ISO/IEC PDTS 17961Editors Report

1. Add informative Annex D "Decidability of Rules"
2. For the Netherlands, changed the word "discovering" to "detecting" in section 2.
3. For the Netherlands, appended the following to the last sentence before section 2.1: ". . . and shall document its accuracy with respect to avoiding false positives and false negatives."
4. The following changes were made in response to comments received in SC 22 N4780 JISC:

JP 1	throughout the document		ge	Each rule section consists of Rule, Rationale (if any), Example(s) and Exception (if any), in this order. Rule is obviously normative, and Rationale and Example(s) are informative, but we cannot judge whether Exception is normative or informative. Some of examples seem to be related to exceptions, so we might think that exceptions are normative. However, exceptions are given after informative examples, and have appearances similar to examples, so we are inclined to consider that they are informative.	Added to the end of Section 1 Scope: Some rules in this document have exceptions. Exceptions are part of the specification of these rules and are normative.
JP 2	throughout the document		te	In the description of some of the rules, we cannot judge whether the rule refers to the dynamic behavior of programs, or is limited to static text of programs. For example, 5.10 (Converting a pointer to integer or integer to pointer) says "shall be diagnosed if the resulting pointer is incorrectly aligned", thus this rule explicitly refers to the dynamic behavior. On the other hand, 5.11 (Converting	Some of the rules refer to dynamic program behavior in cases where there was no static solution that had a sufficiently low level of false positives.

				pointer values to more strictly aligned pointer types) says "Converting a pointer value to a pointer type that is more strictly aligned than the type the value actually points to shall be diagnosed". This can be interpreted as a rule on the static relation between two types. This interpretation would not be correct, since Example 2 refers to the dynamic behavior of programs, but anyway, the description of the rule is ambiguous in some sense. RESPONSE: Some of the rules refer to dynamic program behavior in cases where there was no static solution that had a sufficiently low level of false positives.	
JP 3	Introduction		te	Explanations on completeness and soundness are given in Introduction, and the concept "quality of implementation issue" is given here. Standards often have descriptions like "This Standard does not specify the following …" in the Conformance clauses. We think that such a position is appropriate for completeness and soundness.	Before the last sentence at the End of Section 2.1 added the following sentence: This technical specification does not specify that a conforming analyser be complete or sound when diagnosing rule violations.
JP 4	Introduction	2nd paragraph of Completeness and soundness	ed	"undecideable" should be changed to "undecidable". Also for "undecideability".	Changes made.
JP 5	Introduction	Table 1	ed	The lower-right item should be changed from "Unsound" to "Complete and unsound".	Changes made.

JP 6	Introduction	last line of Taint and tainted sources	ed	"value" in "assigned to any value" should be changed to "variable".	Changed : In this regard, taint is an attribute of a value that is assigned to any value originating from a tainted source. To: In this regard, taint is an attribute of a value originating from a tainted source.
JP 7	1 Scope	page number	ed	Why is the number of the first page "2"?	Template-related problem-fixed.
JP 8	3 Normative references	1st paragraph	ed	The term "the C Secure Coding Rules" should be changed to "this Technical Specification".	Change made.
JP 9	4 Terms and definitions	1st line	ed	The term "this document" should be changed to "this Technical Specification" which appears in the third line.	Change made.
JP 10	5.11	Rationale	ed	The word "that" in "strictly aligned that the value" should be changed to "than".	Change made.
JP 11	5.13	1st line of EXAMPLE 4	ed	The comma in "section, 5.2.4.1" should be deleted.	Change made.
JP 12	5.16	last line in Example	ed	The space between left parenthesis and "string" should be deleted.	Change made.
JP 13	5.20	EXAMPLE 5	te	The rule of 5.20 refers to “a C library function with a pair of arguments”. However, “malloc” function mentioned in this example has only one argument, so the rule is not applicable to this example.	Broke out 5.20.4 into a new rule 5.21 Allocating insufficient memory [insufmem]
JP 14	5.21	EXAMPLE 7	ed	The word "noncompliant" should be deleted in "In this noncompliant compliant example".	Change made

JP 15	5.23	EXAMPLE 3	te	In the last call of "fprintf", the variable "msg" is not used as a format string, and this example is unrelated to the rule given in 5.23.	Changed: fprintf(stderr, "%s\n", msg); // diagnostic required to: fprintf(stderr, msg); // diagnostic required
JP 16	5.25	EXAMPLE 3 and 4	ed	The word "solution" should be changed to "example" in "In this compliant solution".	Change made
JP 17	5.20.4	4th example	ed	The serial number of the example is missing.	Inserted " 4" after "EXAMPLE"
NL 1	2		te	- remove the penultimate paragraph of Section 2 ('For each rule, the analyzer shall report ...').	This was discussed at length, there was no consensus to remove this wording, but during this discussion it was noted that the new MISRA C standard has attached a tag to label each rule as decidable or undecidable. There was consensus to add decidable/undecidable tag to each of the rules
NL 2	2		te	- remove the last paragraph of Section 2 ('For each rule, the analyzer shall document ...').	In the 3rd paragraph change the word "discovering" to "detecting"
NL 3	2		te	- replace the 3rd paragraph of Section 2 ('A conforming analyzer shall produce ...') by the following text: When analyzing a program text, a conforming analyser shall, except for the special cases specified in this section, produce a diagnostic message for each occurrence of a violation of a rule specified in this Technical Specification. The special exceptional cases are: - when a code fragment in a program text violates multiple rules simultaneously, a conforming analyser may aggregate diagnostic messages but shall	Change the last paragraph to: For each rule, the analyzer shall document whether its analysis is guaranteed to report all violations of that rule and shall document its accuracy with respect to avoiding false positives and false negatives.

				produce at least one diagnostic message; - when a code fragment in a program text violates the same rule repeatedly, a conforming analyzer may aggregate diagnostic messages but shall produce at least one diagnostic message; - when the analyzer decides for whatever reason (size, complexity) not to (fully) analyze parts of the program text, a conforming analyzer shall have the option to report on this fact.	