```
To: J3
From: Malcolm Cohen
Subject: Interpretation Update Pages: Standing Document 018 (08-018r1)
Date: 2008/05/13
```

This document contains insertions for every interpretation edit that has been published as a corrigendum to 04-007.

The following pages are intended for insertion into a loose-leaf binder version of 04-007. This document needs to be printed single-sided for this to work.

Most edits are followed by a "making the whole paragraph read" summary; in such summaries deleted text appears struck-out ~~like this~~ and new text is wavy-underlined like this. (NB: Some summaries might be missing this feature.)

Interp **F03/0020**, Status: *Corrigendum 2.*

Ref: 4.4.1, 1st paragraph, 3rd sentence, [36:14]

After the 3rd sentence,
Before "The kind type parameter..."
Insert the following sentence:

      The kind type parameter is of type default integer.

36.1

Interp **F03/0020**, Status: *Corrigendum 2.*

Ref: 4.4.2, 1st paragraph, 3rd sentence, [37:30]

After the 3rd sentence,
Before "The kind type parameter..."
Insert the following sentence:

> The kind type parameter is of type default integer.

Interp **F03/0029**, Status: *Corrigendum 2.*

Ref: 4.4.2, 2nd paragraph, [38:2]

Before "equivalent"
Insert "mathematically",
Making the whole paragraph read:

The real type includes a zero value. Processors that distinguish between positive and negative zeros shall treat them as mathematically equivalent

(1)   in all relational operations,
(2)   as actual arguments to intrinsic procedures other than those for which it is explicitly specified that negative zero is distinguished, and
(3)   as the *scalar-numeric-expr* in an arithmetic IF.

38.1

Interp **F03/0020**, Status: *Corrigendum 2.*

Ref: 4.4.3, 2nd paragraph, 2nd sentence, [39:15]

After the 2nd sentence,
Before "The kind type parameter..."
Insert the following sentence:

> The kind type parameter is of type default integer.

Interp **F03/0020**, Status: *Corrigendum 2.*

Ref: 4.4.4, 1st paragraph, 4th sentence, [40:10]

Before "its value",
Insert "its kind is processor-dependent and",
Making the whole sentence read:

> The length is a type parameter; its kind is processor-dependent and its value is greater than or equal to zero.

Interp **F03/0020**, Status: *Corrigendum 2.*

Ref: 4.4.4, 2nd paragraph, 2nd sentence, [40:14]

After the 2nd sentence,
Before "The kind type parameter..."
Insert the following sentence:

> The kind type parameter is of type default integer.

40.1

Interp **F03/0027**, Status: *Corrigendum 2.*

Ref: 4.4.4.1, constraint C416, [41:9,9+]

At the end of list item (3),
Delete "or",
And add a new list item immediately afterwards as follows:

  (3.5) in the *type-spec* or *derived-type-spec* of a type guard statement (8.1.5), or

Interp **F03/0027**, Status: *Corrigendum 2.*

Ref: 4.4.4.1, last paragraph, [41:33+]

After list item (3),
Insert a new list item as follows:

  (3.5) If used in the *type-spec* of a type guard statement, the associating entity assumes its length from
      the selector.

Interp **F03/0020**, Status: *Corrigendum 2.*

Ref: 4.4.5, 2nd paragraph, 2nd sentence, [44:2]

After the 2nd sentence,
Before "The kind type parameter..."
Insert the following sentence:

> The kind type parameter is of type default integer.

44.1

Interp **F03/0072**, Status: *Corrigendum 2.*

Ref: 4.5.3, before R445, [50:40+]

Insert a new constraint as follows:

C447a (R440) If *component-initialization* appears, every type parameter and array bound of the component shall be an initialization expression.

50.1

Interp **F03/0009**, Status: *Corrigendum 1.*

Ref: 4.5.3.3, constraint C453, [53:1]

Append to constraint: "It shall not have the VALUE attribute.",
Making the whole constraint read:

C453   The passed-object dummy argument shall be a scalar, nonpointer, nonallocatable dummy data
        object with the same declared type as the type being defined; all of its length type parameters
        shall be assumed; it shall be polymorphic (5.1.1.2) if and only if the type being defined is
        extensible (4.5.6). It shall not have the VALUE attribute.

Interp **F03/0094**, Status: *Corrigendum 3.*

Ref: 4.5.5, C473, [58:14]

After "The dummy argument shall not"
Change "be INTENT(OUT)"
To "have the INTENT(OUT) or VALUE attribute",
Making the whole constraint read:

C473   (R454) A *final-subroutine-name* shall be the name of a module procedure with exactly one
        dummy argument. That argument shall be nonoptional and shall be a nonpointer, nonallocat-
        able, nonpolymorphic variable of the derived type being defined. All length type parameters
        of the dummy argument shall be assumed. The dummy argument shall not ~~be~~ have the IN-
        TENT(OUT) or VALUE attribute.

Interp **F03/0062**, Status: *Corrigendum 2.*

Ref: 4.5.5.2, 4th paragraph, [59:27]

After the first occurrence of "structure constructor",
insert "or array constructor".
On the same line, delete the second occurrence of "structure".
This makes that whole paragraph read:

> If an executable construct references a structure constructor or array constructor, the entity created by the constructor is finalized after execution of the innermost executable construct containing the reference.

Interp **F03/0007**, Status: *Corrigendum 1.*

Ref: 4.5.5.2, 5th paragraph, [59:30]

Replace "first executable statement"
By "executable constructs",
Making the whole paragraph read:

> If a specification expression in a scoping unit references a function, the result is finalized before execution of the ~~first executable statement~~ executable constructs within the scoping unit.

Interp **F03/0007**, Status: *Corrigendum 1.*

Ref: 4.5.5.2, after the 5th paragraph, [59:30+]

~~Insert new paragraph:~~

> ~~If a specification expression in a scoping unit references a structure constructor, the entity created by the structure constructor is finalized before execution of the executable constructs in the scoping unit.~~

Interp **F03/0062**, Status: *Corrigendum 2.*

Ref: 4.5.5.2, after the 5th paragraph, [59:30+]

In the (struck-out) new paragraph inserted by interp F03/0007 above,
After the first occurrence of "structure constructor",
insert "or array constructor".
In the same sentence, delete the second occurrence of "structure".
This makes the inserted paragraph read:

> If a specification expression in a scoping unit references a structure constructor or array constructor, the entity created by the constructor is finalized before execution of the executable constructs in the scoping unit.

Interp **F03/0013**, Status: *Corrigendum 1.*

Ref: 5.1, constraint C509, [72:23]

Append to constraint: "It shall not have the VALUE attribute.",
Making the whole constraint read:

C509   (R501) An entity declared with the CLASS keyword shall be a dummy argument or have the ALLOCATABLE or POINTER attribute. It shall not have the VALUE attribute.

Interp **F03/0012**, Status: *Corrigendum 2.*

Ref: 5.1, C512, [72:28]

Delete ", EXTERNAL",
Making the whole constraint read:

C512   (R501) If the POINTER attribute is specified, the ALLOCATABLE, TARGET, ~~EXTERNAL,~~ or INTRINSIC attribute shall not be specified.

72.1

Interp **F03/0012**, Status: *Corrigendum 2.*

Ref: 5.1, C521, [73:7]

After "dummy procedure"
Insert ", a procedure pointer",
Making the whole constraint read:

C521   (R504) The *function-name* shall be the name of an external function, an intrinsic function, a
        function dummy procedure, a procedure pointer, or a statement function.

Interp **F03/0012**, Status: *Corrigendum 2.*

Ref: 5.1, C536, [73:35-36]

Replace C536 with the following:

C536   (R501) If the PROTECTED attribute is specified, the INTRINSIC or PARAMETER attribute
        shall not be specified. If the PROTECTED and EXTERNAL attributes are specified, the
        POINTER attribute shall also be specified.

Interp **F03/0014**, Status: *Corrigendum 1.*

Ref: 5.1.2.5.1, constraint C542, [78:21-22]

Replace "a dummy... procedure"
By "declared only in a subprogram or interface body",
Making the whole constraint read:

C542   (R511) An explicit-shape array whose bounds are not initialization expressions shall be ~~a dummy argument, a function result, or an automatic array of a procedure~~ declared only in a subprogram or interface body.

Interp **F03/0014**, Status: *Corrigendum 1.*

Ref: 5.1.2.5.1, paragraph after constraint C542, [78:23]

After "subprogram" insert "or interface body",
Making the whole paragraph read:

> An **automatic array** is an explicit-shape array that is declared in a subprogram or interface body, is not a dummy argument, and has bounds that are not initialization expressions.

Interp **F03/0045**, Status: *Corrigendum 2.*

Ref: 5.1.2.5.4, constraint C544, [80:9]

Before "of a type"
Insert "polymorphic, of a finalizable type, of a type with an ultimate allocatable component, or",
Making the whole constraint read:

C544   An assumed-size array with INTENT (OUT) shall not be polymorphic, of a finalizable type, of
        a type with an ultimate allocatable component, or of a type for which default initialization is
        specified.

80.1

Interp **F03/0012**, Status: *Corrigendum 2.*

Ref: 5.2 1st paragraph, last sentence, [73:35-36]

Replace whole sentence "This also applies to PROCEDURE, EXTERNAL, and INTRINSIC statements.",
By the following whole sentence:

> This also applies to procedure declaration statements, and to EXTERNAL and INTRINSIC statements.

86.1

Interp **F03/0012**, Status: *Corrigendum 2.*

Ref: 5.2.10, C568, [91:5]

Replace C568 "A *proc-entity-name* shall also be declared in a *procedure-declaration-stmt.*"
By the following whole constraint:

C568   (R541) The EXTERNAL attribute (5.1.2.6) shall be explicitly specified for a *proc-entity-name.*

Interp **F03/0011**, Status: *Corrigendum 1.*

Ref: 6.3.1, [111:11-12]

Replace "unlimited polymorphic" in constraint C625
By "unlimited polymorphic or is of abstract type",
Making the whole constraint:

C625   (R623) If any *allocate-object* is unlimited polymorphic or is of abstract type, either *type-spec* or
         SOURCE= shall appear.

Interp **F03/0007**, Status: *Corrigendum 1.*

Ref: 6.3.3.1, 2nd paragraph after Note 6.24, [116:8]

Replace "first executable statement"
By "executable constructs",
Making the whole paragraph read:

> If a specification expression in a scoping unit references a function whose result is either allocatable
> or a structure with a subobject that is allocatable, and the function reference is executed, an
> allocatable result and any subobject that is an allocated allocatable entity in the result returned by
> the function is deallocated before execution of the ~~first executable statement~~ executable constructs
> in the scoping unit.

116.1

Interp **F95/0030**, Status: *Corrigendum 1.*

Ref: 7.1.6, immediately before Note 7.10, [126:19+]

Insert new paragraph:

If a specification expression in a module includes a reference to a generic, that generic shall have no specific procedures defined in the module subsequent to the specification expression.

126.1

Interp **F95/000030**, Status: *Corrigendum 1.*

Ref: 7.1.7, immediately before Note 7.11, [127:33+]

Insert new paragraph:

If an initialization expression in a module includes a reference to a generic, that generic shall have no specific procedures defined in the module subsequent to the initialization expression.

Interp **F03/0006**, Status: *Corrigendum 1.*

Ref: 7.4.1.3, 1st paragraph, [139:17]

Replace "the evaluation of all operations in *expr* and *variable*"
By "the evaluation of *expr* and the evaluation of all expressions in *variable*",
Making the whole paragraph read:

> Execution of an intrinsic assignment causes, in effect, the evaluation of the expression *expr* and all expressions within *variable* (7.1.8), the possible conversion of *expr* to the type and type parameters of *variable* (Table 7.9), and the definition of *variable* with the resulting value. The execution of the assignment shall have the same effect as if the evaluation of ~~all operations in *expr* and~~ *expr* and the evaluation of all expressions in *variable* occurred before any portion of *variable* is defined by the assignment. The evaluation of expressions within *variable* shall neither affect nor be affected by the evaluation of *expr*. No value is assigned to *variable* if *variable* is of type character and zero length, or is an array of size zero.

Interp **F03/0093**, Status: *Corrigendum 3.*

Ref: 7.4.1.3, 3rd paragraph, [139:22,25]

The paragraph begins "If *variable* is an allocated allocatable";
At beginning of paragraph
Insert new sentence "If *variable* is an unallocated allocatable array, *expr* shall have the same rank as *variable*.",
In the last sentence of the paragraph,
After "equal to the corresponding type"
Change "parameters" to "parameter",
Before "with the shape of *expr*"
Change "," to "." and
Insert "If *variable* is an array and *expr* is scalar it is allocated with the same bounds as before, otherwise it is allocated",
Making the whole paragraph read:

> If *variable* is an unallocated allocatable array, *expr* shall be an array of the same rank as *variable*. If *variable* is an allocated allocatable variable, it is deallocated if *expr* is an array of different shape or any of the corresponding length type parameter values of *variable* and *expr* differ. If *variable* is or becomes an unallocated allocatable variable, then it is allocated with each deferred type parameter equal to the corresponding type parameters of *expr*. If *variable* is an array and *expr* is scalar it is allocated with the same bounds as before, otherwise it is allocated with the shape of *expr*, and with each lower bound equal to the corresponding element of LBOUND(*expr*).

Interp **F03/0006**, Status: *Corrigendum 1.*

Ref: 7.4.1.3, list item (2) after Note 7.39, [141:20,21,22]

Insert "the value of" before each of the three occurrences of "*expr*",
Making the whole paragraph read:

(2)   If the component of the value of *expr* is allocated, the corresponding component of *variable* is allocated with the same dynamic type and type parameters as the component of the value of *expr*. If it is an array, it is allocated with the same bounds. The value of the component of the value of *expr* is then assigned to the corresponding component of *variable* using defined assignment if the declared type of the component has a type-bound defined assignment consistent with the component, and intrinsic assignment for the dynamic type of that component otherwise.

Interp **F03/0008**, Status: *Corrigendum 2.*

Ref: 7.4.2, R736, C722, R741, C725; [143:12,24,35,37]

Four times, change "*variable*"
To "*scalar-variable*",

Making those two BNF rules and two constraints, in whole:

R736   *data-pointer-object*   **is**   *variable-name*
                                             or   *scalar-variable* % *data-pointer-component-name*

C722   (R736) A *data-pointer-component-name* shall be the name of a component of *scalar-variable* that is a data pointer.

R741   *proc-component-ref*   **is**   *scalar-variable* % *procedure-component-name*

C725   (R741) the *procedure-component-name* shall be the name of a procedure pointer component of the declared type of *scalar-variable.*

Interp **F03/0092**, Status: *Corrigendum 3.*

Ref: 7.4.2.2, penultimate paragraph, [145:5]

After "the same type"
Insert "or both be unlimited polymorphic",
Making the whole paragraph read:

> If *proc-target* and *proc-pointer-object* are functions, they shall have the same type or both be unlimited polymorphic; corresponding type parameters shall either both be deferred or both have the same value.

Interp **F03/0015**, Status: *Corrigendum 1.*

Ref: 8.1.4.3, 1st paragraph, [161:18-19]

Delete ", TARGET,", and after "the attribute." insert new sentence:
"The associating entity has the TARGET attribute if and only if the selector is a variable and has either the TARGET or POINTER attribute.",
Making the whole paragraph read:

> Within a SELECT TYPE or ASSOCIATE construct, each associating entity has the same rank as its associated selector. The lower bound of each dimension is the result of the intrinsic function LBOUND (13.7.60) applied to the corresponding dimension of *selector*. The upper bound of each dimension is one less than the sum of the lower bound and the extent. The associating entity has the ASYNCHRONOUS, ~~TARGET,~~ or VOLATILE attribute if and only if the selector is a variable and has the attribute. The associating entity has the TARGET attribute if and only if the selector is a variable and has either the TARGET or POINTER attribute. If the associating entity is polymorphic, it assumes the dynamic type and type parameter values of the selector. If the selector has the OPTIONAL attribute, it shall be present.

Interp **F03/0025 and F03/0026**, Status: *Corrigendum 2.*

Ref: 8.1.5.1, R823, C814, C815, C816, [162:17,19,20,21]

In R823 *type-guard-stmt*,
Replace the line

        or CLASS IS (*type-spec*) [*select-construct-name*]

By

        or CLASS IS (*derived-type-spec*) [*select-construct-name*]

In C814, C815 and C816 (thus three times),
After "*type-spec*"
Insert "or *derived-type-spec*",

Making the whole BNF R823 and three constraints read:

R823	*type-guard-stmt*	**is**	TYPE IS (*type-spec*) [*select-construct-name*]
		or	CLASS IS (*derived-type-spec*) [*select-construct-name*]
		or	CLASS DEFAULT [*select-construct-name*]

C814   (R823) The *type-spec* or *derived-type-spec* shall specify that each length type parameter is assumed.

C815   (R823) The *type-spec* or *derived-type-spec* shall not specify a sequence derived type or a type with the BIND attribute.

C816   (R823) If *selector* is not unlimited polymorphic, the *type-spec* or *derived-type-spec* shall specify an extension of the declared type of *selector*.

162.1

Interp **F03/0070**, Status: *Corrigendum 2.*

Ref: 9.5.1.3, 2nd and last sentences, [189:7,9]

Replace "this input/output statement"
By "a nonchild input/output statement",
Replace "from an input/output statement"
By "from a nonchild input/output statement",
And append a new sentence "A formatted child input/output statement is a nonadvancing input/output statement, and any ADVANCE= specifier is ignored.",
Making the whole paragraph read:

> The *scalar-default-char-expr* shall evaluate to YES or NO. The ADVANCE= specifier determines whether advancing input/output occurs for ~~this~~ a nonchild input/output statement. If YES is specified, advancing input/output occurs. If NO is specified, nonadvancing input/output occurs (9.2.3.1). If this specifier is omitted from ~~an~~ a nonchild input/output statement that allows the specifier, the default value is YES. A formatted child input/output statement is a nonadvancing input/output statement, and any ADVANCE= specifier is ignored.

Interp **F03/0050**, Status: *Corrigendum 3.*

Ref: 9.5.3.4, after the 7th paragraph, [196:29+]

Insert new paragraphs:

> During the execution of an output statement that specifies an internal file, no part of that internal file shall be referenced, defined, or become undefined as the result of evaluating any output list item.

> During the execution of an input statement that specifies an internal file, no part of that internal file shall be defined or become undefined as the result of transferring a value to any input list item.

196.1

Interp **F95/0096**, Status: *Corrigendum 1.*

Ref: 9.5.3.4.2, 8th paragraph, [198:12]

Replace "input item and its corresponding data edit descriptor"
By "effective input item and its corresponding data edit descriptors",
Making the whole paragraph read:

> During nonadvancing input when the pad mode has the value YES, blank characters are supplied by the processor if an effective input item and its corresponding data edit descriptors require more characters from the record than the record contains. If the record is incomplete, an end-of-file condition occurs; otherwise an end-of-record condition occurs.

198.1

Interp **F03/0070**, Status: *Corrigendum 2.*

Ref: 9.5.3.7.1, last paragraph, after the 1st bullet item, [199:8+]

Insert a new bullet item as follows:

- Any ADVANCE= specifier in a child input/output statement is ignored.

Interp **F03/0106**, Status: *Corrigendum 3.*

Ref: 9.9.1.8, 9.9.1.9, 9.9.1.12, [212:15,21,36]

NOTE: This interp also has edits on pages 213-216.

Edit subclauses 9.9.1.8, 9.9.1.9, and 9.9.1.12 as shown below.

9.9.1.8   DIRECT= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the DIRECT= specifier is assigned the value YES if DIRECT is included in the set of allowed access methods for the file, NO if DIRECT is not included in the set of allowed access methods for the file, and UNKNOWN if the processor is unable to determine whether or not DIRECT is included in the set of allowed access methods for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.9   ENCODING= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the ENCODING= specifier is assigned the value UTF-8 if the file is connected for formatted input/output with an encoding form of UTF-8, and is assigned the value UNDEFINED if the file is connected for unformatted input/output. If there is no connection, it is assigned the value UTF-8 if the processor is able to determine that the encoding form of the file is UTF-8. If the processor is unable to determine the encoding form of the file or if the unit specified by UNIT= is not connected to a file, the variable is assigned the value UNKNOWN.

> **NOTE 9.62**
>
> The value assigned may be something other than UTF-8, UNDEFINED, or UNKNOWN if the processor supports other specific encoding forms (e.g. UTF-16BE).

9.9.1.12   FORMATTED= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the FORMATTED= specifier is assigned the value YES if FORMATTED is included in the set of allowed forms for the file, NO if FORMATTED is not included in the set of allowed forms for the file, and UNKNOWN if the processor is unable to determine whether or not FORMATTED is included in the set of allowed forms for the file or if the unit specified by UNIT= is not connected to a file.

212.1

Interp **F03/0106**, Status: *Corrigendum 3.*

Ref: 9.9.1.16, 9.9.1.17, [213:15,16,20-21]

Edit subclauses 9.9.1.16 and 9.9.1.17 as shown below.

NOTE: This interp has other edits on pages 212-216.

9.9.1.16   NEXTREC= specifier in the INQUIRE statement

The *scalar-int-variable* in the NEXTREC= specifier is assigned the value $n + 1$, where n is the record number of the last record read from or written to the file connected for direct access. If the file is connected but no records have been read or written since the connection, the *scalar-int-variable* is assigned the value 1. If the file is not connected for direct access, ~~or~~ if the position of the file is indeterminate because of a previous error condition, or if the unit specified by UNIT= is not connected to a file, the *scalar-int-variable* becomes undefined. If there are pending data transfer operations for the specified unit, the value assigned is computed as if all the pending data transfers had already completed.

9.9.1.17   NUMBER= specifier in the INQUIRE statement

~~The *scalar-int-variable* in the NUMBER= specifier is assigned the value of the external unit number of the unit that is connected to the file. If there is no unit connected to the file, the value −1 is assigned.~~Execution of an INQUIRE by file statement causes the *scalar-int-variable* in the NUMBER= specifier to be assigned the value of the external unit number of the unit that is connected to the file. If there is no unit connected to the file, the value −1 is assigned. Execution of an INQUIRE by unit statement causes the *scalar-int-variable* to be assigned the value specified by UNIT=.

Interp **F03/0106**, Status: *Corrigendum 3.*

Ref: 9.9.1.21, [214:19,20]

Edit subclause 9.9.1.21 as shown below.

NOTE: This interp has other edits on pages 212-216.

9.9.1.21   POS= specifier in the INQUIRE statement

The *scalar-int-variable* in the POS= specifier is assigned the number of the file storage unit immediately following the current position of a file connected for stream access. If the file is positioned at its terminal position, the variable is assigned a value one greater than the number of the highest-numbered file storage unit in the file. If the file is not connected for stream access ~~or~~, if the position of the file is indeterminate because of previous error conditions, or if the unit specified by UNIT= is not connected to a file, the variable becomes undefined.

214.1

Interp **F03/0106**, Status: *Corrigendum 3*.

Ref: 9.9.1.23, 9.9.1.24, 9.9.1.27, 9.9.1.29, [215:2,7,26,34]

Edit subclauses 9.9.1.23, 9.9.1.24, 9.9.1.27 and 9.9.1.29 as shown below:

9.9.1.23   READ= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the READ= specifier is assigned the value YES if READ is included in the set of allowed actions for the file, NO if READ is not included in the set of allowed actions for the file, and UNKNOWN if the processor is unable to determine whether or not READ is included in the set of allowed actions for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.24   READWRITE= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the READWRITE= specifier is assigned the value YES if READ-WRITE is included in the set of allowed actions for the file, NO if READWRITE is not included in the set of allowed actions for the file, and UNKNOWN if the processor is unable to determine whether or not READWRITE is included in the set of allowed actions for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.27   SEQUENTIAL= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the SEQUENTIAL= specifier is assigned the value YES if SEQUEN-TIAL is included in the set of allowed access methods for the file, NO if SEQUENTIAL is not included in the set of allowed access methods for the file, and UNKNOWN if the processor is unable to determine whether or not SEQUENTIAL is included in the set of allowed access methods for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.29   SIZE= specifier in the INQUIRE statement

The *scalar-int-variable* in the SIZE= specifier is assigned the size of the file in file storage units. If the file size cannot be determined or if the unit specified by UNIT= is not connected to a file, the variable is assigned the value -1.

For a file that may be connected for stream access, the file size is the number of the highest-numbered file storage unit in the file.

For a file that may be connected for sequential or direct access, the file size may be different from the number of storage units implied by the data in the records; the exact relationship is processor-dependent.

NOTE: This interp has other edits on pages 212-216.

Interp **F03/0106**, Status: *Corrigendum 3.*

Ref: 9.9.1.30, 9.9.1.31, 9.9.1.32, [216:5,10,15]

Edit subclauses 9.9.1.30, 9.9.1.31 and 9.9.1.32 as shown below:

9.9.1.30   STREAM= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the STREAM= specifier is assigned the value YES if STREAM is included in the set of allowed access methods for the file, NO if STREAM is not included in the set of allowed access methods for the file, and UNKNOWN if the processor is unable to determine whether or not STREAM is included in the set of allowed access methods for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.31   UNFORMATTED= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the UNFORMATTED= specifier is assigned the value YES if UN-FORMATTED is included in the set of allowed forms for the file, NO if UNFORMATTED is not included in the set of allowed forms for the file, and UNKNOWN if the processor is unable to determine whether or not UNFORMATTED is included in the set of allowed forms for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.32   WRITE= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the WRITE= specifier is assigned the value YES if WRITE is included in the set of allowed actions for the file, NO if WRITE is not included in the set of allowed actions for the file, and UNKNOWN if the processor is unable to determine whether or not WRITE is included in the set of allowed actions for the file or if the unit specified by UNIT= is not connected to a file.

NOTE: This interp also has other edits on pages 212-215.

216.1

Interp **F95/0096**, Status: *Corrigendum 1.*

Ref: 9.10.3, list item (1), [218:6-7]

Replace "input list item (9.5.3.4.2) and corresponding data edit descriptor that requires"
By "effective input item (9.5.2) and its corresponding data edit descriptors that require",
Making the whole paragraph read:

(1)   If the pad mode has the value YES, the record is padded with blanks to satisfy the ~~input list item (9.5.3.4.2)~~ effective input item (9.5.2) and its corresponding data edit descriptors that requires more characters than the record contains. If the pad mode has the value NO, the input list item becomes undefined.

218.1

Interp **F04/0079**, Status: *Corrigendum 3.*

Ref: 10.6.1, numbered list, [227:15+]

Add new list item:

    (7)   On output of a real zero value, the digits in the exponent field shall all be zero.

Interp **F03/0028**, Status: *Corrigendum 2.*

Ref: 10.9.1, penultimate paragraph, last sentence, [240:13]

Replace "blank, comma, slash"
By "blank, comma (if the decimal edit mode is POINT), semicolon (if the decimal edit mode is COMMA), slash",
Making the whole last sentence read:

> If the delimiters are omitted, the character sequence is terminated by the first blank, comma (if the decimal edit mode is POINT), semicolon (if the decimal edit mode is COMMA), slash, or end of record; in this case apostrophes and quotation marks within the datum are not to be doubled.

240.1

Interp **F03/0101**, Status: *Corrigendum 3.*

Ref: 10.9.2, 1st paragraph, [241:5]

Append new sentences to paragraph:

> The form of the values produced by a user-defined derived type output routine invoked during list-directed output is specified by the invoked routine. This form need not be compatible with list-directed input.

NOTE: This interp also has edits on page 246.

Interp **F03/0097**, Status: *Corrigendum 3.*

Ref: 10.10, paragraph before 10.10.1, [243:5]

Replace paragraph in its entirety,
Making it read:

> A value separator for namelist formatting is ~~the same as~~ a value separator for list-directed formatting (10.9), or one or more contiguous blanks between a nonblank value and the following object designator or "!" comment initiator.

Interp **F03/0057**, Status: *Corrigendum 2.*

Ref: 10.10.1.2, 3rd paragraph, [244:8,10]

After "expanded into a sequence of scalar list items"
Delete "of intrinsic data types",
After "format specifications for the"
Delete "intrinsic",
Making the whole paragraph read:

> When the name in the input record represents an array variable or a variable of derived type, the effect is as if the variable represented were expanded into a sequence of scalar list items ~~of intrinsic data types~~, in the same way that formatted input/output list items are expanded (9.5.2). Each input value following the equals shall then be acceptable to format specifications for the ~~intrinsic~~ type of the list item in the corresponding position in the expanded sequence, except as noted in 10.10.1.3. The number of values following the equals shall not exceed the number of list items in the expanded sequence, but may be less; in the latter case, the effect is as if sufficient null values had been appended to match any remaining list items in the expanded sequence.

Interp **F03/0028**, Status: *Corrigendum 2.*

Ref: 10.10.1.3, 2nd paragraph, last sentence, [244:29,30,32,33]

After "separated by a comma",
Insert "(if the decimal edit mode is POINT) or a semicolon (if the decimal edit mode is COMMA),",
Before "The first numeric input field..."
Insert "The separator is a comma if the decimal edit mode is POINT; it is a semicolon if the decimal edit mode is COMMA.",
And in the last sentence of the paragraph, replace both occurrences of "comma" by "separator",
Making the whole paragraph read:

> When the next effective item is of type complex, the input form of the input value consists of a left parenthesis followed by an ordered pair of numeric input fields separated by a comma (if the decimal edit mode is POINT) or a semicolon (if the decimal edit mode is COMMA), and followed by a right parenthesis. The separator is a comma if the decimal edit mode is POINT; it is a semicolon if the decimal edit mode is COMMA. The first numeric input field is the real part of the complex constant and the second part is the imaginary part. Each of the numeric input fields may be preceded or followed by any number of blanks and ends of records. The end of a record may occur between the real part and the ~~comma~~ separator or between the ~~comma~~ separator and the imaginary part.

244.1

Interp **F03/0028**, Status: *Corrigendum 2.*

Ref: 10.10.1.3, 5th paragraph, last sentence, [245:4]

After "comma,"
Insert "semicolon,",
Making the whole sentence read:

> The characters blank, comma, semicolon, and slash may appear in such character sequences.

Interp **F03/0101**, Status: *Corrigendum 3.*

Ref: 10.10.2, 1st paragraph, [246:4,7]

After "and logical values"
Insert ", and output produced by user-defined derived-type output",
Add new sentences to end of paragraph,
Making the whole paragraph read:

> The form of the output produced is the same as that required for input, except for the forms of real, character, and logical values, and output produced by user-defined derived-type output. The name in the output is in upper case. With the exception of adjacent undelimited character values, the values are separated by one or more blanks or by a comma, or a semicolon if the decimal edit mode is COMMA, optionally preceded by one or more blanks and optionally followed by one or more blanks. The form of the output produced by a user-defined derived-type output routine invoked during namelist output is specified by the invoked routine. This form need not be compatible with namelist input.

NOTE: This interp also has an edit on page 241.

246.1

Interp **F03/0068**, Status: *Corrigendum 2.*

Ref: 10.10.2.2, last paragraph, [247:33]

Between "Except for" and "continuation of",
Insert "new records created by explicit formatting within a user-defined derived-type output procedure or by",
Making the whole paragraph:

> Except for new records created by explicit formatting within a user-defined derived-type output procedure or by continuation of delimited character sequences, each output record begins with a blank character.

Interp **F03/0088**, Status: *Corrigendum 3.*

Ref: 12.3.2.1.1, 1st paragraph, [262:12]

After "the second argument."
Append new sentence to paragraph:

> All restrictions and constraints that apply to actual arguments in a reference to the function also apply to the corresponding operands in the expression as if they were used as actual arguments.

NOTE: This interp also has an edit on p263.

Interp **F03/0069**, Status: *Corrigendum 2.*

Ref: 12.3.2.1.2, 2nd paragraph, 2nd sentence, [263:6]

Replace entire sentence "Each argument shall be nonoptional."
By "The dummy arguments shall be nonoptional dummy data objects.",
(See below for resulting paragraph.)

Interp **F03/0088**, Status: *Corrigendum 3.*

Ref: 12.3.2.1.2, 2nd paragraph, [263:12]

After "the second argument."
Insert the following new sentence:

> All restrictions and constraints that apply to actual arguments in a reference to the subroutine also apply to the left-hand-side and to the right-hand-side enclosed in parentheses as if they were used as actual arguments.

Together with the previous interp on this page, making the whole paragraph read:

> Each of these subroutines shall have exactly two dummy arguments. ~~Each argument shall be nonoptional.~~ The dummy arguments shall be nonoptional dummy data objects. The first argument shall have INTENT (OUT) or INTENT (INOUT) and the second argument shall have INTENT (IN). Either the second argument shall be an array whose rank differs from that of the first argument, the declared types and kind type parameters of the arguments shall not conform as specified in Table 7.8, or the first argument shall be of derived type. A defined assignment is treated as a reference to the subroutine, with the left-hand side as the first argument and the right-hand side enclosed in parentheses as the second argument. All restrictions and constraints that apply to actual arguments in a reference to the subroutine also apply to the left-hand-side and to the right-hand-side enclosed in parentheses as if they were used as actual arguments. The ASSIGNMENT generic specification specifies that assignment is extended or redefined.

Interp **F03/0044**, Status: *Corrigendum 1.*

Ref: 12.3.2.5, [266:8]

Change "referenced" to "invoked",
Making the whole paragraph read:

> In a scoping unit where the interface of a function is implicit, the type and type parameters of
> the function result are specified by an implicit or explicit type specification of the function name.
> The type, type parameters, and shape of dummy arguments of a procedure ~~referenced~~ invoked
> from a scoping unit where the interface of the procedure is implicit shall be such that the actual
> arguments are consistent with the characteristics of the dummy arguments.

Interp **F03/0016**, Status: *Corrigendum 1.*

Ref: 12.4, after constraint C1224, [266:24+]

Insert new constraint:

C1224a (R1219) If *data-ref* is an array, the referenced type-bound procedure shall have the PASS attribute.

266.1

Interp **F03/0043**, Status: *Corrigendum 1.*

Ref: 12.4.1.1, [268:17]

After "procedure" insert "or a procedure pointer component",
Making the whole paragraph read:

> In a reference to a type-bound procedure or a procedure pointer component that has a passed-object dummy argument (4.5.3.3), the *data-ref* of the *function-reference* or *call-stmt* is associated, as an actual argument, with the passed-object dummy argument.

Interp **F03/0010**, Status: *Corrigendum 1, MODIFIED.*

Ref: 12.4.1.2, 1st paragraph, [268:23]

Before "the declared"
Insert "either both the actual and dummy arguments shall be unlimited polymorphic, or",
Making the whole paragraph read:

> If a dummy argument is neither allocatable nor a pointer, it shall be type compatible (5.1.1.2) with the associated actual argument. If a dummy argument is allocatable or a pointer, the associated actual argument shall be polymorphic if and only if the dummy argument is polymorphic, and either both the actual and dummy arguments shall be unlimited polymorphic, or the declared type of the actual argument shall be the same as the declared type of the dummy argument.

268.1

Interp **F03/0005**, Status: *Corrigendum 1.*

Ref: 12.4.1.2, paragraph after Note 12.22, [270:1-2]

Replace "associated with an actual argument that is"
By "used as an actual argument that is associated with",
Making the whole paragraph read:

> If the dummy argument does not have the TARGET or POINTER attribute, any pointers associated with the actual argument do not become associated with the corresponding dummy argument on invocation of the procedure. If such a dummy argument is ~~associated with an actual argument that is~~ used as an actual argument that is associated with a dummy argument with the TARGET attribute, whether any pointers associated with the original actual argument become associated with the dummy argument with the TARGET attribute is processor dependent.

Interp **F03/0061**, Status: *Corrigendum 2.*

Ref: 12.4.1.2, 16th paragraph, [270:27]

Replace "assumed-shape or pointer"
By "assumed-shape, pointer, or polymorphic",
Making the whole paragraph read:

> If the actual argument is scalar, the corresponding dummy argument shall be scalar unless the actual argument is of type default character, of type character with the C character kind (15.1), or is an element or substring of an element of an array that is not an assumed-shape, pointer, or polymorphic array. If the procedure is nonelemental and is referenced by a generic name or as a defined operator or defined assignment, the ranks of the actual arguments and corresponding dummy arguments shall agree.

270.1

Interp **F95/0078**, Status: *Corrigendum 1.*

Ref: 12.4.4.1, end of subclause, [278:5+]

Append new list item:

(5)   If (1), (2), (3) and (4) do not apply, the name is that of an intrinsic procedure, and the reference is consistent with the interface of that intrinsic procedure, then the reference is to that intrinsic procedure.

Interp **F03/0086**, Status: *Corrigendum 3.*

Ref: 12.5.2.1, constraint C1242, [280:6-7]

Replace constraint with:

C1242   An elemental procedure shall not have the BIND attribute.

280.1

Interp **F90/000207**, Status: *Corrigendum 1.*

Ref: 13.3, last sentence, [293:5-6]

Delete the last sentence of this subclause, which currently reads:

In particular, whereas the models are identical for $w_{z-1} = 0$, they do not correspond for $w_{z-1} = 1$ and the interpretation of bits in such objects is processor dependent.

Interp **F03/0054**, Status: *Corrigendum 1.*

Ref: 13.7.37, Result Value paragraph, [316:5-6]

Replace "model representation (13.4) for the value of X"
By "representation for the value of X in the model (13.4) that has the radix of X but no limits on exponent values",
Making the whole paragraph read:

Result Value. The result has a value equal to the exponent e of the ~~model~~ representation ~~(13.4)~~ for the value of X in the model (13.4) that has the radix of X but no limits on exponent values, provided X is nonzero and e is within the range for default integers. If X has the value zero, the result has the value zero. If X is an IEEE infinity or NaN, the result has the value HUGE(0).

316.1

Interp **F03/0054**, Status: *Corrigendum 1.*

Ref: 13.7.40, Result Value paragraph, [317:8]

Replace "model ... X"
By "representation for the value of X in the model that has the radix of X but no limits on exponent values",
Making the whole paragraph read:

Result Value. The result has the value $X \times b^{-e}$, where b and e are as defined in (13.4) for the representation of X in the model that has the radix of X but no limits on exponent values. If X has the value zero, the result has the value zero. If X is an IEEE infinity, the result is that infinity. If X is an IEEE NaN, the result is that NaN.

Interp **F03/0055**, Status: *Corrigendum 1.*

Ref: 13.7.100, Result Value paragraph, [347:22]

Replace "the model representation of X."
By "the value nearest to X in the model for real values whose kind type parameter is that of X; if there are two such values, the value of greater absolute value is taken.",
Making the whole paragraph read:

Result Value. The result has the value $|Y \times b^{-e}| \times b^p$, where b, e, and p are as defined in 13.4 for the value nearest to X in the model for real values whose kind type parameter is that of X; if there are two such values, the value of greater absolute value is taken. If X is an IEEE infinity, the result is zero. If X is an IEEE NaN, the result is that NaN.

Interp **F03/0054**, Status: *Corrigendum 1.*

Ref: 13.7.107, Result Value paragraph, [351:5]

Replace "model ... X"
By "representation for the value of X in the model that has the radix of X but no limits on exponent values",
Making the whole paragraph read:

Result Value. The result has the value $X \times b^{I-e}$, where b and e are as defined in 13.4 for the representation for the value of X in the model that has the radix of X but no limits on exponent values. If X has the value zero, the result has value zero.

Interp **F03/0055**, Status: *Corrigendum 1.*

Ref: 13.7.113, Result Value paragraph, [353:9]

Replace "the model representation of X."
By "the value nearest to X in the model for real values whose kind type parameter is that of X; if there are two such values, the value of greater absolute value is taken.",
Making the whole paragraph read:

Result Value. If X does not have the value zero, the result has the value $b^{\max(e-p,\,e_{\mathrm{MIN}}-1)}$, where b, e, and p are as defined in 13.4 for the value nearest to X in the model for real values whose kind type parameter is that of X; if there are two such values, the value of greater absolute value is taken. If X has the value zero, the result is the same as that of TINY (X). If X is an IEEE infinity, the result is positive infinity. If X is an IEEE NaN, the result is that NaN.

Interp **F03/0107**, Status: *Corrigendum 3.*

Ref: 14.9.2, 1st paragraph, [370:8-9]

Edit paragraph as shown below:

> The module IEEE_ARITHMETIC contains the following elemental functions for all reals X and Y ~~for which IEEE_SUPPORT_DATATYPE(X) and IEEE_SUPPORT_DATATYPE(Y) are true~~:

370.1

Interp **F03/0023**, Status: *Corrigendum 2.*

Ref: 14.10.7, Argument paragraph, 1st sentence, [374:21]

After "shall be"
Insert "scalar and",
Making the whole paragraph read:

> GRADUAL shall be scalar and of type default logical. It is an INTENT(OUT) argument. The value is true if the current underflow mode is gradual underflow, and false if the current underflow mode is abrupt underflow.

374.1

Interp **F03/0023**, Status: *Corrigendum 2.*

Ref: 14.10.22, Argument paragraph, 1st sentence, [380:13]

After "shall be"
Insert "scalar and",
Making the whole paragraph read:

> GRADUAL shall be scalar and of type default logical. If it is true, the current underflow mode is set to gradual underflow. If it is false, the current underflow mode is set to abrupt underflow.

380.1

Interp **F03/0089**, Status: *Corrigendum 3.*

Ref: 15.2.3, 2nd paragraph, [398:9]

After "A Fortran derived type is interoperable with a C struct type if"
Insert "and only if",
Making the whole paragraph:

> A Fortran derived type is interoperable with a C struct type if and only if the derived-type
> definition of the Fortran type specifies BIND(C) (4.5.1), the Fortran derived type and the C
> struct type have the same number of components, and the components of the Fortran derived
> type have types and type parameters that are interoperable with the types of the corresponding
> components of the struct type. A component of a Fortran derived type and a component of a C
> struct type correspond if they are declared in the same relative position in their respective type
> definitions.

398.1