PRQA
Programming Research

RELEASE NOTES

QA·C 9.0.1

March, 2016

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 , QA·C, QA·C++ and High Integrity C++ (HIC++) are trademarks of *Programming Research* Ltd.
"MISRA", "MISRA C" and "MISRA C++" are registered trademarks of MIRA Limited, held on behalf of the MISRA Consortium.
Yices is a registered trademark of SRI International.
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) 1932 888 080
by fax on +44 (0) 1932 888 081
or by webpage: www.programmingresearch.com/services/contact-support/

Contents

1 Introduction **1**

2 QA·C Changes Overview **2**
2.1 Functional Changes in QA·C 9.0.1 . . . 2
2.1.1 Dataflow . . . 2
2.1.1.1 Dataflow Timeout . . . 2
2.2 Functional Changes in QA·C 9.0.0 . . . 2
2.2.1 New Storage Format . . . 2
2.2.1.1 General . . . 2
2.2.1.2 Upgrading from a previous version . . . 2
2.2.2 Dataflow . . . 2
2.2.2.1 Inter Translation Unit Dataflow . . . 2
2.2.3 Internationalization Support . . . 3
2.2.3.1 General . . . 3
2.2.3.2 Specifying Paths and Filenames . . . 4
2.2.3.3 Encoding of source files . . . 4
2.2.4 Updates to Suppression Code Annotations . . . 5
2.2.4.1 General . . . 5
2.2.4.2 Continuous Suppressions and Include Directives: Ticket 11645 . . . 5
2.2.4.3 Messages generated against shared header files: Ticket 22550 . . . 5
2.2.4.4 Relaxation of Restrictions on Suppression Tag Names: Ticket 22552 . . . 6
2.2.4.5 Mapping of #pragma suppressions to comment annotations: Ticket 22553 . . . 7
2.2.4.6 Suppressing ALL messages: Ticket 22554 . . . 7
2.2.4.7 Changes to suppressions in force included header files . . . 8

3 QA·C Messages **9**
3.1 Messages for QA·C 9.0.1 . . . 9
3.1.1 New Messages . . . 9
3.1.2 Removed Messages . . . 9
3.1.3 Messages with Modified Behavior . . . 9
3.1.4 Message Text Changes . . . 9
3.2 Messages for QA·C 9.0.0 . . . 9
3.2.1 New Messages . . . 9
3.2.2 Removed Messages . . . 11
3.2.3 Messages with Modified Behavior . . . 12
3.2.4 Message Text Changes . . . 18

4 QA·C Ticket Summary **19**
4.1 Ticket Summary for QA·C 9.0.1 19
4.2 Ticket Summary for QA·C 9.0.0 19

List of Figures

List of Tables

2.1 Supported Character Sets 4
3.1 Messages Added for QA·C 9.0.0 9
3.2 Messages Modified for QA·C 9.0.0 12
4.1 Ticket Summary for QA·C 9.0.1 19
4.2 Ticket Summary for QA·C 9.0.0 20

1 Introduction

Version 9.0.1 release of QA·C is a minor release.

This document provides information on the feature additions as well as all feature fixes made in this release.

2 QA·C Changes Overview

2.1 Functional Changes in QA·C 9.0.1

2.1.1 Dataflow

2.1.1.1 Dataflow Timeout

Dataflow analysis will no longer unexpectedly time out on simple functions.

2.2 Functional Changes in QA·C 9.0.0

2.2.1 New Storage Format

2.2.1.1 General

In order to improve scalability and performance over large projects, QA·C stores diagnostic and suppression information in a database.

See the PRQA Framework release notes for more information.

2.2.1.2 Upgrading from a previous version

The database model for storing diagnostics is incompatible with the legacy `.err` output files. As a result, if upgrading from a previous version of QA·C a reanalysis of all output may be required. In particular baseline information will need to be regenerated.

2.2.2 Dataflow

2.2.2.1 Inter Translation Unit Dataflow

Previous versions of QA·C performed inter-function analysis but only within a single translation unit. As a result of the changes for New Storage Format, dataflow stores the function bodies of a translation unit so that they can be read in when processing other translation units.

For example one translation unit may contain:

```
// t1.cc
void f1 (int i)
```

```
{
  1 / i;
}
```

And the other:

```
// t2.cc
void f1 (int i);
void b1 ()
{

  int i = 0;
  f1(i);
}
```

After parsing the body of `f1`, dataflow stores the processed function body. Later, when analyzing `b1`, dataflow loads the function body for `f1`, enabling inter function dataflow to take place and resulting in a definite message being generated for the division by zero.

Cross translation unit inter-function analysis can be achieved by enabling `-prodoption df::cma` along with the desired setting for `-prodoption df::inter`. See the QA·C user manual for more information..

Order of Analysis:

In previous versions of QA·C, the order of analysis did not impact analysis results. However, when `-prodoption df::cma` is enabled the results for one translation may be different as functions called by this translation unit may not yet have been processed. For example, if `t2.cc` is analyzed before `t1.cc` then the definition of `f1` will not be available.

The option `--repeat=<n>` has been added to `qacli` to enable scripting of multiple analysis runs. Each subsequent run results in more functions being available for inlining. The total number of runs required depends on the project structure and the order that files are analyzed. See the Dataflow section in the QA·C manual for more information.

2.2.3 Internationalization Support

2.2.3.1 General

QA·C has been updated to encode strings internally in UTF-8 ensuring a better and wider support of locales. Furthermore the following indirect benefits have been gained:

- Temporary files are no longer generated as part of the trans-coding process.
- It is now possible to specify the name of a supported codec in the `-encoding` option, for example: `-en UTF-8`.

The following are examples of supported character set names[1] :

Table 2.1: Supported Character Sets

EUC-KR	EUC-JP	GB18030	GBK
ISO-2022-JP	ISO-8859-1	KOI8-U	KOI8-R
Shift_JIS	UTF-8	windows-1252	

2.2.3.2 Specifying Paths and Filenames

By default, QA·C reads command line parameters and the contents of `.via` files using the default system character set.

Paths that include characters not in the system character set can be specified to QA·C in a `.via` file encoded with one of the UTF encodings. The `.via` file should be saved with an appropriate BOM[2]. Most editors provide a mechanism for encoding and saving files with such a signature.

2.2.3.3 Encoding of source files

The QA·C configuration option `-encoding` allows a user to specify that the source files use a different character set to that of the system. This is useful where source code is located on a shared drive of a remote machine. In the past, QA·C provided support for different Japanese character sets using special shorthand identifiers:

- ASC: ASCII encoding, resulting in no input encoding being used.
- EUC: Japanese Extended Unix Codes.
- NEWJ: Japanese New JIS encoding.
- OLDJ: Japanese Old JIS encoding.
- NECJ: Japanese NEC JIS encoding.
- SJ: Japanese Shift-JIS encoding.

It is now possible to specify the name of any supported character set to QA·C using the `-encoding` configuration option:

```
-encoding "KOI8-U"
```

[1] Character sets may also have variants not listed, for example: ISO-8859-14, windows-1250, UTF-32LE, etc.

[2] Byte Order Mark

Note: The option is ignored for source files that use a BOM. These files will always be read using using the UTF variant specified by the BOM.

2.2.4 Updates to Suppression Code Annotations

2.2.4.1 General

Suppression annotations are now being stored alongside diagnostics in the New Storage Format. As part of this move, improvements and changes have been made to the underlying logic. This section describes the main areas of improvement.

2.2.4.2 Continuous Suppressions and Include Directives: Ticket 11645

Continuous suppressions were added as an alternative to the `#pragma PRQA_MESSAGES_ON/OFF` functionality. A feature of this is the ability to disable suppressions within a header without impacting the set of messages suppressed by the outer file. For example:

```
// source.cc
// PRQA S 100 ++
#include "header.h"
// Message 100 suppressed here


// header.h
// Message 100 suppressed here
// PRQA S 100 --
// Message 100 not suppressed here
```

A limitation of how previous versions of QA·C implemented this feature was that messages generated on the `#include` line itself were not suppressed.

The modelling of suppressions in QA·C has been improved resulting in this issue being fixed.

2.2.4.3 Messages generated against shared header files: Ticket 22550

From the outset, a primary goal of C++ was to maintain interoperability with C, especially the ability to call and use existing code and libraries written in C.

C++ programs achieve this by including `extern` blocks around the C declarations:

```
#if defined(__cplusplus)
 extern "C" {
```

```
#endif

/* ... */

#if defined(__cplusplus)
}
#endif
```

Where both QA·C and QA·C++ are used in a given project, it may arise that both tools produce analysis results for the same header file. If a message is to be suppressed for one of the tools, doing so with a plain suppression could result in the suppression of an unrelated message in the other tool:

```
inline void f(int * i, int * j)
{
  if (i && (++(*i)))   // PRQA S 3230
  {
    ++(*j);
  }
}
```

The above suppression, applies equally to the QA·C++ message:

```
3230: The right hand side of this operator has side effects.
```

and to the QA·C message:

```
3230: Address of automatic object assigned to local pointer
      with static storage duration.
```

To avoid such issues, the message specification syntax has been extended to allow for messages to be associated with the analyzer that generates them:

```
// PRQA S QAC( 1234 )
// PRQA S QACPP( 1234 )
```

2.2.4.4 Relaxation of Restrictions on Suppression Tag Names: Ticket 22552

In previous versions of QA·C, the tag name of a suppression was restricted such that it must start with a letter and then could only be formed of letters, numbers and underscores. In contrast, suppressions in QA·Verify can be formed of any characters including non ASCII characters and spaces.

It is now possible to specify a suppression tag as a quoted string, allowing tags to be formed as required:

```
// PRQA S:1 1234                    // Not OK - generates 4822
```

```
// PRQA S:"1" 1234                // OK
// PRQA S:"Deviation 1234" 1234   // OK
```

2.2.4.5 Mapping of #pragma suppressions to comment annotations: Ticket 22553

With the introduction of suppressions through comment based annotations, the use of `#pragma` based suppressions was deprecated.

In previous versions of QA·C, `#pragma` suppressions resulted in diagnostics being marked as suppressed as they were written out.

A `#pragma` suppression is now mapped to a comment based annotation with approximately the same meaning[3]. The suppression is now applied at display time and so will apply to diagnostics generated by other components in the analysis tool chain.

For example, the following `#pragma`:

```
#pragma PRQA_MESSAGES_OFF 1234
```

Is mapped to:

```
// PRQA S 1234 ++
```

Note: `#pragma` suppressions in previous versions of QA·C applied from the line after the directive. Suppression annotations include the line containing the comment. This may result in a difference where a message is both generated and suppressed by the `#pragma` directive:

```
                          /*  Old QAC  *  New QAC  */
#pragma PRQA_MESSAGES_OFF /*   3619    *           */
#pragma PRQA_MESSAGES_ON  /*           *   3619    */
```

2.2.4.6 Suppressing ALL messages: Ticket 22554

As part of mapping `#pragma` functionality, a new message specifier `ALL` is now supported in the comment annotation syntax. This can be used to specify that the annotation applies to any message[4] generated by the tools.

```
// PRQA S ALL ++    // Suppresses all messages from this line onwards
```

This new messages specifier can also be used for other forms of suppressions:

```
// PRQA S ALL L1    // Suppresses all messages from this line until location L1
```

[3] One significant difference is the special handling of continuous suppressions in `-forceinclude` files. See Changes to suppressions in force included header files and QA·C user manual for details

[4] With the exception of hard errors which can never be suppressed

```
// PRQA L:L1
```

2.2.4.7 Changes to suppressions in force included header files

In previous versions of QA·C, all annotation suppression appearning in a `-forceinclude` file were modeled as if they appeared on line 0 of the main source file[5], for example:

```
/* 1 */ fi.h
/* 2 */
/* 3 */ Single line suppression // PRQA S 1234
/* 4 */

/* 0 */        <--- Suppression moved to here and
/* 1 */ main.c      therefore applied to all of 'fi.h'
/* 2 */
```

For most types of suppression annotations this results in unexpected behavior. As part of migrating to the New Storage Format the behavior has been limited to apply to enabling continuous suppressions only, for example:

```
/* 1 */ fi.h
/* 2 */
/* 3 */ Single line suppression // PRQA S 1234
/* 4 */
/* 5 */ // PRQA S 4321 ++
/* 6 */
/* 7 */

/* 0 */ Include location for "fi.h"
/* 1 */ main.c <--- PRQA S 4321 ++ moved here
/* 2 */
```

In the above, messages 1234 is suppressed on line 3 of `fi.h` and message 4321 is suppressed from line 1 of `main.c` onwards.

[5] The same line used for the include location of the `-forceinclude` header file itself

3 QA·C Messages

3.1 Messages for QA·C 9.0.1

3.1.1 New Messages

There are no new messages for QA·C 9.0.1.

3.1.2 Removed Messages

There are no removed messages for QA·C 9.0.1.

3.1.3 Messages with Modified Behavior

There are no messages with modified behavior for QA·C 9.0.1.

3.1.4 Message Text Changes

There are no message text changes for QA·C 9.0.1.

3.2 Messages for QA·C 9.0.0

3.2.1 New Messages

The following table lists messages which are new in QA·C 9.0.0.

Table 3.1: Messages Added for QA·C 9.0.0

Msg. Id.	Description
3119	This statement has no side-effect - it can be removed. 22372 *Message 3112 will be split so that it will no longer be generated when the statement without side effects is composed of an expression which is cast to void. In these circumstances a new message 3119 will be generated instead. Consequently, message 3112 will map more precisely to MISRA C:2012 Rule 2.2 and message 3119 will report exceptions to Rule 2.2, where it is still possible and may still be beneficial to remove the statement without side effects.*

continued on next page

Table 3.1 – continued from previous page

Msg. Id.	Description
4580	An operand of 'essentially character' type is being added to another operand of 'essentially character' type. 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4581	An operand of 'essentially character' type is being subtracted from an operand of 'essentially signed' type. 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4582	An operand of 'essentially character' type is being subtracted from an operand of 'essentially unsigned' type. 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4583	An expression of 'essentially character' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4584	An expression of 'essentially character' type (%1s) is being used as the %2s operand of this bitwise operator (%3s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4585	An expression of 'essentially character' type (%1s) is being used as the left-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4586	An expression of 'essentially character' type (%1s) is being used as the right-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4587	An expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4588	A non-negative constant expression of 'essentially signed' type (%1s) is being used as the %2s operand of this bitwise operator (%3s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4589	A non-negative constant expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*

continued on next page

Table 3.1 – continued from previous page

Msg. Id.	Description
4590	A non-negative constant expression of 'essentially signed' type (%1s) is being used as the right-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4813	The location annotation is not preceded by a suppression annotation which refers to it. 22551 *New messages to highlight invalid location annotations.*
4814	A location annotation cannot itself have a location specifier. 22551 *New messages to highlight invalid location annotations.*
4815	The specified product name is invalid. 22550 *For annotations in a shared C/C++ header file it is now possible to specify if the message is for QAC or QAC++. See Updates to Suppression Code Annotations.*
4829	Invalid character in message specifier. 22550 *For annotations in a shared C/C++ header file it is now possible to specify if the message is for QAC or QAC++. See Updates to Suppression Code Annotations.*
4830	Unexpected left bracket in message specifier. 22550 *For annotations in a shared C/C++ header file it is now possible to specify if the message is for QAC or QAC++. See Updates to Suppression Code Annotations.*
4831	Unexpected right bracket in message specifier. 22550 *For annotations in a shared C/C++ header file it is now possible to specify if the message is for QAC or QAC++. See Updates to Suppression Code Annotations.*
4832	Unexpected left bracket in tag name. 22550 *For annotations in a shared C/C++ header file it is now possible to specify if the message is for QAC or QAC++. See Updates to Suppression Code Annotations.*
4833	Unexpected right bracket in tag name. 22550 *For annotations in a shared C/C++ header file it is now possible to specify if the message is for QAC or QAC++. See Updates to Suppression Code Annotations.*
4834	Expected left bracket in product specifier. 22550 *For annotations in a shared C/C++ header file it is now possible to specify if the message is for QAC or QAC++. See Updates to Suppression Code Annotations.*
4835	Expected right bracket in product specifier. 22550 *For annotations in a shared C/C++ header file it is now possible to specify if the message is for QAC or QAC++. See Updates to Suppression Code Annotations.*

3.2.2 Removed Messages

There are no removed messages for QA·C 9.0.0.

3.2.3 Messages with Modified Behavior

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

Table 3.2: Messages Modified for QA·C 9.0.0

Msg. Id.	Description
0543	[U] 'void' expressions have no value and may not be used in expressions. 13237 *(1) Message 0543 will also be generated when a void expression is the operand of a '!' or (prefix or postfix) '++' or '–' (unary) operator or the 1st operand of a '? :' operator or the left or right operand of a '%', '<<', '>>', '|', '&', '^', '||', '&&', '-', '+', '*', '/', '*=', '/=', '<<=', '&=', '|=', '^=' or '%=' (binary) operator or the right operand of a '+=' operator. (2) Message 0543 will also be generated when a void expression is cast to a non void type. (3) Message 0543 will no longer be incorrectly generated when a void expression is the 3rd operand of a conditional (ternary) operator whose result is not used. (4) Message 0543 will always be shown against the applicable void expression (instead of being shown against outermost containing expressions). (5) If an expression contains multiple void sub-expressions, message 0543 will always be generated multiple times, once for each void sub-expression. (Note that even if the scope of message 0543 will be significantly extended, all the newly identified situations for the generation of 0543 are still leading to the additional generation of one or more constraint error messages such as: 0436, 0446, 0451, 0453, 0456, 0466, 0467, 0468, 0469, 0481, 0493, 0494, 0495, 0496, 0536, 0537, 0540, 0541, 0542, 0555, 0557, 0559, 0560, 0563, 0564, 0565, 0746, 0756, 0758. Message 0543 will therefore provide additional useful information on issues that are already being otherwise detected by QAC.)*

continued on next page

Table 3.2 – continued from previous page

Msg. Id.	Description
0616	[C] Illegal combination of type specifiers or storage class specifiers. 11261 *Message 0616 will be generated (instead of false positive 1027) when the type specifiers 'long' or 'long long' are preceded by other incompatible type specifiers such as 'char' or 'short'. Noise will also be reduced, within the same declaration: (1) message 0616 will only be generated for the first incompatible type specifier and the first incompatible storage class specifier encountered and (2) message 1027 will be generated at most once and only if message 0616 is not already generated for incompatibilities with any preceding type specifiers.*
0634	[I] Bit-fields in this struct/union have not been declared explicitly as unsigned or signed. 12656 *Messages 0634, 0635 and 3663 will be generated where applicable even if the bit-fields happen to be defined within non-tagged unions.*
0635	[E] Bit-fields in this struct/union have been declared with types other than int, signed int, unsigned int or _Bool. 12656 *Messages 0634, 0635 and 3663 will be generated where applicable even if the bit-fields happen to be defined within non-tagged unions.*
0671	[C] Initializer for object of arithmetic type is not of arithmetic type. 11975 *Some existing checks on initialisers and expressions will be extended as applicable to situations where objects of automatic storage duration are initialised using non constant expressions. Messages such as for instance 0671, 0672, 0692, 4434, 4453, etc. may be generated to notify potential issues in these circumstances.*
0672	[U] The initializer for a 'struct', 'union' or array is not enclosed in braces. 11975 *Some existing checks on initialisers and expressions will be extended as applicable to situations where objects of automatic storage duration are initialised using non constant expressions. Messages such as for instance 0671, 0672, 0692, 4434, 4453, etc. may be generated to notify potential issues in these circumstances.*
0675	[C] Initializer is not of compatible 'struct'/'union' type. 22286 *Constraint error message 0675 will no longer be incorrectly generated under certain circumstances involving nested compound literals.*
0684	[C] Too many initializers. 22290 *Message 0684 will no longer be incorrectly generated in certain situations where initialisers contain nested compound literals.*

continued on next page

Table 3.2 – continued from previous page

Msg. Id.	Description
0692	Union initializer is missing the optional {. 11975 *Some existing checks on initialisers and expressions will be extended as applicable to situations where objects of automatic storage duration are initialised using non constant expressions. Messages such as for instance 0671, 0672, 0692, 4434, 4453, etc. may be generated to notify potential issues in these circumstances.*
0823	[S] Unexpected '#else' or '#elif' directive follows '#else'. 22493 *The detection of #else or #elif directives following an #else directive will be extended, so that message 0823 will also be generated in certain circumstances, for instance when an #elif follows an active #else branch. As part of this upgrade, side effects on message 3318 will also be removed, so that message 3318 will no longer be incorrectly generated in certain situations where for instance a file contains an #if ... #elif #endif directive with the arguments of both #if and #elif evaluating to 0, and that file is included from another file, and the #include is within an #if ... #endif branch.*
0862	This #include "%s" directive is redundant. 21995 *In the Windows environment QAC will handle paths as case-insensitive and in particular will output the same messages irrespective of whether an include option folder path begins with 'C:\' or with 'c:\'.* 20865 *Message 0862 will no longer be incorrectly generated in certain situations, which may involve for instance multiple inclusions of the same header interleaved with source code which uses features from that header or from other headers included from that header. Message 0862 will be generated also when the unused header happens to contain functions definitions with statements. Message 0862 will no longer be generated when a function definition is partly implemented in a header and partly in its including file.*
1027	[C99] Use of type 'long long'. 11261 *Message 0616 will be generated (instead of false positive 1027) when the type specifiers 'long' or 'long long' are preceded by other incompatible type specifiers such as 'char' or 'short'. Noise will also be reduced, within the same declaration: (1) message 0616 will only be generated for the first incompatible type specifier and the first incompatible storage class specifier encountered and (2) message 1027 will be generated at most once and only if message 0616 is not already generated for incompatibilities with any preceding type specifiers.*

continued on next page

Table 3.2 – continued from previous page

Msg. Id.	Description
1577	Next seen here. 21588 *Removal of confusing sub messages for messages such as apparent use of unset pointer*
1810	An operand of 'essentially character' type is being added to another operand of 'essentially character' type. 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
1811	An operand of 'essentially character' type is being subtracted from an operand of 'essentially signed' type. 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
1812	An operand of 'essentially character' type is being subtracted from an operand of 'essentially unsigned' type. 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
2791	Definite: Right hand operand of shift operator is negative or too large. 14002 *False negative message where the right operand of a shift is equal to the number of bits of the left hand operand.*
2810	Constant: Dereference of NULL pointer. 21590 *False positive message when passing NULL to 'wctomb'*
2920	Constant: Left shift operation on expression of unsigned type results in loss of high order bits. 14002 *False negative message where the right operand of a shift is equal to the number of bits of the left hand operand.*
2921	Definite: Left shift operation on expression of unsigned type results in loss of high order bits. 14002 *False negative message where the right operand of a shift is equal to the number of bits of the left hand operand.*
2961	Definite: Using value of uninitialized automatic object '%s'. 20159 *False positive use of unset message for object passed by address to an ellipsis parameter.* 21893 *False positive use of unset message for array object initialized from elements of another array.* 22205 *False positive use of uninitialized value where address of class object converted to void.*
2962	Apparent: Using value of uninitialized automatic object '%s'. 21588 *Removal of confusing sub messages for messages such as apparent use of unset pointer*
2995	The result of this logical operation is always 'true'. 21970 *False positive messages due to array element modelling treating distinct elements as the same element.*

continued on next page

Table 3.2 – continued from previous page

Msg. Id.	Description
2996	The result of this logical operation is always 'false'. 21970 *False positive messages due to array element modelling treating distinct elements as the same element.*
3112	This statement has no side-effect - it can be removed. 22372 *Message 3112 will be split so that it will no longer be generated when the statement without side effects is composed of an expression which is cast to void. In these circumstances a new message 3119 will be generated instead. Consequently, message 3112 will map more precisely to MISRA C:2012 Rule 2.2 and message 3119 will report exceptions to Rule 2.2, where it is still possible and may still be beneficial to remove the statement without side effects.*
3318	'#else'/'#elif'/'#endif' in included file matched '#if...' in parent file. This is probably an error. 22493 *The detection of #else or #elif directives following an #else directive will be extended, so that message 0823 will also be generated in certain circumstances, for instance when an #elif follows an active #else branch. As part of this upgrade, side effects on message 3318 will also be removed, so that message 3318 will no longer be incorrectly generated in certain situations where for instance a file contains an #if ... #elif #endif directive with the arguments of both #if and #elif evaluating to 0, and that file is included from another file, and the #include is within an #if ... #endif branch.*
3663	Unnamed bit-field defined with non-zero width. 12656 *Messages 0634, 0635 and 3663 will be generated where applicable even if the bit-fields happen to be defined within non-tagged unions.*
4434	A non-constant expression of 'essentially signed' type (%1s) is being converted to unsigned type, '%2s' on assignment. 11975 *Some existing checks on initialisers and expressions will be extended as applicable to situations where objects of automatic storage duration are initialised using non constant expressions. Messages such as for instance 0671, 0672, 0692, 4434, 4453, etc. may be generated to notify potential issues in these circumstances.*

continued on next page

Table 3.2 – continued from previous page

Msg. Id.	Description
4453	An expression of 'essentially floating' type (%1s) is being converted to signed type, '%2s' on assignment. 11975 *Some existing checks on initialisers and expressions will be extended as applicable to situations where objects of automatic storage duration are initialised using non constant expressions. Messages such as for instance 0671, 0672, 0692, 4434, 4453, etc. may be generated to notify potential issues in these circumstances.*
4501	An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4502	An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this bitwise operator (%3s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4503	An expression of 'essentially Boolean' type (%1s) is being used as the left-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4504	An expression of 'essentially Boolean' type (%1s) is being used as the right-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4505	An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this relational operator (%3s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4511	An expression of 'essentially character' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4512	An expression of 'essentially character' type (%1s) is being used as the %2s operand of this bitwise operator (%3s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4513	An expression of 'essentially character' type (%1s) is being used as the left-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*

continued on next page

Table 3.2 – continued from previous page

Msg. Id.	Description
4514	An expression of 'essentially character' type (%1s) is being used as the right-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4524	An expression of 'essentially enum' type (%1s) is being used as the right-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4532	An expression of 'essentially signed' type (%1s) is being used as the %2s operand of this bitwise operator (%3s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4533	An expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4534	An expression of 'essentially signed' type (%1s) is being used as the right-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4543	A non-negative constant expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4544	A non-negative constant expression of 'essentially signed' type (%1s) is being used as the right-hand operand of this shift operator (%2s). 21790 *Distinguish essential type messages between preprocessing and non-preprocessing expressions.*
4828	Invalid usage of predefined location tag. 22551 *New messages to highlight invalid location annotations.*

3.2.4 Message Text Changes

There are no message text changes for QA·C 9.0.0.

4 QA·C Ticket Summary

4.1 Ticket Summary for QA·C 9.0.1

The following table summarizes the tickets that were closed in QA·C 9.0.1.

Tickets are been categorized into 3 types:

E – Enhancement to an existing feature.

F – A fix of a bug or problem feature.

N – New functionality has been introduced.

Table 4.1: Ticket Summary for QA·C 9.0.1

Ticket	Type	Description
22787	F	Dataflow analysis times out unexpectedly on expressions that can never overflow or wraparound.

4.2 Ticket Summary for QA·C 9.0.0

The following table summarizes the tickets that were closed in QA·C 9.0.0.

Tickets are been categorized into 3 types:

E – Enhancement to an existing feature.

F – A fix of a bug or problem feature.

N – New functionality has been introduced.

Table 4.2: Ticket Summary for QA·C 9.0.0

Ticket	Type	Description
11261	F	Message 0616 will be generated (instead of false positive 1027) when the type specifiers 'long' or 'long long' are preceded by other incompatible type specifiers such as 'char' or 'short'. Noise will also be reduced, within the same declaration: (1) message 0616 will only be generated for the first incompatible type specifier and the first incompatible storage class specifier encountered and (2) message 1027 will be generated at most once and only if message 0616 is not already generated for incompatibilities with any preceding type specifiers. 0616 *[C] Illegal combination of type specifiers or storage class specifiers.* 1027 *[C99] Use of type 'long long'.*
11645	F	Diagnostics appearing on the same line as an include directive were not suppressed by a continuous suppression for the message that included this include line. See Updates to Suppression Code Annotations.
11975	E	Some existing checks on initialisers and expressions will be extended as applicable to situations where objects of automatic storage duration are initialised using non constant expressions. Messages such as 0671, 0672, 0692, 4434, 4453, etc. may be generated to notify potential issues in these circumstances. 0671 *[C] Initializer for object of arithmetic type is not of arithmetic type.* 0672 *[U] The initializer for a 'struct', 'union' or array is not enclosed in braces.* 0692 *Union initializer is missing the optional {.* 4434 *A non-constant expression of 'essentially signed' type (%1s) is being converted to unsigned type, '%2s' on assignment.* 4453 *An expression of 'essentially floating' type (%1s) is being converted to signed type, '%2s' on assignment.*
12656	E	Messages 0634, 0635 and 3663 will be generated where applicable even if the bit-fields happen to be defined within non-tagged unions. 0634 *[I] Bit-fields in this struct/union have not been declared explicitly as unsigned or signed.* 0635 *[E] Bit-fields in this struct/union have been declared with types other than int, signed int, unsigned int or _Bool.* 3663 *Unnamed bit-field defined with non-zero width.*

continued on next page

Table 4.2 – continued from previous page

Ticket	Type	Description
13237	E	(1) Message 0543 will also be generated when a void expression is the operand of a '!' or (prefix or postfix) '++' or '–' (unary) operator or the 1st operand of a '? :' operator or the left or right operand of a '%', '<<', '>>', '|', '&', '^', '||', '&&', '-', '+', '*', '/', '*=', '/=', '<<=', '&=', '|=', '^=' or '%=' (binary) operator or the right operand of a '+=' operator. (2) Message 0543 will also be generated when a void expression is cast to a non void type. (3) Message 0543 will no longer be incorrectly generated when a void expression is the 3rd operand of a conditional (ternary) operator whose result is not used. (4) Message 0543 will always be shown against the applicable void expression (instead of being shown against outermost containing expressions). (5) If an expression contains multiple void sub-expressions, message 0543 will always be generated multiple times, once for each void sub-expression. (Note that even if the scope of message 0543 will be significantly extended, all the newly identified situations for the generation of 0543 are still leading to the additional generation of one or more constraint error messages such as: 0436, 0446, 0451, 0453, 0456, 0466, 0467, 0468, 0469, 0481, 0493, 0494, 0495, 0496, 0536, 0537, 0540, 0541, 0542, 0555, 0557, 0559, 0560, 0563, 0564, 0565, 0746, 0756, 0758. Message 0543 will therefore provide additional useful information on issues that are already being otherwise detected by QAC.) 0543 *[U] 'void' expressions have no value and may not be used in expressions.*
13538	F	Correcting calculation of STUNR metric.
14002	F	False negative message where the right operand of a shift is equal to the number of bits of the left hand operand. 2791 *Definite: Right hand operand of shift operator is negative or too large.* 2920 *Constant: Left shift operation on expression of unsigned type results in loss of high order bits.* 2921 *Definite: Left shift operation on expression of unsigned type results in loss of high order bits.*
20159	F	False positive use of unset message for object passed by address to an ellipsis parameter. 2961 *Definite: Using value of uninitialized automatic object '%s'.*
20249	E	Sub message generation has been updated to ensure it is deterministic.
20434	N	When QAC is run with the '-AlwaysSearchIncludeDirectoriesFirst+' option, on processing an #include directive whose file path is enclosed in double quotes, QAC will search for the file in the current folder only if the file is not found first in any of the specified include folders.

continued on next page

Table 4.2 – continued from previous page

Ticket	Type	Description
20865	F	Message 0862 will no longer be incorrectly generated in certain situations, which may involve for instance multiple inclusions of the same header interleaved with source code which uses features from that header or from other headers included from that header. Message 0862 will be generated also when the unused header happens to contain functions definitions with statements. Message 0862 will no longer be generated when a function definition is partly implemented in a header and partly in its including file. 0862 *This #include "%s" directive is redundant.*
21226	F	Memory consumption when parsing code based annotations has been improved in comparison to the previous versions.
21588	F	Removal of confusing sub messages for messages such as apparent use of unset pointer 1577 *Next seen here.* 2962 *Apparent: Using value of uninitialized automatic object '%s'.*
21590	F	False positive message when passing NULL to 'wctomb' 2810 *Constant: Dereference of NULL pointer.*

continued on next page

Table 4.2 – continued from previous page

Ticket	Type	Description
21790	E	When analysing essential types in expressions that are used in conditional preprocessing (#if or #elif) directives, QAC will: (1) no longer generate messages 1810, 1811, 1812, 4511, 4512, 4513, 4514, 4533, 4524, 4543 and 4544, (2) generate equivalent new messages under the same circumstances, respectively 4580, 4581, 4582, 4583, 4584, 4585, 4586, 4587, 4588, 4589, and 4590, (3) no longer generate messages 4501, 4502, 4503, 4504, 4505, 4532 and 4534. Note: Existing messages for essential types are mapped by the compliance modules to MISRA C:2012 (or MISRA C:2004) rules that are based on the essential (or underlying) type model. This model is only intended to be applicable after preprocessing. Restricting existing messages will therefore prevent the generation of false positives within a MISRA C compliance context. Nevertheless, the essential type model can also partially apply to preprocessing expressions, and generating new equivalent messages (which are not mapped to MISRA C) in these situations preserves the scope and consistency of existing QAC's analysis. Messages at (3) are instead simply restricted because other messages 4104, 4105, 4111, 2790 and 2890 already cover equivalent conditions at preprocessing. 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.* 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).* 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).*

continued on next page

Table 4.2 – continued from previous page

Ticket	Type	Description
21790	E	Continued. 4514 *An expression of 'essentially character' type (%1s) is being used as the right-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).* 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).* 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).* 4580 *An operand of 'essentially character' type is being added to another operand of 'essentially character' type.* 4581 *An operand of 'essentially character' type is being subtracted from an operand of 'essentially signed' type.* 4582 *An operand of 'essentially character' type is being subtracted from an operand of 'essentially unsigned' type.* 4583 *An expression of 'essentially character' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s).* 4584 *An expression of 'essentially character' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).* 4585 *An expression of 'essentially character' type (%1s) is being used as the left-hand operand of this shift operator (%2s).* 4586 *An expression of 'essentially character' type (%1s) is being used as the right-hand operand of this shift operator (%2s).* 4587 *An expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s).* 4588 *A non-negative constant expression of 'essentially signed' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).* 4589 *A non-negative constant expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s).* 4590 *A non-negative constant expression of 'essentially signed' type (%1s) is being used as the right-hand operand of this shift operator (%2s).*

continued on next page

Table 4.2 – continued from previous page

Ticket	Type	Description
21893	F	False positive use of unset message for array object initialized from elements of another array. 2961 *Definite: Using value of uninitialized automatic object '%s'.*
21905	F	Arguments to a function declared with an empty parameter list are now handled correctly.
21970	F	False positive messages due to array element modelling treating distinct elements as the same element. 2995 *The result of this logical operation is always 'true'.* 2996 *The result of this logical operation is always 'false'.*
21995	F	In the Windows environment QAC will handle paths as case-insensitive and in particular will output the same messages irrespective of whether an include option folder path begins with 'C:\' or with 'c:\'. 0862 *This #include "%s" directive is redundant.*
22205	F	False positive use of uninitialized value where address of member object converted to void. 2961 *Definite: Using value of uninitialized automatic object '%s'.*
22207	F	QAC will not abort with a SIGSEGV fault under certain circumstances involving the parsing of consecutive designated initialisers with nested compound literals used as initialising expressions.
22265	F	QAC will no longer abort with a SIGSEGV fault when parsing syntactically incorrect designated initialisers containing compound literals. Also, QAC will no longer abort with a SIGSEGV fault when parsing compound literals containing designated initialisers which are implemented using statement expressions. In this second case, QAC will instead generate message 0907 and exit with code 1 (parser hard error). Note that statement expressions are a GCC language extension whose semantic is currently not fully supported by QAC. 0907 *[S] Unexpected token.*
22286	F	Constraint error message 0675 will no longer be incorrectly generated under certain circumstances involving nested compound literals. 0675 *[C] Initializer is not of compatible 'struct'/'union' type.*
22290	F	Message 0684 will no longer be incorrectly generated in certain situations where initialisers contain nested compound literals. 0684 *[C] Too many initializers.*

continued on next page

Table 4.2 – continued from previous page

Ticket	Type	Description
22372	E	Message 3112 will be split so that it will no longer be generated when the statement without side effects is composed of an expression which is cast to void. In these circumstances a new message 3119 will be generated instead. Consequently, message 3112 will map more precisely to MISRA C:2012 Rule 2.2 and message 3119 will report exceptions to Rule 2.2, where it is still possible and may still be beneficial to remove the statement without side effects. 3112 *This statement has no side-effect - it can be removed.* 3119 *This statement has no side-effect - it can be removed.*
22493	E	The detection of #else or #elif directives following an #else directive will be extended, so that message 0823 will also be generated in certain circumstances, for instance when an #elif follows an active #else branch. As part of this upgrade, side effects on message 3318 will also be removed, so that message 3318 will no longer be incorrectly generated in certain situations where for instance a file contains an #if ... #elif #endif directive with the arguments of both #if and #elif evaluating to 0, and that file is included from another file, and the #include is within an #if ... #endif branch. 0823 *[S] Unexpected '#else' or '#elif' directive follows '#else'.* 3318 *'#else'/'#elif'/'#endif' in included file matched '#if...' in parent file. This is probably an error.*
22543	F	If the operand to the -Q option ends with a directory separator then only files under that directory will be suppressed.
22545	F	QAC has been updated to encode strings internally using UTF-8. See Internationalization Support.
22550	N	For annotations in a shared C/C++ header file it is now possible to specify if the message is for QAC or QAC++. See Updates to Suppression Code Annotations. 4815 *The specified product name is invalid.* 4829 *Invalid character in message specifier.* 4830 *Unexpected left bracket in message specifier.* 4831 *Unexpected right bracket in message specifier.* 4832 *Unexpected left bracket in tag name.* 4833 *Unexpected right bracket in tag name.* 4834 *Expected left bracket in product specifier.* 4835 *Expected right bracket in product specifier.*
22551	F	New messages to highlight invalid location annotations. 4813 *The location annotation is not preceded by a suppression annotation which refers to it.* 4814 *A location annotation cannot itself have a location specifier.* 4828 *Invalid usage of predefined location tag.*

continued on next page

Table 4.2 – continued from previous page

Ticket	Type	Description
22552	E	A suppression tag may now be specified as a quoted string allowing characters such as spaces or non ASCII to be used as part of a tag name. See Updates to Suppression Code Annotations.
22553	E	Suppressions specified using #pragma PRQA_MESSAGES_OFF/ON are implemented by mapping them to a continuous suppression annotation. See Updates to Suppression Code Annotations.
22554	E	A new message specifier 'ALL' has been added for comment based suppressions. See Updates to Suppression Code Annotations.
22708	F	Option added to qacli to allow multiple whole program runs of dataflow. See Dataflow.