Used QAC Messages

This table lists all QAC messages incorporated in the M3CM-1.0-QAC-8.1.2 compliance module.

Msg No	Message	Rule
160	[U] Using unsupported conversion specifier number %s.	Rule-1.3
161	[U] Unknown length modifier used with 'i' or 'd' conversion specifier, number %s.	Rule-1.3
162	[U] Unknown length modifier used with 'o' conversion specifier, number %s.	Rule-1.3
163	[U] Unknown length modifier used with 'u' conversion specifier, number %s.	Rule-1.3
164	[U] Unknown length modifier used with 'x' conversion specifier, number %s.	Rule-1.3
165	[U] Unknown length modifier used with 'X' conversion specifier, number %s.	Rule-1.3
166	[U] Unknown length modifier used with 'f' conversion specifier, number %s.	Rule-1.3
167	[U] Unknown length modifier used with 'e' conversion specifier, number %s.	Rule-1.3
168	[U] Unknown length modifier used with 'E' conversion specifier, number %s.	Rule-1.3
169	[U] Unknown length modifier used with 'g' conversion specifier, number %s.	Rule-1.3
170	[U] Unknown length modifier used with 'G' conversion specifier, number %s.	Rule-1.3
171	[U] Unknown length modifier used with 'c' conversion specifier, number %s.	Rule-1.3
172	[U] Unknown length modifier used with '%%' conversion specifier, number %s.	Rule-1.3
173	[U] Unknown length modifier used with 's' conversion specifier, number %s.	Rule-1.3
174	[U] Unknown length modifier used with 'n' conversion specifier, number %s.	Rule-1.3
175	[U] Unknown length modifier used with 'p' conversion specifier, number %s.	Rule-1.3
176	[U] Incomplete conversion specifier, number %s.	Rule-1.3
177	[U] Field width of format conversion specifier exceeds 509 characters.	Rule-1.3
178	[U] Precision of format conversion specifier exceeds 509 characters.	Rule-1.3
179	[U] Argument type does not match conversion specifier number %s.	Rule-1.3
180	[C99] Use of ll for conversion specifier.	Dir-1.1
184	[U] Insufficient arguments to satisfy conversion specifier, number %s.	Rule-1.3
185	[U] Call contains more arguments than conversion specifiers.	Rule-1.3
186	[U] A call to this function must include at least one argument.	Rule-1.3
190	[U] Using unsupported conversion specifier number %s.	Rule-1.3
191	[U] Unknown length modifier used with 'd/i/n' conversion specifier, number %s.	Rule-1.3
192	[U] Unknown length modifier used with 'o' conversion specifier, number %s.	Rule-1.3
193	[U] Unknown length modifier used with 'u' conversion specifier, number %s.	Rule-1.3
194	[U] Unknown length modifier used with 'x/X' conversion specifier, number %s.	Rule-1.3
195	[U] Unknown length modifier used with 'e/E/f/F/g/G' conversion specifier, number %s.	Rule-1.3
196	[U] Unknown length modifier used with 's' conversion specifier, number %s.	Rule-1.3
197	[U] Unknown length modifier used with 'p' conversion specifier, number %s.	Rule-1.3
198	[U] Unknown length modifier used with '%%' conversion specifier, number %s.	Rule-1.3
199	[U] Unknown length modifier used with '[' conversion specifier, number %s.	Rule-1.3
200	[U] Unknown length modifier used with 'c' conversion specifier, number %s.	Rule-1.3
201	[U] Incomplete conversion specifier, number %s.	Rule-1.3
202	[I] '-' character in '[]' conversion specification is implementation defined.	Dir-1.1
203	[U] Value of character prior to '-' in '[]' is greater than following character.	Rule-1.3
204	[U] Field width of format conversion specifier exceeds 509 characters.	Rule-1.3
206	[U] Argument type does not match conversion specifier number %s.	Rule-1.3
207	[U] 'scanf' expects address of objects being stored into.	Rule-1.3
208	[U] Same character occurs in scanset more than once.	Rule-1.3
232	[C] Value of hex escape sequence is not representable in type 'unsigned char'.	Rule-1.1
233	[C] Value of octal escape sequence is not representable in type 'unsigned char'.	Rule-1.1
235	[U] Unknown escape sequence.	Rule-1.3
240	[E] This file contains the control-M character at the end of a line.	Rule-1.2
241	[E] This file contains the control-Z character - was this transferred from a PC?	Rule-1.2
244	[C] Value of character constant is not representable in type 'int'.	Rule-1.1
246	[E] Binary integer constants are a language extension.	Rule-1.2
261	[C] Comment still open at end of included file.	Rule-1.1
275	[U] Floating value is out of range for conversion to destination type.	Rule-1.3
284	[I] Multiple character constants have implementation defined values.	Dir-1.1

285	[I] Character constant contains character which is not a member of the basic source character set.	Dir-1.1
286	[I] String literal contains character which is not a member of the basic source character set.	Dir-1.1
287	[I] Header name contains character which is not a member of the basic source character set.	Dir-1.1
288	[I] Source file '%s' has comments containing characters which are not members of the basic source character set.	Dir-1.1
289	[I] Source file '%s' has preprocessing tokens containing characters which are not members of the basic source character set.	Dir-1.1
292	[I] Source file '%s' has comments containing one of the characters '$', '@' or '`'.	Dir-1.1
299	[I] Source file '%s' includes #pragma directives containing characters which are not members of the basic source character set.	Dir-1.1
301	[u] Cast between a pointer to object and a floating type.	Rule-1.3
301	[u] Cast between a pointer to object and a floating type.	Rule-11.7
302	[u] Cast between a pointer to function and a floating type.	Rule-1.3
302	[u] Cast between a pointer to function and a floating type.	Rule-11.1
303	[I] Cast between a pointer to volatile object and an integral type.	Rule-11.4
304	[U] The address of an array declared 'register' may not be computed.	Rule-1.3
305	[I] Cast between a pointer to function and an integral type.	Rule-11.1
306	[I] Cast between a pointer to object and an integral type.	Rule-11.6
306	[I] Cast between a pointer to object and an integral type.	Rule-11.4
307	[u] Cast between a pointer to object and a pointer to function.	Rule-1.3
307	[u] Cast between a pointer to object and a pointer to function.	Rule-11.1
308	Non-portable cast involving pointer to an incomplete type.	Rule-11.2
309	[U] Integral type is not large enough to hold a pointer value.	Rule-1.3
310	Casting to different object pointer type.	Rule-11.3
311	Dangerous pointer cast results in loss of const qualification.	Rule-11.8
312	Dangerous pointer cast results in loss of volatile qualification.	Rule-11.8
313	Casting to different function pointer type.	Rule-11.1
316	[I] Cast from a pointer to void to a pointer to object type.	Rule-11.5
317	[I] Implicit conversion from a pointer to void to a pointer to object type.	Rule-11.5
320	[C99] Declaration within 'for' statement.	Dir-1.1
321	[C] Declaration within 'for' statement defines an identifier '%s' which is not an object.	Rule-1.1
322	[C] Illegal storage class specifier used in 'for' statement declaration.	Rule-1.1
336	Macro defined as an octal constant.	Rule-7.1
337	[U] String literal has undefined value. This may be a result of using '#' on \\.	Rule-1.3
338	[C] Octal or hex escape sequence value is too large for 'unsigned char' or 'wchar_t' type.	Rule-1.1
339	Octal constant used.	Rule-7.1
341	Using the stringify operator '#'.	Rule-20.10
342	Using the glue operator '##'.	Rule-20.10
360	An expression of pointer type is being converted to type _Bool on assignment.	Rule-11.4
361	An expression of pointer type is being cast to type _Bool.	Rule-11.4
362	An expression of essentially Boolean type is being cast to a pointer.	Rule-11.4
371	[L] Nesting levels of blocks exceeds 127 - program does not conform strictly to ISO:C99.	Dir-1.1
372	[L] More than 63 levels of nested conditional inclusion - program does not conform strictly to ISO:C99.	Dir-1.1
375	[L] Nesting of parenthesized expressions exceeds 63 - program does not conform strictly to ISO:C99.	Dir-1.1
380	[L] Number of macro definitions exceeds 4095 - program does not conform strictly to ISO:C99.	Dir-1.1
388	[L] '#include "%s"' causes nesting to exceed 15 levels - program does not conform strictly to ISO:C99.	Dir-1.1
390	[L] Number of members in 'struct' or 'union' exceeds 1023 - program does not conform strictly to ISO:C99.	Dir-1.1
391	[L] Number of enumeration constants exceeds 1023 - program does not conform strictly to ISO:C99.	Dir-1.1
392	[L] Nesting of 'struct' or 'union' types exceeds 63 - program does not conform strictly to ISO:C99.	Dir-1.1
400	[U] '%s' is modified more than once between sequence points - evaluation order unspecified.	Rule-1.3
400	[U] '%s' is modified more than once between sequence points - evaluation order unspecified.	Rule-13.2
401	[U] '%s' may be modified more than once between sequence points - evaluation order unspecified.	Rule-1.3
401	[U] '%s' may be modified more than once between sequence points - evaluation order unspecified.	Rule-13.2
402	[U] '%s' is modified and accessed between sequence points - evaluation order unspecified.	Rule-1.3
402	[U] '%s' is modified and accessed between sequence points - evaluation order unspecified.	Rule-13.2

403	[U] '%s' may be modified and accessed between sequence points - evaluation order unspecified.	Rule-13.2
403	[U] '%s' may be modified and accessed between sequence points - evaluation order unspecified.	Rule-1.3
410	[L] Nesting of parentheses exceeds 32 - program does not conform strictly to ISO:C90.	Dir-1.1
422	[C] Function call contains fewer arguments than prototype specifies.	Rule-1.1
423	[C] Function call contains more arguments than prototype specifies.	Rule-1.1
426	[C] Called function has incomplete return type.	Rule-1.1
427	[C] Object identifier used as if it were a function or a function pointer identifier.	Rule-1.1
429	[C] Function argument is not of arithmetic type.	Rule-1.1
430	[C] Function argument is not of compatible 'struct'/'union' type.	Rule-1.1
431	[C] Function argument points to a more heavily qualified type.	Rule-1.1
432	[C] Function argument is not of compatible pointer type.	Rule-1.1
435	[C] The 'struct'/'union' member '%s' does not exist.	Rule-1.1
436	[C] Left operand of '.' must be a 'struct' or 'union' object.	Rule-1.1
437	[C] Left operand of '->' must be a pointer to a 'struct' or 'union' object.	Rule-1.1
446	[C] Operand of ++/-- must have scalar (arithmetic or pointer) type.	Rule-1.1
447	[C] Operand of ++/-- must be a modifiable object.	Rule-1.1
448	[C] Operand of ++/-- must not be a pointer to an object of unknown size.	Rule-1.1
449	[C] Operand of ++/-- must not be a pointer to a function.	Rule-1.1
450	[C] An expression of array type cannot be cast.	Rule-1.1
451	[C] Subscripting requires a pointer (or array lvalue).	Rule-1.1
452	[C] Cannot subscript a pointer to an object of unknown size.	Rule-1.1
453	[C] An array subscript must have integral type.	Rule-1.1
454	[C] The address-of operator '&' cannot be applied to an object declared with 'register'.	Rule-1.1
456	[C] This expression does not have an address - '&' may only be applied to an lvalue or a function designator.	Rule-1.1
457	[C] The address-of operator '&' cannot be applied to a bit-field.	Rule-1.1
458	[C] Indirection operator '*' requires operand of pointer type.	Rule-1.1
466	[C] Unary '+' requires arithmetic operand.	Rule-1.1
467	[C] Operand of '!' must have scalar (arithmetic or pointer) type.	Rule-1.1
468	[C] Unary '-' requires arithmetic operand.	Rule-1.1
469	[C] Bitwise not '~' requires integral operand.	Rule-1.1
475	[u] Operand of 'sizeof' is an expression designating a bit-field.	Rule-1.3
476	[C] 'sizeof' cannot be applied to a bit-field.	Rule-1.1
477	[C] 'sizeof' cannot be applied to a function.	Rule-1.1
478	[C] 'sizeof' cannot be applied to an object of unknown size.	Rule-1.1
481	[C] Only scalar expressions may be cast to other types.	Rule-1.1
482	[C] Expressions may only be cast to 'void' or scalar types.	Rule-1.1
483	[C] A pointer to an object of unknown size cannot be the operand of an addition operator.	Rule-1.1
484	[C] A pointer to an object of unknown size cannot be the operand of a subtraction operator.	Rule-1.1
485	[C] Only integral expressions may be added to pointers.	Rule-1.1
486	[C] Only integral expressions and compatible pointers may be subtracted from pointers.	Rule-1.1
487	[C] If two pointers are subtracted, they must be pointers that address compatible types.	Rule-1.1
488	Performing pointer arithmetic.	Rule-18.4
493	[C] Type of left operand is not compatible with this operator.	Rule-1.1
494	[C] Type of right operand is not compatible with this operator.	Rule-1.1
495	[C] Left operand of '%', '<<', '>>', '&', '^' or '|' must have integral type.	Rule-1.1
496	[C] Right operand of '%', '<<', '>>', '&', '^' or '|' must have integral type.	Rule-1.1
499	Right operand of shift operator is greater than or equal to the width of the essential type of the left operand.	Rule-12.2
513	[C] Relational operator used to compare pointers to incompatible types.	Rule-1.1
514	[C] Relational operator used to compare a pointer with an incompatible operand.	Rule-1.1
515	[C] Equality operator used to compare a pointer with an incompatible operand.	Rule-1.1
536	[C] First operand of '&&', '||' or '?' must have scalar (arithmetic or pointer) type.	Rule-1.1
537	[C] Second operand of '&&' or '||' must have scalar (arithmetic or pointer) type.	Rule-1.1
540	[C] 2nd and 3rd operands of conditional operator '?' must have compatible types.	Rule-1.1
541	[C] Argument no. %s does not have object type.	Rule-1.1
542	[C] Controlling expression must have scalar (arithmetic or pointer) type.	Rule-1.1
543	[U] 'void' expressions have no value and may not be used in expressions.	Rule-1.3
544	[U] The value of an incomplete 'union' may not be used.	Rule-1.3

545	[U] The value of an incomplete 'struct' may not be used.	Rule-1.3
546	[C] 'enum %s' has unknown content. Use of an enum tag with undefined content is not permitted.	Rule-1.1
547	[C] This declaration of tag '%s' conflicts with a previous declaration.	Rule-1.1
550	[C] Left operand of '+=' or '-=' is a pointer to an object of unknown size.	Rule-1.1
551	[E] Cast may not operate on the left operand of the assignment operator.	Rule-1.2
554	[C] 'static %s()' has been declared and called but no definition has been given.	Rule-1.1
555	[C] Invalid assignment to object of void type or array type.	Rule-1.1
556	[C] Left operand of assignment must be a modifiable object.	Rule-1.1
557	[C] Right operand of assignment is not of arithmetic type.	Rule-1.1
558	[C] Right operand of '+=' or '-=' must have integral type when left operand is a pointer.	Rule-1.1
559	[C] Right operand of '<<=', '>>=', '&=', '|=', '^=' or '%=' must have integral type.	Rule-1.1
560	[C] Left operand of '<<=', '>>=', '&=', '|=', '^=' or '%=' must have integral type.	Rule-1.1
561	[C] Right operand of assignment is not of compatible 'struct'/'union' type.	Rule-1.1
562	[C] Right operand of assignment points to a more heavily qualified type.	Rule-1.1
563	[C] Right operand of assignment is not of compatible pointer type.	Rule-1.1
564	[C] Left operand of assignment must be an lvalue (it must designate an object).	Rule-1.1
565	[C] Left operand of '+=' or '-=' must be of arithmetic or pointer to object type.	Rule-1.1
570	This switch case label of 'essential type' '%1s', is not consistent with a controlling expression of essential type '%2s'.	Rule-10.3
572	This switch case label of 'essential type' '%1s' is not consistent with a controlling expression which has an essential type of lower rank (%2s).	Rule-10.3
580	[C] Constant is too large to be representable.	Rule-1.1
581	[I] Floating-point constant may be too small to be representable.	Dir-1.1
588	[C] Width of bit-field must be an integral constant expression.	Rule-1.1
589	[C] Enumeration constant must be an integral constant expression.	Rule-1.1
590	[C] Array bound must be an integral constant expression.	Rule-1.1
591	[C] A 'case' label must be an integral constant expression.	Rule-1.1
594	Negative 'case' label expression is incompatible with unsigned controlling expression in 'switch' statement.	Rule-2.1
601	[E] Function 'main()' is not of type 'int (void)' or 'int (int, char *[])'.	Rule-1.2
602	[U] The identifier '%s' is reserved for use by the library.	Rule-21.2
602	[U] The identifier '%s' is reserved for use by the library.	Rule-1.3
604	[C99] Declaration appears after statements in a compound statement.	Dir-1.1
605	[C] A declaration must declare a tag or an identifier.	Rule-1.1
609	[L] More than 12 pointer, array or function declarators modifying a declaration - program does not conform strictly to ISO:C90.	Dir-1.1
611	[L] Nesting of 'struct' or 'union' types exceeds 15 - program does not conform strictly to ISO:C90.	Dir-1.1
612	[L] Size of object '%s' exceeds 32767 bytes - program does not conform strictly to ISO:C90.	Dir-1.1
614	[L] More than 127 block scope identifiers defined within a block - program does not conform strictly to ISO:C90.	Dir-1.1
616	[C] Illegal combination of type specifiers or storage class specifiers.	Rule-1.1
617	[C99] 'const' qualifier has been duplicated.	Dir-1.1
618	[C99] 'volatile' qualifier has been duplicated.	Dir-1.1
619	[C] The identifier '%s' has already been defined in the current scope within the ordinary identifier namespace.	Rule-1.1
620	[C] Cannot initialize '%s' because it has unknown size.	Rule-1.1
621	[C] The struct/union '%s' cannot be initialized because it has unknown size.	Rule-1.1
622	[C] The identifier '%s' has been declared both with and without linkage in the same scope.	Rule-1.1
623	[U] '%s' has incomplete type and no linkage - this is undefined.	Rule-1.3
624	Function '%s' is declared using typedefs which are different to those in a previous declaration.	Rule-8.3
625	[U] '%s' has been declared with both internal and external linkage - the behaviour is undefined.	Rule-1.3
626	[U] '%s' has different type to previous declaration (which is no longer in scope).	Rule-1.3
627	[C] '%s' has different type to previous declaration in the same scope.	Rule-1.1
628	[C] '%s' has different type to previous declaration at wider scope.	Rule-1.1
629	[C] More than one definition of '%s' (with internal linkage).	Rule-1.1
630	[U] More than one definition of '%s' (with external linkage).	Rule-8.6
630	[U] More than one definition of '%s' (with external linkage).	Rule-1.3
631	[C] More than one declaration of '%s' (with no linkage).	Rule-1.1
632	[U] Tentative definition of '%s' with internal linkage cannot have unknown size.	Rule-1.3
633	[E] Empty structures and unions are a language extension.	Rule-1.2

634	[I] Bit-fields in this struct/union have not been declared explicitly as unsigned or signed.	Rule-6.1
634	[I] Bit-fields in this struct/union have not been declared explicitly as unsigned or signed.	Dir-1.1
635	[E] Bit-fields in this struct/union have been declared with types other than int, signed int or unsigned int.	Rule-1.2
635	[E] Bit-fields in this struct/union have been declared with types other than int, signed int or unsigned int.	Rule-6.1
636	[U] There are no named members in this 'struct' or 'union'.	Rule-1.3
638	[C] Duplicate member name '%s' in 'struct' or 'union'.	Rule-1.1
639	[L] Number of members in 'struct' or 'union' exceeds 127 - program does not conform strictly to ISO:C90.	Dir-1.1
640	[C] '%s' in 'struct' or 'union' type may not have 'void' type.	Rule-1.1
641	[C] '%s' in 'struct' or 'union' type may not have function type.	Rule-1.1
642	[C] '%s' in 'struct' or 'union' type may not be an array of unknown size.	Rule-1.1
643	[C] '%s' in 'struct' or 'union' type may not be a 'struct' or 'union' with unknown content.	Rule-1.1
644	[C] Width of bit-field must be no bigger than the width of an 'int'.	Rule-1.1
645	[C] A zero width bit-field cannot be given a name.	Rule-1.1
646	[C] Enumeration constants must have values representable as 'int's.	Rule-1.1
647	[L] Number of enumeration constants exceeds 127 - program does not conform strictly to ISO:C90.	Dir-1.1
649	[C] K&R style declaration of parameters is not legal after a function header that includes a parameter list.	Rule-1.1
650	[C] Illegal storage class specifier on named function parameter.	Rule-1.1
651	[C] Missing type specifiers in function declaration.	Rule-1.1
653	[C] Duplicate definition of 'struct', 'union' or 'enum' tag '%s'.	Rule-1.1
654	[U] Using 'const' or 'volatile' in a function return type is undefined.	Rule-1.3
655	[C] Illegal storage class specifier on unnamed function parameter.	Rule-1.1
656	[C] Function return type cannot be function or array type, or an incomplete struct/union (for function definition).	Rule-1.1
657	[C] Unnamed parameter specified in function definition.	Rule-1.1
658	[U] Parameter cannot have 'void' type.	Rule-1.3
659	[C] The identifier '%s' was not given in the parameter list.	Rule-1.1
660	[E] Defining an unnamed member in a struct or union. This is a language extension.	Rule-1.2
661	[U] '%s()' may not have a storage class specifier of 'static' when declared at block scope.	Rule-1.3
662	[E] Accessing a member of an unnamed struct or union member in this way is a language extension.	Rule-1.2
664	[C] Parameter specified with type 'void'.	Rule-1.1
665	[C] Two parameters have been declared with the same name '%s'.	Rule-1.1
667	[U] '%s' is declared as a typedef and may not be redeclared as an object at an inner scope without an explicit type specifier.	Rule-1.3
668	[U] '%s' is declared as a typedef and may not be redeclared as a member of a 'struct' or 'union' without an explicit type specifier.	Rule-1.3
671	[C] Initializer for object of arithmetic type is not of arithmetic type.	Rule-1.1
672	[U] The initializer for a 'struct', 'union' or array is not enclosed in braces.	Rule-1.3
673	[C] Initializer points to a more heavily qualified type.	Rule-1.1
674	[C] Initializer for pointer is of incompatible type.	Rule-1.1
675	[C] Initializer is not of compatible 'struct'/'union' type.	Rule-1.1
676	[u] Array element is of function type. Arrays cannot be constructed from function types.	Rule-1.3
677	[C] Array size is negative, or unrepresentable.	Rule-1.1
678	[u] Array element is array of unknown size. Arrays cannot be constructed from incomplete types.	Rule-1.3
680	[u] Array element is 'void' or an incomplete 'struct' or 'union'. Arrays cannot be constructed from incomplete types.	Rule-1.3
682	[C] Initializer for object of a character type is a string literal.	Rule-1.1
683	[C] Initializer for object of a character type is a wide string literal.	Rule-1.1
684	[C] Too many initializers.	Rule-1.1
685	[C] Initializer for any object with static storage duration must be a constant expression.	Rule-1.1
686	Array has fewer initializers than its declared size. Default initialization is applied to the remainder of the array elements.	Rule-9.3
690	[C] String literal contains too many characters to initialize object.	Rule-1.1
693	Struct initializer is missing the optional {.	Rule-9.2
694	Array initializer is missing the optional {.	Rule-9.2
698	[C] String literal used to initialize an object of incompatible type.	Rule-1.1
699	[C] String literal used to initialize a pointer of incompatible type.	Rule-1.1

706	[U] Label '%s' is not unique within this function.	Rule-1.3
708	[C] No definition found for the label '%s' in this function.	Rule-1.1
709	[C] Initialization of locally declared 'extern %s' is illegal.	Rule-1.1
715	[L] Nesting of control structures (statements) exceeds 15 - program does not conform strictly to ISO:C90.	Dir-1.1
735	Using relational or logical operators in a 'switch' expression is usually a programming error.	Rule-16.7
736	[C] 'case' label does not have unique value within this 'switch' statement.	Rule-1.1
737	[C] More than one 'default' label found in 'switch' statement.	Rule-1.1
738	[C] Controlling expression in a 'switch' statement must have integral type.	Rule-1.1
739	[L] Number of 'case' labels exceeds 257 - program does not conform strictly to ISO:C90.	Dir-1.1
745	[U] 'return;' found in '%s()', which has been defined with a non-'void' return type.	Rule-1.3
745	[U] 'return;' found in '%s()', which has been defined with a non-'void' return type.	Rule-17.4
746	[C] 'return exp;' found in '%s()' whose return type is 'void'.	Rule-1.1
747	[C] 'return exp;' found in '%s()' whose return type is qualified 'void'.	Rule-1.1
750	A union type specifier has been defined.	Rule-19.2
752	String literal passed as argument to function whose parameter is not a 'pointer to const'.	Rule-7.4
753	String literal assigned to pointer which is not a 'pointer to const'.	Rule-7.4
755	[C] 'return' expression is not of arithmetic type.	Rule-1.1
756	[C] 'return' expression is not of compatible 'struct'/'union' type.	Rule-1.1
757	[C] 'return' expression points to a more heavily qualified type.	Rule-1.1
758	[C] 'return' expression is not of compatible pointer type.	Rule-1.1
759	An object of union type has been defined.	Rule-19.2
766	[C] 'continue' statement found outside an iteration statement.	Rule-1.1
767	[C] 'break' statement found outside a 'switch' or iteration statement.	Rule-1.1
768	[C] 'case' or 'default' found outside a 'switch' statement.	Rule-1.1
771	More than one 'break' statement has been used to terminate this iteration statement.	Rule-15.4
774	[C] 'auto' may not be specified on global declaration of '%s'.	Rule-1.1
775	[C] 'register' may not be specified on global declaration of '%s'.	Rule-1.1
777	[U] External identifier does not differ from other identifier(s) (e.g. '%s') within the specified number of significant characters.	Rule-5.1
777	[U] External identifier does not differ from other identifier(s) (e.g. '%s') within the specified number of significant characters.	Rule-1.3
779	[U] Identifier does not differ from other identifier(s) (e.g. '%s') within the specified number of significant characters.	Rule-5.2
779	[U] Identifier does not differ from other identifier(s) (e.g. '%s') within the specified number of significant characters.	Rule-1.3
801	[C] The '##' operator may not be the first token in a macro replacement list.	Rule-1.1
802	[C] The '##' operator may not be the last token in a macro replacement list.	Rule-1.1
803	[C] The '#' operator may only appear before a macro parameter.	Rule-1.1
804	[C] Macro parameter '%s' is not unique.	Rule-1.1
809	[U] The '#include' preprocessing directive has not been followed by <h-char-sequence> or "s-char-sequence".	Rule-1.3
809	[U] The '#include' preprocessing directive has not been followed by <h-char-sequence> or "s-char-sequence".	Rule-20.3
810	[L] '#include "%s"' causes nesting to exceed 8 levels - program does not conform strictly to ISO:C90.	Dir-1.1
811	[C] The glue operator '##' may only appear in a '#define' preprocessing directive.	Rule-1.1
812	[C] Header name token '<text>' found outside '#include' preprocessing directive.	Rule-1.1
813	[U] Using any of the characters ' " or /* in '#include <%s>' gives undefined behaviour.	Rule-20.2
813	[U] Using any of the characters ' " or /* in '#include <%s>' gives undefined behaviour.	Rule-1.3
814	[U] Using the characters ' or /* in '#include "%s"' gives undefined behaviour.	Rule-1.3
814	[U] Using the characters ' or /* in '#include "%s"' gives undefined behaviour.	Rule-20.2
821	[C] '#include %s' does not identify a header or source file that can be processed.	Rule-1.1
828	[L] More than 8 levels of nested conditional inclusion - program does not conform strictly to ISO:C90.	Dir-1.1
830	[E] Unrecognized text encountered after a preprocessing directive.	Rule-1.2
831	[E] Use of '\\' in this '#include' line is a PC extension - this usage is non-portable.	Rule-20.2
831	[E] Use of '\\' in this '#include' line is a PC extension - this usage is non-portable.	Rule-1.2
834	[C] Function-like macro '%s()' is being redefined as an object-like macro.	Rule-1.1
835	[C] Macro '%s' is being redefined with different parameter names.	Rule-1.1
836	[U] Definition of macro named 'defined'.	Rule-21.1

836	[U] Definition of macro named 'defined'.	Rule-1.3
837	[U] Use of '#undef' to remove the operator 'defined'.	Rule-1.3
841	Using '#undef'.	Rule-20.5
844	[C] Macro '%s' is being redefined with a different replacement list.	Rule-1.1
845	[C] Object-like macro '%s' is being redefined as a function-like macro.	Rule-1.1
848	[U] Attempting to #undef '%s', which is a predefined macro name.	Rule-21.1
848	[U] Attempting to #undef '%s', which is a predefined macro name.	Rule-1.3
850	[C99] Macro argument is empty.	Dir-1.1
851	[C] More arguments in macro call than specified in definition.	Rule-1.1
852	[C] Unable to find the ')' that marks the end of the macro call.	Rule-1.1
853	[U] Macro arguments contain a sequence of tokens that has the form of a preprocessing directive.	Rule-20.6
853	[U] Macro arguments contain a sequence of tokens that has the form of a preprocessing directive.	Rule-1.3
854	[U] Attempting to #define '%s', which is a predefined macro name.	Rule-1.3
854	[U] Attempting to #define '%s', which is a predefined macro name.	Rule-21.1
856	[C] Fewer arguments in macro call than specified in definition.	Rule-1.1
857	[L] Number of macro definitions exceeds 1024 - program does not conform strictly to ISO:C90.	Dir-1.1
858	[L] Number of macro parameters exceeds 31 - program does not conform strictly to ISO:C90.	Dir-1.1
859	[L] Number of arguments in macro call exceeds 31 - program does not conform strictly to ISO:C90.	Dir-1.1
864	[U] '#line' directive specifies line number which is not in the range 1 to 32767.	Rule-1.3
865	[U] '#line' directive is badly formed.	Rule-1.3
866	[C] The string literal in a '#line' directive cannot be a 'wide string literal'.	Rule-1.1
867	[U] '#line' has not been followed by a line number.	Rule-1.3
872	[U] Result of '##' operator is not a legal preprocessing token.	Rule-1.3
873	[C] Preprocessing token cannot be converted to an actual token.	Rule-1.1
874	[U] Character string literal and wide character string literal are adjacent.	Rule-1.3
875	[L] String literal exceeds 509 characters - program does not conform strictly to ISO:C90.	Dir-1.1
877	[C] '#if' and '#elif' expressions may contain only integral constants.	Rule-1.1
883	Include file code is not protected against repeated inclusion	Dir-4.10
885	[U] The token 'defined' is generated in the expansion of this macro.	Rule-1.3
887	[U] Use of 'defined' must match either 'defined(identifier)' or 'defined identifier'.	Rule-1.3
888	[U] 'defined' requires an identifier as an argument.	Rule-1.3
899	[E] Unrecognized preprocessing directive has been ignored - assumed to be a language extension.	Rule-1.2
914	[U] Source file does not end with a newline character.	Rule-1.3
915	[U] Source file ends with a backslash character followed by a newline.	Rule-1.3
930	[C99] Trailing comma at the end of an enumerator-list.	Dir-1.1
940	[C] Illegal usage of a variably modified type.	Rule-1.1
941	[C] A variable length array may not be initialized.	Rule-1.1
942	[U] A * can only be used to specify array size within function prototype scope.	Rule-1.3
943	[C] Jump to label '%s' is a jump into the scope of an identifier with variably modified type.	Rule-1.1
944	[C] The label '%s' is inside the scope of an identifier with variably modified type.	Rule-1.1
945	[C99] WARNING. Operand of sizeof is an expression of variable length array type.	Rule-18.8
1001	[E] '#include %s' is a VMS extension.	Rule-1.2
1002	[E] '%s' is not a legal identifier in ISO C.	Rule-1.2
1003	[E] '#%s' is a language extension for in-line assembler. All statements located between #asm and #endasm will be ignored.	Dir-4.2
1003	[E] '#%s' is a language extension for in-line assembler. All statements located between #asm and #endasm will be ignored.	Rule-1.2
1006	[E] This in-line assembler construct is a language extension. The code has been ignored.	Rule-1.2
1006	[E] This in-line assembler construct is a language extension. The code has been ignored.	Dir-4.2
1008	[E] '#%s' is not a legal ISO C preprocessing directive.	Rule-1.2
1011	[C99] Use of '//' comment.	Dir-1.1
1012	[E] Use of a C++ reference type ('type &') will be treated as a language extension.	Rule-1.2
1014	[E] Non-standard type specifier - this will be treated as a language extension.	Rule-1.2
1015	[E] '%s' is not a legal keyword in ISO C - this will be treated as a language extension.	Rule-1.2
1018	[C99] Use of LL suffix.	Dir-1.1
1019	[E] '@ address' is not supported in ISO C - this will be treated as a language extension.	Rule-1.2
1020	[E] '__typeof__' is not supported in ISO C, and is treated as a language extension.	Rule-1.2
1021	[E] A statement expression is not supported in ISO C, and is treated as a language extension.	Rule-1.2

1022	[E] '__alignof__' is not supported in ISO C, and is treated as a language extension.	Rule-1.2
1023	[C] Using '__alignof__' on function types is illegal.	Rule-1.1
1024	[C] Using '__alignof__' on incomplete types is illegal.	Rule-1.1
1025	[C] Using '__alignof__' on bit-fields is illegal.	Rule-1.1
1026	[E] The indicated @word construct has been ignored.	Rule-1.2
1027	[C99] Use of type 'long long'.	Dir-1.1
1028	[E] Use of the sizeof operator in a preprocessing directive is a language extension.	Rule-1.2
1029	[E] Whitespace encountered between backslash and new-line has been ignored.	Rule-1.2
1030	[C99] Macro defined with variable argument list.	Dir-1.1
1031	[C99] Initializer for 'struct', 'union' or array type is not a constant expression.	Dir-1.1
1033	[C] The identifier __VA_ARGS__ may only be used in the replacement list of a variadic macro.	Rule-1.1
1034	[E] Macro defined with named variable argument list. This is a language extension.	Rule-1.2
1035	[E] No macro arguments supplied for variable argument list. This is a language extension.	Rule-1.2
1036	[E] Comma before ## ignored in expansion of variadic macro. This is a language extension.	Rule-1.2
1037	[E] Arrays of length zero are a language extension.	Rule-1.2
1038	[E] The sequence ", ##__VA_ARGS__" is a language extension.	Rule-1.2
1041	[E] Empty aggregate initializers are a language extension.	Rule-1.2
1042	[E] Using I64 or UI64 as an integer constant suffix. This is a language extension.	Rule-1.2
1043	[E] Defining an anonymous union object. This is a language extension.	Rule-1.2
1044	[E] Defining an anonymous struct object. This is a language extension.	Rule-1.2
1045	[E] Use of the #include_next preprocessing directive is a language extension.	Rule-1.2
1046	[E] Function is being declared with default argument syntax. This is a language extension.	Rule-1.2
1047	[C] Function is being declared with default argument syntax after a previous call to the function. This is not allowed.	Rule-1.1
1048	[C] Default argument values are missing for some parameters in this function declaration. This is not allowed.	Rule-1.1
1051	[C99] A variable length array has been declared.	Rule-18.8
1052	[C99] A variable length array of unspecified size has been declared.	Rule-18.8
1053	[C99] Designators have been used in this initialization list.	Dir-1.1
1054	[C99] A compound literal has been used.	Dir-1.1
1055	[C99] The keyword 'inline' has been used.	Dir-1.1
1056	[C99] The keyword '_Bool' has been used.	Dir-1.1
1257	An integer constant suffixed with L or LL is being converted to a type of lower rank on assignment.	Rule-10.3
1264	A suffixed floating constant is being converted to a different floating type on assignment.	Rule-10.3
1265	An unsuffixed floating constant is being converted to a different floating type on assignment.	Rule-10.3
1266	A floating constant is being converted to integral type on assignment.	Rule-10.3
1280	A lowercase letter L (l) has been used in an integer or floating suffix.	Rule-7.3
1281	Integer literal constant is of an unsigned type but does not include a "U" suffix.	Rule-7.2
1291	An integer constant of 'essentially unsigned' type is being converted to signed type on assignment.	Rule-10.3
1292	An integer constant of 'essentially signed' type is being converted to type char on assignment.	Rule-10.3
1293	An integer constant of 'essentially unsigned' type is being converted to type char on assignment.	Rule-10.3
1294	An integer constant of 'essentially signed' type is being converted to type _Bool on assignment.	Rule-10.3
1295	An integer constant of 'essentially unsigned' type is being converted to type _Bool on assignment.	Rule-10.3
1296	An integer constant of 'essentially signed' type is being converted to enum type on assignment.	Rule-10.3
1297	An integer constant of 'essentially unsigned' type is being converted to enum type on assignment.	Rule-10.3
1298	An integer constant of 'essentially signed' type is being converted to floating type on assignment.	Rule-10.3
1299	An integer constant of 'essentially unsigned' type is being converted to floating type on assignment.	Rule-10.3
1330	The parameter identifiers in this function declaration differ from those in a previous declaration.	Rule-8.3
1331	Type or number of arguments doesn't match previous use of the function.	Rule-1.3
1332	Type or number of arguments doesn't match prototype found later.	Rule-1.3
1333	Type or number of arguments doesn't match function definition found later.	Rule-1.3
1335	Parameter identifiers missing in function prototype declaration.	Rule-8.2
1336	Parameter identifiers missing in declaration of a function type.	Rule-8.2
1337	Function defined with a variable number of parameters.	Rule-17.1
1460	'Switch' label value, %s, not contained in enum type.	Rule-2.1
1503	The function '%1s' is defined but is not used within this project.	Rule-2.1
1504	The object '%1s' is only referenced in the translation unit where it is defined.	Rule-8.7
1505	The function '%1s' is only referenced in the translation unit where it is defined.	Rule-8.7
1506	The identifier '%1s' is declared as a typedef and is used elsewhere for a different kind of declaration.	Rule-5.6

1507	'%1s' is used as a typedef for different types.	Rule-5.6
1508	The typedef '%1s' is declared in more than one location.	Rule-5.6
1509	'%1s' has external linkage and has multiple definitions.	Rule-1.3
1509	'%1s' has external linkage and has multiple definitions.	Rule-8.6
1510	'%1s' has external linkage and has incompatible declarations.	Rule-1.3
1513	Identifier '%1s' with external linkage has separate non-defining declarations in more than one location.	Rule-8.5
1514	The object '%1s' is only referenced by function '%2s', in the translation unit where it is defined	Rule-8.9
1520	Functions are indirectly recursive.	Rule-17.2
1525	Object/function with external linkage has same identifier as another object/function with internal linkage.	Rule-5.8
1525	Object/function with external linkage has same identifier as another object/function with internal linkage.	Rule-5.9
1526	Object with no linkage has same identifier as another object/function with external linkage.	Rule-5.8
1527	Object/function with internal linkage has same identifier as another object/function with internal linkage.	Rule-5.9
1528	Object with no linkage has same identifier as another object/function with internal linkage.	Rule-5.9
1800	The %1s operand (essential type: '%2s') will be implicitly converted to a floating type, '%3s', in this arithmetic operation.	Rule-10.4
1802	The %1s operand (essential type: '%2s') will be implicitly converted to a floating type, '%3s', in this relational operation.	Rule-10.4
1803	The %1s operand (essential type: '%2s') will be implicitly converted to a floating type, '%3s', in this equality operation.	Rule-10.4
1804	The %1s operand (essential type: '%2s') will be implicitly converted to a floating type, '%3s', in this conditional operation.	Rule-10.4
1810	An operand of 'essentially character' type is being added to another operand of 'essentially character' type.	Rule-10.2
1811	An operand of 'essentially character' type is being subtracted from an operand of 'essentially signed' type.	Rule-10.2
1812	An operand of 'essentially character' type is being subtracted from an operand of 'essentially unsigned' type.	Rule-10.2
1813	An operand of 'essentially character' type is being balanced with an operand of 'essentially floating' type in this arithmetic operation.	Rule-10.2
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4
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.	Rule-10.4

1850	The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this arithmetic operation.	Rule-10.4
1851	The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this bitwise operation.	Rule-10.4
1852	The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this relational operation.	Rule-10.4
1853	The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this equality operation.	Rule-10.4
1854	The %1s operand is 'essentially unsigned' (%2s) but will be implicitly converted to a signed type (%3s) in this conditional operation.	Rule-10.4
1860	The operands of this arithmetic operator are of different 'essential signedness' but will generate a result of type 'signed int'.	Rule-10.4
1861	The operands of this bitwise operator are of different 'essential signedness' but will generate a result of type 'signed int'.	Rule-10.4
1862	The operands of this relational operator are of different 'essential signedness' but will both be promoted to 'signed int' for comparison.	Rule-10.4
1863	The operands of this equality operator are of different 'essential signedness' but will both be promoted to 'signed int' for comparison.	Rule-10.4
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'.	Rule-10.4
1880	The operands of this relational operator are expressions of different 'essential type' categories (%1s and %2s).	Rule-10.4
1881	The operands of this equality operator are expressions of different 'essential type' categories (%1s and %2s).	Rule-10.4
1882	The 2nd and 3rd operands of this conditional operator are expressions of different 'essential type' categories (%1s and %2s).	Rule-10.4
1890	A composite expression of 'essentially signed' type (%1s) is being implicitly converted to a wider signed type, '%2s'.	Rule-10.7
1891	A composite expression of 'essentially unsigned' type (%1s) is being implicitly converted to a wider unsigned type, '%2s'.	Rule-10.7
1892	A composite expression of 'essentially floating' type (%1s) is being implicitly converted to a wider floating type, '%2s'.	Rule-10.7
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.	Rule-10.7
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.	Rule-10.7
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.	Rule-10.7
2001	A 'goto' statement has been used.	Rule-15.1
2002	No 'default' label found in this 'switch' statement.	Rule-16.4
2003	The preceding 'switch' clause is not empty and does not end with a 'jump' statement. Execution will fall through.	Rule-16.3
2004	No concluding 'else' exists in this 'if'-'else'-'if' statement.	Rule-15.7
2008	Code statements precede the first label in this 'switch' construct.	Rule-16.1
2009	This 'default' label is not the final 'case' label within the 'switch' block.	Rule-16.5
2019	'Switch' label is located within a nested code block.	Rule-16.2
2020	Final 'switch' clause does not end with an explicit 'jump' statement.	Rule-16.3
2050	The 'int' type specifier has been omitted from a function declaration.	Rule-8.1
2051	The 'int' type specifier has been omitted from an object declaration.	Rule-8.1
2212	Body of control statement is not enclosed within braces.	Rule-15.6
2214	Body of control statement is on the same line and is not enclosed within braces.	Rule-15.6
2461	Loop control variable, %s, has file scope.	Rule-14.2
2462	The variable initialized in the first expression of this 'for' statement is not the variable identified as the 'loop control variable' (%s).	Rule-14.2
2463	The variable incremented in the third expression of this 'for' statement is not the variable identified as the 'loop control variable' (%s).	Rule-14.2
2464	Loop control variable, %s, modified twice in for-loop header.	Rule-14.2
2467	Loop control variable, %s, is not modified inside loop.	Rule-14.2
2469	Loop control variable in this 'for' statement, %s, is modified in the body of the loop.	Rule-14.2
2471	Unable to identify a loop control variable.	Rule-14.2
2472	More than one possible loop control variable.	Rule-14.2
2547	This declaration of tag '%s' hides a more global declaration.	Rule-5.3
2742	This 'if' controlling expression is a constant expression and its value is 'false'.	Rule-2.1

2744	This 'while' or 'for' loop controlling expression is a constant expression and its value is 'false'. The loop will not be entered.	Rule-2.1
2771	Definite: These pointers address different objects.	Rule-18.2
2771	Definite: These pointers address different objects.	Rule-18.3
2772	Apparent: These pointers address different objects.	Rule-18.3
2772	Apparent: These pointers address different objects.	Rule-18.2
2776	Definite: Copy between overlapping objects.	Rule-19.1
2777	Apparent: Copy between overlapping objects.	Rule-19.1
2790	Constant: Right hand operand of shift operator is negative or too large.	Rule-12.2
2791	Definite: Right hand operand of shift operator is negative or too large.	Dir-4.1
2792	Apparent: Right hand operand of shift operator is negative or too large.	Dir-4.1
2800	Constant: Overflow in signed arithmetic operation.	Rule-1.3
2801	Definite: Overflow in signed arithmetic operation.	Dir-4.1
2802	Apparent: Overflow in signed arithmetic operation.	Dir-4.1
2810	Constant: Dereference of NULL pointer.	Rule-1.3
2811	Definite: Dereference of NULL pointer.	Dir-4.1
2812	Apparent: Dereference of NULL pointer.	Dir-4.1
2820	Constant: Arithmetic operation on NULL pointer.	Rule-1.3
2821	Definite: Arithmetic operation on NULL pointer.	Dir-4.1
2822	Apparent: Arithmetic operation on NULL pointer.	Dir-4.1
2830	Constant: Division by zero.	Rule-1.3
2831	Definite: Division by zero.	Dir-4.1
2832	Apparent: Division by zero.	Dir-4.1
2840	Constant: Dereference of an invalid pointer value.	Rule-1.3
2841	Definite: Dereference of an invalid pointer value.	Dir-4.1
2842	Apparent: Dereference of an invalid pointer value.	Dir-4.1
2845	Constant: Maximum number of characters to be written is larger than the target buffer size.	Dir-4.1
2846	Definite: Maximum number of characters to be written is larger than the target buffer size.	Dir-4.1
2847	Apparent: Maximum number of characters to be written is larger than the target buffer size.	Dir-4.1
2850	Constant: Implicit conversion to a signed integer type of insufficient size.	Rule-10.3
2851	Definite: Implicit conversion to a signed integer type of insufficient size.	Rule-10.3
2852	Apparent: Implicit conversion to a signed integer type of insufficient size.	Rule-10.3
2855	Constant: Casting to a signed integer type of insufficient size.	Dir-1.1
2856	Definite: Casting to a signed integer type of insufficient size.	Dir-1.1
2857	Apparent: Casting to a signed integer type of insufficient size.	Dir-1.1
2860	Constant: Implementation-defined value resulting from left shift operation on expression of signed type.	Dir-1.1
2861	Definite: Implementation-defined value resulting from left shift operation on expression of signed type.	Dir-1.1
2862	Apparent: Implementation-defined value resulting from left shift operation on expression of signed type.	Dir-1.1
2871	Infinite loop identified.	Dir-4.1
2872	This loop, if entered, will never terminate.	Dir-4.1
2877	This loop will never be executed more than once.	Dir-4.1
2880	This code is unreachable.	Rule-2.1
2882	This 'switch' statement will bypass the initialization of local variables.	Rule-2.1
2883	This 'goto' statement will always bypass the initialization of local variables.	Rule-9.1
2887	Function 'main' ends with an implicit 'return' statement.	Rule-17.4
2888	This function has been declared with a non-void 'return' type but ends with an implicit 'return ;' statement.	Rule-17.4
2889	This function has more than one 'return' path.	Rule-15.5
2890	Constant: Negative value implicitly converted to an unsigned type.	Dir-1.1
2891	Definite: Negative value implicitly converted to an unsigned type.	Dir-1.1
2892	Apparent: Negative value implicitly converted to an unsigned type.	Dir-1.1
2895	Constant: Negative value cast to an unsigned type.	Dir-1.1
2896	Definite: Negative value cast to an unsigned type.	Dir-1.1
2897	Apparent: Negative value cast to an unsigned type.	Dir-1.1
2900	Constant: Positive integer value truncated by implicit conversion to a smaller unsigned type.	Rule-10.3
2901	Definite: Positive integer value truncated by implicit conversion to a smaller unsigned type.	Rule-10.3
2902	Apparent: Positive integer value truncated by implicit conversion to a smaller unsigned type.	Rule-10.3

2910	Constant: Wraparound in unsigned arithmetic operation.	Rule-12.4
2930	Constant: Computing an invalid pointer value.	Rule-18.1
2931	Definite: Computing an invalid pointer value.	Rule-18.1
2932	Apparent: Computing an invalid pointer value.	Rule-18.1
2961	Definite: Using value of uninitialized automatic object '%s'.	Rule-9.1
2962	Apparent: Using value of uninitialized automatic object '%s'.	Rule-9.1
2971	Definite: Passing address of uninitialized object '%s' to a function parameter declared as a pointer to const.	Rule-9.1
2972	Apparent: Passing address of uninitialized object '%s' to a function parameter declared as a pointer to const.	Rule-9.1
2980	The value of this function parameter is never used before being modified.	Rule-2.2
2981	This initialization is redundant. The value of this object is never used before being modified.	Rule-2.2
2982	This assignment is redundant. The value of this object is never used before being modified.	Rule-2.2
2983	This assignment is redundant. The value of this object is never subsequently used.	Rule-2.2
2984	This operation is redundant. The value of the result is always '%1s'.	Rule-2.2
2985	This operation is redundant. The value of the result is always that of the left-hand operand.	Rule-2.2
2986	This operation is redundant. The value of the result is always that of the right-hand operand.	Rule-2.2
2990	The value of this loop controlling expression is always 'true'.	Rule-14.3
2991	The value of this 'if' controlling expression is always 'true'.	Rule-14.3
2992	The value of this 'if' controlling expression is always 'false'.	Rule-14.3
2993	The value of this 'do - while' loop controlling expression is always 'false'. The loop will only be executed once.	Rule-14.3
2994	The value of this 'while' or 'for' loop controlling expression is always 'false'. The loop will not be entered.	Rule-14.3
2995	The result of this logical operation is always 'true'.	Rule-2.2
2996	The result of this logical operation is always 'false'.	Rule-2.2
3001	Function has been declared with an empty parameter list.	Rule-8.2
3002	Defining '%s()' with an identifier list and separate parameter declarations is an obsolescent feature.	Rule-8.2
3003	This character constant is being interpreted as a NULL pointer constant.	Rule-11.9
3004	This integral constant expression is being interpreted as a NULL pointer constant.	Rule-11.9
3006	This function contains a mixture of in-line assembler statements and C statements.	Dir-4.3
3007	"void" has been omitted when defining a function with no parameters.	Rule-8.2
3101	Unary '-' applied to an operand of type unsigned int or unsigned long gives an unsigned result.	Rule-10.1
3102	Unary '-' applied to an operand whose underlying type is unsigned.	Rule-10.1
3108	Nested comments are not recognized in the ISO standard.	Rule-3.1
3110	The left-hand operand of this ',' has no side effects.	Rule-2.2
3112	This statement has no side-effect - it can be removed.	Rule-2.2
3113	[U] 'return' statement includes no expression but function '%s()' is implicitly of type 'int'.	Rule-1.3
3113	[U] 'return' statement includes no expression but function '%s()' is implicitly of type 'int'.	Rule-17.4
3114	[U] Function '%s()' is implicitly of type 'int' but ends without returning a value.	Rule-17.4
3114	[U] Function '%s()' is implicitly of type 'int' but ends without returning a value.	Rule-1.3
3115	Unrecognized preprocessing directive has been ignored because of conditional inclusion directives.	Rule-20.13
3116	Unrecognized #pragma arguments '%s' This #pragma directive has been ignored.	Dir-1.1
3200	'%s' returns a value which is not being used.	Rule-17.7
3202	The label '%s:' is not used in this function and could be removed.	Rule-2.6
3206	The parameter '%s' is not used in this function.	Rule-2.7
3217	Address of automatic object exported to a pointer with linkage or wider scope.	Rule-18.6
3218	File scope static, '%s', is only accessed in one function.	Rule-8.9
3219	Static function '%s()' is not used within this translation unit.	Rule-2.1
3221	Function with external linkage declared at block scope.	Rule-8.5
3222	Object with external linkage declared at block scope.	Rule-8.5
3224	This identifier has previously been declared with internal linkage but is not declared here with the static storage class specifier.	Rule-8.8
3225	Address of automatic object exported using a function parameter.	Rule-18.6
3226	The result of an assignment is being used in an arithmetic operation or another assigning operation.	Rule-13.4
3230	Address of automatic object assigned to local pointer with static storage duration.	Rule-18.6
3234	Declarations precede the first label in this 'switch' construct.	Rule-16.1

3236	[C] 'inline' may not be applied to function 'main'.	Rule-1.1
3237	[C] inline function '%1s' has external linkage and is defining an object, '%2s', with static storage duration.	Rule-1.1
3238	[C] inline function '%1s' has external linkage and is referring to an object, '%2s', with internal linkage.	Rule-1.1
3239	[U] inline function '%1s' has external linkage, but is not defined within this translation unit.	Rule-1.3
3240	inline function '%s' is being defined with external linkage.	Rule-8.10
3243	inline function '%s' is also an 'external definition'.	Rule-8.10
3244	[C] 'inline' may only be used in the declaration of a function identifier.	Rule-1.1
3260	Typedef defined with more than 2 levels of indirection.	Rule-18.5
3261	Member of struct/union defined with more than 2 levels of indirection.	Rule-18.5
3262	Object defined or declared with more than 2 levels of indirection.	Rule-18.5
3263	Function defined or declared with a return type which has more than 2 levels of indirection.	Rule-18.5
3305	Pointer cast to stricter alignment.	Rule-11.3
3307	The operand of 'sizeof' is an expression with implied side effects, but they will not be evaluated.	Rule-13.6
3311	[u] An earlier jump to this statement will bypass the initialization of local variables.	Rule-15.3
3311	[u] An earlier jump to this statement will bypass the initialization of local variables.	Rule-1.3
3312	[u] This goto statement will jump into a previous block and bypass the initialization of local variables.	Rule-1.3
3315	This 'switch' statement contains only a single path - it is redundant.	Rule-16.6
3317	'#if...' not matched by '#endif' in included file. This is probably an error.	Rule-20.14
3318	'#else'/'#elif'/'#endif' in included file matched '#if...' in parent file. This is probably an error.	Rule-20.14
3319	[U] Function called with number of arguments which differs from number of parameters in definition.	Rule-1.3
3320	Type of argument no. %s differs from its type in definition of function.	Rule-1.3
3326	The result of an assignment is being used in a logical operation.	Rule-13.4
3332	The macro '%s' used in this '#if' or '#elif' expression is not defined.	Rule-20.9
3334	This declaration of '%s' hides a more global declaration.	Rule-5.3
3335	No function declaration. Implicit declaration inserted: 'extern int %s();'.	Rule-17.3
3340	Floating point variable used as 'for' loop control variable.	Rule-14.1
3342	Controlling expression of 'for' loop is a floating point comparison.	Rule-14.1
3344	Controlling expression is not an 'essentially Boolean' expression.	Rule-14.4
3389	Extra parentheses recommended to clarify the ordering of a % operator and another arithmetic operator (* / % + -).	Rule-12.1
3391	Extra parentheses recommended. A conditional operation is the operand of another conditional operator.	Rule-12.1
3392	Extra parentheses recommended. A shift, relational or equality operation is the operand of a second identical operator.	Rule-12.1
3394	Extra parentheses recommended. A shift, relational or equality operation is the operand of a different operator with the same precedence.	Rule-12.1
3395	Extra parentheses recommended. A * or / operation is the operand of a + or - operator.	Rule-12.1
3396	Extra parentheses recommended. A binary operation is the operand of a conditional operator.	Rule-12.1
3397	Extra parentheses recommended. A binary operation is the operand of a binary operator with different precedence.	Rule-12.1
3402	Braces are needed to clarify the structure of this 'if'-'if'-'else' statement.	Rule-15.6
3406	Object/function '%s', with external linkage, has been defined in a header file.	Rule-8.6
3408	'%s' has external linkage and is being defined without any previous declaration.	Rule-8.4
3410	Macro parameter not enclosed in ().	Rule-20.7
3415	Right hand operand of '&&' or '||' is an expression with possible side effects.	Rule-13.5
3417	The comma operator has been used outside a 'for' statement.	Rule-12.3
3418	The comma operator has been used in a 'for' statement.	Rule-12.3
3425	One branch of this conditional operation is a redundant expression.	Rule-2.2
3426	Right hand side of comma expression has no side effect and its value is not used.	Rule-2.2
3427	Right hand side of logical operator has no side effect and its value is not used.	Rule-2.2
3437	[u] The assert macro has been suppressed to call a function of that name.	Rule-1.3
3438	[U] #undef'ing the assert macro to call a function of that name causes undefined behaviour.	Rule-1.3
3439	Macro redefines a keyword.	Rule-20.4
3440	Using the value resulting from a ++ or -- operation.	Rule-13.3
3447	'%s' is being declared with external linkage but this declaration is not in a header file.	Rule-8.5
3448	Declaration of typedef '%s' is not in a header file although it is used in a definition or declaration with external linkage.	Rule-5.6
3451	The global identifier '%s' has been declared in more than one file.	Rule-8.5

3453	A function could probably be used instead of this function-like macro.	Dir-4.9
3601	Trigraphs (??x) are an ISO feature.	Rule-4.2
3628	Octal escape sequences used in a character constant or string literal.	Rule-7.1
3660	Named bit-field consisting of a single bit declared with a signed type.	Rule-6.2
3664	[E] Using a dot operator to access an individual bit is a language extension.	Rule-1.2
3665	Unnamed bit-field consisting of a single bit declared with a signed type.	Rule-6.2
3670	Recursive call to function containing this call.	Rule-17.2
3673	The object addressed by the pointer parameter '%s' is not modified and so the pointer could be of type 'pointer to const'.	Rule-8.13
3675	Function parameter declared with type qualification which differs from previous declaration.	Rule-8.3
3684	Array declared with unknown size.	Rule-8.11
4140	Address of automatic object exported in function return value.	Rule-18.6
4301	An expression of 'essentially Boolean' type (%1s) is being cast to character type '%2s'.	Rule-10.5
4302	An expression of 'essentially Boolean' type (%1s) is being cast to enum type '%2s'.	Rule-10.5
4303	An expression of 'essentially Boolean' type (%1s) is being cast to signed type '%2s'.	Rule-10.5
4304	An expression of 'essentially Boolean' type (%1s) is being cast to unsigned type '%2s'.	Rule-10.5
4305	An expression of 'essentially Boolean' type (%1s) is being cast to floating type '%2s'.	Rule-10.5
4310	An expression of 'essentially character' type (%1s) is being cast to Boolean type, '%2s'.	Rule-10.5
4312	An expression of 'essentially character' type (%1s) is being cast to enum type, '%2s'.	Rule-10.5
4315	An expression of 'essentially character' type (%1s) is being cast to floating type, '%2s'.	Rule-10.5
4320	An expression of 'essentially enum' type (%1s) is being cast to Boolean type, '%2s'.	Rule-10.5
4322	An expression of 'essentially enum' type (%1s) is being cast to a different enum type, '%2s'.	Rule-10.5
4330	An expression of 'essentially signed' type (%1s) is being cast to Boolean type '%2s'.	Rule-10.5
4332	An expression of 'essentially signed' type (%1s) is being cast to enum type, '%2s'.	Rule-10.5
4340	An expression of 'essentially unsigned' type (%1s) is being cast to Boolean type '%2s'.	Rule-10.5
4342	An expression of 'essentially unsigned' type (%1s) is being cast to enum type '%2s'.	Rule-10.5
4350	An expression of 'essentially floating' type (%1s) is being cast to Boolean type '%2s'.	Rule-10.5
4351	An expression of 'essentially floating' type (%1s) is being cast to character type '%2s'.	Rule-10.5
4352	An expression of 'essentially floating' type (%1s) is being cast to enum type, '%2s'.	Rule-10.5
4390	A composite expression of 'essentially signed' type (%1s) is being cast to a wider signed type, '%2s'.	Rule-10.8
4391	A composite expression of 'essentially unsigned' type (%1s) is being cast to a wider unsigned type, '%2s'.	Rule-10.8
4392	A composite expression of 'essentially floating' type (%1s) is being cast to a wider floating type, '%2s'.	Rule-10.8
4393	A composite expression of 'essentially signed' type (%1s) is being cast to a different type category, '%2s'.	Rule-10.8
4394	A composite expression of 'essentially unsigned' type (%1s) is being cast to a different type category, '%2s'.	Rule-10.8
4395	A composite expression of 'essentially floating' type (%1s) is being cast to a different type category, '%2s'.	Rule-10.8
4397	An expression which is the result of a ~ or << operation has not been cast to its essential type.	Rule-10.7
4398	An expression which is the result of a ~ or << operation has been cast to a different essential type category.	Rule-10.8
4399	An expression which is the result of a ~ or << operation has been cast to a wider type.	Rule-10.8
4401	An expression of 'essentially Boolean' type (%1s) is being converted to character type, '%2s' on assignment.	Rule-10.3
4402	An expression of 'essentially Boolean' type (%1s) is being converted to enum type, '%2s' on assignment.	Rule-10.3
4403	An expression of 'essentially Boolean' type (%1s) is being converted to signed type, '%2s' on assignment.	Rule-10.3
4404	An expression of 'essentially Boolean' type (%1s) is being converted to unsigned type, '%2s' on assignment.	Rule-10.3
4405	An expression of 'essentially Boolean' type (%1s) is being converted to floating type, '%2s' on assignment.	Rule-10.3
4410	An expression of 'essentially character' type (%1s) is being converted to Boolean type, '%2s' on assignment.	Rule-10.3
4412	An expression of 'essentially character' type (%1s) is being converted to enum type, '%2s' on assignment.	Rule-10.3
4413	An expression of 'essentially character' type (%1s) is being converted to signed type, '%2s' on assignment.	Rule-10.3
4414	An expression of 'essentially character' type (%1s) is being converted to unsigned type, '%2s' on	Rule-10.3

	assignment.	
4415	An expression of 'essentially character' type (%1s) is being converted to floating type, '%2s' on assignment.	Rule-10.3
4420	An expression of 'essentially enum' type (%1s) is being converted to Boolean type, '%2s' on assignment.	Rule-10.3
4421	An expression of 'essentially enum' type (%1s) is being converted to character type, '%2s' on assignment.	Rule-10.3
4422	An expression of 'essentially enum' type (%1s) is being converted to a different enum type, '%2s' on assignment.	Rule-10.3
4423	An expression of 'essentially enum' type (%1s) is being converted to signed type, '%2s' on assignment.	Rule-10.3
4424	An expression of 'essentially enum' type (%1s) is being converted to unsigned type, '%2s' on assignment.	Rule-10.3
4425	An expression of 'essentially enum' type (%1s) is being converted to floating type, '%2s' on assignment.	Rule-10.3
4430	An expression of 'essentially signed' type (%1s) is being converted to Boolean type, '%2s' on assignment.	Rule-10.3
4431	An expression of 'essentially signed' type (%1s) is being converted to character type, '%2s' on assignment.	Rule-10.3
4432	An expression of 'essentially signed' type (%1s) is being converted to enum type, '%2s' on assignment.	Rule-10.3
4434	A non-constant expression of 'essentially signed' type (%1s) is being converted to unsigned type, '%2s' on assignment.	Rule-10.3
4435	A non-constant expression of 'essentially signed' type (%1s) is being converted to floating type, '%2s' on assignment.	Rule-10.3
4436	A constant expression of 'essentially signed' type (%1s) is being converted to unsigned type, '%2s' on assignment.	Rule-10.3
4437	A constant expression of 'essentially signed' type (%1s) is being converted to floating type, '%2s' on assignment.	Rule-10.3
4440	An expression of 'essentially unsigned' type (%1s) is being converted to Boolean type, '%2s' on assignment.	Rule-10.3
4441	An expression of 'essentially unsigned' type (%1s) is being converted to character type, '%2s' on assignment.	Rule-10.3
4442	An expression of 'essentially unsigned' type (%1s) is being converted to enum type, '%2s' on assignment.	Rule-10.3
4443	A non-constant expression of 'essentially unsigned' type (%1s) is being converted to a wider signed type, '%2s' on assignment.	Rule-10.3
4445	An expression of 'essentially unsigned' type (%1s) is being converted to floating type, '%2s' on assignment.	Rule-10.3
4446	A non-constant expression of 'essentially unsigned' type (%1s) is being converted to signed type, '%2s' on assignment.	Rule-10.3
4447	A constant expression of 'essentially unsigned' type (%1s) is being converted to signed type, '%2s' on assignment.	Rule-10.3
4450	An expression of 'essentially floating' type (%1s) is being converted to Boolean type, '%2s' on assignment.	Rule-10.3
4451	An expression of 'essentially floating' type (%1s) is being converted to character type, '%2s' on assignment.	Rule-10.3
4452	An expression of 'essentially floating' type (%1s) is being converted to enum type, '%2s' on assignment.	Rule-10.3
4453	An expression of 'essentially floating' type (%1s) is being converted to signed type, '%2s' on assignment.	Rule-10.3
4454	An expression of 'essentially floating' type (%1s) is being converted to unsigned type, '%2s' on assignment.	Rule-10.3
4460	A non-constant expression of 'essentially signed' type (%1s) is being converted to narrower signed type, '%2s' on assignment.	Rule-10.3
4461	A non-constant expression of 'essentially unsigned' type (%1s) is being converted to narrower unsigned type, '%2s' on assignment.	Rule-10.3
4462	A non-constant expression of 'essentially floating' type (%1s) is being converted to narrower floating type, '%2s' on assignment.	Rule-10.3
4463	A constant expression of 'essentially signed' type (%1s) is being converted to narrower signed type, '%2s' on assignment.	Rule-10.3
4464	A constant expression of 'essentially unsigned' type (%1s) is being converted to narrower unsigned type, '%2s' on assignment.	Rule-10.3
4465	A constant expression of 'essentially floating' type (%1s) is being converted to narrower floating type, '%2s' on assignment.	Rule-10.3

4490	A composite expression of 'essentially signed' type (%1s) is being converted to wider signed type, '%2s' on assignment.	Rule-10.6
4491	A composite expression of 'essentially unsigned' type (%1s) is being converted to wider unsigned type, '%2s' on assignment.	Rule-10.6
4492	A composite expression of 'essentially floating' type (%1s) is being converted to wider floating type, '%2s' on assignment.	Rule-10.6
4499	An expression which is the result of a ~ or << operation has been converted to a wider essential type on assignment.	Rule-10.6
4500	An expression of 'essentially Boolean' type (%1s) is being used as an array subscript.	Rule-10.1
4501	An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s).	Rule-10.1
4502	An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).	Rule-10.1
4503	An expression of 'essentially Boolean' type (%1s) is being used as the left-hand operand of this shift operator (%2s).	Rule-10.1
4504	An expression of 'essentially Boolean' type (%1s) is being used as the right-hand operand of this shift operator (%2s).	Rule-10.1
4505	An expression of 'essentially Boolean' type (%1s) is being used as the %2s operand of this relational operator (%3s).	Rule-10.1
4507	An expression of 'essentially Boolean' type (%1s) is being used as the operand of this increment/decrement operator (%2s).	Rule-10.1
4510	An expression of 'essentially character' type (%1s) is being used as an array subscript.	Rule-10.1
4511	An expression of 'essentially character' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s).	Rule-10.1
4512	An expression of 'essentially character' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).	Rule-10.1
4513	An expression of 'essentially character' type (%1s) is being used as the left-hand operand of this shift operator (%2s).	Rule-10.1
4514	An expression of 'essentially character' type (%1s) is being used as the right-hand operand of this shift operator (%2s).	Rule-10.1
4518	An expression of 'essentially character' type (%1s) is being used as the %2s operand of this logical operator (%3s).	Rule-10.1
4519	An expression of 'essentially character' type (%1s) is being used as the first operand of this conditional operator (%2s).	Rule-10.1
4521	An expression of 'essentially enum' type (%1s) is being used as the %2s operand of this arithmetic operator (%3s).	Rule-10.1
4522	An expression of 'essentially enum' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).	Rule-10.1
4523	An expression of 'essentially enum' type (%1s) is being used as the left-hand operand of this shift operator (%2s).	Rule-10.1
4524	An expression of 'essentially enum' type (%1s) is being used as the right-hand operand of this shift operator (%2s).	Rule-10.1
4527	An expression of 'essentially enum' type is being used as the operand of this increment/decrement operator.	Rule-10.1
4528	An expression of 'essentially enum' type (%1s) is being used as the %2s operand of this logical operator (%3s).	Rule-10.1
4529	An expression of 'essentially enum' type (%1s) is being used as the first operand of this conditional operator (%2s).	Rule-10.1
4532	An expression of 'essentially signed' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).	Rule-10.1
4533	An expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s).	Rule-10.1
4534	An expression of 'essentially signed' type (%1s) is being used as the right-hand operand of this shift operator (%2s).	Rule-10.1
4538	An expression of 'essentially signed' type (%1s) is being used as the %2s operand of this logical operator (%3s).	Rule-10.1
4539	An expression of 'essentially signed' type (%1s) is being used as the first operand of this conditional operator (%2s).	Rule-10.1
4542	A non-negative constant expression of 'essentially signed' type (%1s) is being used as the %2s operand of this bitwise operator (%3s).	Rule-10.1
4543	A non-negative constant expression of 'essentially signed' type (%1s) is being used as the left-hand operand of this shift operator (%2s).	Rule-10.1
4544	A non-negative constant expression of 'essentially signed' type (%1s) is being used as the right-hand operand of this shift operator (%2s).	Rule-10.1

4548	A non-negative constant expression of 'essentially signed' type (%1s) is being used as the %2s operand of this logical operator (%3s).	Rule-10.1
4549	A non-negative constant expression of 'essentially signed' type (%1s) is being used as the first operand of this conditional operator (%2s).	Rule-10.1
4558	An expression of 'essentially unsigned' type (%1s) is being used as the %2s operand of this logical operator (%3s).	Rule-10.1
4559	An expression of 'essentially unsigned' type (%1s) is being used as the first operand of this conditional operator (%2s).	Rule-10.1
4568	An expression of 'essentially floating' type (%1s) is being used as the %2s operand of this logical operator (%3s).	Rule-10.1
4569	An expression of 'essentially floating' type (%1s) is being used as the first operand of this conditional operator (%2s).	Rule-10.1
4600	The macro '%1s' is also defined in '<%2s>'.	Rule-21.1
4601	The macro '%1s' is the name of an identifier in '<%2s>'.	Rule-21.1
4602	The identifier '%1s' is declared as a macro in '<%2s>'.	Rule-21.2
4603	The object/function '%1s'is being defined with the same name as an ordinary identifier defined in '<%2s>'.	Rule-21.2
4604	The object/function '%1s' is being declared with the same name as an ordinary identifier defined in '<%2s>'.	Rule-21.2
4605	The typedef '%1s' is also defined in '<%2s>'.	Rule-21.2
4606	The typedef '%1s' has the same name as another ordinary identifier in '<%2s>'.	Rule-21.2
4607	The enum constant '%1s' has the same name as another ordinary identifier in '<%2s>'.	Rule-21.2
4608	The tag '%1s' is also defined in '<%2s>'.	Rule-21.2
5087	Use of #include directive after code fragment.	Rule-20.1
5118	Use of memory allocation or deallocation function: calloc, malloc, realloc or free.	Rule-21.3
5123	Use of standard header file <signal.h>.	Rule-21.5
5124	Use of standard header file <stdio.h>.	Rule-21.6
5125	Use of function: atof, atoi, atol or atoll.	Rule-21.7
5126	Use of function: abort, exit, getenv or system.	Rule-21.8
5127	Use of standard header file <time.h>.	Rule-21.10
5130	Use of standard header file <stdarg.h>.	Rule-17.1
5131	Use of standard header file <tgmath.h>.	Rule-21.11
5132	Use of standard header file <setjmp.h>.	Rule-21.4
5133	Comment delimiter /* or // found within comment.	Rule-3.1
5134	C++ style comment uses line splicing.	Rule-3.2
5135	Use of function: bsearch or qsort.	Rule-21.9
5136	Use of exception handling identifier: feclearexcept, fegetexceptflag, feraiseexcept, fesetexceptflag or fetestexcept.	Rule-21.12
5137	Use of 'restrict' type qualifier.	Rule-8.14
5209	Use of basic type '%s'.	Dir-4.6

Unused QAC Messages

This table lists all QAC messages that are not referenced in the M3CM-1.0-QAC-8.1.2 compliance module.

Msg No	Message
9	[Q] The length of this preprocessed source code line has exceeded the size of an internal buffer.
40	[Q] Definition of size_t differs from configured type.
41	[Q] Definition of ptrdiff_t differs from configured type.
42	[Q] Definition of wchar_t differs from configured type.
97	Recoverable dataflow problem. Analysis continues. Please inform Programming Research.
159	[Q] The maximum number of errors set by -maxerr has been exceeded - the analysis has been aborted.
231	[S] Character constant contains an invalid hex escape sequence.
234	[S] String literal contains an invalid hex escape sequence.
245	[S] Empty character constant.
249	[S] Character constant contains a new-line character.
250	[S] Character constant contains a single backslash character.
251	[S] Character constant contains a trigraph representation of a single backslash character.
257	[S] String literal is not terminated. A trigraph has been used in the construction of an escape sequence.
258	[S] String literal is not terminated. A backslash character has been used to define an escape sequence.

259	[S] String literal is not terminated.
268	[S] Comment open at end of translation unit.
269	[S] Unexpected end of file.
271	[U] Left shift operation on constant signed expression generating an undefined value.
272	[I] Apparent conversion of integer expression to a signed integer type which cannot represent the value.
273	[I] Definite conversion of integer expression to a signed integer type which cannot represent the value.
274	[I] Implicit conversion of integer constant expression to a signed integer type which cannot represent the value.
277	Implicit conversion of a constant negative value to an unsigned type.
278	[C] Overflow in signed arithmetic operation on constant operands.
280	Implicit conversion of constant negative value which assumes a two's complement representation of signed values.
281	Definite use of a negative value which assumes a two's complement representation of signed values.
282	Apparent use of a negative value which assumes a two's complement implementation of signed values.
290	Definite conversion of a negative value to an unsigned type.
291	Apparent conversion of a negative value to an unsigned type.
294	[U] Definite signed left shift operation generating an undefined value.
295	[U] Apparent signed left shift operation generating an undefined value.
296	[U] Definite overflow in signed arithmetic operation.
297	[U] Apparent overflow in signed arithmetic operation.
314	[I] Cast from a pointer to object type to a pointer to void.
315	[I] Implicit conversion from a pointer to object type to a pointer to void.
318	Redundant type qualifier used in cast
340	Using the '#error' preprocessing directive.
343	Using string literal concatenation.
344	Using function prototype syntax.
345	Using the keyword 'void'.
346	Using one of the keywords 'signed', 'const' or 'volatile'.
347	Using 'extern' in the definition of a global variable.
348	Using a bit-field.
349	Using the unary '+' operator.
350	Using the keyword 'enum'.
351	Using an initializer when defining an automatic object of struct, union or array type.
352	Using the '#elif' preprocessing directive.
428	Function identifier is not followed by () but a function call may be intended.
434	[S] The identifier '%s' has not been declared.
473	[Q] Result of 'sizeof' operation will not fit in configured type for 'size_t'.
489	The integer value 1 is being added or subtracted from a pointer.
490	Relational operator used to compare two pointers.
491	Array subscripting applied to an object of pointer type.
492	Array subscripting applied to a function parameter declared as a pointer.
500	[U] Right operand of shift operator is negative - this is undefined.
501	[U] Right operand of shift operator is too large - this is undefined.
502	A right shift on signed data may be an arithmetic or a logical shift.
503	[U] Dereference of constant NULL pointer.
504	[U] Definite dereference of NULL pointer.
505	[U] Apparent dereference of NULL pointer.
506	Possible dereference of NULL pointer.
507	[u] Arithmetic operation on constant NULL pointer.
508	[u] Definite arithmetic operation on NULL pointer.
509	[u] Apparent arithmetic operation on NULL pointer.
510	Possible arithmetic operation on NULL pointer.
553	Translation unit contains no object or function definitions with external linkage.
571	This switch case label of 'essential type' '%1s' is not consistent with a controlling expression which has an essential type of higher rank (%2s).
584	Possible division by zero.
585	[U] Apparent division by zero.
586	[U] Division by constant zero.
587	[U] Definite division by zero.
596	Value of 'case' label expression is too large for type of controlling expression in 'switch' statement.
597	Type of 'case' label expression is not consistent with type of controlling expression in 'switch' statement.

606 Object '%s' is declared using typedefs which are different to those in a previous declaration.
637 [S] Storage class specifier is illegal on 'struct' or 'union' member.
648 [S] Unexpected '{' encountered at file scope.
652 Identifiers have been provided for some but not all of the parameters in a function prototype.
666 [S] Only type qualifiers 'const' or 'volatile' are legal here in a pointer declaration.
670 [S] Function defined with invalid function header syntax.
687 Array has been initialized using concatenated strings and has fewer initializers than its declared size.
688 Array size determined by number of initializers which include concatenated string literals.
689 [u] 'Switch' statement will bypass the initialization of this local variable.
695 Type given in sizeof is not compatible with the pointed to type used to cast malloc.
696 The size of the allocated memory block is smaller than the size of the object type addressed by the pointer cast.
697 The size of the allocated memory block is not an integral multiple of the size of the object type addressed by the pointer cast.
702 Result of sizeof operator is not explicitly cast.
703 Structure has fewer initializers than its declared size. Default initialization is applied to the remainder of the members.
722 Enum constant not explicitly initialized, but a previous constant has been.
723 Initialize none, first only, or all entries in this enumerator list.
744 [U] '%s()' has been declared with a non void return type but ends with an implicit 'return ;' statement.
769 A 'break' statement has been used to terminate an iteration statement.
770 A 'continue' statement has been used.
776 [L] External identifier matches other external identifier(s) (e.g. '%s') in first 6 characters - program does not conform strictly to ISO:C90.
778 [L] Identifier matches other identifier(s) (e.g. '%s') in first 31 characters - program does not conform strictly to ISO:C90.
780 Another identifier '%s' is already in scope in a different namespace.
781 '%s' is being used as a structure/union member as well as being a label, tag or ordinary identifier.
782 This identifier, '%s', will still be in scope when the same identifier is declared later in a different namespace.
783 A subsequent declaration of '%s' means that it is being used both as a structure/union member and also as a label, tag or ordinary identifier.
790 This translation unit makes use of floating types.
805 [S] Invalid identifier encountered in macro parameter list.
806 [S] Comma missing in macro parameter list.
807 [S] Identifier missing in macro parameter list.
808 [Q] '#include "%s"' causes itself to be included recursively. QA C terminates execution after 8 occurrences.
815 [L] #include <...> file name does not conform strictly to ISO:C90.
816 [L] #include "..." file name does not conform strictly to ISO:C90.
817 [S] Closing quote or bracket '>' missing from include filename.
818 [Q] Cannot find '%s' - Perhaps the appropriate search path was not given ?
819 [Q] Cannot open '%s' - Perhaps the appropriate search path was not given ?
820 [S] '#include' requires a header name.
822 [Q] Cannot find forceinclude file '%s'.
823 [S] Unexpected '#else' or '#elif' directive follows '#else'.
824 [S] Unexpected '#else' or '#elif' directive found outside a '#if' block.
825 [S] Unexpected '#endif' found outside a '#if' block.
826 [S] Unexpected ':' found without a preceding '?' in a '#if' or '#elif' expression.
827 [S] Missing ':' after '?' in a '#if' or '#elif' expression.
829 [S] Could not find '#endif' preprocessing directive to close '#if...'.
832 Macro substitution in #include preprocessing directive.
833 [S] '#ifdef' or '#ifndef' must be followed by an identifier.
838 File '%1s' has already been included directly from within file '%2s'.
839 File '%1s' has already been included indirectly from within file '%2s'.
842 Using #define or #undef inside a function.
843 / character used in an include file name.
846 [S] '#define' must be followed by an identifier.
847 [S] '#undef' must be followed by an identifier.
849 [S] Preprocessing directive appears in the middle of a line.
855 Preprocessing results in a sequence of tokens that has the form of a preprocessing directive.
861 This #include <%s> directive is redundant.
862 This #include "%s" directive is redundant.
863 [S] '#line' encountered without a following line number.

868	An absolute path has been specified in a #include statement.
869	[Q] '#error' directive: %s Analysis terminated due to '#error' directive.
878	Using wide character or string literals.
879	[S] Illegal operator in '#if' or '#elif' expression.
880	Using # and ## operators in the same macro definition.
881	Using multiple ## operators in the same macro definition.
882	The forceinclude file '%s' is redundant in this translation unit.
884	Using multiple # operators in the same macro definition.
886	[S] Missing or invalid expression in '#if' or '#elif' directive.
889	#undef refers to a macro that has previously been #undef'd.
890	#ifdef refers to a macro that has been #undef'd in a previous #if block.
891	#ifdef refers to a macro that has been #undef'd previously.
896	[S] Missing operand in '#if' or '#elif' expression.
897	[S] Missing operator in '#if' or '#elif' expression.
898	[S] Unexpected ')' or ':' in '#if' or '#elif' expression.
901	Source skipped to this point after error in top level declaration.
902	Source skipped to this point after error in parameter declaration.
903	Source skipped to this point after error in member declaration.
904	[S] Invalid designator.
907	[S] Unexpected token.
911	Inserted '%s' token in an attempt to continue parsing.
913	Source skipped to this point after error in 'case' label.
917	[S] Unable to recover from syntax errors in this source file.
918	[Q] Source too complex. Please report this problem to Programming Research Technical Support.
920	Source skipped to this point after error in expression.
925	[S] Unexpected end of file.
926	[S] Expected:%s.
927	[S] Unexpected ')'.
928	[S] Missing comma ',' between enumerators.
970	[Q] Unable to create preprocessed file.
974	[I] Cast of integer constant expression to a signed integer type which cannot represent the value.
977	Cast of a constant negative value to an unsigned type.
980	Cast of constant negative value which assumes a two's complement representation of signed values.
1004	[S] '#endasm' found without matching '#asm' preprocessing directive.
1005	[S] End-of-file encountered when expecting '#%s' preprocessing directive.
1009	[S] Unrecognizable expression in '#if' preprocessing directive.
1010	[S] '#%s' is not a valid form of '#if' syntax .
1013	[S] Use of a C++ reference type ('type &').
1100	The label '%s' is also declared as a typedef.
1250	Unsuffixed integer constant causes implicit conversion of other operand.
1251	Suffixed integer constant causes implicit conversion of other operand.
1252	Suffixed integer constant implicitly converted to different integer type.
1253	Unsuffixed integer constant implicitly converted to different integer type.
1254	Suffix is not consistent with a type of unsigned long.
1255	Unsuffixed integer constant is not of type int.
1256	An integer constant suffixed with L is being converted to type signed or unsigned long long on assignment.
1258	Suffixed integer constant cast to a different integral type.
1259	Unsuffixed integer constant cast to a different integral type.
1260	Integer constant implicitly converted to a floating type.
1261	Suffixed floating constant implicitly converted to different floating type.
1262	Unsuffixed floating constant implicitly converted to different floating type.
1263	Floating constant causes implicit conversion of other (integral) operand.
1267	Suffixed floating constant cast to another type.
1268	Unsuffixed floating constant cast to another type.
1269	Floating constant cast to integral type.
1271	Using a non-int expression to define an enum constant.
1272	Redundant leading zeroes on a numeric constant.
1274	Unsuffixed floating constant causes implicit conversion of other (floating) operand.
1275	Suffixed floating constant causes implicit conversion of other (floating) operand.

1276 An integer constant is being converted to floating type on assignment.
1277 Hex constant does not include a "U" suffix.
1278 Decimal constant includes a suffix.
1279 Hex constant includes an "L" suffix.
1290 An integer constant of 'essentially signed' type is being converted to unsigned type on assignment.
1300 '%s' is a keyword in C++.
1301 '%s' is a keyword in some C++ implementations.
1302 '%s()' must be declared before use in C++.
1303 An empty parameter list in a function type has a different meaning in C++.
1304 Old style definition of function '%s()' is not portable to C++.
1305 The global object '%s' declared 'const' has external linkage in C but internal linkage in C++.
1306 Multiple tentative definitions of '%s'. This is not allowed in C++.
1307 Unnamed 'struct' and 'union' types may cause portability problems when moving to C++.
1308 The tag '%s' would clash with an existing typedef in C++.
1309 The typedef '%s' would clash with an existing 'struct' / 'union' / 'enum' tag in C++.
1310 '%s' is used as a tag and a typedef for the same 'struct' / 'union' / 'enum'.
1311 'void *' and 'const T *' pointers used as operands to an equality or conditional operator.
1312 The array being initialized is not large enough to hold a terminating null byte for the string initializer.
1313 Executing 'goto %s' will cause local initialization to be skipped.
1314 The tag '%s' is defined within another 'struct' / 'union'.
1315 The 'static' qualifier has been used in the declaration of a tag.
1317 Value of constant expression is not in the enum type to which it is being converted.
1318 Object of enum type is being modified with a compound assignment operator.
1319 Object of enum type is being modified with an increment or decrement operator.
1322 In C, sizeof('c') == sizeof(int), but in C++, sizeof('c') == sizeof(char) == 1.
1323 The sizeof operator has been applied to an enum constant.
1324 Function 'main' cannot be called or have its address taken in C++.
1325 '%s()' is defined with a non-void return type but contains 'return;'. This is not allowed in C++.
1327 The macro __STDC__ has been used. It may not be present in a C++ environment.
1328 Tag type defined within a function declaration.
1329 The global object '%s' declared 'const' has been defined without an explicit initializer. This is not allowed in C++.
1334 The parameter identifiers in the prototypes of these functions/function pointers are different.
1400 Enum object is being compared with an enum constant of a different enum type using an equality operator.
1401 An enum constant is being passed as argument to a function parameter of a different enum type.
1402 An enum constant is being assigned to an object of a different enum type.
1403 An enum constant is being returned from a function with a different enum return type.
1404 Enum object is being compared with an enum constant of a different enum type using a relational operator.
1410 Enum object is being compared with a constant, non-enum expression using an equality operator.
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.
1414 Enum object is being compared with a constant, non-enum expression using a relational operator.
1420 Enum object is being compared with an object of a different enum type using an equality operator.
1421 An enum object is being passed as argument to a function parameter of a different enum type.
1422 An enum object is being assigned to an object of a different enum type.
1423 An enum object is being returned from a function with a different enum return type.
1424 Enum object is being compared with an object of a different enum type using a relational operator.
1431 An enum object is being passed as argument to a function parameter of non-enum type.
1432 An enum object is being assigned to an object of non-enum type.
1433 An enum object is being returned from a function with a non-enum return type.
1434 This enum constant is not representable in a 16 bit integer type.
1440 Enum object is being compared with a non-constant, non-enum expression using an equality operator.
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.
1444 Enum object is being compared with a non-constant, non-enum expression using a relational operator.
1461 Value of constant expression is not in the enum type to which it is being converted, but is bitwise OR of constants in the enum type.
1470 Numeric constant used as 'case' label with 'switch' expression of enum type.

1472	Enum constant used as 'case' label with 'switch' expression of different enum type.
1473	The 2nd and 3rd operands of this conditional operator (? :) are of different enum types.
1474	Object of enum type is being modified with a bitwise compound assignment operator.
1475	Range of possible enum values suggests this test is always true.
1476	Range of possible enum values suggests this test is always false.
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.
1479	Object of enum type is being modified with an arithmetic compound assignment operator.
1480	Objects or constants of different enum types are operands of a bitwise operator.
1481	Object of enum type is being modified with an increment or decrement operator.
1482	Non-constant expression cast to enum type.
1483	Enum object or constant passed as argument to function declared in K&R style.
1484	Constant expression cast to enum type.
1500	The object '%1s' is declared but is not used within this project.
1501	The function '%1s' is declared but is not used within this project.
1502	The object '%1s' is defined but is not used within this project.
1512	Identifier '%1s' with external linkage has separate declarations in multiple translation units.
1515	Different files are being used which have the same filename '%1s'.
1516	Different files are being used which have similar filenames '%1s', but with different case.
1517	Different files are being used which have the same filename '%1s', and one includes the other.
1529	Object with no linkage has same identifier as another object with static storage duration but no linkage.
1531	The object '%1s' is referenced in only one translation unit - but not the one in which it is defined.
1532	The function '%1s' is only referenced in one translation unit - but not the one in which it is defined.
1533	The object '%1s' is only referenced by function '%2s'.
1540	Unable to open file '%1s'.
1569	Issue for message '%1s' found here (even if not visible).
1570	Called from here.
1571	Occurs on %1s iteration.
1572	Return statement here.
1573	Last unreachable statement.
1574	Viable path.
1575	Variable '%1s' previously seen here. (Specimen value: '%2s').
1576	Label statement here.
1577	Next seen here.
1578	Found in directory: '%1s'.
1579	Referenced here.
1580	Function '%1s' calls '%2s' here.
1581	%1s %2s here.
1582	Previously seen here.
1585	Next set here.
1586	Previously set here.
1590	Offending operand.
1592	'%1s' declared in %2s '%3s'.
1593	'%1s' defined here.
1594	'%1s' declared here.
1599	Included from here.
1690	Null preprocessing directive used.
1691	Null preprocessing directive used in an excluded section of code.
2000	No 'else' clause exists for this 'if' statement.
2005	A 'continue' statement has been used.
2006	'%s()' has more than one 'return' path.
2007	'auto' does not add information to a declaration, and is best avoided.
2010	The function '%s()' must not be called.
2011	The 'register' storage class specifier has been used.
2015	A statement 'label' has been used.
2016	This 'switch' statement 'default' clause is empty.
2017	Comment spans more than one line.
2018	This 'switch' 'default' label is probably unreachable.

2021	This tentative definition is interpreted as a declaration. Is this intended ?
2022	A tentative definition is being used. Is it appropriate to include an explicit initializer ?
2100	Integral promotion : unsigned char promoted to signed int.
2101	Integral promotion : unsigned short promoted to signed int.
2102	Integral promotion : unsigned char promoted to unsigned int.
2103	Integral promotion : unsigned short promoted to unsigned int.
2104	Integral promotion : signed char promoted to signed int.
2105	Integral promotion : signed short promoted to signed int.
2106	Integral promotion : plain char promoted to signed int.
2107	Integral promotion : plain char promoted to unsigned int.
2109	Integral promotion : _Bool promoted to signed int.
2110	Default argument promotion : unsigned char promoted to signed int.
2111	Default argument promotion : unsigned short promoted to signed int.
2112	Default argument promotion : unsigned char promoted to unsigned int.
2113	Default argument promotion : unsigned short promoted to unsigned int.
2114	Default argument promotion : signed char promoted to signed int.
2115	Default argument promotion : signed short promoted to signed int.
2116	Default argument promotion : plain char promoted to signed int.
2117	Default argument promotion : plain char promoted to unsigned int.
2118	Default argument promotion : float promoted to double.
2119	Default argument promotion : _Bool promoted to signed int.
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.
2200	Indentation of this line is to the left of the line above.
2201	This indentation is not consistent with previous indentation in this file.
2203	This closing brace is not aligned appropriately with the matching opening brace.
2204	'%s' is not aligned to match its controlling 'switch' statement.
2205	More than one declaration or statement on the same line.
2207	This brace style is not consistent with 'K&R' style.
2208	This brace style is not consistent with 'indented' style.
2209	This brace style is not consistent with 'exdented' style.
2210	Tab character encountered in this line.
2211	'%s' is not aligned with the previously declared identifier.
2213	Matching braces appear on the same line - proper indentation would be preferred.
2215	This indentation is not consistent with configured depth.
2216	More than one structure or union member declared on the same line.
2217	Line length exceeds %s characters.
2465	This 'for' loop will only be executed once.
2466	The value of this controlling expression is always 'false'. The contained code is unreachable.
2468	Loop control variable, %s, is not modified inside loop but has file scope.
2470	Taking address of loop control variable, %s.
2740	This loop controlling expression is a constant expression and its value is 'true'.
2741	This 'if' controlling expression is a constant expression and its value is 'true'.
2743	This 'do - while' loop controlling expression is a constant expression and its value is 'false'. The loop will only be executed once.
2750	Internal dataflow problem. Dataflow analysis continues with the next function. Please inform Programming Research.
2751	This function is too complex. Dataflow analysis continues with the next function.
2752	This '%1s' results in the function being too complex. Dataflow analysis continues with the next function.
2753	As a result of error message '%s', dataflow analysis of the remainder of this function is not possible.
2754	As a result of error message '%s', dataflow analysis of the remainder of this translation unit is not possible.
2755	Analysis time of function '%1s' has exceeded the configured maximum: '%2sms'. Dataflow analysis continues with the next function.
2756	Could not expand function call to '%1s' with maximum '-po df::inter' value.
2757	Could not analyze function '%1s'.
2773	Suspicious: These pointers address different objects.

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.
2814 Possible: Dereference of NULL pointer.
2823 Suspicious: Arithmetic operation on NULL pointer.
2824 Possible: Arithmetic operation on NULL pointer.
2833 Suspicious: Division by zero.
2834 Possible: Division by zero.
2843 Suspicious: Dereference of an invalid pointer value.
2848 Suspicious: Maximum number of characters to be written is larger than the target buffer size.
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.
2870 Infinite loop construct with constant control expression.
2881 The code in this 'default' clause is unreachable.
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.
2905 Constant: Positive integer value truncated by cast to a smaller unsigned type.
2906 Definite: Positive integer value truncated by cast to a smaller unsigned type.
2907 Apparent: Positive integer value truncated by cast to a smaller unsigned type.
2908 Suspicious: Positive integer value truncated by cast to a smaller unsigned type.
2911 Definite: Wraparound in unsigned arithmetic operation.
2912 Apparent: Wraparound in unsigned arithmetic operation.
2913 Suspicious: Wraparound in unsigned arithmetic operation.
2920 Constant: Left shift operation on expression of unsigned type results in loss of high order bits.
2921 Definite: Left shift operation on expression of unsigned type results in loss of high order bits.
2922 Apparent: Left shift operation on expression of unsigned type results in loss of high order bits.
2923 Suspicious: Left shift operation on expression of unsigned type results in loss of high order bits.
2933 Suspicious: Computing an invalid pointer value.
2940 Constant: Result of implicit conversion is only representable in a two's complement implementation.
2941 Definite: Result of implicit conversion is only representable in a two's complement implementation.
2942 Apparent: Result of implicit conversion is only representable in a two's complement implementation.
2943 Suspicious: Result of implicit conversion is only representable in a two's complement implementation.
2945 Constant: Result of cast is only representable in a two's complement implementation.
2946 Definite: Result of cast is only representable in a two's complement implementation.
2947 Apparent: Result of cast is only representable in a two's complement implementation.
2948 Suspicious: Result of cast is only representable in a two's complement implementation.
2950 Constant: Negative value used in array subscript or pointer arithmetic operation.
2951 Definite: Negative value used in array subscript or pointer arithmetic operation.
2952 Apparent: Negative value used in array subscript or pointer arithmetic operation.
2953 Suspicious: Negative value used in array subscript or pointer arithmetic operation.
2963 Suspicious: Using value of uninitialized automatic object '%s'.
2973 Suspicious: Passing address of uninitialized object '%s' to a function parameter declared as a pointer to const.
3103 Result of signed division or remainder operation may be implementation defined.
3104 [S] #pragma '%s' has invalid arguments and has been ignored.
3105 [S] A #pragma block has not been closed with a #pragma '%s'.
3106 This preprocessing directive has been ignored because it lies within a #pragma block. #pragma '%s' expected.
3107 [S] A #pragma '%s' has been found without a matching #pragma block start directive.
3109 Null statement follows other code on the same line.
3111 Redundant comma at end of braced initializer will be ignored.
3117 Expected 'on' or 'off' after '#pragma' flag.
3118 Using the #line preprocessing directive.
3120 Hard-coded 'magic' integer constant, '%s'.
3121 Hard-coded 'magic' floating constant, '%s'.
3122 Hard-coded 'magic' string literal, %s.
3123 Hard coded 'magic' character constant, %s.
3131 Hard coded 'magic' number, '%s', used to define the size of a bit-field.

3132 Hard coded 'magic' number, '%s', used to define the size of an array.
3138 Null statement is located close to other code or comments.
3139 Null statement is obscured by code or comment on the same line.
3140 Null statement occurs on a line by itself.
3141 Null statement does not occur on a line by itself.
3195 The function parameter '%s' is always modified before use.
3196 The variable '%s' is never set.
3197 This initialization is redundant. The value of '%s' is never used before being modified.
3198 This assignment is redundant. The value of '%s' is never used before being modified.
3199 This assignment is redundant. The value of '%s' is never subsequently used.
3201 This statement is unreachable.
3203 The variable '%s' is set but never used.
3204 The variable '%s' is only set once and so it could be declared with the 'const' qualifier.
3205 The identifier '%s' is not used and could be removed.
3207 File scope static, '%s', is not used, and could be removed.
3208 '%s()' returns a value which is sometimes ignored.
3209 '%s()' returns a value which is always ignored.
3210 The global identifier '%s' is declared here but is not used in this translation unit.
3211 The global identifier '%s' is defined here but is not used in this translation unit.
3212 This cast is redundant.
3216 Address of local static object exported to a pointer with linkage or wider scope.
3220 Identifier declared at a nested level of block scope.
3223 Object with static storage duration declared at block scope.
3227 The parameter '%s' is never modified and so it could be declared with the 'const' qualifier.
3228 Storage class specifier not positioned at the beginning of declaration.
3229 File scope static, '%s', is written but never used.
3231 Address of local static object exported using a function parameter.
3232 File scope static, '%s', is never modified. It could be declared const.
3233 File scope static, '%s', is not explicitly initialized but its value is used.
3290 Truncation of positive constant integer value during cast to a smaller unsigned type.
3296 Definite truncation of positive integer value during implicit conversion to a smaller unsigned type.
3297 Apparent truncation of positive integer value during implicit conversion to a smaller unsigned type.
3301 Truncation of bits in an unsigned left shift operation on a constant value.
3302 Wraparound past zero in unsigned subtraction of constant operands.
3303 Wraparound past zero in unsigned addition of constant operands.
3304 Wraparound past zero in unsigned multiplication of constant operands.
3306 Truncation of positive constant integer value during implicit conversion to a smaller unsigned type.
3308 'static %s()' has been declared but no definition has been provided. This is redundant.
3309 The value of the controlling expression in this 'switch' statement is constant.
3313 No definition has been found for structure/union tag '%s'.
3314 This controlling expression is an assignment.
3316 An unsigned value can never be less than zero - this test is always false.
3321 [U] The variable '%s' is definitely unset at this point.
3322 Operand of a logical ! operator is a constant expression which is not a 'Boolean' value.
3323 This controlling expression has a constant 'true' value.
3324 An unsigned value is always greater than or equal to zero - this test is always true.
3325 This 'while' or 'for' loop controlling expression has a constant 'false' value.
3328 An unsigned value is being compared with a negative constant - this is dangerous.
3329 This 'if' controlling expression has a constant 'false' value.
3330 '%s()' has been called with a variable number of arguments.
3333 A 'break' statement has been used in the middle of a 'switch' 'case'/'default' clause.
3337 The array '%s[]' is defined with a single element because no size has been specified.
3341 Comparing floating point expressions for equality (with '==' or '!=').
3343 Logical NOT being performed on one operand of a comparison.
3345 Statement contains more than one access to objects that are volatile.
3346 The controlling expression in this 'if' statement has a constant 'true' value.
3347 [U] The variable '%s' is apparently unset at this point.
3348 Definite use of unset pointer as an argument to a function which expects a read-only pointer.
3349 Apparent use of unset pointer as an argument to a function which expects a read-only pointer.

3352	This 'switch' statement contains only two execution paths.
3353	The variable '%s' is possibly unset at this point.
3354	Possible use of unset pointer as an argument to a function which expects a read-only pointer.
3355	The result of this logical operation is always 'true'.
3356	The result of this logical operation is always 'false'.
3357	The value of this loop controlling expression is always 'true'.
3358	The value of this 'if' controlling expression is always 'true'.
3359	The value of this controlling expression is always 'false'.
3360	The value of this 'do - while' controlling expression is always 'false'. The loop will only be executed once.
3361	Using a 'do-while-zero' construct. The loop will only be executed once.
3371	Definite truncation of bits in an unsigned left shift operation.
3372	Definite wraparound past zero in an unsigned arithmetic operation.
3377	Operand of a logical && or || operator is a constant expression which is not a 'Boolean' value.
3381	Apparent truncation of bits in an unsigned left shift operation.
3382	Apparent wraparound past zero in an unsigned arithmetic operation.
3390	Expression is equivalent to: "%s"
3393	Extra parentheses recommended. An arithmetic operation (* / + -) is the operand of a different operator with the same precedence.
3398	Extra parentheses recommended. A function call, array subscript, or member operation is the operand of a logical && or ||.
3399	Extra parentheses recommended. A unary operation is the operand of a logical && or ||.
3400	Extra parentheses recommended. A binary operation is the operand of a logical && or ||.
3401	Possible precedence confusion: extra parentheses are recommended here.
3403	This sequence of operators is difficult to read without intervening spaces.
3404	Statement contains a redundant * operator at top level. *p++ means *(p++) not (*p)++.
3405	Index[Array] is equivalent to Array[Index] but more confusing.
3407	String literal compared using a relational or equality operator.
3409	The replacement list of function-like macro '%s' is not enclosed in ().
3411	Macro defined with unbalanced brackets, parentheses or braces.
3412	Macro defines an unrecognized code-fragment.
3413	Macro definition could be replaced by a typedef.
3414	Macro defines a storage-class or function specifier keyword.
3416	Logical operation performed on expression with possible side effects.
3419	Initialization expression of 'for' statement has no side effects.
3420	Increment expression of 'for' statement has no side effects.
3422	Statement contains a redundant operator at top level.
3423	Statement contains a redundant cast at top level.
3424	Statement contains a redundant & or | at top level.
3428	Macro defines a type qualifier keyword.
3429	A function-like macro is being defined.
3430	Macro argument expression may require parentheses.
3431	Macro defines an operator, a punctuator or a control statement keyword.
3435	Parameter '%s' occurs more than once in the replacement list of this macro.
3436	Macro definition hides previously declared identifier.
3441	Function call argument is an expression with possible side effects.
3442	Operator other than & (address-of) or = (assignment) applied to a volatile object.
3443	Macro '%s' has a replacement list which appears recursive.
3446	The 2nd or 3rd operand of this conditional operator is an expression with possible side effects.
3450	Function '%s', with internal linkage, is being defined without a previous declaration.
3452	The replacement list of object-like macro '%s' is not enclosed in ().
3454	Macro argument contains an increment, decrement or assignment operator.
3455	Macro argument contains a function call.
3456	Parameter '%s' will be evaluated more than once when this macro is used.
3457	Macro defines a braced initializer.
3458	Macro defines a braced code statement block.
3459	Macro defines a 'do-while-zero' construct.
3460	Macro defines a type specifier keyword.
3461	Macro defines a storage-class specifier/type qualifier sequence.
3470	The operand of 'sizeof' is not an expression which designates either an object or a type.

3480	Object/function '%s', with internal linkage, has been defined in a header file.
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.
3501	Absence of space between assignment operator and following unary operator may cause confusion.
3600	Using the unary '+' operator.
3602	Using one of the keywords 'signed', 'const', 'volatile' or 'enum'.
3603	Using 'extern' in the definition of a global variable.
3604	Using an initializer when defining an automatic object of struct, union or array type.
3605	Type of 'switch' controlling expression cannot be represented in type 'int'.
3606	This unsuffixed decimal constant has type 'long', but had type 'unsigned int' in K&R C.
3607	Parameter '%s' declared with function type in a K&R style function definition.
3608	Using the '#elif' preprocessing directive.
3609	Using the alert escape sequence '\\a'.
3610	Hexadecimal escape sequence used.
3611	Non-portable comparison of plain 'char' with negative constant.
3612	Nonportable comparison of plain 'char' with zero.
3613	Some pre-ISO compilers would treat this 8 or 9 as an octal digit.
3614	Macro '%s' is being defined more than once without using #undef to remove the previous definition.
3615	Using 'entry' as an identifier.
3616	Character constants may have different values in preprocessor arithmetic than in actual code.
3617	Assignment of a struct/union by value.
3618	Whitespace used after '#' at the start of this preprocessing directive.
3619	Whitespace used before '#' at the start of this preprocessing directive.
3620	'register' may be illegal on array and 'struct' / 'union' types in some compilers.
3621	A bit-field is being defined as a member of a struct or union.
3623	Passing a struct/union by value as a function argument.
3624	Function returns a struct/union by value.
3625	Type 'char' has been used in the declaration of an object or a function.
3626	Double-quote character in a character constant is not preceded by a backslash character.
3627	Single-quote character in a string literal is not preceded by a backslash character.
3629	Union contains member of floating type.
3631	Type 'char' has been used in a cast.
3632	Type 'char' has been used in the declaration of a typedef.
3633	Type 'char' has been used in the operand of the sizeof operator.
3635	Function identifier used as a pointer without a preceding & operator.
3650	Typedef defines an array type of unknown size.
3651	Using a typedef for an array of unknown size can lead to unexpected results.
3659	Unnamed zero-width bit-field declared with a signed type.
3661	Plain int bit-field compared with zero.
3662	Plain int bit-field compared with negative constant.
3663	Unnamed bit-field defined with non-zero width.
3671	Function called via pointer to function.
3672	Using non-const pointer to function.
3674	Array size defined implicitly by the number of initializers.
3680	[U] Access outside bounds of array using a constant array subscript.
3681	Use of a constant negative value in array subscript operation or pointer arithmetic.
3685	[U] Definite access outside bounds of array.
3686	Definite use of a negative value in array subscript operation or pointer arithmetic.
3689	[U] Apparent access outside bounds of array.
3690	Apparent use of a negative value in array subscript operation or pointer arithmetic.
3700	Implicit conversion: char to signed char.
3701	Implicit conversion: char to unsigned char.
3702	Implicit conversion: char to short.
3703	Implicit conversion: char to unsigned short.
3704	Implicit conversion: char to int.
3705	Implicit conversion: char to unsigned int.
3706	Implicit conversion: char to long.
3707	Implicit conversion: char to unsigned long.

3708 Implicit conversion: char to float.
3709 Implicit conversion: char to double.
3710 Implicit conversion: char to long double.
3711 Implicit conversion: unsigned char to char.
3712 Implicit conversion: unsigned char to signed char.
3713 Implicit conversion: unsigned char to short.
3715 Implicit conversion: unsigned char to int.
3717 Implicit conversion: unsigned char to long.
3719 Implicit conversion: unsigned char to float.
3720 Implicit conversion: unsigned char to double.
3721 Implicit conversion: unsigned char to long double.
3722 Implicit conversion: signed char to char.
3723 Implicit conversion: signed char to unsigned char.
3725 Implicit conversion: signed char to unsigned short.
3727 Implicit conversion: signed char to unsigned int.
3729 Implicit conversion: signed char to unsigned long.
3730 Implicit conversion: signed char to float.
3731 Implicit conversion: signed char to double.
3732 Implicit conversion: signed char to long double.
3733 Implicit conversion: short to char.
3734 Implicit conversion: short to signed char.
3735 Implicit conversion: short to unsigned char.
3736 Implicit conversion: short to unsigned short.
3738 Implicit conversion: short to unsigned int.
3740 Implicit conversion: short to unsigned long.
3741 Implicit conversion: short to float.
3742 Implicit conversion: short to double.
3743 Implicit conversion: short to long double.
3744 Implicit conversion: unsigned short to char.
3745 Implicit conversion: unsigned short to signed char.
3746 Implicit conversion: unsigned short to unsigned char.
3747 Implicit conversion: unsigned short to short.
3748 Implicit conversion: unsigned short to int.
3750 Implicit conversion: unsigned short to long.
3752 Implicit conversion: unsigned short to float.
3753 Implicit conversion: unsigned short to double.
3754 Implicit conversion: unsigned short to long double.
3755 Implicit conversion: int to char.
3756 Implicit conversion: int to signed char.
3757 Implicit conversion: int to unsigned char.
3758 Implicit conversion: int to short.
3759 Implicit conversion: int to unsigned short.
3760 Implicit conversion: int to unsigned int.
3762 Implicit conversion: int to unsigned long.
3763 Implicit conversion: int to float.
3764 Implicit conversion: int to double.
3765 Implicit conversion: int to long double.
3766 Implicit conversion: unsigned int to char.
3767 Implicit conversion: unsigned int to signed char.
3768 Implicit conversion: unsigned int to unsigned char.
3769 Implicit conversion: unsigned int to short.
3770 Implicit conversion: unsigned int to unsigned short.
3771 Implicit conversion: unsigned int to int.
3772 Implicit conversion: unsigned int to long.
3774 Implicit conversion: unsigned int to float.
3775 Implicit conversion: unsigned int to double.
3776 Implicit conversion: unsigned int to long double.
3777 Implicit conversion: long to char.
3778 Implicit conversion: long to signed char.

3779 Implicit conversion: long to unsigned char.
3780 Implicit conversion: long to short.
3781 Implicit conversion: long to unsigned short.
3782 Implicit conversion: long to int.
3783 Implicit conversion: long to unsigned int.
3784 Implicit conversion: long to unsigned long.
3785 Implicit conversion: long to float.
3786 Implicit conversion: long to double.
3787 Implicit conversion: long to long double.
3788 Implicit conversion: unsigned long to char.
3789 Implicit conversion: unsigned long to signed char.
3790 Implicit conversion: unsigned long to unsigned char.
3791 Implicit conversion: unsigned long to short.
3792 Implicit conversion: unsigned long to unsigned short.
3793 Implicit conversion: unsigned long to int.
3794 Implicit conversion: unsigned long to unsigned int.
3795 Implicit conversion: unsigned long to long.
3796 Implicit conversion: unsigned long to float.
3797 Implicit conversion: unsigned long to double.
3798 Implicit conversion: unsigned long to long double.
3799 Implicit conversion: float to char.
3800 Implicit conversion: float to signed char.
3801 Implicit conversion: float to unsigned char.
3802 Implicit conversion: float to short.
3803 Implicit conversion: float to unsigned short.
3804 Implicit conversion: float to int.
3805 Implicit conversion: float to unsigned int.
3806 Implicit conversion: float to long.
3807 Implicit conversion: float to unsigned long.
3810 Implicit conversion: double to char.
3811 Implicit conversion: double to signed char.
3812 Implicit conversion: double to unsigned char.
3813 Implicit conversion: double to short.
3814 Implicit conversion: double to unsigned short.
3815 Implicit conversion: double to int.
3816 Implicit conversion: double to unsigned int.
3817 Implicit conversion: double to long.
3818 Implicit conversion: double to unsigned long.
3819 Implicit conversion: double to float.
3821 Implicit conversion: long double to char.
3822 Implicit conversion: long double to signed char.
3823 Implicit conversion: long double to unsigned char.
3824 Implicit conversion: long double to short.
3825 Implicit conversion: long double to unsigned short .
3826 Implicit conversion: long double to int.
3827 Implicit conversion: long double to unsigned int.
3828 Implicit conversion: long double to long.
3829 Implicit conversion: long double to unsigned long.
3830 Implicit conversion: long double to float.
3831 Implicit conversion: long double to double.
3832 Implicit conversion: char to long long.
3833 Implicit conversion: char to unsigned long long.
3834 Implicit conversion: unsigned char to long long.
3837 Implicit conversion: signed char to unsigned long long.
3839 Implicit conversion: short to unsigned long long.
3840 Implicit conversion: unsigned short to long long.
3843 Implicit conversion: int to unsigned long long.
3844 Implicit conversion: unsigned int to long long.
3847 Implicit conversion: long to unsigned long long.

3848 Implicit conversion: unsigned long to long long.
3850 Implicit conversion: long long to char.
3851 Implicit conversion: long long to signed char.
3852 Implicit conversion: long long to unsigned char.
3853 Implicit conversion: long long to short.
3854 Implicit conversion: long long to unsigned short.
3855 Implicit conversion: long long to int.
3856 Implicit conversion: long long to unsigned int.
3857 Implicit conversion: long long to long
3858 Implicit conversion: long long to unsigned long.
3859 Implicit conversion: long long to unsigned long long.
3860 Implicit conversion: long long to float.
3861 Implicit conversion: long long to double.
3862 Implicit conversion: long long to long double.
3863 Implicit conversion: unsigned long long to char.
3864 Implicit conversion: unsigned long long to signed char.
3865 Implicit conversion: unsigned long long to unsigned char.
3866 Implicit conversion: unsigned long long to short.
3867 Implicit conversion: unsigned long long to unsigned short.
3868 Implicit conversion: unsigned long long to int.
3869 Implicit conversion: unsigned long long to unsigned int.
3870 Implicit conversion: unsigned long long to long.
3871 Implicit conversion: unsigned long long to unsigned long.
3872 Implicit conversion: unsigned long long to long long.
3873 Implicit conversion: unsigned long long to float.
3874 Implicit conversion: unsigned long long to double.
3875 Implicit conversion: unsigned long long to long double.
3876 Implicit conversion: float to long long.
3877 Implicit conversion: float to unsigned long long.
3878 Implicit conversion: double to long long.
3879 Implicit conversion: double to unsigned long long.
3880 Implicit conversion: long double to long long.
3881 Implicit conversion: long double to unsigned long long.
3892 The result of this cast is implicitly converted to another type.
3900 char value returned from signed char %s().
3901 char value returned from unsigned char %s().
3902 char value returned from short %s().
3903 char value returned from unsigned short %s().
3904 char value returned from int %s().
3905 char value returned from unsigned int %s().
3906 char value returned from long %s().
3907 char value returned from unsigned long %s().
3908 char value returned from float %s().
3909 char value returned from double %s().
3910 char value returned from long double %s().
3911 unsigned char value returned from char %s().
3912 unsigned char value returned from signed char %s().
3913 unsigned char value returned from short %s().
3915 unsigned char value returned from int %s().
3917 unsigned char value returned from long %s().
3919 unsigned char value returned from float %s().
3920 unsigned char value returned from double %s().
3921 unsigned char value returned from long double %s().
3922 signed char value returned from char %s().
3923 signed char value returned from unsigned char %s().
3925 signed char value returned from unsigned short %s().
3927 signed char value returned from unsigned int %s().
3929 signed char value returned from unsigned long %s().
3930 signed char value returned from float %s().

3931 signed char value returned from double %s().
3932 signed char value returned from long double %s().
3933 short value returned from char %s().
3934 short value returned from signed char %s().
3935 short value returned from unsigned char %s().
3936 short value returned from unsigned short %s().
3938 short value returned from unsigned int %s().
3940 short value returned from unsigned long %s().
3941 short value returned from float %s().
3942 short value returned from double %s().
3943 short value returned from long double %s().
3944 unsigned short value returned from char %s().
3945 unsigned short value returned from signed char %s().
3946 unsigned short value returned from unsigned char %s().
3947 unsigned short value returned from short %s().
3948 unsigned short value returned from int %s().
3950 unsigned short value returned from long %s().
3952 unsigned short value returned from float %s().
3953 unsigned short value returned from double %s().
3954 unsigned short value returned from long double %s().
3955 int value returned from char %s().
3956 int value returned from signed char %s().
3957 int value returned from unsigned char %s().
3958 int value returned from short %s().
3959 int value returned from unsigned short %s().
3960 int value returned from unsigned int %s().
3962 int value returned from unsigned long %s().
3963 int value returned from float %s().
3964 int value returned from double %s().
3965 int value returned from long double %s().
3966 unsigned int value returned from char %s().
3967 unsigned int value returned from signed char %s().
3968 unsigned int value returned from unsigned char %s().
3969 unsigned int value returned from short %s().
3970 unsigned int value returned from unsigned short %s().
3971 unsigned int value returned from int %s().
3972 unsigned int value returned from long %s().
3974 unsigned int value returned from float %s().
3975 unsigned int value returned from double %s().
3976 unsigned int value returned from long double %s().
3977 long value returned from char %s().
3978 long value returned from signed char %s().
3979 long value returned from unsigned char %s().
3980 long value returned from short %s().
3981 long value returned from unsigned short %s().
3982 long value returned from int %s().
3983 long value returned from unsigned int %s().
3984 long value returned from unsigned long %s().
3985 long value returned from float %s().
3986 long value returned from double %s().
3987 long value returned from long double %s().
3988 unsigned long value returned from char %s().
3989 unsigned long value returned from signed char %s().
3990 unsigned long value returned from unsigned char %s().
3991 unsigned long value returned from short %s().
3992 unsigned long value returned from unsigned short %s().
3993 unsigned long value returned from int %s().
3994 unsigned long value returned from unsigned int %s().
3995 unsigned long value returned from long %s().

3996 unsigned long value returned from float %s().
3997 unsigned long value returned from double %s().
3998 unsigned long value returned from long double %s().
3999 float value returned from char %s().
4000 float value returned from signed char %s().
4001 float value returned from unsigned char %s().
4002 float value returned from short %s().
4003 float value returned from unsigned short %s().
4004 float value returned from int %s().
4005 float value returned from unsigned int %s().
4006 float value returned from long %s().
4007 float value returned from unsigned long %s().
4010 double value returned from char %s().
4011 double value returned from signed char %s().
4012 double value returned from unsigned char %s().
4013 double value returned from short %s().
4014 double value returned from unsigned short %s().
4015 double value returned from int %s().
4016 double value returned from unsigned int %s().
4017 double value returned from long %s().
4018 double value returned from unsigned long %s().
4019 double value returned from float %s().
4021 long double value returned from char %s().
4022 long double value returned from signed char %s().
4023 long double value returned from unsigned char %s().
4024 long double value returned from short %s().
4025 long double value returned from unsigned short %s().
4026 long double value returned from int %s().
4027 long double value returned from unsigned int %s().
4028 long double value returned from long %s().
4029 long double value returned from unsigned long %s().
4030 long double value returned from float %s().
4031 long double value returned from double %s().
4032 char value returned from long long %s().
4033 char value returned from unsigned long long %s().
4034 unsigned char value returned from long long %s().
4037 signed char value returned from unsigned long long %s().
4039 short value returned from unsigned long long %s().
4040 unsigned short value returned from long long %s().
4043 int value returned from unsigned long long %s().
4044 unsigned int value returned from long long %s().
4047 long value returned from unsigned long long %s().
4048 unsigned long value returned from long long %s().
4050 long long value returned from char %s().
4051 long long value returned from signed char %s().
4052 long long value returned from unsigned char %s().
4053 long long value returned from short %s().
4054 long long value returned from unsigned short %s().
4055 long long value returned from int %s().
4056 long long value returned from unsigned int %s().
4057 long long value returned from long %s().
4058 long long value returned from unsigned long %s().
4059 long long value returned from unsigned long long %s().
4060 long long value returned from float %s().
4061 long long value returned from double %s().
4062 long long value returned from long double %s().
4063 unsigned long long value returned from char %s().
4064 unsigned long long value returned from signed char %s().
4065 unsigned long long value returned from unsigned char %s().

4066	unsigned long long value returned from short %s().
4067	unsigned long long value returned from unsigned short %s().
4068	unsigned long long value returned from int %s().
4069	unsigned long long value returned from unsigned int %s().
4070	unsigned long long value returned from long %s().
4071	unsigned long long value returned from unsigned long %s().
4072	unsigned long long value returned from long long %s().
4073	unsigned long long value returned from float %s().
4074	unsigned long long value returned from double %s().
4075	unsigned long long value returned from long double %s().
4076	float value returned from long long %s().
4077	float value returned from unsigned long long %s().
4078	double value returned from long long %s().
4079	double value returned from unsigned long long %s().
4080	long double value returned from long long %s().
4081	long double value returned from unsigned long long %s().
4100	Illegal user error number %1s
4101	Both operands of & operator are 'Boolean' expressions.
4102	Both operands of | operator are 'Boolean' expressions.
4103	Both operands of arithmetic or bitwise operator are 'Boolean' expressions.
4104	Left hand operand of arithmetic or bitwise operator is a 'Boolean' expression.
4105	Right hand operand of arithmetic or bitwise operator is a 'Boolean' expression.
4106	Both operands of && operator are arithmetic or bitwise expressions.
4107	Both operands of || operator are arithmetic or bitwise expressions.
4108	Left hand operand of logical operator is an arithmetic or bitwise expression.
4109	Right hand operand of logical operator is an arithmetic or bitwise expression.
4110	Operand of ! operator is an arithmetic or bitwise expression.
4111	Right hand operand of relational operator is a 'Boolean' expression.
4112	Left hand operand of relational operator is a 'Boolean' expression.
4113	Both operands of relational operator are 'Boolean' expressions.
4114	Operand of ~ operator is a 'Boolean' expression.
4115	Operand of logical && or || operator is not an 'essentially Boolean' expression.
4116	Operand of logical ! operator is not an 'essentially Boolean' expression.
4117	Result of integer division operation implicitly converted to a floating type.
4118	Result of integer division operation cast to a floating type.
4119	Result of floating point operation cast to an integral type.
4120	Implicit conversion: complex expression of integral type to wider type.
4121	Cast of complex expression of integral type to wider type.
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.
4126	Cast of complex expression of type float to type double.
4127	Cast of complex expression of type float to type long double.
4128	Cast of complex expression of type double to type long double.
4130	Bitwise operations on signed data will give implementation defined results.
4131	Left shift operation on signed operand.
4139	Address of local static object exported in function return value.
4150	The identifier '%s' may cause confusion because it contains only the letter 'l' and and the number '1'.
4151	The identifier '%s' may cause confusion because it contains only the letter 'O' and the number '0'.
4152	The identifier '%s' may cause confusion because the letter 'l' and the number '1' are adjacent.
4153	The identifier '%s' may cause confusion because the letter 'O' and the number '0' are adjacent.
4325	An expression of 'essentially enum' type (%1s) is being cast to floating type, '%2s'.
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'.
4498	An expression which is the result of a ~ or << operation has been converted to a different essential type category on assignment.
4517	An expression of 'essentially character' type (%1s) is being used as the operand of this increment/decrement operator (%2s).
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'.
4620	The macro '%1s' may also be defined as a macro in '<%2s>'.
4621	The macro '%1s' may also be defined as a typedef in '<%2s>'.
4622	The identifier '%1s' may be defined as a macro in '<%2s>'.
4623	The typedef '%1s' may also be defined in '<%2s>'.
4624	The ordinary identifier '%1s' may be defined as a typedef in '<%2s>'.
4640	The macro '%1s' could conflict in the future with the name of a macro in '<%2s>'.
4641	The identifier '%1s' could conflict in the future with the name of a macro in '<%2s>'.
4642	The macro '%1s' could conflict in the future with the name of a function in '<%2s>'.
4643	The identifier '%1s' could conflict in the future with the name of a function in '<%2s>'.
4644	The macro '%1s' could conflict in the future with the name of a typedef in '<%2s>'.
4645	The identifier '%1s' could conflict in the future with the name of a typedef in '<%2s>'.
4700	Metric value out of threshold range: %s.
4800	The identifier '%1s' does not conform to the name rule
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.