constant, 'essentially signed' (%2s) and non-negative but will be implicitly converted to an unsigned type (%3s) in this conditional operation.*	14285
1850	*The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this arithmetic operation.*	14281 14285
1851	*The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this bitwise operation.*	14285
1852	*The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this relational operation.*	14285
1853	*The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this equality operation.*	14285
1854	*The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this conditional operation.*	14285

Msg	Message Text	CR
1860	*The operands of this arithmetic operator are of different 'essential signedness' but will generate a result of type 'signed int'.*	14285
1861	*The operands of this bitwise operator are of different 'essential signedness' but will generate a result of type 'signed int'.*	14285
1862	*The operands of this relational operator are of different 'essential signedness' but will both be promoted to 'signed int' for comparison.*	14285
1863	*The operands of this equality operator are of different 'essential signedness' but will both be promoted to 'signed int' for comparison.*	14285
1864	*The 2nd and 3rd operands of this conditional operator are of different 'essential signedness'. The result will be in the promoted type 'signed int'.*	14285
1880	*The operands of this relational operator are expressions of different 'essential type' categories (%1s and %2s).*	14287
1881	*The operands of this equality operator are expressions of different 'essential type' categories (%1s and %2s).*	14287
1882	*The 2nd and 3rd operands of this conditional operator are expressions of different 'essential type' categories (%1s and %2s).*	14287
1890	*A composite expression of 'essentially signed' type (%1s) is being implicitly converted to a wider signed type, '%2s'.*	14303
1891	*A composite expression of 'essentially unsigned' type (%1s) is being implicitly converted to a wider unsigned type, '%2s'.*	14303
1892	*A composite expression of 'essentially floating' type (%1s) is being implicitly converted to a wider floating type, '%2s'.*	14303
1893	*The 2nd and 3rd operands of this conditional operator are both 'essentially signed' ('%1s' and '%2s') but one is a composite expression of a narrower type than the other.*	14303
1894	*The 2nd and 3rd operands of this conditional operator are both 'essentially unsigned' ('%1s' and '%2s') but one is a composite expression of a narrower type than the other.*	14303
1895	*The 2nd and 3rd operands of this conditional operator are both 'essentially floating' ('%1s' and '%2s') but one is a composite expression of a narrower type than the other.*	14303
2000	*No 'else' clause exists for this 'if' statement.*	11075
2109	*Integral promotion : _Bool promoted to signed int.*	14059
2119	*Default argument promotion : _Bool promoted to signed int.*	14059
2120	*Integral promotion : unsigned bit-field promoted to signed int.*	10391
2122	*Integral promotion : unsigned bit-field promoted to unsigned int.*	10391
2124	*Integral promotion : signed bit-field promoted to signed int.*	10391
2130	*Default argument promotion : unsigned bit-field promoted to signed int.*	10391
2132	*Default argument promotion : unsigned bit-field promoted to unsigned int.*	10391
2134	*Default argument promotion : signed bit-field promoted to signed int.*	10391
2756	*Could not expand function call to '%1s' with maximum '-po df::inter' value.*	13872
2757	*Could not analyse function '%1s'. Try a smaller '-po df::inter' value?*	13872
2845	*Constant: Maximum number of characters to be written is larger than the target buffer size.*	14409
2846	*Definite: Maximum number of characters to be written is larger than the target buffer size.*	14409
2847	*Apparent: Maximum number of characters to be written is larger than the target buffer size.*	14409
2848	*Suspicious: Maximum number of characters to be written is larger than the target buffer size.*	14409
2984	*This operation is redundant. The value of the result is always '%1s'.*	13640
2985	*This operation is redundant. The value of the result is always that of the left-hand operand.*	13640

Msg	Message Text	CR
2986	*This operation is redundant. The value of the result is always that of the right-hand operand.*	13640
3004	*This integral constant expression is being interpreted as a NULL pointer constant.*	14121
3236	*[C] 'inline' may not be applied to function 'main'.*	10977
3237	*[C] inline function '%1s' has external linkage and is defining an object, '%2s', with static storage duration.*	10977
3238	*[C] inline function '%1s' has external linkage and is referring to an object, '%2s', with internal linkage.*	10977
3239	*[U] inline function '%1s' has external linkage, but is not defined within this translation unit.*	10977
3240	*inline function '%s' is being defined with external linkage.*	10977
3243	*inline function '%s' is also an 'external definition'.*	10977
3244	*[C] 'inline' may only be used in the declaration of a function identifier.*	10977
3491	*Using conditional operator in a macro.*	10947
3492	*Using conditional operator outside a macro.*	10947
3495	*Using a conditional operator in place of a selection statement.*	10947
4301	*An expression of 'essentially Boolean' type (%1s) is being cast to character type '%2s'.*	14231
4302	*An expression of 'essentially Boolean' type (%1s) is being cast to enum type '%2s'.*	14231
4303	*An expression of 'essentially Boolean' type (%1s) is being cast to signed type '%2s'.*	14231
4304	*An expression of 'essentially Boolean' type (%1s) is being cast to unsigned type '%2s'.*	14231
4305	*An expression of 'essentially Boolean' type (%1s) is being cast to floating type '%2s'.*	14231
4310	*An expression of 'essentially character' type (%1s) is being cast to Boolean type, '%2s'.*	14231
4312	*An expression of 'essentially character' type (%1s) is being cast to enum type, '%2s'.*	14231
4315	*An expression of 'essentially character' type (%1s) is being cast to floating type, '%2s'.*	14231
4320	*An expression of 'essentially enum' type (%1s) is being cast to Boolean type, '%2s'.*	14231
4322	*An expression of 'essentially enum' type (%1s) is being cast to a different enum type, '%2s'.*	14231
4325	*An expression of 'essentially enum' type (%1s) is being cast to floating type, '%2s'.*	14231
4330	*An expression of 'essentially signed' type (%1s) is being cast to Boolean type '%2s'.*	14231
4332	*An expression of 'essentially signed' type (%1s) is being cast to enum type, '%2s'.*	14231
4340	*An expression of 'essentially unsigned' type (%1s) is being cast to Boolean type '%2s'.*	14231
4342	*An expression of 'essentially unsigned' type (%1s) is being cast to enum type '%2s'.*	14231
4350	*An expression of 'essentially floating' type (%1s) is being cast to Boolean type '%2s'.*	14231
4351	*An expression of 'essentially floating' type (%1s) is being cast to character type '%2s'.*	14231
4352	*An expression of 'essentially floating' type (%1s) is being cast to enum type, '%2s'.*	14231
4390	*A composite expression of 'essentially signed' type (%1s) is being cast to a wider signed type, '%2s'.*	14146
4391	*A composite expression of 'essentially unsigned' type (%1s) is being cast to a wider unsigned type, '%2s'.*	14146
4392	*A composite expression of 'essentially floating' type (%1s) is being cast to a wider floating type, '%2s'.*	14146
4393	*A composite expression of 'essentially signed' type (%1s) is being cast to a different type category, '%2s'.*	14146
4394	*A composite expression of 'essentially unsigned' type (%1s) is being cast to a different type category, '%2s'.*	14146
4395	*A composite expression of 'essentially floating' type (%1s) is being cast to a different type category, '%2s'.*	14146
4397	*An expression which is the result of a ~ or << operation has not been cast to its essential type.*	11155

Msg	Message Text	CR
4398	*An expression which is the result of a ~ or << operation has been cast to a different essential type category.*	11155
4399	*An expression which is the result of a ~ or << operation has been cast to a wider type.*	11155
4401	*An expression of 'essentially Boolean' type (%1s) is being converted to character type, '%2s' on assignment.*	14234
4402	*An expression of 'essentially Boolean' type (%1s) is being converted to enum type, '%2s' on assignment.*	14234
4403	*An expression of 'essentially Boolean' type (%1s) is being converted to signed type, '%2s' on assignment.*	14234
4404	*An expression of 'essentially Boolean' type (%1s) is being converted to unsigned type, '%2s' on assignment.*	14234
4405	*An expression of 'essentially Boolean' type (%1s) is being converted to floating type, '%2s' on assignment.*	14234
4410	*An expression of 'essentially character' type (%1s) is being converted to Boolean type, '%2s' on assignment.*	14234
4412	*An expression of 'essentially character' type (%1s) is being converted to enum type, '%2s' on assignment.*	14234
4413	*An expression of 'essentially character' type (%1s) is being converted to signed type, '%2s' on assignment.*	14234
4414	*An expression of 'essentially character' type (%1s) is being converted to unsigned type, '%2s' on assignment.*	14234
4415	*An expression of 'essentially character' type (%1s) is being converted to floating type, '%2s' on assignment.*	14234
4420	*An expression of 'essentially enum' type (%1s) is being converted to Boolean type, '%2s' on assignment.*	14234
4421	*An expression of 'essentially enum' type (%1s) is being converted to character type, '%2s' on assignment.*	14234
4422	*An expression of 'essentially enum' type (%1s) is being converted to a different enum type, '%2s' on assignment.*	14234
4423	*An expression of 'essentially enum' type (%1s) is being converted to signed type, '%2s' on assignment.*	14234
4424	*An expression of 'essentially enum' type (%1s) is being converted to unsigned type, '%2s' on assignment.*	14234
4425	*An expression of 'essentially enum' type (%1s) is being converted to floating type, '%2s' on assignment.*	14234
4430	*An expression of 'essentially signed' type (%1s) is being converted to Boolean type, '%2s' on assignment.*	14234
4431	*An expression of 'essentially signed' type (%1s) is being converted to character type, '%2s' on assignment.*	14234
4432	*An expression of 'essentially signed' type (%1s) is being converted to enum type, '%2s' on assignment.*	14234
4434	*A non-constant expression of 'essentially signed' type (%1s) is being converted to unsigned type, '%2s' on assignment.*	14234 13036
4435	*A non-constant expression of 'essentially signed' type (%1s) is being converted to floating type, '%2s' on assignment.*	14234
4436	*A constant expression of 'essentially signed' type (%1s) is being converted to unsigned type, '%2s' on assignment.*	14234
4437	*A constant expression of 'essentially signed' type (%1s) is being converted to floating type, '%2s' on assignment.*	14234
4440	*An expression of 'essentially unsigned' type (%1s) is being converted to Boolean type, '%2s' on assignment.*	14234

Msg	Message Text	CR
4441	*An expression of 'essentially unsigned' type (%1s) is being converted to character type, '%2s' on assignment.*	14234
4442	*An expression of 'essentially unsigned' type (%1s) is being converted to enum type, '%2s' on assignment.*	14234
4443	*A non-constant expression of 'essentially unsigned' type (%1s) is being converted to a wider signed type, '%2s' on assignment.*	14234
4445	*An expression of 'essentially unsigned' type (%1s) is being converted to floating type, '%2s' on assignment.*	14234
4446	*A non-constant expression of 'essentially unsigned' type (%1s) is being converted to signed type, '%2s' on assignment.*	14234
4447	*A constant expression of 'essentially unsigned' type (%1s) is being converted to signed type, '%2s' on assignment.*	14234
4450	*An expression of 'essentially floating' type (%1s) is being converted to Boolean type, '%2s' on assignment.*	14234
4451	*An expression of 'essentially floating' type (%1s) is being converted to character type, '%2s' on assignment.*	14234
4452	*An expression of 'essentially floating' type (%1s) is being converted to enum type, '%2s' on assignment.*	14234
4453	*An expression of 'essentially floating' type (%1s) is being converted to signed type, '%2s' on assignment.*	14234
4454	*An expression of 'essentially floating' type (%1s) is being converted to unsigned type, '%2s' on assignment.*	14234
4460	*A non-constant expression of 'essentially signed' type (%1s) is being converted to narrower signed type, '%2s' on assignment.*	14236
4461	*A non-constant expression of 'essentially unsigned' type (%1s) is being converted to narrower unsigned type, '%2s' on assignment.*	14236
4462	*A non-constant expression of 'essentially floating' type (%1s) is being converted to narrower floating type, '%2s' on assignment.*	14236
4463	*A constant expression of 'essentially signed' type (%1s) is being converted to narrower signed type, '%2s' on assignment.*	14236
4464	*A constant expression of 'essentially unsigned' type (%1s) is being converted to narrower unsigned type, '%2s' on assignment.*	14236
4465	*A constant expression of 'essentially floating' type (%1s) is being converted to narrower floating type, '%2s' on assignment.*	14236
4470	*A non-constant expression of 'essentially signed' type (%1s) is being passed to a function parameter of wider signed type, '%2s'.*	14238
4471	*A non-constant expression of 'essentially unsigned' type (%1s) is being passed to a function parameter of wider unsigned type, '%2s'.*	14238
4472	*A non-constant expression of 'essentially floating' type (%1s) is being passed to a function parameter of wider floating type, '%2s'.*	14238
4480	*A non-constant expression of 'essentially signed' type (%1s) is being returned from a function defined with a wider signed return type, '%2s'.*	14238
4481	*A non-constant expression of 'essentially unsigned' type (%1s) is being returned from a function defined with a wider unsigned return type, '%2s'.*	14238
4482	*A non-constant expression of 'essentially floating' type (%1s) is being returned from a function defined with a wider floating return type, '%2s'.*	14238
4490	*A composite expression of 'essentially signed' type (%1s) is being converted to wider signed type, '%2s' on assignment.*	14239
4491	*A composite expression of 'essentially unsigned' type (%1s) is being converted to wider unsigned type, '%2s' on assignment.*	14239

Msg	Message Text	CR
4492	*A composite expression of 'essentially floating' type (%1s) is being converted to wider floating type, '%2s' on assignment.*	14239
4498	*An expression which is the result of a ~ or << operation has been converted to a different essential type category on assignment.*	11155
4499	*An expression which is the result of a ~ or << operation has been converted to a wider essential type on assignment.*	11155
4500	*An expression of 'essentially Boolean' type (%1s) is being used as an array subscript.*	14034 14145
4501	*An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s).*	14036 14145
4502	*An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).*	14035 14145
4503	*An expression of 'essentially Boolean' type (%1s) is being used as the left-hand operand of this shift operator (%2s).*	14034 14145
4504	*An expression of 'essentially Boolean' type (%1s) is being used as the right-hand operand of this shift operator (%2s).*	14034 14145
4505	*An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this relational operator (%3s).*	14034 14145
4507	*An expression of 'essentially Boolean' type (%1s) is being used as the operand of this increment/decrement operator (%2s).*	14280
4510	*An expression of 'essentially character' type (%1s) is being used as an array subscript.*	14145
4511	*An expression of 'essentially character' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s).*	14145
4512	*An expression of 'essentially character' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).*	14145
4513	*An expression of 'essentially character' type (%1s) is being used as the left-hand operand of this shift operator (%2s).*	14145
4514	*An expression of 'essentially character' type (%1s) is being used as the right-hand operand of this shift operator (%2s).*	14145
4517	*An expression of 'essentially character' type (%1s) is being used as the operand of this increment/decrement operator (%2s).*	14280
4518	*An expression of 'essentially character' type (%1s) is being used as the %2s operand of this logical operator (%3s).*	14145
4519	*An expression of 'essentially character' type (%1s) is being used as the first operand of this conditional operator (%2s).*	14145
4521	*An expression of 'essentially enum' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s).*	14145
4522	*An expression of 'essentially enum' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).*	14145
4523	*An expression of 'essentially enum' type (%1s) is being used as the left-hand operand of this shift operator (%2s).*	14145
4524	*An expression of 'essentially enum' type (%1s) is being used as the right-hand operand of this shift operator (%2s).*	14145
4527	*An expression of 'essentially enum' type is being used as the operand of this increment/decrement operator.*	14280
4528	*An expression of 'essentially enum' type (%1s) is being used as the %2s operand of this logical operator (%3s).*	14145
4529	*An expression of 'essentially enum' type (%1s) is being used as the first operand of this conditional operator (%2s).*	14145
4532	*An expression of 'essentially signed' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).*	14145

Msg	Message Text	CR
4533	*An expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s).*	14145
4534	*An expression of 'essentially signed' type (%1s) is being used as the right-hand operand of this shift operator (%2s).*	14145
4538	*An expression of 'essentially signed' type (%1s) is being used as the %2s operand of this logical operator (%3s).*	14145
4539	*An expression of 'essentially signed' type (%1s) is being used as the first operand of this conditional operator (%2s).*	14145
4542	*A non-negative constant expression of 'essentially signed' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).*	14145
4543	*A non-negative constant expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s).*	14145
4544	*A non-negative constant expression of 'essentially signed' type (%1s) is being used as the right-hand operand of this shift operator (%2s).*	14145
4548	*A non-negative constant expression of 'essentially signed' type (%1s) is being used as the %2s operand of this logical operator (%3s).*	14145
4549	*A non-negative constant expression of 'essentially signed' type (%1s) is being used as the first operand of this conditional operator (%2s).*	14145
4558	*An expression of 'essentially unsigned' type (%1s) is being used as the %2s operand of this logical operator (%3s).*	14145
4559	*An expression of 'essentially unsigned' type (%1s) is being used as the first operand of this conditional operator (%2s).*	14145
4568	*An expression of 'essentially floating' type (%1s) is being used as the %2s operand of this logical operator (%3s).*	14145
4569	*An expression of 'essentially floating' type (%1s) is being used as the first operand of this conditional operator (%2s).*	14145
4570	*The operand of this ~ operator has an 'essential type' which is narrower than type 'int'.*	11155
4571	*The left-hand operand of this << operator has an 'essential type' which is narrower than type 'int'.*	11155

6. REMOVED MESSAGES

Msg	Message Text	CR
4240	*The value of an 'effectively Boolean' expression is being assigned to an object of type 'char'.*	14033
4241	*The value of an 'effectively Boolean' expression is being passed to a function parameter of type 'char'.*	14033
4242	*The value of an 'effectively Boolean' expression is being returned from a function with a return type of 'char'.*	14033
4243	*The value of an 'effectively Boolean' expression is being assigned to an object of floating type.*	14033
4244	*The value of an 'effectively Boolean' expression is being passed to a function parameter of floating type.*	14033
4245	*The value of an 'effectively Boolean' expression is being returned from a function with a floating return type.*	14033

7. 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
0268	*[S] Comment open at end of translation unit.*	14288
0336	*Macro defined as an octal constant.*	14442
0339	*Octal constant used.*	14442
0671	*[C] Initializer for object of arithmetic type is not of arithmetic type.*	14333
0672	*[U] The initializer for a 'struct', 'union' or array is not enclosed in braces.*	13875
0684	*[C] Too many initializers.*	13875
0693	*Struct initializer is missing the optional {.*	13875 14333
0818	*[Q] Cannot find '%s' - Perhaps the appropriate search path was not given ?*	14506
1006	*[E] This in-line assembler construct is a language extension. The code has been ignored.*	11675
1037	*[E] Arrays of length zero are a language extension.*	14011
1256	*An integer constant suffixed with L is being converted to type signed or unsigned long long on assignment.*	11047 11328 14312
1257	*An integer constant suffixed with L or LL is being converted to a type of lower rank on assignment.*	11047 11328 14312
1264	*A suffixed floating constant is being converted to a different floating type on assignment.*	11047 14312
1265	*An unsuffixed floating constant is being converted to a different floating type on assignment.*	11047 14312
1266	*A floating constant is being converted to integral type on assignment.*	11047 14312
1276	*An integer constant is being converted to floating type on assignment.*	11047 14312
1411	*A constant expression of non-enum type is being passed as argument to a function parameter of enum type.*	14323
1412	*A constant expression of non-enum type is being assigned to an object of enum type.*	14323
1413	*A constant expression of non-enum type is being returned from a function with an enum return type.*	14323
1441	*A non-constant expression of non-enum type is being passed as argument to a function parameter of enum type.*	14323
1442	*A non-constant expression of non-enum type is being assigned to an object of enum type.*	14323
1443	*A non-constant expression of non-enum type is being returned from a function with an enum return type.*	14323
1475	*Range of possible enum values suggests this test is always true.*	15536
1476	*Range of possible enum values suggests this test is always false.*	15536
1477	*Object of enum type is being implicitly compared against zero in a controlling expression.*	15538
1478	*Object of an enum type which does not include a zero value, is being implicitly compared against zero in a controlling expression.*	15538
2100	*Integral promotion : unsigned char promoted to signed int.*	11462

2210	*Tab character encountered in this line.*	11821
2214	*Body of control statement is on the same line and is not enclosed within braces.*	11867
2752	*This '%1s' results in the function being too complex. Dataflow analysis continues with the next function.*	15537
2755	*Analysis time of function '%1s' has exceeded the configured maximum: '%2sms'. Dataflow analysis continues with the next function.*	14429
2801	*Definite: Overflow in signed arithmetic operation.*	14196
2813	*Suspicious: Dereference of NULL pointer.*	15519
2830	*Constant: Division by zero.*	12847 14474
2832	*Apparent: Division by zero.*	14419
2834	*Possible: Division by zero.*	14419
2841	*Definite: Dereference of an invalid pointer value.*	14235
2890	*Constant: Negative value implicitly converted to an unsigned type.*	13517 14624
2895	*Constant: Negative value cast to an unsigned type.*	13517 14624
2900	*Positive integer value truncated by implicit conversion to a smaller unsigned type.*	13517 14624
2905	*Positive integer value truncated by cast to a smaller unsigned type.*	13517 14624
2911	*Definite: Wraparound in unsigned arithmetic operation.*	14196
2931	*Definite: Computing an invalid pointer value.*	14235
2962	*Apparent: Using value of uninitialized automatic object '%s'.*	14365
2963	*Suspicious: Using value of uninitialized automatic object '%s'.*	14365
2972	*Apparent: Passing address of uninitialized object '%s' to a function parameter declared as a pointer to const.*	14365
2973	*Suspicious: Passing address of uninitialized object '%s' to a function parameter declared as a pointer to const.*	14365
2983	*This assignment is redundant. The value of this object is never subsequently used.*	13958 14211 14212
2984	*This operation is redundant. The value of the result is always '%1s'.*	14228
2985	*This operation is redundant. The value of the result is always that of the left-hand operand.*	14228
2986	*This operation is redundant. The value of the result is always that of the right-hand operand.*	14228
2961	*Definite: Using value of uninitialized automatic object '%s'.*	14314
2995	*The result of this logical operation is always 'true'.*	14649 15658
2996	*The result of this logical operation is always 'false'.*	14649 15658
3001	*Function has been declared with an empty parameter list.*	13167
3103	*Result of signed division or remainder operation may be implementation defined.*	12847
3104	*[S] #pragma '%s' has invalid arguments and has been ignored.*	11675
3199	*This assignment is redundant. The value of '%s' is never subsequently used.*	11788
3212	*This cast is redundant.*	13516
3317	*'#if...' not matched by '#endif' in included file. This is probably an error.*	12303
3318	*'#else'/'#elif'/'#endif' in included file matched '#if...' in parent file. This is probably an error.*	12303
3408	*'%s' has external linkage and is being defined without any previous declaration.*	13898

3447	*'%s' is being declared with external linkage but this declaration is not in a header file.*	13898
3799-3807, 3876,3877, 3810-3818, 3878,3879, 3821-3829, 3880,3881	*Implicit conversion:* **x** *to* **y**. Where **x** is one of float, double, long double and **y** is one of char, signed char, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, long long, unsigned long long.	14313
3999-4007, 4076,4077, 4010-4018, 4078,4079, 4021-4029, 4080,4081	**x** *value returned from* **y** *%s().* Where **x** is one of float, double, long double and **y** is one of char, signed char, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, long long, unsigned long long.	14313
4810	*Invalid annotation: tag '%1s' is not defined subsequently in this file.*	13819
4811	*The start of the range '%1s', starts and ends at the same location.*	13819
4812	*'%1s' not allowed here.*	13819
4820	*Annotation kind is expected.*	13819
4821	*Colon is expected.*	13819
4822	*Tag name is expected.*	13819
4823	*Annotation syntax error.*	13819
4824	*Invalid character in tag name.*	13819
4825	*Unexpected character.*	13819
4826	*Message specification is incomplete or missing.*	13819
4827	*Tag name is not allowed in the message specification of continuous suppression annotation.*	13819
4828	*Invalid usage of predefined location tag.*	13819
2778	*Suspicious: Copy between overlapping objects.*	15519
2793	*Suspicious: Right hand operand of shift operator is negative or too large.*	15519
2803	*Suspicious: Overflow in signed arithmetic operation.*	15519
2813	*Suspicious: Dereference of NULL pointer.*	15519
2823	*Suspicious: Arithmetic operation on NULL pointer.*	15519
2833	*Suspicious: Division by zero.*	15519
2843	*Suspicious: Dereference of an invalid pointer value.*	15519
2853	*Suspicious: Implicit conversion to a signed integer type of insufficient size.*	15519
2858	*Suspicious: Casting to a signed integer type of insufficient size.*	15519
2863	*Suspicious: Implementation-defined value resulting from left shift operation on expression of signed type.*	15519
2893	*Suspicious: Negative value implicitly converted to an unsigned type.*	15519
2898	*Suspicious: Negative value cast to an unsigned type.*	15519
2903	*Suspicious: Positive integer value truncated by implicit conversion to a smaller unsigned type.*	15519
2908	*Suspicious: Positive integer value truncated by cast to a smaller unsigned type.*	15519
2913	*Suspicious: Wraparound in unsigned arithmetic operation.*	15519
2923	*Suspicious: Left shift operation on expression of unsigned type results in loss of high order bits.*	15519

2933	*Suspicious: Computing an invalid pointer value.*	15519
2943	*Suspicious: Result of implicit conversion is only representable in a two's complement implementation.*	15519
2948	*Suspicious: Result of cast is only representable in a two's complement implementation.*	15519
2953	*Suspicious: Negative value used in array subscript or pointer arithmetic operation.*	15519

8. CR SUMMARY

The following table summarises the change requests (CRs) that have been implemented in QA·C 8.1. CRs have been 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.

Some CRs are associated with more than one category and, in some cases, the distinction between categories is a little indistinct. The classification should therefore be treated as a guide only.

CR	T	Resolution
10391 (1880)	N	New messages have been introduced to identify integral promotion and default argument promotion of bit-fields: *2120: Integral promotion : unsigned bit-field promoted to signed int.* *2122: Integral promotion : unsigned bit-field promoted to unsigned int.* *2124: Integral promotion : signed bit-field promoted to signed int.* *2130: Default argument promotion : unsigned bit-field promoted to signed int.* *2132: Default argument promotion : unsigned bit-field promoted to unsigned int.* *2134:Default argument promotion : signed bit-field promoted to signed int.*
10947 (1147)	N	New messages have been introduced to identify use of the conditional operator: *3491: Using conditional operator in a macro.* *3492: Using conditional operator outside a macro.* *3495: Using a conditional operator in place of a selection statement.*
10977	N	'inline' functions as specified in the ISO:C99 standard are now fully supported and the semantic behaviour is addressed with the following messages. *1055: [C99] The keyword 'inline' has been used.* *3236: [C] 'inline' may not be applied to function 'main'.* *3237: [C] inline function '%1s' has external linkage and is defining an object, '%2s', with static storage duration.* *3238: [C] inline function '%1s' has external linkage and is referring to an object, '%2s', with internal linkage.* *3239: [U] inline function '%1s' has external linkage, but is not defined within this translation unit.* *3240: inline function '%s' is being defined with external linkage.* *3243: inline function '%s' is also an 'external definition'.* *3244: [C] 'inline' may only be used in the declaration of a function identifier.*
11047	C/ N	The set of messages associated with assignment of literal constants has been rationalised and extended. The messages identify situations where a literal constant undergoes type conversion in an initialisation, assignment, function argument or function return context. Note also: • The text of the pre-existing messages has been modified. • The pre-existing messages previously were not generated on function arguments. See CR 14312. • The specification of messages 1256 and 1257 has been modified. See CR 11328 **Pre-existing Messages:** *1256: An integer constant suffixed with L is being converted to type signed or unsigned long long on assignment.* *1257: An integer constant suffixed with L or LL is being converted to a type of lower rank on assignment.*

CR	T	Resolution
		1264: A suffixed floating constant is being converted to a different floating type on assignment. *1265: An unsuffixed floating constant is being converted to a different floating type on assignment.* *1266: A floating constant is being converted to integral type on assignment.* *1276: An integer constant is being converted to floating type on assignment.* **New messages:** *1290: An integer constant of 'essentially signed' type is being converted to unsigned type on assignment.* *1291: An integer constant of 'essentially unsigned' type is being converted to signed type on assignment.* *1292: An integer constant of 'essentially signed' type is being converted to type char on assignment.* *1293: An integer constant of 'essentially unsigned' type is being converted to type char on assignment.* *1294: An integer constant of 'essentially signed' type is being converted to type _Bool on assignment.* *1295: An integer constant of 'essentially unsigned' type is being converted to type _Bool on assignment.* *1296: An integer constant of 'essentially signed' type is being converted to enum type on assignment.* *1297: An integer constant of 'essentially unsigned' type is being converted to enum type on assignment.* *1298: An integer constant of 'essentially signed' type is being converted to floating type on assignment.* *1299: An integer constant of 'essentially unsigned' type is being converted to floating type on assignment.*
11075	N	A new message has been introduced to identify an "if" statement which has no corresponding "else" clause. *2000: No 'else' clause exists for this 'if' statement.*
11155	N	New messages have been introduced to support enforcement of MISRA-C:2004 Rule 10.5. *4397: An expression which is the result of a ~ or << operation has not been cast to its essential type.* *4398: An expression which is the result of a ~ or << operation has been cast to a different essential type category.* *4399: An expression which is the result of a ~ or << operation has been cast to a wider type.* *4498: An expression which is the result of a ~ or << operation has been converted to a different essential type category on assignment.* *4499: An expression which is the result of a ~ or << operation has been converted to a wider essential type on assignment.* *4570: The operand of this ~ operator has an 'essential type' which is narrower than type 'int'.* *4571: The left-hand operand of this << operator has an 'essential type' which is narrower than type 'int'.*
11278	F	This CR is not implemented however it is superseded following changes in QA·C 8.1. Message 4121 not generated when expected. *4121: Cast of complex expression of integral type to wider type.* This message has been superseded. See CR 14146.
11300	N	New QA·C message introduced to identify redundant type qualifiers in casts *0318: Redundant type qualifier used in cast.*

CR	T	Resolution
11328	C	The specification and text of messages 1256 and 1257 has been modified. See also CR 11047 and CR 14312. **Previously:** *1256: Suffixed integer constant implicitly converted to larger integer type on assignment.* *1257: Suffixed integer constant implicitly converted to smaller integer type on assignment.* Message 1256 was generated when a) an L suffixed constant was converted to any type of higher rank or an unsigned type of the same rank (i.e. unsigned long) b) a U suffixed constant was converted to any type of higher rank c) a UL suffixed constant was converted to any type of higher rank Message 1257 was generated when a) an L suffixed constant being converted to any type of lower rank b) an LL suffixed constant being converted to any type of lower rank c) a U suffixed constant being converted to any type of lower rank or a signed type of the same rank (int) d) a UL suffixed constant being converted to any type of lower rank or a signed type of the same rank (long) e) a ULL suffixed constant being converted to any type of lower rank or a signed type of the same rank (long long) **QA·C 8.1** *1256: An integer constant suffixed with L is being converted to type signed or unsigned long long on assignment.* *1257: An integer constant suffixed with L or LL is being converted to a type of lower rank on assignment.* Message 1256 is now generated when a) an L suffixed constant is converted to type signed long long or unsigned long long b) a UL suffixed constant is converted to type signed long long or unsigned long long Message 1257 is now generated when a) an L suffixed constant is converted to any type of lower rank b) a UL suffixed constant is converted to any type of lower rank See also CR 11047.
11396	C	Handle UTF-8 Byte Order Mark (BOM) on Windows. This CR is implemented as part of CR 14176.
11405	F	This CR is not implemented however it is affected by changes in QA·C 8.1. The following messages have been superseded and are now classified as obsolete. See CR 14303: *4123: Implicit conversion: complex expression of type float to type double.* *4124: Implicit conversion: complex expression of type float to type long double.* *4125: Implicit conversion: complex expression of type double to type long double.*

CR	T	Resolution
11462	F	Integral promotion messages were generated incorrectly when performing pointer arithmetic or an array subscript operation. See also CR 12244. *2100: Integral promotion : unsigned char promoted to signed int.*
11675	N	Support is provided for parsing of source files which include functions containing inline assembler code as defined in Renesas compilers. These functions are declared in a #pragma statement of the form: #pragma inline_asm(func_name) *1006: [E] This in-line assembler construct is a language extension. The code has been ignored.* *3104: [S] #pragma '%s' has invalid arguments and has been ignored.*
11788	F	Message 3199 is now generated also in the case where an object is subject to a pre-increment or pre-decrement, and the value is not subsequently used. *3199: This assignment is redundant. The value of '%s' is never subsequently used.*
11821	C	Message 2210 is only generated when the option "–tabstop 0" is supplied. The text and specification of the message has been modified so that it is generated (once) on any line which includes a tab character anywhere within the line. Previously: *2210: Tab character encountered at the beginning of a line.* Now: *2210: Tab character encountered in this line.*
11867	F	A bug is fixed, so that the inclusion of a file whose last compound statement is a do while loop will no longer prevent the subsequent generation of message 2214 on the including file. *2214: Body of control statement is on the same line and is not enclosed within braces.*
11924	F	This CR is not implemented however it is superseded following changes in QA·C 8.1. Noisy instances of message 3760 are generated when comparing the value of an unsigned bitfield with an unsigned constant. Message 3760 is now classified as obsolete. New messages have been implemented under CR 14285. See also CRs 12089, 13391 and 13410. *3760: Implicit conversion: int to unsigned int.*
12089	F	This CR is not implemented however it is superseded following changes in QA·C 8.1. Noisy instances of message 3760 are generated when comparing the value of an unsigned bitfield with an unsigned constant. Message 3760 is now classified as obsolete. New messages have been implemented under CR 14285. See also CRs 11924, 13391 and 13410. *3760: Implicit conversion: int to unsigned int.*
12244	F	Integral promotion messages were generated incorrectly when performing pointer arithmetic or an array subscript operation. See also CR 11462.
12303	F	Message 3317 was not issued if the code at the end of the included file is located in an unreachable branch. Message 3318 was not issued if the included file contains a #elif or #else as well as the #endif. Message 3318 was not issued if the #else or #elif is in an included file but the #endif is in the parent file. *3317: '#if...' not matched by '#endif' in included file. This is probably an error.* *3318: '#else'/'#elif'/'#endif' in included file matched '#if...' in parent file. This is probably an error.*
12824	F	Messages Browser now correctly handles UNC paths in the message help.
12847	F	Messages 586/2830 and 3103 were not issued in compound assignment operations. *2830: Constant: Division by zero.* *3103: Result of signed division or remainder operation may be implementation defined.*

CR	T	Resolution
12998	F	"OV" was generated instead of "OE" for the "space" field in the met file DEFINE record for an enumeration constant.
13036	F	Addressed under CR 14234. Assignment of a non-constant signed expression to an unsigned bit-field was not identified. Now identified with: *4434: A non-constant expression of 'essentially signed' type (%1s) is being converted to unsigned type, '%2s' on assignment.*
13167	F	Message 3001 was not generated in the declaration of a function pointer with an empty parameter list. *3001: Function has been declared with an empty parameter list.*
13322	F	Inconsistent representation of pointer to void in met file. The met file symbolic type code representation of a "pointer to void" was sometimes "g" and sometimes "p." It is now always represented as "p.".
13378	C	Message Browser: In admin mode, the (-n, -nomsg) and (-o, -only) values were ignored.
13391	F	This CR is not implemented however it is affected by changes in QA·C 8.1. Noisy instances of message 3760 are generated when comparing the value of an unsigned bitfield with an unsigned constant. Message 3760 is now classified as obsolete. New messages have been implemented under CR 14285. See also CRs 11924,12089 and 13410. *3760: Implicit conversion: int to unsigned int.*
13407	N	The #include_next preprocessing directive supported by the GCC compiler is now supported. Use of this language extension is identified with message: *1045: [E] Use of the #include_next preprocessing directive is a language extension.*
13410	F	This CR is not implemented however it is superseded following changes in QA·C 8.1. Noisy instances of message 3760 are generated when comparing the value of an unsigned bitfield with an unsigned constant. Message 3760 is now classified as obsolete. New messages have been implemented under CR 14285. See also CRs 11924,12089 and 13391. *3760: Implicit conversion: int to unsigned int.*
13433	F	This CR is not implemented however it is affected by changes in QA·C 8.1. Message 0597 is generated incorrectly when type *char* is configured as unsigned, the controlling expression is of type char and the switch label is a character constant. The message has been superseded by message 0570. See CR 14325. *0570: This switch case label of 'essential type' '%1s', is not consistent with a controlling expression of essential type '%2s'.* *0597: Type of 'case' label expression is not consistent with type of controlling expression in 'switch' statement.*
13446	N	Addressed under CR 11047. Messages have been implemented to identify when an object of unsigned type is initialised with a constant of signed type (e.g. 0 rather than 0u).
13468	N	Support is now included for default argument syntax as defined in C++ and Renesas compilers. The following messages have been implemented: *1046: [E] Function is being declared with default argument syntax. This is a language extension.* *1047: [C] Function is being declared with default argument syntax after a previous call to the function. This is not allowed.* *1048: [C] Default argument values are missing for some parameters in this function declaration. This is not allowed.*
13516	C	Message 3212 was previously generated when applying a cast which was of the same type as the expression to which it was applied. Now the message is only generated if the 'essential type' of the expression also remains unchanged.

CR	T	Resolution
		3212: This cast is redundant.
13517	C	Noisy instances of messages 2890, 2895, 2900 and 2905 were generated when observing the requirements of MISRA-C:2004 Rule 10.5 – i.e. to apply a cast to the underlying type following a ~ or << operation. Messages 2890 and 2895 are no longer generated if the operand is essentially unsigned. Messages 2895 and 2905 are no longer generated unless the value represented in the essential type is truncated. *2890: Constant: Negative value implicitly converted to an unsigned type.* *2895: Constant: Negative value cast to an unsigned type.* *2900: Positive integer value truncated by implicit conversion to a smaller unsigned type.* *2905: Positive integer value truncated by cast to a smaller unsigned type.*
13527	F	The positioning of the message browser is improved to avoid situations where the window re-opens off screen after the screen configuration is changed.
13546	F	Windows GUI: there was a wrong application of a trailing backslash for a relative path at the current directory.
13590	N	A new message has been implemented to identify when a function is being defined with a variable number of parameters. *1337: Function defined with a variable number of parameters.*
13640	N	New messages have been implemented to identify when the result of an arithmetic operation is invariant. *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.*
13709	C	Improved checks are implemented for strings with formats passed to the C API (e.g. via sprintf, snprintf, strftime), based on calculation of minimum required buffer size.
13819	F	Annotation messages from QA·C did not have 'included from' information. As a result: (1) The messages were not displayed by errdsp for the first include and (2) The messages were not suppressed by the source level suppression. *4810: Invalid annotation: tag '%1s' is not defined subsequently in this file.* *4811: The start of the range '%1s', starts and ends at the same location.* *4812: '%1s' not allowed here.* *4820: Annotation kind is expected.* *4821: Colon is expected.* *4822: Tag name is expected.* *4823: Annotation syntax error.* *4824: Invalid character in tag name.* *4825: Unexpected character.* *4826: Message specification is incomplete or missing.* *4827: Tag name is not allowed in the message specification of continuous suppression annotation.* *4828: Invalid usage of predefined location tag.*
13872	N	Inter-function dataflow analysis is now conducted across all functions within a single translation unit.The following new messages have been introduced. *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?*
13875	F	QA·C was failing to parse certain forms of designated initializers. *0672: [U] The initializer for a 'struct', 'union' or array is not enclosed in braces.* *0684: [C] Too many initializers.*

CR	T	Resolution
		0693: Struct initializer is missing the optional {.
13898	F	Message 3408 was sometimes generated incorrectly on a function declaration. Message 3447 was not always generated when expected. *3408: '%s' has external linkage and is being defined without any previous declaration.* *3447: '%s' is being declared with external linkage but this declaration is not in a header file.*
13930	F	The message browser will no longer fail to obtain a licence when there are multiple servers and one of the servers is down.
13958	F	Message 2983 is no longer incorrectly generated when the address of a member is passed to an unknown function. *2983: This assignment is redundant. The value of this object is never subsequently used.*
13964	F	Messages that are hidden will remain hidden when the message browser is closed and re-opened.
14011	F	The __alignof__ operator was returning incorrect values. *1037: [E] Arrays of length zero are a language extension.*
14033	C	The following messages have been removed, since they have been superseded by messages 4401 and 4405 introduced in CR 14234: *4240: The value of an 'effectively Boolean' expression is being assigned to an object of type 'char'.* *4241: The value of an 'effectively Boolean' expression is being passed to a function parameter of type 'char'.* *4242: The value of an 'effectively Boolean' expression is being returned from a function with a return type of 'char'.* *4243: The value of an 'effectively Boolean' expression is being assigned to an object of floating type.* *4244: The value of an 'effectively Boolean' expression is being passed to a function parameter of floating type.* *4245: The value of an 'effectively Boolean' expression is being returned from a function with a floating return type.*
14034	N	New messages to identify misuse of an effectively Boolean expression in array subscript, relational and shift operations. *4500: An expression of 'essentially Boolean' type (%1s) is being used as an array subscript.* *4503: An expression of 'essentially Boolean' type (%1s) is being used as the left-hand operand of this shift operator (%2s).* *4504: An expression of 'essentially Boolean' type (%1s) is being used as the right-hand operand of this shift operator (%2s).* *4505: An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this relational operator (%3s).*
14035	N	New messages to identify misuse of an effectively Boolean expression in bitwise operations. *4502: An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).* [where %2s is either 'left-hand', 'right-hand' or, in the case of the bitwise complement operator is blank]
14036	N	New messages to identify misuse of an effectively Boolean expression in arithmetic operations. *4501: An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s).*

CR	T	Resolution
		[where %1s is either "left-hand", "right-hand" or an empty string "" and %2s is the relevant operator]
14058	F	Motif GUI: The Auto Create Window was too tall and so was truncated in low resolution screens.
14059	N	Support has been added for parsing of type _Bool as specified in ISO:C99. The following new messages have been implemented: *0360: An expression of pointer type is being converted to type _Bool on assignment.* *0361: An expression of pointer type is being cast to type _Bool.* *0362: An expression of essentially Boolean type is being cast to a pointer.* *1056: [C99] The keyword '_Bool' has been used.* *2109: Integral promotion : _Bool promoted to signed int.* *2119: Default argument promotion : _Bool promoted to signed int.*
14062	F	Motif GUI: Changed date format for file to match that of the Windows GUI.
14071	F	Dataflow was sometimes disabled for first function after exit from a header file.
14073	F	Windows GUI: A trailing '\' was required when specifying the starting directory for an Auto-create with sub-path output location.
14082	F	Motif GUI: Allow view source window size to be specified in the resource file (Japanese only).
14084	F	Windows GUI: Incorrect handling of trailing backslash for Auto Create in SJIS environments.
14121	N	A new message has been implemented to identify when an integral constant expression with value 0 is being interpreted as a null pointer constant. *3004: This integral constant expression is being interpreted as a NULL pointer constant.*
14145	N	New messages have been implemented to identify situations where the operand of an operator may be of inappropriate or unexpected type: *4500: An expression of 'essentially Boolean' type (%1s) is being used as an array subscript.* *4501: An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s).* *4502: An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).* *4503: An expression of 'essentially Boolean' type (%1s) is being used as the left-hand operand of this shift operator (%2s).* *4504: An expression of 'essentially Boolean' type (%1s) is being used as the right-hand operand of this shift operator (%2s).* *4505: An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this relational operator (%3s).* *4510: An expression of 'essentially character' type (%1s) is being used as an array subscript.* *4511: An expression of 'essentially character' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s).* *4512: An expression of 'essentially character' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).* *4513: An expression of 'essentially character' type (%1s) is being used as the left-hand operand of this shift operator (%2s).* *4514: An expression of 'essentially character' type (%1s) is being used as the right-hand operand of this shift operator (%2s).* *4518: An expression of 'essentially character' type (%1s) is being used as the %2s operand of this logical operator (%3s).* *4519: An expression of 'essentially character' type (%1s) is being used as the first*

CR	T	Resolution
		operand of this conditional operator (%2s). *4521: An expression of 'essentially enum' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s).* *4522: An expression of 'essentially enum' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).* *4523: An expression of 'essentially enum' type (%1s) is being used as the left-hand operand of this shift operator (%2s).* *4524: An expression of 'essentially enum' type (%1s) is being used as the right-hand operand of this shift operator (%2s).* *4528: An expression of 'essentially enum' type (%1s) is being used as the %2s operand of this logical operator (%3s).* *4529: An expression of 'essentially enum' type (%1s) is being used as the first operand of this conditional operator (%2s).* *4532: An expression of 'essentially signed' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).* *4533: An expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s).* *4534: An expression of 'essentially signed' type (%1s) is being used as the right-hand operand of this shift operator (%2s).* *4538: An expression of 'essentially signed' type (%1s) is being used as the %2s operand of this logical operator (%3s).* *4539: An expression of 'essentially signed' type (%1s) is being used as the first operand of this conditional operator (%2s).* *4542: A non-negative constant expression of 'essentially signed' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).* *4543: A non-negative constant expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s).* *4544: A non-negative constant expression of 'essentially signed' type (%1s) is being used as the right-hand operand of this shift operator (%2s).* *4548: A non-negative constant expression of 'essentially signed' type (%1s) is being used as the %2s operand of this logical operator (%3s).* *4549: A non-negative constant expression of 'essentially signed' type (%1s) is being used as the first operand of this conditional operator (%2s).* *4558: An expression of 'essentially unsigned' type (%1s) is being used as the %2s operand of this logical operator (%3s).* *4559: An expression of 'essentially unsigned' type (%1s) is being used as the first operand of this conditional operator (%2s).* *4568: An expression of 'essentially floating' type (%1s) is being used as the %2s operand of this logical operator (%3s).* *4569: An expression of 'essentially floating' type (%1s) is being used as the first operand of this conditional operator (%2s).*
14146	N	New messages have been implemented to identify when a complex expression is being cast to a wider essential type: *4390: A composite expression of 'essentially signed' type (%1s) is being cast to a wider signed type, '%2s'.* *4391: A composite expression of 'essentially unsigned' type (%1s) is being cast to a wider unsigned type, '%2s'.* *4392: A composite expression of 'essentially floating' type (%1s) is being cast to a wider floating type, '%2s'.* *4393: A composite expression of 'essentially signed' type (%1s) is being cast to a different type category, '%2s'.* *4394: A composite expression of 'essentially unsigned' type (%1s) is being cast to a*

CR	T	Resolution
		different type category, '%2s'. *4395: A composite expression of 'essentially floating' type (%1s) is being cast to a different type category, '%2s'.*
14176	C	Support has been implemented for parsing of files in UTF8 encoding.
14196	F	Dataflow generated false positive first iteration overflow/wraparound messages due to incorrect modelling of loops involving the != operator. *2801: Definite: Overflow in signed arithmetic operation.* *2911: Definite: Wraparound in unsigned arithmetic operation.*
14202	N	The Windows GUI is updated to support the inter-function analysis capability in dataflow.
14203	N	The Motif GUI is updated to support the inter-function analysis capability in dataflow.
14211	F	Dataflow did not issue message 2983 if the last use of a variable was in a pre-increment or pre-decrement expression. *2983: This assignment is redundant. The value of this object is never subsequently used.*
14212	F	Dataflow message 2983 was not generated where the assignment took place in the middle or right hand operand of a ternary operator. *2983: This assignment is redundant. The value of this object is never subsequently used.*
14228	F	Determining loop directions for a loop variable in dataflow sometimes resulted in an incorrect result for complex conditional expressions. *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.*
14231	N	New messages have been implemented to identify certain cast operations to a different essential type: *4301: An expression of 'essentially Boolean' type (%1s) is being cast to character type '%2s'.* *4302: An expression of 'essentially Boolean' type (%1s) is being cast to enum type '%2s'.* *4303: An expression of 'essentially Boolean' type (%1s) is being cast to signed type '%2s'.* *4304: An expression of 'essentially Boolean' type (%1s) is being cast to unsigned type '%2s'.* *4305: An expression of 'essentially Boolean' type (%1s) is being cast to floating type '%2s'.* *4310: An expression of 'essentially character' type (%1s) is being cast to Boolean type, '%2s'.* *4312: An expression of 'essentially character' type (%1s) is being cast to enum type, '%2s'.* *4315: An expression of 'essentially character' type (%1s) is being cast to floating type, '%2s'.* *4320: An expression of 'essentially enum' type (%1s) is being cast to Boolean type, '%2s'.* *4322: An expression of 'essentially enum' type (%1s) is being cast to a different enum type, '%2s'.* *4325: An expression of 'essentially enum' type (%1s) is being cast to floating type, '%2s'.* *4330: An expression of 'essentially signed' type (%1s) is being cast to Boolean type*

CR	T	Resolution
		'%2s'. *4332: An expression of 'essentially signed' type (%1s) is being cast to enum type, '%2s'.* *4340: An expression of 'essentially unsigned' type (%1s) is being cast to Boolean type '%2s'.* *4342: An expression of 'essentially unsigned' type (%1s) is being cast to enum type '%2s'.* *4350: An expression of 'essentially floating' type (%1s) is being cast to Boolean type '%2s'.* *4351: An expression of 'essentially floating' type (%1s) is being cast to character type '%2s'.* *4352: An expression of 'essentially floating' type (%1s) is being cast to enum type, '%2s'.*
14234	N	New messages have been implemented to identify implicit conversions to a different essential type which occur in assigning operations (i.e. initialization, assignment, function arguments, functions returns): *4401: An expression of 'essentially Boolean' type (%1s) is being converted to character type, '%2s' on assignment.* *4402: An expression of 'essentially Boolean' type (%1s) is being converted to enum type, '%2s' on assignment.* *4403: An expression of 'essentially Boolean' type (%1s) is being converted to signed type, '%2s' on assignment.* *4404: An expression of 'essentially Boolean' type (%1s) is being converted to unsigned type, '%2s' on assignment.* *4405: An expression of 'essentially Boolean' type (%1s) is being converted to floating type, '%2s' on assignment.* *4410: An expression of 'essentially character' type (%1s) is being converted to Boolean type, '%2s' on assignment.* *4412: An expression of 'essentially character' type (%1s) is being converted to enum type, '%2s' on assignment.* *4413: An expression of 'essentially character' type (%1s) is being converted to signed type, '%2s' on assignment.* *4414: An expression of 'essentially character' type (%1s) is being converted to unsigned type, '%2s' on assignment.* *4415: An expression of 'essentially character' type (%1s) is being converted to floating type, '%2s' on assignment.* *4420: An expression of 'essentially enum' type (%1s) is being converted to Boolean type, '%2s' on assignment.* *4421: An expression of 'essentially enum' type (%1s) is being converted to character type, '%2s' on assignment.* *4422: An expression of 'essentially enum' type (%1s) is being converted to a different enum type, '%2s' on assignment.* *4423: An expression of 'essentially enum' type (%1s) is being converted to signed type, '%2s' on assignment.* *4424: An expression of 'essentially enum' type (%1s) is being converted to unsigned type, '%2s' on assignment.* *4425: An expression of 'essentially enum' type (%1s) is being converted to floating type, '%2s' on assignment.* *4430: An expression of 'essentially signed' type (%1s) is being converted to Boolean type, '%2s' on assignment.* *4431: An expression of 'essentially signed' type (%1s) is being converted to character type, '%2s' on assignment.*

CR	T	Resolution
		4432: An expression of 'essentially signed' type (%1s) is being converted to enum type, '%2s' on assignment. *4434: A non-constant expression of 'essentially signed' type (%1s) is being converted to unsigned type, '%2s' on assignment.* *4435: A non-constant expression of 'essentially signed' type (%1s) is being converted to floating type, '%2s' on assignment.* *4436: A constant expression of 'essentially signed' type (%1s) is being converted to unsigned type, '%2s' on assignment.* *4437: A constant expression of 'essentially signed' type (%1s) is being converted to floating type, '%2s' on assignment.* *4440: An expression of 'essentially unsigned' type (%1s) is being converted to Boolean type, '%2s' on assignment.* *4441: An expression of 'essentially unsigned' type (%1s) is being converted to character type, '%2s' on assignment.* *4442: An expression of 'essentially unsigned' type (%1s) is being converted to enum type, '%2s' on assignment.* *4443: A non-constant expression of 'essentially unsigned' type (%1s) is being converted to a wider signed type, '%2s' on assignment.* *4445: An expression of 'essentially unsigned' type (%1s) is being converted to floating type, '%2s' on assignment.* *4446: A non-constant expression of 'essentially unsigned' type (%1s) is being converted to signed type, '%2s' on assignment.* *4447: A constant expression of 'essentially unsigned' type (%1s) is being converted to signed type, '%2s' on assignment.* *4450: An expression of 'essentially floating' type (%1s) is being converted to Boolean type, '%2s' on assignment.* *4451: An expression of 'essentially floating' type (%1s) is being converted to character type, '%2s' on assignment.* *4452: An expression of 'essentially floating' type (%1s) is being converted to enum type, '%2s' on assignment.* *4453: An expression of 'essentially floating' type (%1s) is being converted to signed type, '%2s' on assignment.* *4454: An expression of 'essentially floating' type (%1s) is being converted to unsigned type, '%2s' on assignment.*
14235	F	False positives 2841 and 2931 messages were generated when executing a loop controlled with a pointer variable where the operator used in the condition was !=. *2841: Definite: Dereference of an invalid pointer value.* *2931: Definite: Computing an invalid pointer value.*
14236	N	New messages have been implemented to identify implicit conversion of a non-constant expression to an essential type of the same type category but lower rank which occur in assigning operations (i.e. initialization, assignment, function arguments, functions returns): *4460: A non-constant expression of 'essentially signed' type (%1s) is being converted to narrower signed type, '%2s' on assignment.* *4461: A non-constant expression of 'essentially unsigned' type (%1s) is being converted to narrower unsigned type, '%2s' on assignment.* *4462: A non-constant expression of 'essentially floating' type (%1s) is being converted to narrower floating type, '%2s' on assignment.* *4463: A constant expression of 'essentially signed' type (%1s) is being converted to narrower signed type, '%2s' on assignment.* *4464: A constant expression of 'essentially unsigned' type (%1s) is being converted to narrower unsigned type, '%2s' on assignment.*

CR	T	Resolution
		4465: A constant expression of 'essentially floating' type (%1s) is being converted to narrower floating type, '%2s' on assignment.
14237	N	64 bit builds are provided for post-analysis and reports tools to extend the CMA analysis or report generation beyond the 3 GB Windows limit.
14238	N	New messages have been implemented to identify implicit widening type conversions occurring in function arguments and function return expressions: *4470: A non-constant expression of 'essentially signed' type (%1s) is being passed to a function parameter of wider signed type, '%2s'.* *4471: A non-constant expression of 'essentially unsigned' type (%1s) is being passed to a function parameter of wider unsigned type, '%2s'.* *4472: A non-constant expression of 'essentially floating' type (%1s) is being passed to a function parameter of wider floating type, '%2s'.* *4480: A non-constant expression of 'essentially signed' type (%1s) is being returned from a function defined with a wider signed return type, '%2s'.* *4481: A non-constant expression of 'essentially unsigned' type (%1s) is being returned from a function defined with a wider unsigned return type, '%2s'.* *4482: A non-constant expression of 'essentially floating' type (%1s) is being returned from a function defined with a wider floating return type, '%2s'.*
14239	N	New messages have been implemented to identify implicit conversion of a composite expression to a wider type in assigning operations (i.e. initialization, assignment, function arguments, functions returns): *4490: A composite expression of 'essentially signed' type (%1s) is being converted to wider signed type, '%2s' on assignment.* *4491: A composite expression of 'essentially unsigned' type (%1s) is being converted to wider unsigned type, '%2s' on assignment.* *4492: A composite expression of 'essentially floating' type (%1s) is being converted to wider floating type, '%2s' on assignment.*
14273	F	Message Browser: fixed a failure when a message on line 0 was selected.
14280	N	New messages have been implemented to identify usage of an increment or decrement operator on an object of inappropriate type: *4507: An expression of 'essentially Boolean' type (%1s) is being used as the operand of this increment/decrement operator (%2s).* *4517: An expression of 'essentially character' type (%1s) is being used as the operand of this increment/decrement operator (%2s).* *4527: An expression of 'essentially enum' type is being used as the operand of this increment/decrement operator.*
14281	C	Compound assignment operations may be considered to consist of 2 distinct operations, a binary operation followed by an assignment operation. The messages associated with misuse of these operators will be generated for the corresponding compound assignment operations. *1820: The %1s operand is non-constant and 'essentially signed' (%2s) but will be implicitly converted to an unsigned type (%3s) in this arithmetic operation.* *1850: The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this arithmetic operation.* etc.
14284	N	New messages have been implemented to identify operations which balance a floating operand with a signed or unsigned operand: *1800: The %1s operand (essential type: '%2s') will be implicitly converted to a floating type, '%3s', in this arithmetic operation.* *1802: The %1s operand (essential type: '%2s') will be implicitly converted to a floating type, '%3s', in this relational operation.* *1803: The %1s operand (essential type: '%2s') will be implicitly converted to a floating*

CR	T	Resolution
		type, '%3s', in this equality operation. *1804: The %1s operand (essential type: '%2s') will be implicitly converted to a floating type, '%3s', in this conditional operation.*
14285	N	New messages have been implemented to identify operations which balance signed and unsigned operands: *1820: The %1s operand is non-constant and 'essentially signed' (%2s) but will be implicitly converted to an unsigned type (%3s) in this arithmetic operation.* *1821: The %1s operand is non-constant and 'essentially signed' (%2s) but will be implicitly converted to an unsigned type (%3s) in this bitwise operation.* *1822: The %1s operand is non-constant and 'essentially signed' (%2s) but will be implicitly converted to an unsigned type (%3s) in this relational operation.* *1823: The %1s operand is non-constant and 'essentially signed' (%2s) but will be implicitly converted to an unsigned type (%3s) in this equality operation.* *1824: The %1s operand is non-constant and 'essentially signed' (%2s) but will be implicitly converted to an unsigned type (%3s) in this conditional operation.* *1830: The %1s operand is constant, 'essentially signed' (%2s) and negative but will be implicitly converted to an unsigned type (%3s) in this arithmetic operation.* *1831: The %1s operand is constant, 'essentially signed' (%2s) and negative but will be implicitly converted to an unsigned type (%3s) in this bitwise operation.* *1832: The %1s operand is constant, 'essentially signed' (%2s) and negative but will be implicitly converted to an unsigned type (%3s) in this relational operation.* *1833: The %1s operand is constant, 'essentially signed' (%2s) and negative but will be implicitly converted to an unsigned type (%3s) in this equality operation.* *1834: The %1s operand is constant, 'essentially signed' (%2s) and negative but will be implicitly converted to an unsigned type (%3s) in this conditional operation.* *1840: The %1s operand is constant, 'essentially signed' (%2s) and non-negative but will be implicitly converted to an unsigned type (%3s) in this arithmetic operation.* *1841: The %1s operand is constant, 'essentially signed' (%2s) and non-negative but will be implicitly converted to an unsigned type (%3s) in this bitwise operation.* *1842: The %1s operand is constant, 'essentially signed' (%2s) and non-negative but will be implicitly converted to an unsigned type (%3s) in this relational operation.* *1843: The %1s operand is constant, 'essentially signed' (%2s) and non-negative but will be implicitly converted to an unsigned type (%3s) in this equality operation.* *1844: The %1s operand is constant, 'essentially signed' (%2s) and non-negative but will be implicitly converted to an unsigned type (%3s) in this conditional operation.* *1850: The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this arithmetic operation.* *1851: The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this bitwise operation.* *1852: The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this relational operation.* *1853: The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this equality operation.* *1854: The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this conditional operation.* *1860: The operands of this arithmetic operator are of different 'essential signedness' but will generate a result of type 'signed int'.* *1861: The operands of this bitwise operator are of different 'essential signedness' but will generate a result of type 'signed int'.* *1862: The operands of this relational operator are of different 'essential signedness' but will both be promoted to 'signed int' for comparison.* *1863: The operands of this equality operator are of different 'essential signedness' but*

CR	T	Resolution
		will both be promoted to 'signed int' for comparison. *1864: The 2nd and 3rd operands of this conditional operator are of different 'essential signedness'. The result will be in the promoted type 'signed int'.*
14286	N	New messages have been implemented to identify certain operations which balance an operand of essentially character type: *1810: An operand of 'essentially character' type is being added to another operand of 'essentially character' type.* *1811: An operand of 'essentially character' type is being subtracted from an operand of 'essentially signed' type.* *1812: An operand of 'essentially character' type is being subtracted from an operand of 'essentially unsigned' type.* *1813: An operand of 'essentially character' type is being balanced with an operand of 'essentially floating' type in this arithmetic operation.*
14287	N	New messages have been implemented to identify specific operations which balance operands of different essential type category: *1880: The operands of this relational operator are expressions of different 'essential type' categories (%1s and %2s).* *1881: The operands of this equality operator are expressions of different 'essential type' categories (%1s and %2s).* *1882: The 2nd and 3rd operands of this conditional operator are expressions of different 'essential type' categories (%1s and %2s).*
14288	F	QA·C failed under certain conditions when parsing a file which finished with an open comment. *0268: [S] Comment open at end of translation unit.*
14303	N	New messages required to identify balancing operations which result in the implicit conversion of a composite expression to a wider type. *1890: A composite expression of 'essentially signed' type (%1s) is being implicitly converted to a wider signed type, '%2s'.* *1891: A composite expression of 'essentially unsigned' type (%1s) is being implicitly converted to a wider unsigned type, '%2s'.* *1892: A composite expression of 'essentially floating' type (%1s) is being implicitly converted to a wider floating type, '%2s'.* *1893: The 2nd and 3rd operands of this conditional operator are both 'essentially signed' ('%1s' and '%2s') but one is a composite expression of a narrower type than the other.* *1894: The 2nd and 3rd operands of this conditional operator are both 'essentially unsigned' ('%1s' and '%2s') but one is a composite expression of a narrower type than the other.* *1895: The 2nd and 3rd operands of this conditional operator are both 'essentially floating' ('%1s' and '%2s') but one is a composite expression of a narrower type than the other.*
14312	F	Messages associated with assignment of literal constants were not generated for function arguments. See also CR 11047 and CR 11328. *1256: An integer constant suffixed with L is being converted to type signed or unsigned long long on assignment.* *1257: An integer constant suffixed with L or LL is being converted to a type of lower rank on assignment.* *1264: A suffixed floating constant is being converted to a different floating type on assignment.* *1265: An unsuffixed floating constant is being converted to a different floating type on assignment.* *1266: A floating constant is being converted to integral type on assignment.*

CR	T	Resolution
		1276: An integer constant is being converted to floating type on assignment.
14313	F	'Standard' implicit conversion messages are no longer generated when a floating constant is assigned to an object of integer type. 3799-3807,3876,3877,3810-3818,3878,3879,3821-3829,3880,3881: *Implicit conversion:* **x** *to* **y**. 3999-4007,4076,4077,4010-4018,4078,4079,4021-4029,4080,4081: **x** *value returned from* **y** *%s()*. Where **x** is one of *float*, *double*, *long double* and **y** is one of *char*, *signed char*, *unsigned char*, *short*, *unsigned short*, *int*, *unsigned int*, *long*, *unsigned long*, *long long*, *unsigned long long*. These messages duplicated the function of message 1266. *1266: A floating constant is being converted to integral type on assignment.*
14314	C	Dataflow no longer generates message 2961 for usage of an object that has no members. *2961: Definite: Using value of uninitialized automatic object '%s'.*
14323	F	Incorrect messages were sometimes generated when assigning the value of a floating constant to an enum object: *1411: A constant expression of non-enum type is being passed as argument to a function parameter of enum type.* *1412: A constant expression of non-enum type is being assigned to an object of enum type.* *1413: A constant expression of non-enum type is being returned from a function with an enum return type.* *1441: A non-constant expression of non-enum type is being passed as argument to a function parameter of enum type.* *1442: A non-constant expression of non-enum type is being assigned to an object of enum type.* *1443: A non-constant expression of non-enum type is being returned from a function with an enum return type.*
14325	N	New messages have been implemented to identify when the type of a switch case label is not consistent with the essential type of the controlling expression. *0570: This switch case label of 'essential type' '%1s', is not consistent with a controlling expression of essential type '%2s'.* *0571: This switch case label of 'essential type' '%1s' is not consistent with a controlling expression which has an essential type of higher rank (%2s).* *0572: This switch case label of 'essential type' '%1s' is not consistent with a controlling expression which has an essential type of lower rank (%2s).*
14333	F	Incorrect error messages will no longer be generated on certain valid forms of designated initializers: *0671: [C] Initializer for object of arithmetic type is not of arithmetic type.* *0693: Struct initializer is missing the optional {.*
14365	F	The implementation of dataflow messages which identify the use of uninitialized data has been improved so that variable inter-dependency is now correctly modelled. *2962: Apparent: Using value of uninitialized automatic object '%s'.* *2963: Suspicious: Using value of uninitialized automatic object '%s'.* *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.*
14409	F	New messages have been implemented to identify potential problems when supplying string arguments to standard library functions. *2845: Constant: Maximum number of characters to be written is larger than the target*

CR	T	Resolution
		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.*
14419	F	Incorrect dataflow messages were generated for variables used in the right hand side of a logical operator in the condition of a branch statement. *2832: Apparent: Division by zero.* *2834: Possible: Division by zero.*
14429	F	When specifying large values for dataflow timeout where function_timeout was smaller than query_timeout, message 2755 was not always generated as expected. *2755: Analysis time of function '%1s' has exceeded the configured maximum: '%2sms'. Dataflow analysis continues with the next function.*
14442	F	Messages 0336 and 0339 were not generated for an octal constant which includes a suffix. *0336: Macro defined as an octal constant.* *0339: Octal constant used.*
14474	F	The evaluation of && and || operations in constant expressions was incorrect. *2830: Constant: Division by zero.*
14485	F	Memory usage when analysing deeply nested code in dataflow analysis has been reduced, thereby reducing the risk of failing to complete dataflow analysis.
14496	F	Memory usage has been modified to reduce the risk of analysis failure when analysing code with very large numbers of objects.
14501	F	The Windows GUI was not showing correct metrics values for file static entities when these metrics were calculated from the Cross-Module Analysis phase.
14503	F	A bug is fixed which caused the Motif GUI to crash in some cases where files were added and deleted from projects.
14506	F	A bug has been fixed in the include file search algorithm. *0818: [Q] Cannot find '%s' - Perhaps the appropriate search path was not given ?*
14517	N	Implemented automatic retry to acquire a licence from the licence server in case of communication failure, to address cases of timeout due to slow connections.
14545	F	Modifications have been made to the x_print utility to allow printing on all supported Unix platforms.
14624	F	Noisy instances of messages 2890, 2895, 2900 and 2905 were generated when observing the requirements of MISRA-C:2004 Rule 10.5 – i.e. to apply a cast to the underlying type following a ~ or << operation. Messages 2890 and 2895 are no longer generated if the operand is essentially unsigned. Messages 2895 and 2905 are no longer generated unless the value represented in the essential type is truncated. *2890: Constant: Negative value implicitly converted to an unsigned type.* *2895: Constant: Negative value cast to an unsigned type.* *2900: Positive integer value truncated by implicit conversion to a smaller unsigned type.* *2905: Positive integer value truncated by cast to a smaller unsigned type.*
14625	F	The metrics description in the Windows and Unix metrics browser is improved.
14639	F	The -q (-quiet) option will suppress messages also from files included using angle brackets and located in the directories that are specified with the –i option.
14649	F	Dataflow was incorrectly modeling values if multiple calls to same function modify them via global pointers.

CR	T	Resolution
		2995: The result of this logical operation is always 'true'. *2996: The result of this logical operation is always 'false'.*
14655	N	The Motif GUI is updated to support the UTF encoding option.
14657	N	The Windows GUI is updated to support the UTF encoding option.
15519	F	Dataflow suspicious messages will also be generated for variables that are modified within a loop body. *2778: Suspicious: Copy between overlapping objects.* *2793: Suspicious: Right hand operand of shift operator is negative or too large.* *2803: Suspicious: Overflow in signed arithmetic operation.* *2813: Suspicious: Dereference of NULL pointer.* *2823: Suspicious: Arithmetic operation on NULL pointer.* *2833: Suspicious: Division by zero.* *2843: Suspicious: Dereference of an invalid pointer value.* *2853: Suspicious: Implicit conversion to a signed integer type of insufficient size.* *2858: Suspicious: Casting to a signed integer type of insufficient size.* *2863: Suspicious: Implementation-defined value resulting from left shift operation on expression of signed type.* *2893: Suspicious: Negative value implicitly converted to an unsigned type.* *2898: Suspicious: Negative value cast to an unsigned type.* *2903: Suspicious: Positive integer value truncated by implicit conversion to a smaller unsigned type.* *2908: Suspicious: Positive integer value truncated by cast to a smaller unsigned type.* *2913: Suspicious: Wraparound in unsigned arithmetic operation.* *2923: Suspicious: Left shift operation on expression of unsigned type results in loss of high order bits.* *2933: Suspicious: Computing an invalid pointer value.* *2943: Suspicious: Result of implicit conversion is only representable in a two's complement implementation.* *2948: Suspicious: Result of cast is only representable in a two's complement implementation.* *2953: Suspicious: Negative value used in array subscript or pointer arithmetic operation.*
15536	F	Messages 1475 and 1476 are no longer classified as obsolete. *1475: Range of possible enum values suggests this test is always true.* *1476: Range of possible enum values suggests this test is always false.*
15537	F	A bug is fixed, which prevented some cases of forward goto jumping into loop from being detected. *2752: This '%1s' results in the function being too complex. Dataflow analysis continues with the next function.*
15538	F	Messages 1477 and 1478 are no longer classified as obsolete. *1477: Object of enum type is being implicitly compared against zero in a controlling expression.* *1478: Object of an enum type which does not include a zero value, is being implicitly compared against zero in a controlling expression.*
15547	F	This new constraint error message terminates dataflow analysis: 1048. This existing constraint error message now terminates dataflow analysis: 430. These new minor error messages terminate dataflow analysis: 1333, 3319. This existing minor error message now terminates dataflow analysis: 658. These new constraint error messages do not terminate dataflow analysis: 1047, 3236, 3244. These existing constraint error messages no longer terminate dataflow analysis: 232, 233, 261, 338, 427, 431, 436, 437, 446, 447, 448, 449, 450, 452, 454, 458, 466, 467,

CR	T	Resolution
		476, 477, 478, 481, 483, 484, 487, 495, 496, 513, 541, 546, 547, 550, 558, 559, 580, 588, 589, 590, 591, 616, 619, 620, 622, 627, 628, 629, 631, 638, 641, 644, 646, 649, 650, 651, 653, 655, 656, 659, 664, 665, 673, 675, 677, 683, 684, 690, 699, 708, 709, 736, 737, 738, 746, 747, 757, 766, 767, 768, 774, 775, 940, 941, 943, 944, 1023, 1024, 1025, 1033. The "*DF^2 - Deep Flow Dataflow for QA·C 8.1 – Reference Manual*" provides the details of all the messages that terminate dataflow analysis.
15596	F	The Windows GUI will correctly display function parameters of type unsigned char in the Relationship diagram.
15612	N	The Windows GUI will support conversion of project's dataflow information from QA·C 8.0 to QA·C 8.1 format.
15645	C	The format of metrics files is changed: dataflow metrics (STST1, STST2, STST3, STAV1, STAV2, STAV3, STRET and STUNR) will be logged separately from other metrics. Dataflow metrics will not be logged if dataflow is disabled.
15658	F	Invariant conditions of a top level logical OR or AND operator will be detected also when they occur on the right hand side of the operator: *2995: The result of this logical operation is always 'true'.* *2996: The result of this logical operation is always 'false'.*