Subject: Comments on Committee Draft for 1539-1
From: US National Body for Fortran, IR Van Snyder

Summary

The United States National Body for Fortran recommends the following changes in the Committee Draft for the next revision of ISO/IEC 1539-1.

Contents

The paper from J3 meeting 163 in which each change was proposed is shown for reference. J3 meeting 163 papers are available from `ftp://ftp.j3-fortran.org/j3/doc/meeting/163`

1 Problems for which editorial recommendations are provided

Some of the changes in Sections 1.3 – 1.15 were proposed at J3 meeting 163 in paper 02-309. Some were proposed in paper 02-321r2. Unless otherwise noted, the paper in which the remaining changes were proposed is shown at the beginning of each section.

Edits refer to the J3 committee draft, paper 02-007r3. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by associated text, while a page and line number followed by + (-) indicates that associated text is to be inserted after (before) the indicated line. Where a line number is followed by + and another number or range, it refers to a line that doesn't have a number by referring to the number of the nearest previous line that has a number. Remarks and instructions for the Editor appear between [and].

1.1 Changes recommended for the Introduction

[Editor: On the last 2 lines, delete ", and access ... conditions)".] xiii

1.2 Changes recommended for Section 2

[Editor: Delete sentence "In contexts ... of the structure."] 16:32-33

[Editor: "and" ⇒ "or".] 19:5

[Editor: "may be" ⇒ "is" (twice).] 19:23,24

[Editor: "In an actual argument list" ⇒ "as an actual argument".] 20:15

1.3 Changes recommended for Section 4

Most of the following changes were proposed in paper 02-278r2 at J3 meeting 163. The changes for Note 4.45 on page 54 and constraint 480 on page 61 were also proposed in paper 02-306r1 at J3 meeting 163, and part 5 of paper 02-319r2.

[Editor: "Variables may be objects or subobjects." ⇒ "A variable is a data object."] 33:12

[Editor: Unbold "components".] 41:7

[The introductory waffle mostly ignores type-bound procedures, but where it doesn't, it's wrong. Editor: After "components" at [41:7] insert "and type-bound procedures". Delete the first sentence at [41:20] because it's no longer needed. Delete the second sentence at [41:20-21] because it's wrong.] 41:7, 20-21

[Editor: "a bound for an *explicit-shape-spec* or a *type-param-value*" ⇒ "a *type-param-value* or a bound in an *explicit-shape-spec*".] 43:Note 4.18

[Editor: Append sentence "If the kind selector is omitted, the kind type parameter is default integer."] 45:27

[Repeats verbatim material from 4.2, after which it's upside-down: The attribute determines what variety of a parameter it is. All that's necessary here is the last sentence (but alone it would have a dangling antecedent):] 45:28-46:1

The *type-param-attr-spec* explicitly specifies whether a type parameter is a kind parameter or a nonkind parameter.

[Editor: In lines 10-11 of Note 4.24: "TYPE(MEMBER)" ⇒ "TYPE(MEMBER(9))" twice.] 47:Note 4.24

[Sounds like the final subroutine calls the other final subroutines. Editor: Delete "calling" from the second line of Note 4.33 and insert "are called" after "type" on the third line.] 50:18+7-8

[The possibility to use a *type-alias* name for a parent type name was deleted between 02-007r2 and 02-007r3. Compare versions of Constraint 417. Editor: Delete "a *type-alias* name or".] 54:5+2

[Editor: "An" ⇒ "A scalar". This change was recommended in J3 meeting 164 paper 02-332.] 54:13

[Editor: "specific" ⇒ "nongeneric".] 55:5

[Editor: In the last line of Note 4.50: "END FUNCTION POINT_3D" ⇒ "END FUNCTION POINT_3D_LENGTH".] 56

[Editor: Move the bold to the second instance of "finalized".] 59:19

[The sentence "If the object ... undefined" has nothing to do with *when* an object is finalized, but rather what happens if it's not finalized. A more logical place for it is in 4.5.10.0. Editor: "the object" ⇒ "an object"; then move the sentence to [60:13+], making it a separate paragraph.] 60:20-21

[Editor: "which" ⇒ "that" at the first line of Note 4.58.] 61:0+2

[Editor: replace "or" by a comma at [61:10] and insert ", or a previously defined type alias" after "type" at [61:11].] 61:10-11

[A *scalar-int-initialization-expr* cannot possibly be a dummy argument. Editor: "present" ⇒ "specified".] 62:28

1.4 Changes recommended for Section 5

Most of the following changes were proposed in paper 02-279r2 at J3 meeting 163.

[Note 5.3 contradicts 12.1.2.3, where it says "a dummy procedure with the pointer attribute is a dummy procedure pointer." It's also hard to imagine how a dummy procedure can avoid being a dummy argument. Editor: Delete "a dummy argument that is" at [69:1], insert "without the POINTER attribute" after "procedure" at [69:2], and delete Note 5.3.] 69:1-2, Note 5.3

C529a (R501) The VALUE attribute shall not be specified for a dummy procedure. 69:30+

The following change was proposed at J3 meeting 163 in paper 02-305r1, and part 4 of paper 02-319r2.

5.1.1.7 TYPE 73:16-26

A TYPE type specifier may be used to declare entities of a derived type or a type that is aliased by a type alias.

If the *derived-type-spec* contains a *type-name*, then the TYPE type specifier is used to declare entities of the derived type specified by that *type-name*. Derived type parameter values for each such entity may be specified by a *type-param-spec-list* in the *derived-type-spec*. The components of each such entity are declared to be of the types specified by the corresponding *component-def-stmt*s of the *derived-type-def* (4.5.1).

If the *derived-type-spec* is a *type-alias-name*, then the TYPE type specifier is used to declare entities of the type and type parameters aliased by the type alias specified. It is as if the *declaration-type-spec* for which the *type-alias-name* is an alias was used instead.

Where a data entity is declared explicitly using the TYPE type specifier, the specified derived type or type alias shall have been defined previously in the scoping unit or be accessible there by use or host association. If the data entity is a function result, the derived type or type alias may be specified in the FUNCTION statement provided the derived type or type alias is defined within the body of the function or is accessible there by use or host association. If the derived type or type alias is specified in the FUNCTION statement and is defined within the body of the function, it is as if the function result variable was declared with that derived type or type alias immediately following the *derived-type-def* of the specified derived type or the *type-alias-stmt* that defines the specified *type-alias-name*.

A scalar entity of derived type is a **structure**. If a derived type has the SEQUENCE property, a scalar entity of the type is a **sequence structure**.

5.1.1.8 CLASS

[Every other reference to "use association" has "use" in lower case. Editor: "USE" ⇒ "use".] 74:31

[Editor: Insert a comma before " then" (only one other "then" in Section 5 doesn't have one).] 75:13

[What does "Nonkind type parameters can be deferred" do here? Nothing. Editor: Delete the sentence.] 77:19-20

[What does “Nonkind type parameters can be deferred” do here? Nothing. Editor: Delete the sentence.] 77:23

[Editor: Insert “abstract” before “interface”.] 78:35

[Editor: Insert a comma before “ then” (only one other “then” in Section 5 doesn't have one).] 83:7

The next three changes were proposed in paper 02-316r1 at J3 meeting 163.

[Editor: After “association”, add “and to the target when it is accessed through the pointer”.] 83:10

[Editor: Delete Note 5.25.] 83:11-

[Editor: Remove “and”. After “status”, add “and value”.] 83:12

[Editor: There are two consecutive index items with a space between them at around line 1447 in `c05.tex`. This causes an extra blank between “the” and “SAVE”.] 85:26

[The word “become” implies a dynamic event that occurs at some instant during the execution sequence. Storage association is a static condition. Editor: replace “become” with “be” at the following places: [98:1], [98:3], [98:6], [98:10], [98:12], and [98:17].] 98:1-17

1.5 Changes recommended for Section 6

Most of the following changes were proposed in paper 02-280r1 at J3 meeting 163. The change for page 103 was proposed in paper 02-332 at J3 meeting 163.

[Editor: “first” ⇒ “value of the first”.] 102:15

[Editor: Delete “the derived type definition of”.] 103:3

[Editor: “second” ⇒ “value of the second”.] 105:15

[Editor: “values of the starting and ending point expressions” ⇒ “starting and ending points”.] 105:17

[Editor: “an allocated” ⇒ “any allocated” (twice).] 114:9,10

[Editor: Delete “or subobject”.] 114:25

[Why is a discussion of pointer undefinition in subclause 6.3.3.2, entitled “Deallocation of pointer targets?” It belongs in 16.4.2.1.3, at [404:32-33], which is the only place with a cross reference to 6.3.3.2. Also, item(3) ought to mention the SAVE attribute. Delete [115:3-5]. Replace [404:32-33] by:] 115:3-15

(4) A procedure is terminated by execution of a RETURN or END statement and the pointer is declared or accessed in the subprogram that defines the procedure unless the pointer 404:32-33

[Then delete “A pointer” from the first line of each item at [115:6-13], replace “with” by “Has” at [115:6], insert “that has the SAVE attribute or” after “block” at [115:8], delete “that” at [115:13], capitalize the resulting first word (“Is”) in each item at [115:7-13], and move [115:6-13] to be items (a)-(f) in a sub-list of item (4) at [404:33+]. Here's the result:]

(4) A procedure is terminated by execution of a RETURN or END statement and the pointer is declared or accessed in the subprogram that defines the procedure unless the pointer 404:32-33

   (a) Has the SAVE attribute,
   (b) Is in blank common,
   (c) Is in a named common block that appears in at least one other scoping unit that is in execution,
   (d) Is in the scoping unit of a module if the module also is accessed by another scoping unit that is in execution,
   (e) Is accessed by host association, or
   (f) Is the return value of a function declared to have the POINTER attribute.

[Then delete [115:14-15] (it's covered by [404:30-31]).]

1.6 Changes recommended for Section 7

The following change was proposed in e-mail correspondence after meeting 163

[In table 7.1, the third row of the entry for relational operators .GT. etc., states that comparison of character objects is not permitted. This is a typographical error that crept into the draft between 01-007r3 (the last Frame edition) and 01-007r4 (the first LaTeX edition. Editor: “C I,R L,L” ⇒ “C C L”.] 121:7+15

Most of the following changes were proposed in paper 02-281r2 at J3 meeting 163.

[The reference to R502 is uninformative. Editor: “R502” ⇒ “C501” (Putting a `\label` in the constraint and `\ref` for it in the text doesn't work unless you change the `\stepcounter` inside the `dcoms` macro to `\refstepcounter`; then you'll get the constraint number without a C. I don't know what this will break in 007 – I haven't done the experiments yet.)] 125:13

[Editor: “activation record” ⇒ “instance of a procedure”.] 126:Note 7.9

[Editor: “required to evaluate only” ⇒ ”only required to evaluate”.] 131:9

[Editor: “bounds-remapping-list” ⇒ “*bounds-remapping-list*”.] 144:3

1.7 Changes recommended for Section 8

The following changes were proposed in paper 02-282r1 at J3 meeting 163.

[Too wordy. Editor: “the construct is terminated” ⇒ “it terminates”.] 166:18

1.8 Changes recommended for Section 9

Most of the following changes were proposed in paper 02-283r1 at J3 meeting 163.

[The “re-applied” here is the only word in the standard that begins “re-”. Editor: “re-applied” ⇒ “reapplied”.] 192:6

[Editor: “ID= specifier” ⇒ “identifier (9.5.1.8)”.] 194:25

[Editor: “specifer” ⇒ “specifier”] 202:28

[A dummy procedure pointer is a dummy procedure (see [252:9]). So it's not necessary to specify “or dummy procedure pointer”; Editor: Delete “or dummy procedure pointer”. Editor: Insert “to” between “or” and “a” at [203:17].] 203:16-17

[Editor: Delete sentence “The value . . . specified unit.”] 211:30-31

[Editor: Insert “=” after “IOLENGTH”.] 214:16

The following change was proposed at J3 meeting 163 in paper 02-303r2, and in part 2 of paper 02-319r2.

[Replace the sentence “This specifier may . . . access.” with the following sentence: “This specifier may appear in a data transfer statement only if the statement specifies a unit connected for stream access.”] 190:14-15

The following changes were proposed in paper 02-310r1 at J3 meeting 163.

[Editor: Delete the words starting with “or” and replace with, “or a processor-dependent negative value if the flush operation is not supported for the unit specified”.] 207:11-12

[Editor: Delete Note 9.57.] 207:12+

The following change was proposed in paper 02-314r2 at J3 meeting 163.

[Editor: Insert] 188:8+

C929a (R914) If *default-char-expr* is an initialization expression (7.1.7), the leading part of its value shall be a valid *format-specification*.

[Compare this to the dynamic requirement in 10.1.2.]

1.9 Changes recommended for Section 10

The following change was proposed in paper 02-321, and recommended in paper 02-321r1 at J3 meeting 163.

[Editor: Replace hyphens in Notes 10.11 and 10.12 by minus signs.] 227,228

[Editor: "rounded" ⇒ "rounding mode".] 230:26

[Editor: Move ", $w - n$ shall be positive" to after the following table, changing the comma to "The value of" and appending a period.] 231:1

[Editor: After ";" insert "in this case".] 237:2

1.10 Changes recommended for Section 11

The following changes were proposed in paper 02-284r2 at J3 meeting 163.

[Subclause 11.2.1 is vacuous, mostly because it repeats things said better elsewhere, and it's wrong: accessibility attributes control access to names, not entities. Delete it. The substance of another very small part of it will reappear below as a note.] 247:1-13

[Editor: Change the reference from "`\ref{D11:Module reference}`" to "`\ref{D11:The USE statement and use association}`" (the number will stay the same).] 12:9+6

[Editor: Change the reference from "`\ref{D11:Module reference}`" to "`\ref{D11:The USE statement and use association}`" (the number will stay the same).] 20:18

[Editor: Delete "(11.2.1)" Accessibility attributes are in 4.5.1.8, 5.1.2.1 and 5.1.2.12. What's discussed here is their effect on module references.] 459:7

[Start a new paragraph with the following sentence:] 247:15-

The **USE statement** specifies use association. A USE statement is a **module reference** to the module it specifies. At the time a USE statement is processed, the public portions of the specified module shall be available. A module shall not reference itself, either directly or indirectly.

[The USE statement isn't the means, it's use association. Editor: "The **USE statement** ⇒ "**Use association**" and put "use association" (not capitalized) into the index with a bold-face page number using `\mindex{use association|textbf}`, without a space before the macro.] 247:15

[Editor: First "in" ⇒ "from".] 247:17

[Editor: "be" ⇒ "are".] 247:21

NOTE $11.7\frac{1}{2}$ 247:21+

> The accessibility of module entities may be controlled by accessibility attributes (4.5.1.8, 5.1.2.1), and the ONLY option of the USE statement. Definability of module entities can be controlled by the PROTECTED attribute (5.1.2.12).

[Editor: Delete "explicitly".] 249:1

[Has nothing to do with USE. Editor: Move to [247:1-].] 249:1-5

1.11 Changes recommended for Section 12

Most of the following changes were proposed in paper 02-285r2 at J3 meeting 163.

[Editor: Delete the last sentence.] 252:13-14

[Editor: "a subroutine or a function" ⇒ "an external subroutine or an external function"; after "it" insert "; the interface of an external function that does not have a separate result name is implicit within the program unit that defines it".] 253:20,21

[This isn't the definition of "abstract interface." Editor: Disembolden "abstract interface".] 255:31

[Editor: "without a generic specification" ⇒ "with neither ABSTRACT nor a generic specification".] 255:32

[Editor: Delete the last sentence of the paragraph. It duplicates one at [255:32-33].] 255:39-40

[Editor: "all of ... the" ⇒ "an".] 255:41

[The two sentences "The characteristics ... (12.3.2.3)." are irrelevant to generic interfaces, and are already covered by [255:28-29]. Editor: Delete them.] 257:3-5

[Editor: "operators" ⇒ "operations".] 259:5+4

[Editor: Delete ", an abstract interface".] 260:5

[A dummy procedure pointer is a dummy procedure (see [252:9]), so we need to exclude it here. Editor: After "procedure" insert "without the POINTER attribute".] 267:15

[Editor: After "pointer" insert "that is associated with a procedure" twice.] 267:16,17

[Editor: Replace "subobject selector" with "component selector, array section selector, array element selector, or substring selector".] 269:1

[This sentence overlooks DTIO. Editor: "or" ⇒ ","; after "(7.4.1.4)" insert ", or user-defined derived-type input/output (9.5.3.7.1)".] 272:10

[This sentence overlooks DTIO. Editor: "or" ⇒ ","; after second "statement" insert ", or the processing of an input or output list item".] 272:13

[Editor: "specific only" ⇒ "only specific" (twice).] 272:22,32

[Editor: Insert "only" before "specific".] 274:6

[The nearby material isn't about *prefix-spec*s; those discussions are two paragraphs and a note below. Editor: Move this paragraph to be between Note 12.37 and [276:30].] 276:14-18

[The remainder of the paragraph discusses "the function;" this sentence suddenly discusses "a function." Editor, for consistency, "a" ⇒ "the" and the second "the value" ⇒ "that".] 276:24-25

[Too wordy, and sounds like one can't specify INTENT(IN) for pointer arguments. Use the term "dummy data object" for a dummy data object.] 282:1-2

C1267 The *specification-part* of a pure function subprogram shall specify that all nonpointer dummy data objects have INTENT(IN).

[Too wordy, and sounds like one can't specify INTENT for pointer arguments. Use the term "dummy data object" for a dummy data object.] 282:3-5

C1268 The *specification-part* of a pure subroutine subprogram shall specify the intents of all nonpointer dummy data objects.

[Too wordy. Editor: "dummy arguments that are procedure arguments" ⇒ "dummy procedures".] 282:9-10

The following change was proposed at J3 meeting 163 in paper 02-304r1, and in part 3 of paper 02-319r2.

[The sentence beginning with "Otherwise" doesn't parse well as it is, and simply adding "or ALLOCATABLE" would make the problem worse. Editor: Replace this sentence with following: "Otherwise, they are storage associated and shall all be nonpointer, nonallocatable scalars of type default integer, default real, double precision real, default complex, or default logical."] 279:20-22

The following changes were proposed in paper 02-314r1 at J3 meeting 163.

[Editor: Change to] 275:12-14

R1224 *function-stmt* **is** [*prefix*] FUNCTION *function-name* ■
■ ([*dummy-arg-name-list*]) [*suffix*]

[Editor: Insert] 275:38+

R1228a *suffix* **is** *proc-language-binding-spec* [RESULT (*result-name*)]
or RESULT (*result-name*) [*proc-language-binding-spec*]

[For only two items, the above approach to the BNF seems simplest. If there were more, the approach used for *prefix* would be better.]

[Editor: Change to] 277:8-9

R1231 *subroutine-stmt* **is** [*prefix*] SUBROUTINE *subroutine-name* ■
■ [([*dummy-arg-list*]) [*proc-language-binding-spec*]]

[This is slightly different from what was in 02-314r1, in that *prefix* is optional here but was not optional in 02-314r1. Kurt Hirchert has remarked that this was a typographical error in 02-314r1.]

[Editor: Change to] 278:31-35

R1234 *entry-stmt* **is** ENTRY *entry-name* [([*dummy-arg-list*]) [*suffix*]]

[Yes, this really replaces *both* of the existing alternatives.]

1.12 Changes recommended for Section 13

[Editor: "unallocated" ⇒ "unallocated allocatables".] 287:10

[The next two changes were proposed in J3 meeting 163 paper 02-332r1.]

[Editor: "(C)" ⇒ "(C [,KIND])".] 315:14

[Editor: Replace with the following:] 315:18-19

Arguments.

C	shall be of type default character and of length one.
KIND (optional)	shall be a scalar integer initialization expression.

Result Characteristics. Integer. If KIND is present, the kind type parameter is that specified by the value of KIND; otherwise the kind type parameter is that of default integer type.

[Editor: "depedent" ⇒ "dependent".] 350:11

1.13 Changes recommended for Section 14

Most of the following change was proposed in part 8 of paper 02-319r2 at J3 meeting 163.

[Editor: Change "70" to "30".] 369:31

[Editor: In Note 14.12: "LOGICAL MATRIX_ERROR = .FALSE." ⇒ "LOGICAL :: MATRIX_ERROR = .FALSE."] 377

1.14 Changes recommended for Section 15

It is recommended that the description of the procedures in Section 15 in the draft standard be revised. If that occurs, at least the following changes are recommended. Most of these changes were proposed at J3 meeting 163 in paper 02-286r1. The first, third and fourth of these were proposed at J3 meeting 163 in paper 02-314r1.

[Editor: Make a subclause, number 15.1.2.1, out of the description of C_LOC, as we do in Sections 13 and 14.] 382:17

[It already says "scalar" at [382:28] so it isn't needed here. Editor: Delete "scalar".] 383:2

[Editor: Make a subclause, number 15.1.2.2, out of the description of C_ASSOCIATED, as we do in Sections 13 and 14.] 383:11

[Editor: Make a subclause, number 15.1.2.3, out of the description of C_F_POINTER, as we do in Sections 13 and 14.] 384:1

[The space between the heading and body of Table 15.2 needs some polish. Use something like is done in Table 15.1:] 385:17+3

```
\hline
\multicolumn{3}{|l|}{}                          \\[-10pt]
\hline
```

The following changes were proposed at J3 meeting 163 in paper 02-307r1, and part 6 of paper 02-319r2.

[After "C_SIZE_T," insert "C_INT8_T, C_INT16_T, C_INT32_T, C_INT64_T,".] 381:20

[Remove "and".] 381:22

[After "C_INTMAX_T", insert ", and C_INTPTR_T".] 381:22

[After "-1", insert "if the companion C processor defines the corresponding C type and there is no interoperating Fortran processor kind or -2 if the C processor does not define the corresponding C type".] 381:23

[Add the following rows to Table 15.2 below the row for C_SIZE_T:] 385:17+

(Column 1)	(Column 2)	(Column 3)
INTEGER	C_INT8_T	int8_t
INTEGER	C_INT16_T	int16_t
INTEGER	C_INT32_T	int32_t
INTEGER	C_INT64_T	int64_t

[Add the following row to Table 15.2 below the row for C_INTMAX_T:] 385:17+

(Column 1)	(Column 2)	(Column 3)
INTEGER	C_INTPTR_T	intptr_t

The following changes were proposed in paper 02-318r2 at J3 meeting 163.

The error is in the C function prototype given at Note 15.22 at page 390.

The prototype in the note is:

```
short func(int i; double *j; int *k; int l[10]; void *m)
```

This is not valid C. The semicolons must be replaced by commas such as:

```
short func(int i, double *j, int *k, int l[10], void *m);
```

1.15 Changes recommended for Section 16

Most of the following changes were proposed in paper 02-287r2 at J3 meeting 163.

[Editor: Delete "or type-bound".] 397:15

[Editor: In Note 16.6, "may be found" ⇒ "is".] 398

[Editor: Move "as an argument keyword" to the beginning of its sentence, add a comma after it, and adjust capitalization. "As an argument keyword, it" ⇒ "It". (twice).] 398:36-38 399:3-4

[Editor: "contains" ⇒ "is the host of".] 401:28

[Editor: Insert ", a type-bound procedure," after "pointer".] 411:39+5

[Note 16.19 has nothing to do with anything nearby. Editor: Move it to [413:6+].] 412:top

---

[16.5.6 is about events, not enduring conditions. All of the other items say "becomes undefined" instead of "is undefined". Editor: "is undefined" ⇒ "becomes undefined" four times, at [412:29-30, 31, 35, 37].] 412:29-37

[The phrase is mangled. Editor: Insert "of" after "subcomponents".] 412:32

[Item (12c) overlooks components that have default initialization. Editor: Insert "except for any non-pointer default-initialized subcomponents of the argument" after "undefined".] 412:34

[Editor: Insert "of its" after "any"; delete "of the result".] 412:37-38

[16.5.6 is about events, not enduring conditions. Editor: "has completed" ⇒ "completes". All of the index names become undefined. Editor: "*index-name*" ⇒ "*index-names*"; "becomes" ⇒ "become".] 412:41

1.16 Changes recommended for Annex A

[Editor: "and" ⇒ "or".] 418:26

1.17 Changes recommended for Annex C

[Editor: "can be used only" ⇒ "may only be used".] 443:12

1.18 "Implied-DO" shouldn't have a hyphen where it's not an adjective

The term "implied DO" is used without definition. It appears frequently unhyphenated where it ought to be hyphenated (because it's used as an adjective, as in "implied DO variable") and it appears frequently hyphenated where it ought not to be (because it's not used as an adjective).

The correct solution is to use the appropriate syntax term, except maybe in notes.

The following changes were proposed in paper 02-288r2 at J3 meeting 163.

[Editor: "implied DO variable" ⇒ "ac-do-variable".] 64:19

[Editor: "implied-DO" ⇒ "implied DO" in the first line of Note 4.66.] 64:-1

[Editor: "implied-DO variable" ⇒ "*data-i-do-variable*".] 87:6

[Editor: "implied-DO list" ⇒ "*data-implied-do*".] 87:10

[Editor: "the bounds ... expressions" ⇒ "each *scalar-int-expr* of each *ac-implied-do-control* is a restricted expression".] 125:23-24

[Editor: "implied-DO variable"⇒ "*ac-do-variable*".] 125:36

[Editor: ""the bounds ... expressions" ⇒ "each *scalar-int-expr* of the corresponding *ac-implied-do-control* is a restricted expression".] 125:36-37

[Editor: "the bounds ... expressions" ⇒ "each *scalar-int-expr* of each *ac-implied-do-control* is an initialization expression".] 126:28-29

[Editor: "implied-DO variable"⇒ "*ac-do-variable*".] 127:21

[Editor: ""the bounds ... expressions" ⇒ "each *scalar-int-expr* of the corresponding *ac-implied-do-control* is an initialization expression".] 128:21-22

[Editor: "the bounds ... implied-DO" ⇒ "each *scalar-int-expr* of the *ac-implied-do-control* in any *ac-implied-do*".] 129:2-3

[Editor: "implied-DO lists" ⇒ "*io-implied-do*s".] 193:10

[Editor: "implied-DO" ⇒ "*io-implied-do*".] 193:12

[Editor: In the first line of Note 9.37: "implied-DO" ⇒ "implied DO".] 193:14+2

[Editor: "in an implied-DO" ⇒ "as a *scalar-int-expr* in an *io-implied-do-control*".] 194:15

[Editor: "implied DO variables" ⇒ *do-variable*s" thrice.] 215:2, 20, 43

[Editor: In the third line of Note 10.4: "implied-DO" ⇒ "implied DO".] 222:25+4

[Editor: "implied DO variable" ⇒ *do-variable*".] 237:22

[Editor: "DO variable of an implied-DO" ⇒ "*data-i-do-variable*"; before second "an" insert "or an *ac-do-variable* in".] 399:7

[Editor: "a variable ... implied-DO" ⇒ "*data-i-do-variable*" twice.] 399:11

[Editor: "the implied-DO" ⇒ its *data-implied-do*"; before "an" insert "or an *ac-do-variable* in.] 399:12

[Editor: "the DO variable of an implied-DO" ⇒ "a *data-i-do-variable*"; before second "an" insert "or an *ac-do-variable* in.] 399:15

[Editor: "implied-DO list" ⇒ "*io-implied-do*".] 410:14

[Editor: "implied-DO variable" ⇒ "*do-variable*".] 410:15

[Editor: "implied-DOs" ⇒ "*io-implied-do*s".] 412:5

[Editor: "implied-DO variables" ⇒ "*do-variable*s".] 412:6

[Editor: "Implied-DO variables" ⇒ "The *do-variable*s in *io-implied-do*s".] 451:35

1.19 "Unsaved" should be "nonsaved"

The term "unsaved" has a temporal connotation, i.e., becoming "unsaved" is a process. Whether a variable has the SAVE attribute is, however, a static condition. A variable does not become saved, or become unsaved. Where the absence of any other attribute is discussed, it has the "non" prefix, not the "un" prefix. The term "unsaved" is the wrong term; it ought to be "nonsaved".

The following changes were proposed in paper 02-279r2 at J3 meeting 163.

Editor: Replace "unsaved" by "nonsaved" everywhere, preserving case. Many of them also require replacing "an" by "a". The Editor will know what to do. Here are most (maybe all) of the places:

82:12 111:13 111:15 113:9 Note 6.23 on page 113 278:23 404:21
411:9 411:34 411:35 411:38 411:40

1.20 NONKIND is an unfortunate attribute name

What we currently call nonkind type parameters can only ultimately be used for character lengths or array dimensions. So as to allow other attributes of the KIND–NONKIND variety, change NONKIND to something more focused, such as EXTENT.

This change was proposed at J3 meeting 163 in papers 02-279r2, 02-301r1, 02-314r1, and 02-318r2.

[Editor: "a nonkind" ⇒ "an extent" (change the index entry too).] 32:7

[Editor: "A nonkind" ⇒ "An extent".] 32:12

[Editor: "a nonkind" ⇒ "an extent" at the following places: [32:13-14], [32:14+2], [32:22], [33:1], [45:28], [70:21-22], [415:36].]

[Editor: "nonkind" ⇒ "extent" at the following places: [41:11], [44:35], [46:1], [50:12], [110:7], [125:13], [199:15], [269:10], [382:28], [424:26].]

or EXTENT 42:17

[Editor: "NONKIND" ⇒ "EXTENT".] 46:4+5

[Editor: In the fourth line of Note 4.24, "NONKIND" ⇒ "EXTENT".] 47

[Editor: In the first line of Note 4.70, "a nonkind" ⇒ "an extent".] 65:bottom

[Editor: "Nonkind" ⇒ "Extent".] 77:18

[Editor: "Nonkind" ⇒ "Extent".] 77:23

[Editor: "A nonkind" ⇒ "An extent".] 418:7

1.21 Remove the access specification from the extends clause

The following changes were proposed at J3 meeting 163 in paper 02-308r1. Problems addressed by these changes were also noticed in part 9 of paper 02-319r2.

[Remove "[*access-spec* ::]" from the syntax rule.] 42:1

[Remove “the parent component or”.]	408:9
[Remove “public ::” from the extends clause.]	433:8
[Remove these lines.]	433:14-17
[Remove this line.]	433:23
[Replace “,” with “)”.]	433:33
[Remove “&”.]	433:33
[Remove these lines.]	433:34-36

2 Problems for which editorial recommendations are not provided

2.1 Selecting the “declared” component of polymorphic objects

The following was proposed in paper 02-295r3 at J3 meeting 163.

The US National Body recommends that J3 more carefully consider the issue of where polymorphic variables need to be explicitly cast into non-polymorphic ones, and suggest how to clarify this in and improve the consistency of the draft standard if needed.

2.2 Create polymorphic objects without values, and copy them

The following was proposed in paper 02-294r2 at J3 meeting 163.

The US National Body recommends that J3 more carefully consider the issues of CREATE and COPY operations for polymorphic variables, and recommend how to modify the Fortran 2000 Committee Draft if needed.

2.3 Reinstating deferred bindings

The following change was proposed at J3 meeting 163 in paper 02-296r2, and in part 10 of paper 02-319r2.

The US National Body recommends that deferred bindings be added to the draft standard, using the specifications proposed in 02-296r2 as a design guide.

2.4 Note 4.50 error

The following change was proposed in paper 02-301r1 at J3 meeting 163.

The IF/END IF conditional in Note 4.50 on page 56 needs to be replaced with SELECT TYPE in order for this example to be correct:

```
SELECT TYPE(B)
   CLASS IS(POINT_3D)
      ...! Body of IF
END SELECT
```

The US National Body recommends that this edit be made in the Committee Draft.

2.5 Delete the enum feature

In J3 meeting 163 paper 02-321, it was proposed to delete the enum feature. Paper 02-311r1 proposed the following answer.

The US National Body recomments that we leave enums in the standard but require the use of the BIND attribute.

2.6 Cross reference in Annex D

The following change was proposed in paper 02-314r1 at J3 meeting 163.

The US National Body recommends that a cross reference listing for the BNF terms be added to Annex D. The editor already has tools to do this. J3 agreed informally to do this at meeting 162, but a formal vote was not taken.

2.7 Problem with rules in 16.2.3

There appears to be a serious problem with the rules in 16.2.3 in that they are violated by intrinsic assignment as it applies to extensible types.

The source of this problem appears to be that 12.4.1.2 allows an actual argument of an extension type (e.g., TYPE(APPLE)) to correspond with a dummy argument of its parent type (e.g., TYPE(FRUIT)). If the programmer truly wanted this mismatch, the dummy argument could have been declared CLASS(FRUIT) instead of TYPE(FRUIT).

The US National Body agrees that there is a problem here and recommends that it be fixed.

2.8 Problems with "getting a value"

Problems with the way a variable is described to "get a value" are noticed in part 1 of J3 meeting papers 02-319r2 and 02-330r2. The descriptions of the problems in those papers are extensive, and are not quoted here. The reader's attention is drawn to those papers.

These comments raise important questions that The US National Body recommends be addressed.

2.9 Problem with IOSTAT_END and IOSTAT_EOR

Part 7 of J3 meeting 163 paper 02-319r2 noticed problems with the use of constants to represent the conditions described for IOSTAT_END and IOSTAT_EOR.

> The IOSTAT_END constant in the ISO_FORTRAN_ENV module is a useful thing, but requiring that a processor can only have one IOSTAT value for end-of-file is overly restrictive. It invalidates one implementation I can think of.

The US National Body recommends that intrinsic functions IOSTAT_END and IOSTAT_EOR be added to the draft standard. These intrinsic functions replace the named constants in the current draft and return .TRUE. if the actual argument satisfies the condition.

2.10 Problem with "executes normally"

Part 8 of J3 meeting 163 paper 02-319r2 noticed problems with the definition of the term "executes normally".

> Page 358, lines 2-5 of the Committee Draft specify what happens "if an intrinsic procedure or a procedure defined in an intrinsic module executes normally". This is far too vague and could be interpreted in many ways.
>
> Does "executes normally" mean that there was no hardware failure during execution of the procedure? It could be interpreted this way, even if it is a bit extreme. If there is no hardware failure, then even if you pass arguments that are nonsensical for that procedure (for example, IEEE_REM(0.0, 0.0)), it is doing what could be considered "normal" for those arguments.
>
> Does "executes normally" mean that it produces a value that is mathematically valid? This would narrow the scope of the statement and its prescribed behaviour to an acceptable point.
>
> Or does "executes normally" mean that it produces an exact representation of a mathematically valid value? This narrows the scope too much, as almost no floating-point operation would execute "normally".
>
> I believe that my second interpretation was probably the one that was intended. But whether it is or not, the "executes normally" bit is far too vague to provide for any consistent interpretation.

The US National Body recommends that these problems be fixed in the current draft of the standard.

2.11 Wording problems noticed in paper 02-321

Items W8 and W9 in paper 02-321 noticed wording problems. In J3 meeting 163 paper 02-321r2, it was recommended to repair these problems.

W8 The sentence at [399:37-39] apparently mentions three FORALL constructs, but leaves it confused as to what relationship among them is referred to, or even if it is actually 3 separate ones. Ref 02-276, 02-221r2.

W9 The sentence structures in [402:1-9] are so complicated that it is next to impossible to figure out what modifies what, and thus what these sentences actually mean unless you knew the answer before reading them. Ref 02-276, 02-221r2.

The US National Body recommends that the wording problems noticed in items W8 and W9 in paper 02-321 be corrected.

2.12 Technical problems noticed in paper 02-321

Items T5, T6 and T8 in paper 02-321 noticed technical problems. In J3 meeting 163 paper 02-321r2, it was recommended to repair these problems.

T5 The terms “local entity” and “global entity” seem to be misused regularly. Introduction of the term “local name” would probably go a long way towards fixing these confusions. Paper J3/01-163 brought this issue to my attention; I think it was a good start, but didn’t fix these terms.
For example, the section on host association says that if a name appears in any of a bunch of contexts, “then it is the name of a local entity”. This is apparently intended to distinguish it from host-associated names, but of course, it doesn’t. And the phrase “local entity of” (as used in the glossary entry for “automatic data object”) makes no sense. Section 14.1.1 talks about “a name that identifies a global entity” in a way that apparently forgets that global entities may be identified by local names (via rename). (surprised there hasn’t been an interp on that one because it sure sounds like it is saying that a widespread practice is illegal).
I do note that the terms “global ent” and “local ent” (I searched on these short forms to avoid missing plurals) are almost entirely restricted to c14. The only other appearances of “local ent” are two in the glossary - one is the glossary definition of the term itself; the other is the incorrect usage cited above. The only other appearance of “global ent” is one incorrect one in c12 (where it confuses the name and the entity).
I think almost all of the occurances of “local entity” would be better replaced by “local name” (which I see 6 uses of, but no definition). “Global name” also appears 6 places, none of them being in a definition. Heck, seems like many of the occurances of “local entity” already end up looking like “name that identifies a local entity”. Not all the uses can be so replaced (after all, there is brief discussion of local entities that don’t have names), but I think most cases can. The term “local name” seems more intuitively appropriate anyway. At least to my ear, saying that something is a local name in a scoping unit is a lot less likely to give an incorrect impression that the named entity is somehow local *ONLY* to the scoping unit in question.

T6 The section on the procedure declaration statement (12.3.2.3) has a whole bunch of syntax with no associated meaning. It adequately discusses the *proc-interface*, but that’s about all. It says not a single word about what any of the *proc-attr-specs* mean. We don’t need to say much about them here, but we *DO* need to say something. As is, we don’t even say that they specify the procs in question to have those attributes. We ought to at least say that, and then xref the appropriate sections. Should do the xrefs in such a manner as to also pull in all the associated constraints/restrictions. For example, nothing here says that the entity has to be a dummy argument to have the optional or intent attributes - or that the entity had better not be a dummy argument of it has the save attribute, etc.
Also, I see no hint here about what the => null() is supposed to mean or what contexts it is allowed in. Perhaps that one can also be referenced. But if so, note that we also use the *proc-decl* bnf in procedure pointer components, where I think the syntax means something slightly different (default initialization vs object initialization).

T8 The descriptions of C_LOC and C_F_POINTER are so incomprehensible as to constitute technical, rather then mere wording flaws. The descriptions are full of requirements that are assumed rather than stated, and other requirements that contradict each other by failing to state assumed conditions. For the simplest example, [384:20] says without qualification "FPTR shall be scalar", although there is lots of discussion of cases where it is an array; this requirement is clearly intended to apply only in some cases, but that isn't stated. Ref 02-276, 02-230r3.

The US National Body recommends that the technical problems noticed in items T5, T6 and T8 in paper 02-321 be corrected.

2.13 Sizes in bits

The following was proposed in paper 02-320r2 at J3 meeting 163.

The US National Body recommends that constants be added that specify the size in bits of the file storage unit, the numeric storage unit, and the character storage unit.

2.14 Global names of intrinsic and nonintrinsic modules

The US National Body recommends that J3 review whether a program can have both an intrinsic and a nonintrinsic module of the same name.

2.15 Kind type parameter for BOZ constants

The US National Body recommends that J3 revise the description of BOZ literal constants in DATA statements and as the A argument of the INT intrinsic, so that they have a kind type parameter appropriate to the context.

2.16 Examples that may be incorrect for some processors

The US National Body recommends that J3 revise the examples for SELECTED_INT_KIND and SELECTED_REAL_KIND so that they are correct for all processors, or insert a disclaimer that examples in Sections 13-15 are not normative, and may not be accurate for all processors.