To: J3
From: Malcolm Cohen
Subject: Interpretation Update Pages: Standing Document 018 (14-018)
Date: 2014/06/16

This document contains insertions for every interpretation edit that has been published as a corrigendum to ISO/IEC 1539-1:2010.

The following pages are intended for insertion into a loose-leaf binder version of 10-007r1. 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: This has not been done when it might be more confusing than helpful.)

This version contains corrigenda 1 to 3.

Interp **F08/0046**, Status: *Corrigendum 1.*

Ref: Introduction, 2^{nd} paragraph, 4^{th} bullet, [*xv*]

At the end of the 4^{th} bullet (beginning "Data declaration"),
Insert the following sentence:

An array or an object with a nonconstant length type parameter can have the VALUE attribute.

Interp **F08/0042**, Status: *Corrigendum 2.*

Ref: Introduction, 2^{nd} paragraph, 5^{th} bullet, [*xv*]

In the 5^{th} bullet (beginning "Data usage and computation"),
After the second sentence ("SOURCE= in ... an expression.") insert the following sentence:

Multiple allocations are permitted in a single ALLOCATE statement with SOURCE=.

Interp **F08/0051**, Status: *Corrigendum 1.*

Ref: Introduction, 2^{nd} paragraph, last bullet, [xvi]

In the last bullet item (beginning "Programs and procedures"),
Before "An impure"
Insert the following sentence:

> An argument to a pure procedure can have default INTENT if it has the VALUE attribute.

Interp **F08/0037**, Status: *Corrigendum 1.*

Ref: Introduction, 2^{nd} paragraph, last bullet, [xvi]

In the last bullet item (beginning "Programs and procedures"),
Before "The FUNCTION and SUBROUTINE"
Insert the following sentence:

> The PROTECTED attribute can be specified by the procedure declaration statement.

NOTE: This interp also has an edit on page 287.

These two edits make the last bullet paragraph read:

- Programs and procedures:
  An empty CONTAINS section is allowed. An internal procedure can be used as an actual argument or procedure pointer target. ALLOCATABLE and POINTER attributes are used in generic resolution. Procedureness of a dummy argument is used in generic resolution. An actual argument with the TARGET attribute can correspond to a dummy pointer. A null pointer or unallocated allocatable can be used to denote an absent nonallocatable nonpointer optional argument. An argument to a pure procedure can have default INTENT if it has the VALUE attribute. An impure elemental procedure processes array arguments in array element order. The PROTECTED attribute can be specified by the procedure declaration statement. The FUNCTION and SUBROUTINE keywords can be omitted from the END statement for a module or internal subprogram. A line in the program is permitted to begin with a semicolon.

Interp **F03/0053**, Status: *Corrigendum 3*.

Ref: 1.3.147.6, [19:15-16]

Replace the definition of **extensible type** as follows.

1.3.147.6
extensible type
type that ~~has neither the BIND attribute nor the SEQUENCE attribute and which therefore~~ may be extended using the EXTENDS clause (4.5.7.1)

NOTE: This interp also has edits on pages 77 and 431.

Interp **F03/0120, F08/0011, F08/0032-0033, F08/0054-55**, Status: *Corrigenda 1 and 2.*

Ref: 1.6.2, 1^{st} paragraph, 1^{st} sentence, [24:9-10,11+]

Note: Interp F03/0120 also has edits on pages 62 and 63.

Interp F08/0032 also has an edit on page 312.

Interp F08/0033 also has an edit on pages 312 and 313.

Interp F08/0054 also has an edit on page 279.

Interp F08/0055 also has another edit on this page, and edits on pages 25 and 258.

Change the first paragraph as show below and insert new paragraphs afterwards, making the whole subclause read:

1.6.2 Fortran 2003 compatibility

1 ~~This~~Except as identified in this subclause, this part of ISO/IEC 1539 is an upward compatible extension to the preceding Fortran International Standard, ISO/IEC 1539-1:2004 (Fortran 2003). ~~Any~~Except as identified in this subclause, any standard-conforming Fortran 2003 program remains standard-conforming under this part of ISO/IEC 1539.

2 Fortran 2003 permitted a sequence type to have type parameters; that is not permitted by this part of ISO/IEC 1539.

3 Fortran 2003 specified that array constructors and structure constructors of finalizable type are finalized. This part of ISO/IEC 1539 specifies that these constructors are not finalized.

4 The form produced by the G edit descriptor for some values and some I/O rounding modes differs from that specified by Fortran 2003.

5 Fortran 2003 required an explicit interface only for a procedure that was actually referenced in the scope, not merely passed as an actual argument. This part of ISO/IEC 1539 requires an explicit interface for a procedure under the conditions listed in 12.4.2.2, regardless of whether the procedure is referenced in the scope.

6 Fortran 2003 permitted the result variable of a pure function to be a polymorphic allocatable variable, or to be finalizable by an impure final subroutine. These are not permitted by this part of ISO/IEC 1539.

7 Fortran 2003 permitted an INTENT (OUT) argument of a pure subroutine to be polymorphic; that is not permitted by this part of ISO/IEC 1539.

Interp **F08/0055**, Status: *Corrigendum 2.*

Ref: 1.6.3, [24:27+]

Append new bullet item to the list

- The form produced by the G edit descriptor with d equal to zero differs from that specified by Fortran 95 for some values.

NOTE: Interp F08/0055 also has another edit on this page, and edits on pages 25 and 258.

Interp **F08/0055**, Status: *Corrigendum 2*.

Ref: 1.6.4, [25:6]

Change the last full stop in this subclause into a semi-colon, and append a new bullet item to the last, making the entire paragraph read:

4 The following Fortran 90 features have different interpretations in this part of ISO/IEC 1539:

- the result value of the intrinsic function SIGN (when the second argument is a negative real zero);
- formatted output of negative real values (when the output value is zero);
- whether an expression is a constant expression (thus whether a variable is considered to be automatic);
- the G edit descriptor with *d* equal to zero for some values.

NOTE: Interp F08/0055 also has edits on pages 24 and 258.

Interp **F08/0078**, Status: *Corrigendum 2*.

Ref: 4.4.2.3, NOTE 4.8, [54:18+2]

Change "can distinguish" to "distinguishes", making the whole first line of NOTE 4.8 read

On a processor that ~~can~~ distinguishes between 0.0 and −0.0,

NOTE: Interp F08/0078 also has an edit on page 387.

Interp **F03/0120**, Status: *Corrigendum 2.*

Ref: 4.5.2.3, constraint C436, [62:19]

After "a sequence type," insert new condition making the whole constraint read:

C436 (R425) If SEQUENCE appears, each data component shall be declared to be of an intrinsic type or of a sequence type, the derived type shall not have type parameters, and a *type-bound-procedure-part* shall not appear.".

NOTE: Interp F03/0120 also has edits on pages 24 and 63.

Interp **F03/0120**, Status: *Corrigendum 2.*

Ref: 4.5.2.4, 2^{nd} paragraph, [63:9]

Delete "type parameters and", making the whole paragraph read:

2 Two data entities have the same type if they are declared with reference to the same derived-type definition. Data entities also have the same type if they are declared with reference to different derived-type definitions that specify the same type name, all have the SEQUENCE attribute or all have the BIND attribute, have no components with PRIVATE accessibility, and have ~~type parameters and~~ components that agree in order, name, and attributes. Otherwise, they are of different derived types. A data entity declared using a type with the SEQUENCE attribute or with the BIND attribute is not of the same type as an entity of a type that has any components that are PRIVATE.

NOTE: Interp F03/0120 also has edits on pages 24 and 62.

Interp **F08/0072**, Status: *Corrigendum 2*.

Ref: 4.5.6.1, constraint C480, [75:10]

Before "nonpointer, nonallocatable, nonpolymorphic" insert "noncoarray,", making the whole constraint read:

C480 (R452) 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 noncoarray, nonpointer, nonallocatable, nonpolymorphic variable of the derived type being defined. All length type parameters of the dummy argument shall be assumed. The dummy argument shall not have the INTENT (OUT) or VALUE attribute.

Interp **F08/0013**, Status: *Corrigendum 1.*

Ref: 4.5.6.3, 0^{th} and 9^{th} paragraphs, [76:10-,25-26]

Move paragraph 9 of the subclause and Note 4.49 to precede paragraph 1, with the following changes:

Change "the variable is"
to "if the variable is not an unallocated allocatable variable, it is",
and append new sentence to the end of the paragraph:
"If the variable is an allocated allocatable that would be deallocated by intrinsic assignment, the finalization occurs before the deallocation.".

Interp **F08/0013**, Status: *Corrigendum 1.*

Ref: 4.5.6.3, 1^{st} paragraph, [76:10]

After "it is finalized"
insert "unless it is the variable in an intrinsic assignment (7.2.1.3) or a component thereof".

Interp **F08/0081**, Status: *Corrigendum 2.*

Ref: 4.5.6.3, 1^{st} paragraph, [76:10]

Append new sentence "If an error condition occurs during deallocation, it is processor dependent whether finalization occurs."

NOTE: Interp F08/0081 also has edits on pages 131, 459, and 460.

Interp **F08/0011**, Status: *Corrigendum 1.*

Ref: 4.5.6.3, 5^{th} and 7^{th} paragraphs, [76:17-18,21-22]

Delete these paragraphs.

Interp **F03/0085, F08/0034, and F08/0070**, Status: *Corrigenda 1 and 2.*

Ref: 4.5.6.3, 8^{th} paragraph, [76:23-24]

Replace paragraph 8 with:

> When a procedure is invoked, a nonpointer, nonallocatable, INTENT (OUT) dummy argument of that procedure is finalized before it becomes undefined. The finalization caused by INTENT (OUT) is considered to occur within the invoked procedure; so for elemental procedures, an INTENT (OUT) argument will be finalized only if a scalar or elemental final subroutine is available, regardless of the rank of the actual argument.

Interp **F03/0085, F08/0011, F08/0013, F08/0034**, Status: *Corrigendum 1.*

Ref: Entire subclause 4.5.6.3

Edit subclause 4.5.6.3 as shown below. Note that the old paragraph 9 (with its NOTE) has been moved to become paragraph 1, thus renumbering paragraphs 1-8 to become 2-9.

4.5.6.3 When finalization occurs

1 When an intrinsic assignment statement is executed, if the variable is not an unallocated allocatable variable, it is finalized after evaluation of *expr* and before the definition of the variable. If the variable is an allocated allocatable that would be deallocated by intrinsic assignment, the finalization occurs before the deallocation.

NOTE 4.1

If finalization is used for storage management, it often needs to be combined with defined assignment.

2 When a pointer is deallocated its target is finalized. When an allocatable entity is deallocated, it is finalized unless it is the variable in an intrinsic assignment statement (7.2.1.3) or a component thereof. If an error condition occurs during deallocation, it is processor dependent whether finalization occurs.

3 A nonpointer, nonallocatable object that is not a dummy argument or@function result is finalized immediately before it would become undefined due to execution of a RETURN or END statement (16.6.6, item (3)).

4 A nonpointer nonallocatable local variable of a BLOCK construct is finalized immediately before it would become undefined due to termination of the BLOCK construct (16.6.6, item (22)).

5 If an executable construct references a function, the result is finalized after execution of the innermost executable construct containing the reference.

6 ~~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.~~

7 If a specification expression in a scoping unit references a function, the result is finalized before execution of the executable constructs in the scoping unit.

8 ~~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.~~

9 When a procedure is invoked, a nonpointer, nonallocatable ~~object that is an actual argument corresponding to an~~ INTENT (OUT) dummy argument of that procedure is finalized before it becomes undefined. The finalization caused by INTENT (OUT) is considered to occur within the invoked procedure; so for elemental procedures, an INTENT(OUT) argument will be finalized only if a scalar or elemental final subroutine is available, regardless of the rank of the actual argument.

10 If an object is allocated via pointer allocation and later becomes unreachable due to all pointers associated with that object having their pointer association status changed, it is processor dependent whether it is finalized. If it is finalized, it is processor dependent as to when the final subroutines are called.

Interp **F03/0053**, Status: *Corrigendum 3*.

Ref: 4.5.7.1, 1^{st} paragraph, [77:3]

Modify the first paragraph of 4.5.7.1 Concepts as follows:

1 A derived type, other than the type C_PTR or C_FUNPTR from the intrinsic module ISO_C_BINDING, that does not have the BIND attribute or the SEQUENCE attribute is an extensible type.

NOTE: This interp also has edits on pages 19 and 431.

Interp **F08/0052**, Status: *Corrigendum 1.*

Ref: 4.5.7.3, 1^{st} paragraph, [78:4]

Change "as a type-bound"
to "as an accessible type-bound",
making the whole paragraph read

1 If a specific type-bound procedure specified in a type definition has the same binding name as an accessible type-bound procedure from the parent type then the binding specified in the type definition overrides the one from the parent type.

Interp **F08/0080**, Status: *Corrigendum 2*.

Ref: 4.8, constraint C4105, after C4106, and the 2^{nd} and 3^{rd} paragraphs, [109:8-9,10+,13-14,18]

Edit constraint C4105, insert new constraint C4106a, and edit paragraphs 2 and 3 as shown below (note that C4106 and C4107 are not edited but are included in the text shown below).

C4105 (R469) If *type-spec* specifies a derived type, ~~all~~the declared type of each *ac-value* expression~~s~~ in the *array-constructor* shall be ~~of~~ that derived type and shall have the same kind type parameter values as specified by *type-spec*.

C4106 (R472) An *ac-value* shall not be unlimited polymorphic.

C4106a (R472) The declared type of an *ac-value* shall not be abstract.

C4107 (R473) The *ac-do-variable* of an *ac-implied-do* that is in another *ac-implied-do* shall not appear as the *ac-do-variable* of the containing *ac-implied-do*.

2 If *type-spec* is omitted, corresponding length type parameters of the declared type of each *ac-value* expression ~~in the array constructor~~ shall have the same value~~length type parameters~~; in this case, the declared type and type parameters of the array constructor are those of the *ac-value* expressions.

3 If *type-spec* appears, it specifies the declared type and type parameters of the array constructor. Each *ac-value* expression in the *array-constructor* shall be compatible with intrinsic assignment to a variable of this type and type parameters. Each value is converted to the type and type parameters of the *array-constructor* in accordance with the rules of intrinsic assignment (7.2.1.3).

Interp **F08/0061**, Status: *Corrigendum 2.*

Ref: 5.3.7, 1^{st} paragraph, [93:7-8]

Edit paragraph as shown below:

1 The CONTIGUOUS attribute specifies that an assumed-shape array ~~can only be associated with a contiguous effective argument~~is contiguous, or that an array pointer can only be pointer associated with a contiguous target.

Interp **F08/0040**, Status: *Corrigendum 2*.

Ref: 5.3.10, constraint C541, [97:13]

Change "An entity" to "A dummy argument of a nonintrinsic procedure", making the whole constraint read:

C541 ~~An entity~~A dummy argument of a nonintrinsic procedure with the INTENT (OUT) attribute shall not be an allocatable coarray or have a subobject that is an allocatable coarray.

NOTE: Interp F08/0040 also has edits on pages 188 and 372.

Interp **F08/0077**, Status: *Corrigendum 2.*

Ref: 5.4.7, constraint C566, [104:26-27]

Edit constraint as shown below.

C566 (R536) ~~In a *variable* that is a *data-stmt-object*, each~~A *data-stmt-object* that is a *variable* shall be a *designator*. Each subscript, section subscript, substring starting point, and substring ending point in the *variable* shall be a constant expression.

Interp **F03/0123 and F08/0015**, Status: *Corrigendum 1.*

Ref: 5.5, 4^{th} paragraph, [109:21-23]

Delete "The mapping may ... scoping unit.",
change "in the outermost inclusive scope in which it appears"
to "; if the outermost inclusive scope in which it appears is not a type definition, it is declared in that scope, otherwise it is declared in the host of that scope",
making the whole paragraph read

4 Any data entity that is not explicitly declared by a type declaration, is not an intrinsic function, is not a component, and is not accessed by use or host association is declared implicitly to be of the type (and type parameters) mapped from the first letter of its name, provided the mapping is not null. The mapping for the first letter of the data entity shall either have been established by a prior IMPLICIT statement or be the default mapping for the letter. ~~The mapping may be to a derived type that is inaccessible in the local scope if the derived type is accessible in the host scoping unit.~~ The data entity is treated as if it were declared in an explicit type declaration ~~in~~; if the outermost inclusive scope in which it appears is not a type definition, it is declared in that scope, otherwise it is declared in the host of that scope. An explicit type specification in a FUNCTION statement overrides an IMPLICIT statement for the name of the result variable of that function subprogram.

Interp **F08/0002 and F08/0079**, Status: *Corrigenda 1 and 2.*

Ref: 5.6, 5^{th} paragraph, [111:19-20]

Change "type, type parameters, and shape"
to "declared type, kind type parameters of the declared type, and rank",
making the whole paragraph read

A namelist group object shall either be accessed by use or host association or shall have its declared type, kind type parameters of the declared type, and ~~shape~~rank specified by previous specification statements or the procedure heading in the same scoping unit or by the implicit typing rules in effect for the scoping unit. If a namelist group object is typed by the implicit typing rules, its appearance in any subsequent type declaration statement shall confirm the implied type and type parameters.

Interp **F08/0014 and F08/0016**, Status: *Corrigendum 1.*

Ref: 6.5.3.3.2, 2^{nd} paragraph, [124:4-7]

Replace the bullet list with "finalized by a nonelemental final subroutine.",
making the whole paragraph read:

> An array section with a vector subscript shall not be finalized by a nonelemental final subroutine.
>
> ~~• argument associated with a dummy array that is defined or redefined,~~
> ~~• the *data-target* in a pointer assignment statement, or~~
> ~~• an internal file.~~

NOTE: Interp F08/0014 also has an edit on page 295.

Interp **F08/0039**, Status: *Corrigendum 1.*

Ref: 6.5.4.4.2, 3^{rd} paragraph, [124:9]

Edit the paragraph as follows:

> If a vector subscript has two or more elements with the same value, an array section with that vector subscript ~~shall not appear in a variable definition context (16.6.7)~~ is not definable and shall not be defined or become undefined.

Interp **F08/0042**, Status: *Corrigendum 2*.

Ref: 6.7.1.1, constraint C633, [126:31-33]

Split constraint into two constraints and edit as shown below.

C633 (R626) If an *allocate-object* is an array, either *allocate-shape-spec-list* shall appear in its *allocation*, or *source-expr* shall appear in the ALLOCATE statement and have the same rank as the *allocate-object*.

C633a (R626) If *allocate-object* is scalar, *allocate-shape-spec-list* shall not appear.

NOTE: Interp F08/0042 also has edits on pages *xv*, 127, and 128.

Interp **F08/0042**, Status: *Corrigendum 2*.

Ref: 6.7.1.1, constraint C639 and 4^{th} paragraph, [127:5,18-19]

Replace the whole of constraint C639 with the following.

C639 (R626) If *source-expr* appears, the kind type parameters of each *allocate-object* shall have the same values as the corresponding type parameters of *source-expr*.

Edit the 4^{th} paragraph as follows:

4 If an *allocate-object* is a coarray, the ALLOCATE statement shall not have a *source-expr* ~~shall not have~~with a dynamic type of C_PTR, C_FUNPTR, or LOCK_TYPE, or ~~have~~which has a subcomponent whose dynamic type is LOCK_TYPE.

NOTE: Interp F08/0042 also has edits on pages *xv*, 126, and 128.

Interp **F08/0042 and F08/0056**, Status: *Corrigendum 2.*

Ref: 6.7.2.3, 7^{th} paragraph, [128:24-26]

Edit paragraph 7 as follows.

7 If SOURCE= appears, *source-expr* shall be conformable with *allocation*. If the value of a nondeferred length type parameter of *allocate-object* is different from the value of the corresponding type parameter of *source-expr*, an error condition occurs. If an *allocate-object* is not polymorphic and the *source-expr* is polymorphic with a dynamic type that differs from its declared type, the value provided for that *allocate-object* is the ancestor component of the *source-expr* that has the type of the *allocate-object*; otherwise, the value provided is the value of the *source-expr*. On successful allocation, if *allocate-object* and *source-expr* have the same rank the value of *allocate-object* becomes that of ~~*source-expr*~~the value provided, otherwise the value of each element of *allocate-object* becomes that of ~~*source-expr*~~the value provided. The *source-expr* is evaluated exactly once for each execution of an ALLOCATE statement.

NOTE: Interp F08/0042 also has edits on pages *xv*, 126, and 127.

Interp **F08/0010**, Status: *Corrigendum 1.*

Ref: 6.7.3.2, 1^{st} paragraph, [130:23]

Append new sentence to the end of the paragraph:

> An allocatable variable shall not be deallocated if it or any subobject of it is argument associated with a dummy argument or construct associated with an associate name.

NOTE: This interp also has an edit on page 131.

Interp **F08/0081**, Status: *Corrigendum 2.*

Ref: 6.7.3.2, 8^{th} paragraph, [131:12]

Append new sentence to the end of the paragraph:

> If an error condition occurs during deallocation, it is processor dependent whether an allocated allocatable subobject is deallocated.

NOTE: Interp F08/0081 also has edits on pages 76, 459, and 460.

Interp **F08/0010**, Status: *Corrigendum 1.*

Ref: 6.7.3.3, 1^{st} paragraph, [131:27]

Append new sentence to the end of the paragraph:

> A pointer shall not be deallocated if its target or any subobject thereof is argument associated with a dummy argument or construct associated with an associate name.

NOTE: Interp F08/0010 also has an edit on page 130.

Interp **F08/0050**, Status: *Corrigendum 1.*

Ref: 7.1.11, 9^{th} paragraph, [151:13-15]

Edit the paragraph as follows:

~~If~~ A generic entity referenced in a specification expression in the *specification-part* of a ~~module or submodule includes a reference to a generic entity, that generic entity~~ scoping unit shall have no specific procedures defined in ~~the module or submodule~~ that scoping unit, or its host scoping unit, subsequent to the specification expression.

NOTE: This interp also has an edit on page 152.

Interp **F08/0066**, Status: *Corrigendum 2.*

Ref: 7.1.12, 1^{st} paragraph, [152:4,6+]

Edit list item (6) and insert new list item after item (7) as follows:

(6) a reference to a transformational standard intrinsic function other than COMMAND_ARGUMENT_COUNT, NULL, NUM_IMAGES, THIS_IMAGE, or TRANSFER, where each argument is a constant expression,

(7) A reference to the intrinsic function NULL that does not have an argument with a type parameter that is assumed or is defined by an expression that is not a constant expression,

(7a) a reference to the intrinsic function TRANSFER where each argument is a constant expression and each ultimate pointer component of the SOURCE argument is disassociated,

NOTE: The editor fixed the grammar in item 7a by changing "A" to "a" and the final full stop to a comma.

Interp **F08/0050**, Status: *Corrigendum 1.*

Ref: 7.1.12, 3^{rd} paragraph, [152:26-28]

Edit the paragraph as follows:

~~If~~ A generic entity referenced in a constant expression in the *specification-part* of a ~~module or submodule includes a reference to a generic entity, that generic entity~~ scoping unit shall have no specific procedures defined in ~~the module or submodule~~ that scoping unit, or its host scoping unit, subsequent to the constant expression.

NOTE: This interp also has an edit on page 151.

Interp **F08/0060**, Status: *Corrigendum 2*.

Ref: 7.2.2.2, constraint C729, [158:33-159:2]

Edit constraint C729 as follows.

C729 (R740) A *procedure-name* shall be the name of an internal, module, or dummy procedure, a procedure pointer, a specific intrinsic function listed in 13.6 and not marked with a bullet (•), or an external procedure that is accessed by use or host association, ~~and is~~ referenced in the scoping unit as a procedure, or that has the EXTERNAL attribute~~, or a specific intrinsic function listed in 13.6 and not marked with a bullet (•)~~.

NOTE: The effects of this edit include the first two lines of page 159.

Interp **F08/0028**, Status: *Corrigendum 1.*

Ref: 8.1.6.6.4, 1^{st} paragraph, [177:28-29]

In the 4^{th} bullet item,
change "Control is transferred from"
to "A branch occurs",
making that bullet item:

- ~~Control is transferred from~~A branch occurs within the range of a DO construct and the branch target statement is neither the *end-do* nor within the range of the same DO construct.

Interp **F08/0023**, Status: *Corrigendum 1.*

Ref: 8.1.6.7, 1^{st} paragraph, [178:8-9]

Edit the 2^{nd} bullet item as follows:

- A pointer that is ~~referenced~~used in an iteration other than as the pointer in pointer assignment, allocation, or nullification, either shall be previously ~~pointer associated during~~pointer-assigned, allocated, or nullified in that iteration~~,~~ or shall not have its pointer association changed during any iteration. A pointer that has its pointer association changed in more than one iteration has an association status of undefined when the construct terminates.

Interp **F08/0025**, Status: *Corrigendum 1.*

Ref: 8.1.6.7, 1^{st} paragraph, [178:13-14]

Edit the 3^{rd} bullet item (replacing the 2^{nd} sentence) as follows:

- An allocatable object that is allocated in more than one iteration shall be subsequently deallocated during the same iteration in which it was allocated. ~~An object that is allocated or deallocated in only one iteration shall not be deallocated, allocated, referenced, defined, or become undefined in a different iteration.~~ An allocatable object that is referenced, defined, deallocated, or has its allocation status, dynamic type, or a deferred type parameter value inquired about, in any iteration, either shall be previously allocated in that iteration or shall not be allocated or deallocated in any other iteration.

Interp **F08/0022**, Status: *Corrigendum 1.*

Ref: 8.1.6.7, 1^{st} paragraph, [178:15-16]

Edit the 4^{th} bullet item as follows:

- ~~An input/output statement shall not write data to a file record or position in one iteration and read from the same record or position in a different iteration.~~ If data are written to a file record or position in one iteration, that record or position in that file shall not be read from or written to in a different iteration.

Interp **F08/0022**, Status: *Corrigendum 1.*

Ref: 8.1.6.7, 1^{st} paragraph, [178:17-18+]

Delete the 5^{th} bullet item ("Records written ... order.") and make a new paragraph after the list, as follows:

> ~~• Records written by output statements in the range of the loop to a sequential access file appear in the file in an indeterminate order.~~
>
> If records are written to a file connected for sequential access by more than one iteration, the ordering between records written by different iterations is indeterminate.

Interp **F08/0040**, Status: *Corrigendum 2.*

Ref: 8.5.1, 2^{nd} paragraph, [188:23+]

Insert a new bullet item before the penultimate bullet item ("STOP statement") in the list:

- a CALL statement that invokes the intrinsic subroutine MOVE_ALLOC with coarray arguments;

NOTE: Interp F08/0040 also has edits on pages 97 and 372.

Interp **F03/0048**, Status: *Corrigendum 1.*

Ref: 9.6.4.8, 25^{th} and 26^{th} paragraphs, [227:15,17-18]

NOTE: This interp also has an edit on page 487.

In the 25^{th} paragraph, delete "record positioning".

In the 26^{th} paragraph,
change "A record positioning edit descriptor, such as TL and TR,"
to "The edit descriptors T and TL", and
change "record position" to "file position" twice,
making those two paragraphs read:

> Because a child data transfer statement does not position the file prior to data transfer, the child data transfer statement starts transferring data from where the file was positioned by the parent data transfer statement's most recently processed effective item or ~~record positioning~~ edit descriptor. This is not necessarily at the beginning of a record.
>
> ~~A record positioning edit descriptor, such as~~The edit descriptors TL and TR~~,~~ used on `unit` by a child data transfer statement shall not cause the ~~record~~file position to be positioned before the ~~record~~file position at the time the defined input/output procedure was invoked.

Interp **F08/0096**, Status: *Corrigendum 2*.

Ref: 9.12, 5^{th} paragraph, [243:3-5]

Edit the paragraph as shown below.

5 The value of a specifier in an input/output statement shall not depend ~~on any *input-item*, *io-implied-do do-variable*, or~~ on the definition or evaluation of any other specifier in the *io-control-spec-list* or *inquire-spec-list* in that statement. The value of an *internal-file-variable* or of a FMT=, ID=, IOMSG=, IOSTAT= or SIZE= specifier shall not depend on the values of any *input-item* or *io-implied-do do-variable* in the same statement.

Interp **F08/0030**, Status: *Corrigendum 1.*

Ref: 10.3.1, [246:15+]

After constraint C1002, add a new constraint:

C1002A (R1005) An *unlimited-format-item* shall contain at least one data edit descriptor.

NOTE: This interp also has an edit on page 249.

Interp **F08/0030**, Status: *Corrigendum 1.*

Ref: 10.4, 7^{th} and 8^{th} paragraphs, [249:11+,19-20]

Between the 7^{th} and 8^{th} paragraphs, insert a new paragraph 7a:

> If format control encounters the rightmost parenthesis of an unlimited format item, format control reverts to the leftmost parenthesis of that unlimited format item. This reversion of format control has no effect on the changeable modes (9.5.2).

In the last sentence of the 8^{th} paragraph, change "If ..., the" to "The", making the whole paragraph read:

> If format control encounters the rightmost parenthesis of a complete format specification and another effective item is not specified, format control terminates. However, if another effective item is specified, format control then reverts to the beginning of the format item terminated by the last preceding right parenthesis that is not part of a DT edit descriptor. If there is no such preceding right parenthesis, format control reverts to the first left parenthesis of the format specification. If any reversion occurs, the reused portion of the format specification shall contain at least one data edit descriptor. If format control reverts to a parenthesis that is preceded by a repeat specification, the repeat specification is reused. Reversion of format control, of itself, has no effect on the changeable modes (9.5.2). ~~If format control reverts to a parenthesis that is not the beginning of an *unlimited-format-item*, the~~The file is positioned in a manner identical to the way it is positioned when a slash edit descriptor is processed (10.8.2).

NOTE: This interp also has an edit on page 246.

Interp **F03/0100**, Status: *Corrigendum 3.*

Ref: 10.7.2.3.2, 7^{th} and 8^{th} paragraphs, [252:33-34,37]

Replace the last sentence of paragraph 7 and edit paragraph 8 as follows.

7 For an internal value that is an IEEE infinity, the output field consists of blanks, if necessary, followed by a minus sign for negative infinity or an optional plus sign otherwise, followed by the letters 'Inf' or 'Infinity', right justified within the field. ~~If w is less than 3, the field is filled with asterisks; otherwise, if w is less than 8, 'Inf' is produced.~~ The minimum field width required for output of the form 'Inf' is 3 if no sign is produced, and 4 otherwise. If w is greater than zero but less than the minimum required, the field is filled with asterisks. The minimum field width for output of the form 'Infinity' is 8 if no sign is produced and 9 otherwise. If w is greater than or equal to the minimum required for the form 'Infinity', the form 'Infinity' is output. If w is zero or w is less than the minimum required for the form 'Infinity' and greater than or equal to the minimum required for the form 'Inf', the form 'Inf' is output. Otherwise, the field is filled with asterisks.

8 For an internal value that is an IEEE NaN, the output field consists of blanks, if necessary, followed by the letters 'NaN' and optionally followed by one to $w - 5$ alphanumeric processor-dependent characters enclosed in parentheses, right justified within the field. If w is greater than zero and less than 3, the field is filled with asterisks. If w is zero, the output field is 'NaN'.

Interp **F08/0055**, Status: *Corrigendum 2.*

Ref: 10.7.5.2.2, after 2^{nd} paragraph and in the 4^{th} paragraph, [258:13+,15-20+8]

NOTE from the editor: the line number references for this edit were wrong in the interp.

Insert new paragraph after the second paragraph and replace the 4^{th} paragraph, including its two internal tables, as shown below; note that the previously-3^{rd} (now 4^{th}) paragraph is not edited but is shown below. (No markup for the edits as those for the previously-4^{th} paragraph are too extensive.)

3 If d is zero, kPEw.0 or kPEw.0Ee editing is used for Gw.0 editing or Gw.0Ee editing respectively.

4 For an internal value that is an IEEE infinity or NaN, the form of the output field for the G$w.d$ and G$w.d$ Ee edit descriptors is the same as for F$w.d$, and the form of the output field for the G0 and G0.d edit descriptors is the same as for F0.0.

5 Otherwise, the method of representation in the output field depends on the magnitude of the internal value being edited. If the internal value is a zero value, let s be one. If the internal value is a number other than zero, let N be the decimal value that is the result of converting the internal value to d significant digits according to the I/O rounding mode and let s be the integer such that $10^{s-1} \leq N < 10^s$. If $s < 0$ or $s > d$, kPE$w.d$ or kPE$w.d$Ee editing is used for G$w.d$ editing or G$w.d$Ee editing respectively, where k is the scale factor (10.8.5). If $0 \leq s \leq d$, the scale factor has no effect and F(w-n).(d-s),n('b') editing is used where b is a blank and n is 4 for G$w.d$ editing and e+2 for G$w.d$Ee editing.

NOTE: Interp F08/0055 also has edits on pages 24 and 25.

Interp **F08/0054**, Status: *Corrigendum 2.*

Ref: 12.4.2.2, 1^{st} paragraph, [279:19]

Replace the first line of this paragraph "A procedure ... referenced and" by

> Within the scope of a procedure identifier, the procedure shall have an explicit interface if it is not a statement function and

NOTE: Interp F08/0054 also has an edit on page 24.

Interp **F03/0019**, Status: *Corrigendum 2*.

Ref: 12.4.3.2, constraint C1209, [281:11-12]

Replace the whole of constraint C1209 with the following:

C1209 (R1201) An *interface-specification* in a generic interface block shall not specify a procedure that is specified previously in any accessible interface with the same generic identifier.

Interp **F08/0001**, Status: *Corrigendum 1.*

Ref: 12.4.3.4.5, 3^{rd} paragraph, 3^{rd} bullet item, [286:4]

After "the other has the POINTER attribute"
insert "and not the INTENT (IN) attribute",
making the whole paragraph (excluding the constraints that follow it) read:

Two dummy arguments are distinguishable if

- one is a procedure and the other is a data object,
- they are both data objects or known to be functions, and neither is TKR compatible with the other,
- one has the ALLOCATABLE attribute and the other has the POINTER attribute and not the INTENT (IN) attribute, or
- one is a function with nonzero rank and the other is not known to be a function.

Interp **F08/0053 and F08/0082**, Status: *Corrigenda 1 and 2.*

Ref: 12.4.3.4.5, constraint C1214 and 5^{th} paragraph, [286:12-13,38]

In constraint C1214, change "two ... identifier"
to "if two procedures have that generic identifier, their `dtv` arguments (9.6.4.8.3)",
making the whole constraint read:

C1214 Within the scope of a *defined-io-generic-spec*, if two procedures ~~with~~have that generic identifier, their `dtv` arguments (9.6.4.8.3) shall be distinguishable.

In the 5*th* paragraph, change "applies to" to "is consistent with",
making the whole paragraph read:

> Within the scope of a generic name that is the same as the generic name of an intrinsic procedure, the intrinsic procedure is not accessible by its generic name if the procedures in the interface and the intrinsic procedure are not all functions or not all subroutines. If a generic invocation ~~applies to~~is consistent with both a specific procedure from an interface and an accessible intrinsic procedure, it is the specific procedure from the interface that is referenced.

Interp **F08/0037**, Status: *Corrigendum 1.*

Ref: 12.4.3.6, BNF rule R1213, [287:15+]

After the production "**or** POINTER"
insert a new production "**or** PROTECTED",
making the whole rule read:

R1213	*proc-attr-spec*	**is** *access-spec*
		or *proc-language-binding-spec*
		or INTENT (*intent-spec*)
		or OPTIONAL
		or POINTER
		or PROTECTED
		or SAVE

NOTE: This interp also has an edit on page 16.

Interp **F03/0064**, Status: *Corrigendum 3*.

Ref: 12.4.3.6, 2^{nd} paragraph, [288:3]

Append new sentence, making the whole paragraph read

2 If *proc-interface* appears and consists of *interface-name*, it specifies an explicit specific interface (12.4.3.2) for the declared procedures or procedure pointers. The abstract interface (12.4) is that specified by the interface named by interface-name. The interface specified by *interface-name* shall not depend on any characteristic of a procedure identified by a *procedure-entity-name* in the *proc-decl-list* of the same procedure declaration statement.

Interp **F08/0068**, Status: *Corrigendum 2*.

Ref: 12.5.2.3, 2^{nd} paragraph, [292:16]

Edit the paragraph as shown below.

2 If a nonpointer dummy argument without the VALUE attribute corresponds to a pointer actual argument that is pointer associated with a target,

- if the dummy argument is polymorphic, it becomes argument associated with that target;
- if the dummy argument is nonpolymorphic, it becomes argument associated with the declared type part of that target.

Interp **F08/0067**, Status: *Corrigendum 2.*

Ref: 12.5.2.4, 2^{nd} paragraph, [293:5]

Append new sentence to the paragraph, making the whole paragraph read:

2 The dummy argument shall be type compatible with the actual argument. If the actual argument is a polymorphic coindexed object, the dummy argument shall not be polymorphic. If the actual argument is a polymorphic assumed-size array, the dummy argument shall be polymorphic.

Interp **F03/0103**, Status: *Corrigendum 2.*

Ref: 12.5.2.4, 3^{rd} and 4^{th} paragraphs, [293:6,10]

Edit these paragraphs as shown below.

3 The kind type parameter values of the actual argument shall agree with the corresponding ones of the dummy argument. The length type parameter values of ~~the~~a present actual argument shall agree with the corresponding ones of the dummy argument that are not assumed, except for the case of the character length parameter of a default character or character with the C character kind (15.2.2) actual argument associated with a dummy argument that is not assumed shape.

4 If a present scalar dummy argument is default character or of type character with the C character kind, the length *len* of the dummy argument shall be less than or equal to the length of the actual argument. The dummy argument becomes associated with the leftmost *len* characters of the actual argument. If an array dummy argument is default character or of type character with the C character kind and is not assumed shape, it becomes associated with the leftmost characters of the actual argument element sequence (12.5.2.11).

Interp **F08/0069**, Status: *Corrigendum 2.*

Ref: 12.5.2.4, 17^{th} paragraph, [294:40,42-295:2]

After "has INTENT (OUT), the"
change "actual argument" to "effective argument",
and delete the last sentence "If ... undefined.",
making the whole paragraph read:

17 If a dummy argument has INTENT (OUT) or INTENT (INOUT), the ~~actual~~effective argument shall be definable. If a dummy argument has INTENT (OUT), the actual argument becomes undefined at the time the association is established, except for direct components of an object of derived type for which default initialization has been specified. ~~If the dummy argument is not polymorphic and the type of the effective argument is an extension type of the type of the dummy argument, only the part of the effective argument that is of the same type as the dummy argument becomes undefined.~~

NOTE: The edit for Interp F08/0069 on page 294 also deletes lines 1-2 of this page.

Interp **F08/0014**, Status: *Corrigendum 1.*

Ref: 12.5.2.4, 18^{th} paragraph, [295:3]

Between "If" and "the actual argument is an array section having a vector subscript",
insert "the procedure is nonelemental and",
making the paragraph (excluding the notes and constraints that follow it) read:

18 If the procedure is nonelemental and the actual argument is an array section having a vector subscript, the dummy argument is not definable and shall not have the ASYNCHRONOUS, INTENT (OUT), INTENT (INOUT), or VOLATILE attributes.

NOTE: Interp F08/0014 also has an edit on page 124.

Interp **F08/0059**, Status: *Corrigendum 2.*

Ref: 12.5.2.5, 1^{st} paragraph, [295:16-17]

Edit the paragraph as shown below:

1 The requirements in this subclause apply to an actual argument~~s~~ with the ALLOCATABLE or POINTER attribute that corresponds to ~~either allocatable or pointer~~a dummy data ~~objects~~argument with the same attribute.

NOTE: Interp F08/0059 also has an edit on page 296.

Interp **F08/0059**, Status: *Corrigendum 2.*

Ref: 12.5.2.5 4^{th} paragraph, 12.5.2.6, 12.5.2.7, [296:4-5,12+,35]

Move the 4^{th} paragraph of 12.5.2.5 "The values of assumed type parameters ... effective argument.", to follow the 3^{rd} paragraph of 12.5.2.6 ("The corank of the actual ... dummy argument."), and append a copy of it to the 3^{rd} paragraph of 12.5.2.7 ("The nondeferred ... shall agree.").

NOTE: Interp F08/0059 also has an edit on page 295.

Interp **F08/0048**, Status: *Corrigendum 2*.

Ref: 12.5.2.8, 2^{nd} paragraph, [297:5]

Edit the paragraph as follows.

2 If the dummy argument is an array coarray that has the CONTIGUOUS attribute or is not of assumed shape, the corresponding actual argument shall be simply contiguous or an element of a simply contiguous array.

Interp **F08/0058**, Status: *Corrigendum 2*.

Ref: 12.6.2.6, 8^{th} and 9^{th} paragraphs, [310:20,23]

Append new sentences to the ends of these paragraphs as shown below.

8 In a subprogram, a dummy argument specified in an ENTRY statement shall not appear in an executable statement preceding that ENTRY statement, unless it also appears in a FUNCTION, SUBROUTINE, or ENTRY statement that precedes the executable statement. A name that appears as a *result-name* in an ENTRY statement shall not appear in any executable statement that precedes the first RESULT clause with that name.

9 In a subprogram, a name that appears as a dummy argument in an ENTRY statement shall not appear in the expression of a statement function unless the name is also a dummy argument of the statement function, appears in a FUNCTION or SUBROUTINE statement, or appears in an ENTRY statement that precedes the statement function statement. A name that appears as a *result-name* in an ENTRY statement shall not appear in the expression of a statement function that precedes the first RESULT clause with that name unless the name is also a dummy argument of that statement function.

Interp **F08/0065**, Status: *Corrigendum 2.*

Ref: 12.7, 1^{st} paragraph, [312:12+]

Insert new bullet item after the first one in the list:

- a module procedure in an intrinsic module, if it is specified to be pure,

NOTE: Interp F08/0065 also has an edit on page 397.

Interp **F08/0032**, Status: *Corrigendum 2.*

Ref: 12.7, after constraint C1276, [312:19+]

Insert new constraints:

C1276a The result variable of a pure function shall not be such that finalization of a reference to the function would reference an impure procedure.

C1276b A pure function shall not have a polymorphic allocatable result variable.

NOTE: Interp F08/0032 also has an edit on page 24.

Interp **F08/0031**, Status: *Corrigendum 2.*

Ref: 12.7, after constraint C1277, [312:21+]

Insert new constraint:

C1277a An INTENT (OUT) argument of a pure procedure shall not be such that finalization of the actual argument would reference an impure procedure.

Interp **F08/0033**, Status: *Corrigendum 1.*

Ref: 12.7, after Note 12.47, [312:23+]

Insert new constraint

C1278a An INTENT (OUT) dummy argument of a pure procedure shall not be polymorphic.

NOTE: This interp also has edits on pages 24 and 313.

Interp **F08/0033**, Status: *Corrigendum 1.*

Ref: 12.7, after constraint C1284, [313:4+]

Insert new constraint and new note:

C1284a A statement that might result in the deallocation of a polymorphic entity is not permitted in a pure procedure.

NOTE 12.48x

Apart from the DEALLOCATE statement, this includes intrinsic assignment if the variable has a polymorphic allocatable component at any level of component selection that does not involve a pointer component but which might involve one or more allocatable components.

NOTE: This interp also has edits on pages 24 and 312.

Interp **F08/0049**, Status: *Corrigendum 1.*

Ref: 12.8.1, constraint C1290, [314:4-5]

In C1290, delete ", and shall not ... expression", making the whole constraint read:

C1290 The result variable of an elemental function shall be scalar~~,~~ and shall not have the POINTER or ALLOCATABLE attribute~~, and shall not have a type parameter that is defined by an expression that is not a constant expression~~.

NOTE: The editor fixed the grammar in the preceding by changing a comma to "and".

Interp **F08/0024 and F08/0049**, Status: *Corrigendum 1.*

Ref: 12.8.1, after constraint C1290, [314:5+]

Insert new constraints as follows.

C1290a The *specification-part* of an elemental subprogram shall specify the intents of all of its dummy arguments that do not have the VALUE attribute.

C1290b In the *specification-expr* that specifies a type parameter value of the result of an elemental function, an object designator with a dummy argument of the function as the base object shall appear only as the subject of a specification inquiry, and that specification inquiry shall not depend on a property that is deferred.

Interp **F08/0018**, Status: *Corrigendum 1.*

Ref: 12.8.1, 12.8.2, 12.8.3

Insert new paragraph at the end of 12.8.1 [314:5+]:

> In a reference to an elemental procedure, if any argument is an array, all actual arguments that correspond to INTENT (OUT) or INTENT (INOUT) dummy arguments shall be arrays. All actual arguments shall be conformable.

In 12.8.2 [314:9-10], delete the sentence beginning "For those elemental,
making the whole paragraph read:

> If a generic name or a specific name is used to reference an elemental function, the shape of the result is the same as the shape of the actual argument with the greatest rank. If there are no actual arguments or the actual arguments are all scalar, the result is scalar. ~~For those elemental functions that have more than one argument, all actual arguments shall be conformable.~~ In the array case, the values of the elements, if any, of the result are the same as would have been obtained if the scalar function had been applied separately, in array element order, to corresponding elements of each array actual argument.

In 12.8.3 [314:14-17] delete the sentence beginning "In a reference",
making the whole paragraph read:

> An elemental subroutine has only scalar dummy arguments, but may have array actual arguments. ~~In a reference to an elemental subroutine, either all actual arguments shall be scalar, or all actual arguments corresponding to INTENT (OUT) and INTENT (INOUT) dummy arguments shall be arrays of the same shape and the remaining actual arguments shall be conformable with them.~~ In the case that the actual arguments corresponding to INTENT (OUT) and INTENT (INOUT) dummy arguments are arrays, the values of the elements, if any, of the results are the same as would be obtained if the subroutine had been applied separately, in array element order, to corresponding elements of each array actual argument.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.2.4, 1^{st} sentence, 2^{nd} sentence, [316:24-25]

Change "an optional" to "a",
and change ", if present, specifies" to "specify",
making the whole paragraph read:

> Some array intrinsic functions are "reduction" functions; that is, they reduce the rank of an array by collapsing one dimension (or all dimensions, usually producing a scalar result). These functions have ~~an optional~~ a DIM argument that~~, if present,~~ can specifies the dimension to be reduced. The DIM argument of a reduction function is not permitted to be an optional dummy argument.

NOTE: This interp also has edits on pages 319, 322, 323, 328, 329, 338, 360, 374, 377, 392, 394 and 395.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.5, Table 13.1, [319]

In the table lines for ALL and ANY,
change "(MASK [, DIM])" to "(MASK) or (MASK, DIM)",
making those lines of the table read:

Table 13.1: **Standard generic intrinsic procedure summary**

Procedure	Arguments	Class	Description
...			
ALL	(MASK) or (MASK, DIM)	T	Reduce logical array by AND operation.
...			
ANY	(MASK) or (MASK, DIM)	T	Reduce logical array with OR operation.
...			

NOTE: This interp also has edits on pages 316, 322, 323, 328, 329, 338, 360, 374, 377, 392, 394 and 395.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.5, Table 13.1, [322]

In the table line for NORM2,
change "(X [, DIM])" to "(X) or (X, DIM)",
and in the table line for PARITY,
change "(MASK [, DIM])" to "(MASK) or (MASK, DIM)",
making those lines of the table read:

Table 13.1: **Standard generic intrinsic procedure summary**

Procedure	Arguments	Class	Description
...			
NORM2	(X) or (X, DIM)	T	L_2 norm of an array.
...			
PARITY	(MASK) or (MASK, DIM)	T	Reduce array with .NEQV. operation.
...			

NOTE: This interp also has edits on pages 316, 319, 323, 328, 329, 338, 360, 374, 377, 392, 394 and 395.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.5, Table 13.1, [323]

In the second table line for THIS_IMAGE,
change "(COARRAY [, DIM])" to "(COARRAY) or (COARRAY, DIM)",
making the line of the table for that function read:

Table 13.1: **Standard generic intrinsic procedure summary**

Procedure	Arguments	Class	Description
...			
THIS_IMAGE	( )	T	Index of the invoking image.
THIS_IMAGE	(COARRAY) or (COARRAY, DIM)	T	Cosubscript(s) for this image.
...			

NOTE: This interp also has edits on pages 316, 319, 322, 328, 329, 338, 360, 374, 377, 392, 394 and 395.

Interp **F08/0008**, Status: *Corrigendum 2.*

Ref: 13.7.1, 2^{nd} paragraph, [325:7-12]

Break the paragraph in two and edit as shown below.

2 The types and type parameters of standard intrinsic procedure arguments and function results are determined by these specifications. The "Argument(s)" paragraphs specify requirements on the actual arguments of the procedures. The result characteristics are sometimes specified in terms of the characteristics of dummy arguments. A program ~~is prohibited from invoking~~shall not invoke an intrinsic procedure under circumstances where a value to be ~~returned in~~assigned to a subroutine argument or returned as a function result is ~~outside the range of values~~not representable by objects of the specified type and type parameters~~, unless the intrinsic module IEEE_ARITHMETIC (clause 14) is accessible and there is support for an infinite or a NaN result, as appropriate. If an infinite result is returned~~

3 If an IEEE infinity is assigned or returned by an intrinsic procedure, the intrinsic module IEEE_ARITHMETIC is accessible, and the actual arguments were finite numbers, the flag IEEE_OVERFLOW or IEEE_DIVIDE_BY_ZERO shall signal~~; if a NaN result is returned~~. If an IEEE NaN is assigned or returned, the actual arguments were finite numbers, the intrinsic module IEEE_ARITHMETIC is accessible, and the exception IEEE_INVALID is supported, the flag IEEE_INVALID shall signal. ~~Otherwise~~ If no IEEE infinity or NaN is assigned or returned, these flags shall have the same status as when the intrinsic procedure was invoked.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.7.10, [328:2,7,10]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 329, 338, 360, 374, 377, 392, 394 and 395.

Change the subclause heading to "ALL (MASK, DIM) or ALL (MASK)",
in paragraph 3, DIM argument, delete "(optional)",
in paragraph 4, change "is absent" to "does not appear",
making the whole subclause read:

13.7.10 ALL (MASK, DIM) or ALL (MASK)

1 **Description.** Reduce logical array by AND operation.

2 **Class.** Transformational function.

3 **Arguments.**

MASK shall be a logical array.

DIM shall be an integer scalar with value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of MASK. The corresponding actual argument shall not be an optional dummy argument.

4 **Result Characteristics.** The result is of type logical with the same kind type parameter as MASK. It is scalar if DIM does not appear or $n = 1$; otherwise, the result has rank $n - 1$ and shape $[d_1, d_2, \ldots, d_{\text{DIM}-1}, d_{\text{DIM}+1}, \ldots, d_n]$ where $[d_1, d_2, \ldots, d_n]$ is the shape of MASK.

5 **Result Value.**

Case (i): The result of ALL (MASK) has the value true if all elements of MASK are true or if MASK has size zero, and the result has value false if any element of MASK is false.

Case (ii): If MASK has rank one, ALL (MASK, DIM) is equal to ALL (MASK). Otherwise, the value of element $(s_1, s_2, \ldots, s_{\text{DIM}-1}, s_{\text{DIM}+1}, \ldots, s_n)$ of ALL (MASK, DIM) is equal to ALL (MASK $(s_1, s_2, \ldots, s_{\text{DIM}-1}, :, s_{\text{DIM}+1}, \ldots, s_n)$).

6 **Examples.**

Case (i): The value of ALL ([.TRUE., .FALSE., .TRUE.]) is false.

Case (ii): If B is the array $\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$ and C is the array $\begin{bmatrix} 0 & 3 & 5 \\ 7 & 4 & 8 \end{bmatrix}$ then ALL (B /= C, DIM = 1) is [true, false, false] and ALL (B /= C, DIM = 2) is [false, false].

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.7.13, [329:6,11,14]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 338, 360, 374, 377, 392, 394 and 395.

Change the subclause heading to "ANY (MASK, DIM) or ANY (MASK)",
in paragraph 3, DIM argument, delete "(optional)",
in paragraph 4, change "is absent" to "does not appear",
making the whole subclause read:

13.7.13 ANY (MASK, DIM) or ANY (MASK)

1 **Description.** Reduce logical array with OR operation.

2 **Class.** Transformational function.

3 **Arguments.**

MASK shall a logical array.

DIM shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of MASK. The corresponding actual argument shall not be an optional dummy argument.

4 **Result Characteristics.** The result is of type logical with the same kind type parameter as MASK. It is scalar if DIM does not appear or $n = 1$; otherwise, the result has rank $n - 1$ and shape $[d_1, d_2, \ldots, d_{\text{DIM}-1}, d_{\text{DIM}+1}, \ldots, d_n]$ where $[d_1, d_2, \ldots, d_n]$ is the shape of MASK.

5 **Result Value.**

Case (i): The result of ANY (MASK) has the value true if any element of MASK is true and has the value false if no elements are true or if MASK has size zero.

Case (ii): If MASK has rank one, ANY (MASK, DIM) is equal to ANY (MASK). Otherwise, the value of element $(s_1, s_2, \ldots, s_{\text{DIM}-1}, s_{\text{DIM}+1}, \ldots, s_n)$ of ANY (MASK, DIM) is equal to ANY (MASK $(s_1, s_2, \ldots, s_{\text{DIM}-1}, :, s_{\text{DIM}+1}, \ldots, s_n)$).

6 **Examples.**

Case (i): The value of ANY ([.TRUE., .FALSE., .TRUE.]) is true.

Case (ii): If B is the array $\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$ and C is the array $\begin{bmatrix} 0 & 3 & 5 \\ 7 & 4 & 8 \end{bmatrix}$ then ANY (B /= C, DIM = 1) is [true, false, true] and ANY (B /= C, DIM = 2) is [true, true].

Interp **F08/0004**, Status: *Corrigendum 2*.

Ref: 13.7.16, after the 5^{th} paragraph, [330:36+]

Insert a new note:

NOTE 13.8a

The references to TARGET in the above cases are referring to properties that might be possessed by the actual argument, so the case of TARGET being a disassociated pointer will be covered by case *(iii)*, *(vi)*, or *(vii)*.

Interp **F08/0027**, Status: *Corrigendum 1.*

Ref: 13.7.21, 4^{th} paragraph, [332:25]

Change "CALL ATOMIC_REF (I [3], VAL)"
to "CALL ATOMIC_REF (VAL, I [3])",
making the whole paragraph read:

Example. CALL ATOMIC_REF (VAL, I [3]) causes VAL to become defined with the value of I on image 3.

Interp **F08/0019**, Status: *Corrigendum 1.*

Ref: 13.7.24, 3^{rd} paragraph, [333:12-14]

NOTE: This interp also has an edit on page 334.

For the arguments N1 and N2,
change "ot type integer and nonnegative" to "an integer scalar with a nonnegative value",
and for argument X,
after "real" insert "; if the function is transformational, X shall be scalar",
making the whole paragraph read:

Arguments.

N	shall be of type integer and nonnegative.
N1	shall be ~~of type~~ an integer scalar with a ~~and~~ nonnegative value.
N2	shall be ~~of type~~ an integer scalar with a ~~and~~ nonnegative value.
X	shall be of type real; if the function is transformational, X shall be scalar.

Interp **F08/0019**, Status: *Corrigendum 1.*

Ref: 13.7.27, 3^{rd} paragraph, [334:12-14]

NOTE: This interp also has an edit on page 333.

For the arguments N1 and N2,
change "ot type integer and nonnegative" to "an integer scalar with a nonnegative value",
and for argument X,
after "real" insert "; if the function is transformational, X shall be scalar",
making the whole paragraph read:

Arguments.

N	shall be of type integer and nonnegative.
N1	shall be ~~of type~~ an integer scalar with a ~~and~~ nonnegative value.
N2	shall be ~~of type~~ an integer scalar with a ~~and~~ nonnegative value.
X	shall be of type real; if the function is transformational, X shall be scalar.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.7.41, 3^{rd} paragraph, DIM argument, [338:31]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 360, 374, 377, 392, 394 and 395.

After "dummy argument" insert ", a disassociated pointer, or an unallocated allocatable",
making the whole paragraph read:

DIM (optional) shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of MASK. The corresponding actual argument shall not be an optional dummy argument, a disassociated pointer, or an unallocated allocatable.

Interp **F08/0020**, Status: *Corrigendum 1.*

Ref: 13.7.61, 3^{rd} paragraph, VALUE argument, [347:31-32]

Change "relational ... 7.1.5.5.2)" to "the operator == or the operator .EQV.",
making the whole paragraph read:

VALUE shall be scalar and in type conformance with ARRAY, as specified in Table 7.2 for ~~relational intrinsic operations (7.1.5.5.2)~~ the operator == or the operator .EQV..

Interp **F08/0064**, Status: *Corrigendum 2.*

Ref: 13.7.67, 3^{rd} paragraph, [351:18]

In the STATUS argument,
after "either has no value" change "or" to a comma,
after "assigned to VALUE," insert "or the VALUE argument is not present,",
making the description of that argument read:

STATUS (optional) shall be a default integer scalar. It is an INTENT (OUT) argument. If the environment variable exists and either has no value ~~or~~, its value is successfully assigned to VALUE, or the VALUE argument is not present, STATUS is set to zero. STATUS is set to -1 if the VALUE argument is present and has a length less than the significant length of the environment variable. It is assigned the value 1 if the specified environment variable does not exist, or 2 if the processor does not support environment variables. Processor-dependent values greater than 2 may be returned for other error conditions.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.7.90 and 13.7.91, 3^{rd} paragraph of each, DIM argument, [360:4,25]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 374, 377, 392, 394 and 395.

In both subclauses,
after "dummy argument" insert ", a disassociated pointer, or an unallocated allocatable",
this makes the paragraph in 13.7.90 read:

DIM (optional) shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of ARRAY. The corresponding actual argument shall not be an optional dummy argument, a disassociated pointer, or an unallocated allocatable.

and makes the paragraph in 13.7.91 read:

DIM (optional) shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of COARRAY. The corresponding actual argument shall not be an optional dummy argument, a disassociated pointer, or an unallocated allocatable.

Interp **F08/0040**, Status: *Corrigendum 2.*

Ref: 13.7.118, 3^{rd} paragraph and after the 6^{th} paragraph, [372:18,19,29+]

In the description of the FROM argument,
change "type and rank" to "type, rank, and corank",
and in the description of the TO argument,
after "same rank" insert "and corank",
making the whole paragraph read

3 **Arguments.**

FROM may be of any type ~~and~~, rank, and corank. It shall be allocatable. It is an INTENT (INOUT) argument.

TO shall be type compatible (4.3.1.3) with FROM and have the same rank and corank. It shall be allocatable. It shall be polymorphic if FROM is polymorphic. It is an INTENT (OUT) argument. Each nondeferred parameter of the declared type of TO shall have the same value as the corresponding parameter of the declared type of FROM.

Insert new paragraph after the 6^{th} paragraph:

7 When a reference to MOVE_ALLOC is executed for which the FROM argument is a coarray, there is an implicit synchronization of all images. On each image, execution of the segment (8.5.2) following the CALL statement is delayed until all other images have executed the same statement the same number of times.

NOTE: Interp F08/0040 also has edits on pages 97 and 188.

Interp **F03/0003**, Status: *Corrigendum 1.*

Ref: 13.7.123, heading and 3^{rd} and 4^{th} paragraphs, [374:24,29,31]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 360, 377, 392, 394 and 395.

Change "NORM2 (X [, DIM])" to "NORM2 (X, DIM) or NORM2 (X)",
in paragraph 3, DIM argument, delete "(optional)",
in paragraph 4, change "is absent" to "does not appear",
making the whole subclause read:

13.7.123 NORM2 (X, DIM) or NORM2 (X)

1 **Description.** L_2 norm of an array.

2 **Class.** Transformational function.

3 **Arguments.**

X	shall be a real array.
DIM	shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of X. The corresponding actual argument shall not be an optional dummy argument.

4 **Result Characteristics.** The result is of the same type and type parameters as X. It is scalar if DIM does not appear; otherwise the result has rank $n-1$ and shape $[d_1, d_2, \ldots, d_{\text{DIM-1}}, d_{\text{DIM+1}}, \ldots, d_n]$, where n is the rank of X and $[d_1, d_2, \ldots, d_n]$ is the shape of X.

5 **Result Value.**

Case (i): The result of NORM2 (X) has a value equal to a processor-dependent approximation to the generalized L_2 norm of X, which is the square root of the sum of the squares of the elements of X.

Case (ii): The result of NORM2 (X, DIM=DIM) has a value equal to that of NORM2 (X) if X has rank one. Otherwise, the value of element $(s_1, s_2, \ldots, s_{\text{DIM}-1}, s_{\text{DIM}+1}, \ldots\ s_n)$ of the result is equal to NORM2 (X$(s_1, s_2, \ldots, s_{\text{DIM}-1}, :, s_{\text{DIM}+1}, \ldots\ s_n)$).

6 It is recommended that the processor compute the result without undue overflow or underflow.

7 **Example.** The value of NORM2 ([3.0, 4.0]) is 5.0 (approximately). If X has the value $\begin{bmatrix} 1.0 & 2.0 \\ 3.0 & 4.0 \end{bmatrix}$ then the value of NORM2 (X, DIM=1) is [3.162, 4.472] (approximately) and the value of NORM2 (X, DIM=2) is [2.236, 5.0] (approximately).

Interp **F03/0003**, Status: *Corrigendum 1.*

Ref: 13.7.128, heading and 3^{rd} and 4^{th} paragraphs, [377:20,25,28]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 360, 374, 392, 394 and 395.

Change "PARITY (MASK [, DIM])" to "PARITY (MASK, DIM) or PARITY (MASK)",
in paragraph 3, DIM argument, delete "(optional)",
in paragraph 4, change "is absent" to "does not appear",
making the whole subclause read:

13.7.128 PARITY (MASK, DIM) or PARITY (MASK)

1 **Description.** Reduce array with .NEQV. operation.

2 **Class.** Transformational function.

3 **Arguments.**

MASK shall be a logical array.

DIM shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of MASK. The corresponding actual argument shall not be an optional dummy argument.

4 **Result Characteristics.** The result is of type logical with the same kind type parameter as MASK. It is scalar if DIM does not appear; otherwise, the result has rank $n-1$ and shape $[d_1, d_2, \ldots, d_{\text{DIM}-1}, d_{\text{DIM}+1}, \ldots, d_n]$ where $[d_1, d_2, \ldots, d_n]$ is the shape of MASK.

5 **Result Value.**

Case (i): The result of PARITY (MASK) has the value true if an odd number of the elements of MASK are true, and false otherwise.

Case (ii): If MASK has rank one, PARITY (MASK, DIM) is equal to PARITY (MASK). Otherwise, the value of element $(s_1, s_2, \ldots, s_{\text{DIM}-1}, s_{\text{DIM}+1}, \ldots, s_n)$ of PARITY (MASK, DIM) is equal to PARITY (MASK $(s_1, s_2, \ldots, s_{\text{DIM}-1}, :, s_{\text{DIM}+1}, \ldots, s_n)$).

6 **Examples.**

Case (i): The value of PARITY ([T, T, T, F]) is true if T has the value true and F has the value false.

Case (ii): If B is the array $\begin{bmatrix} T & T & F \\ T & T & T \end{bmatrix}$, where T has the value true and F has the value false, then PARITY (B, DIM=1) has the value [F, F, T] and PARITY (B, DIM=2) has the value [F, T].

Interp **F08/0078**, Status: *Corrigendum 2.*

Ref: 13.7.153, 5^{th} paragraph, [387:32]

In case *(iv)*, change "cannot" to "does not",
making the whole paragraph read:

Result Value.

Case (i): If B > 0, the value of the result is |A|.

Case (ii): If B < 0, the value of the result is -|A|.

Case (iii): If B is of type integer and B=0, the value of the result is |A|.

Case (iv): If B is of type real and is zero, then:

- if the processor ~~cannot~~does not distinguish between positive and negative real zero, or if B is positive real zero, the value of the result is |A|;
- if B is negative real zero, the value of the result is -|A|.

NOTE: Interp F08/0078 also has an edit on page 54.

Interp **F08/0021**, Status: *Corrigendum 1.*

Ref: 13.7.160, 3^{rd} paragraph, [390:6]

Around "has any deferred type parameters" insert "is unlimited polymorphic or" and a comma, making the whole paragraph read:

Arguments.

A shall be a scalar or array of any type. If it is polymorphic it shall not be an undefined pointer. If it is unlimited polymorphic or has any deferred type parameters, it shall not be an unallocated allocatable variable or a disassociated or undefined pointer.

KIND (optional) shall be a scalar integer constant expression.

Interp **F03/0003**, Status: *Corrigendum 1.*

Ref: 13.7.165, heading and 3^{rd} paragraph, [392:6,11]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 360, 374, 377, 394 and 395.

Change "THIS_IMAGE (COARRAY [, DIM])"
to "THIS_IMAGE (COARRAY) or THIS_IMAGE(COARRAY, DIM)",
in paragraph 3, DIM argument, delete "(optional)",
the result is shown below; **the editor has made additional changes:**
— **deleted the mistaken paragraph 7 marker,** — **added commas to the heading**.

13.7.165 THIS_IMAGE ( ), THIS_IMAGE (COARRAY), or THIS_IMAGE (COARRAY, DIM)

1 **Description.** Cosubscript(s) for this image.

2 **Class.** Transformational function.

3 **Arguments.**

COARRAY shall be a coarray of any type. If it is allocatable it shall be allocated.

DIM shall be a default integer scalar. Its value shall be in the range $1 \leq \text{DIM} \leq n$, where n is the corank of COARRAY. The corresponding actual argument shall not be an optional dummy argument.

4 **Result Characteristics.** Default integer. It is scalar if COARRAY does not appear or DIM is present; otherwise, the result has rank one and its size is equal to the corank of COARRAY.

5 **Result Value.**

Case (i): The result of THIS_IMAGE ( ) is a scalar with a value equal to the index of the invoking image.

Case (ii): The result of THIS_IMAGE (COARRAY) is the sequence of cosubscript values for COARRAY that would specify the invoking image.

Case (iii): The result of THIS_IMAGE (COARRAY, DIM) is the value of cosubscript DIM in the sequence of cosubscript values for COARRAY that would specify the invoking image.

6 **Examples.** If A is declared by the statement

```
REAL A (10, 20) [10, 0:9, 0:*]
```

then on image 5, THIS_IMAGE ( ) has the value 5 and THIS_IMAGE (A) has the value [5, 0, 0]. For the same coarray on image 213, THIS_IMAGE (A) has the value [3, 1, 2].

The following code uses image 1 to read data. The other images then copy the data.

```
IF (THIS_IMAGE()==1) READ (*,*) P
SYNC ALL
P = P[1]
```

NOTE 13.2

For an example of a module that implements a function similar to the intrinsic function THIS_IMAGE, see subclause C.10.1.

Interp **F03/0003**, Status: *Corrigendum 1.*

Ref: 13.7.171, 3^{rd} paragraph, DIM argument, [394:27]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 360, 374, 377, 392 and 395.

After "dummy argument" insert ", a disassociated pointer, or an unallocated allocatable",
making the whole paragraph read:

DIM ~~(optional)~~ shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of ARRAY. The corresponding actual argument shall not be an optional dummy argument, a disassociated pointer, or an unallocated allocatable.

Interp **F03/0003**, Status: *Corrigendum 1.*

Ref: 13.7.172, 3^{rd} paragraph, DIM argument, [395:11]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 360, 374, 377, 392 and 394.

After "dummy argument" insert ", a disassociated pointer, or an unallocated allocatable",
making the whole paragraph read:

DIM ~~(optional)~~ shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the corank of COARRAY. The corresponding actual argument shall not be an optional dummy argument, a disassociated pointer, or an unallocated allocatable.

Interp **F08/0065**, Status: *Corrigendum 2.*

Ref: 13.8.2.1, 2^{nd} paragraph, [397:7]

Append sentence to paragraph, making the whole paragraph read:

2 The processor shall provide the named constants, derived type, and procedures described in subclause 13.8.2. In the detailed descriptions below, procedure names are generic and not specific. The module procedures described in 13.8.2 are pure.

NOTE: Interp F08/0065 also has an edit on page 312.

Interp **F03/0030**, Status: *Corrigendum 3.*

Ref: 14.3, 1^{st} paragraph, [403:7-9,10-11]

Replace the first and second bullet points, making the whole paragraph read as follows:

1 The exceptions are the following.

- ~~IEEE_OVERFLOW occurs when the result for an intrinsic real operation or assignment has an absolute value greater than a processor-dependent limit, or the real or imaginary part of the result for an intrinsic complex operation has an absolute value greater than a processor-dependent limit.~~
- IEEE_OVERFLOW occurs in an intrinsic real addition, subtraction, multiplication, division, or conversion by the intrinsic function REAL, as specified by IEC 60559:1989 if IEEE_SUPPORT_DATATYPE is true for the operands of the operation or conversion, and as determined by the processor otherwise. It occurs in an intrinsic real exponentiation as determined by the processor. It occurs in a complex operation, or conversion by the intrinsic function CMPLX, if it is caused by the calculation of the real or imaginary part of the result.
- ~~IEEE_DIVIDE_BY_ZERO occurs when a real or complex division has a nonzero numerator and a zero denominator.~~
- IEEE_DIVIDE_BY_ZERO occurs in a real division as specified by IEC 60559:1989 if IEEE_SUPPORT_DATATYPE is true for the operands of the division, and as determined by the processor otherwise. It is processor-dependent whether it occurs in a real exponentiation with a negative exponent. It occurs in a complex division if it is caused by the calculation of the real or imaginary part of the result.
- IEEE_INVALID occurs when a real or complex operation or assignment is invalid; possible examples are SQRT (X) when X is real and has a nonzero negative value, and conversion to an integer (by assignment, an intrinsic procedure, or a procedure defined in an intrinsic module) when the result is too large to be representable.
- IEEE_UNDERFLOW occurs when the result for an intrinsic real operation or assignment has an absolute value less than a processor-dependent limit and loss of accuracy is detected, or the real or imaginary part of the result for an intrinsic complex operation or assignment has an absolute value less than a processor-dependent limit and loss of accuracy is detected.
- IEEE_INEXACT occurs when the result of a real or complex operation or assignment is not exact.

NOTE: This interp also has an edit on page 462.

Interp **F08/0009**, Status: *Corrigendum 1.*

Ref: 14.9, 1^{st} paragraph, [406:15+]

Add a new item after the second item of the bulleted list,
making the whole paragraph read:

The inquiry function IEEE_SUPPORT_DATATYPE can be used to inquire whether IEEE arithmetic is supported for a particular kind of real. Complete conformance with IEC 60559:1989 is not required, but

- the normal numbers shall be exactly those of an IEC 60559:1989 floating-point format,
- for at least one rounding mode, the intrinsic operations of addition, subtraction and multiplication shall conform whenever the operands and result specified by IEC 60559:1989 are normal numbers,
- the IEEE function abs shall be provided by the intrinsic function ABS,
- the IEEE operation rem shall be provided by the function IEEE_REM, and
- the IEEE functions copysign, scalb, logb, nextafter, and unordered shall be provided by the functions IEEE_COPY_SIGN, IEEE_SCALB, IEEE_LOGB, IEEE_NEXT_AFTER, and IEEE_UNORDERED, respectively,

for that kind of real.

Interp **F03/0053**, Status: *Corrigendum 3*.

Ref: 15.3.4, 1^{st} paragraph, [431:6]

Replace the first sentence as follows, making the whole paragraph

1 ~~A Fortran derived type is interoperable if it has the BIND attribute.~~ Interoperability between derived types in Fortran and struct types in C is provided by the BIND attribute on the Fortran type.

Interp **F08/0057**, Status: *Corrigendum 2*.

Ref: 15.3.4, after constraint C1504, [431:10+]

Insert new constraint

C1504a (R425) A derived type with the BIND attribute shall have at least one component.

Interp **F03/0053**, Status: *Corrigendum 3*.

Ref: 15.3.4, NOTE 15.11, [431:12+2]

Insert "with a C struct type" after "is interoperable", making the whole note read:

NOTE 15.11

The syntax rules and their constraints required that a derived type that is interoperable with a C struct type have components that are all data entities that are interoperable. No component is permitted to be allocatable or a pointer, but the value of a component of type C_FUNPTR or C_PTR may be the C address of such an entity.

Interp **F03/0053**, Status: *Corrigendum 3*.

Ref: 15.3.4, 2^{nd} paragraph, [431:13-18]

Change all "Fortran derived type" and "Fortran type" to "derived type", making the whole paragraph read

2 A ~~Fortran~~ derived type is interoperable with a C struct type if and only if the ~~Fortran~~derived type has the BIND attribute (4.5.2), the ~~Fortran~~ derived type and the C struct type have the same number of components, and the components of the ~~Fortran~~ derived type would interoperate with corresponding components of the C struct type as described in 15.3.5 and 15.3.6 if the components were variables. 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.

NOTE: Interp F03/0053 also has edits on pages 19 and 77.

Interp **F03/0124**, Status: *Corrigendum 1.*

Ref: 16.6.6, 1^{st} paragraph, [455:4-10]

Replace the entire item (1) by:

(1) When a scalar variable of intrinsic type becomes defined, all totally associated variables of different type become undefined. When a double precision scalar variable becomes defined, all partially associated scalar variables become undefined. When a scalar variable becomes defined, all partially associated double precision scalar variables become undefined.

Interp **F08/0081**, Status: *Corrigendum 2.*

Ref: A.2, [459:36+]

After bullet item "whether and when an object is finalized ... (4.5.6.3);",
insert new bullet item:

- whether an object is finalized by a deallocation in which an error condition occurs (4.5.6.3);

NOTE: Interp F08/0081 also has edits on pages 76, 131, and 460.

Interp **F08/0081**, Status: *Corrigendum 2.*

Ref: A.2, [460:5+]

After bullet item "the order ... event described in 6.7.3.2;",
insert new bullet item:

- whether an allocated allocatable subobject is deallocated when an error condition occurs in the deallocation of an object (6.7.3.2);

NOTE: Interp F08/0081 also has edits on pages 76, 131, and 459.

Interp **F03/0030**, Status: *Corrigendum 3*.

Ref: A.2, after "supports IEEE arithmetic (14), [462:24+]

Insert new bullet points as follows:

- the conditions under which IEEE_OVERFLOW is raised in a calculation involving non-IEC 60559:1989 floating-point data;
- the conditions under which IEEE_OVERFLOW and IEEE_DIVIDE_BY_ZERO are raised in a floating-point exponentiation operation;
- the conditions under which IEEE_DIVIDE_BY_ZERO is raised in a calculation involving non-IEC 60559:1989 floating-point data;

NOTE: This interp also has an edit on page 403.

Interp **F03/0048**, Status: *Corrigendum 1.*

Ref: C.6.2, 1^{st} paragraph, [487:28]

NOTE: This interp also has an edit on page 227.

Delete "record positioning", making the whole paragraph read:

Data transfer statements affect the positioning of an external file. In FORTRAN 77, if no error or end-of-file condition exists, the file is positioned after the record just read or written and that record becomes the preceding record. This part of ISO/IEC 1539 contains the ~~record positioning~~ ADVANCE= specifier in a data transfer statement that provides the capability of maintaining a position within the current record from one formatted data transfer statement to the next data transfer statement. The value NO provides this capability. The value YES positions the file after the record just read or written. The default is YES.

Interp **F08/0036**, Status: *Corrigendum 1.*

Ref: C.13.3.6, 3^{rd} paragraph, [527:18]

Insert a superscript "2" to square the absolute value of X_i,
making the whole paragraph read:

The L^2-norm of vector X, defined as $\sqrt{\sum_{i=1}^{n} |X_i|^2}$, can be formed using the Fortran expression NORM2 (X).