```
To: J3
From: Malcolm Cohen
Subject: Draft Interpretation Pages
Date: 2008/02/06
```

This document contains the edits, as insert pages, for the interpretations that are still being worked on. The status line for interpretations not included in a corrigendum includes the last paper number in the interp's history.

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.

NOTE: This does not include **any** edit published in a corrigendum, even when there is one on the same page; thus it is meant to be read in conjunction with standing document 018, it is not a replacement thereof.

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.)

Note: This is an ordinary meeting paper at meeting 183; maybe it ought to be a standing document?

Interp **F03/0094**, Status: *Passed by J3 letter ballot.*

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, 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 ~~be~~ have the INTENT(OUT) or VALUE attribute.

Interp **F03/0085**, Status: *J3 consideration in progress: 06-374.*

Ref: 4.5.5.2, 6th paragraph, [59:31-32]

After "When a procedure is invoked,"
Replace "a nonpointer, nonallocatable object that is an actual argument associated with an"
With "an object that becomes argument associated with a nonpointer nonallocatable",
After " dummy argument"
Insert "of that procedure",
Making the whole paragraph read:

> When a procedure is invoked, ~~a nonpointer nonallocatable~~ an object that ~~is an actual~~ becomes argument associated with ~~an~~ a nonpointer nonallocatable INTENT(OUT) dummy argument of that procedure is finalized.

Interp **F03/0090**, Status: *Passed by J3 meeting: 07-250r1/272.*

Ref: 4.7, C494, [67:21]

Between “same” and “type”
Insert “declared”,
Making the whole constraint read:

C494 (R466) If *type-spec* is omitted, each *ac-value* expression in the *array-constructor* shall have the same declared type and kind type parameters.

NOTE: This interp also has an edit on p68.

Interp **F03/0090**, Status: *Passed by J3 meeting: 07-250r1/272.*

Ref: 4.7, 2nd and 3*rd* paragraphs, [68:9,14+]

The 2nd paragraph begins "If *type-spec* is omitted,",
Before "type and"
Insert "declared",
Making the whole paragraph read:

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

After the 3rd paragraph
(which begins "If *type-spec* appears,")
Insert new paragraph as follows:

> The dynamic type of the array constructor is the same as its declared type.

NOTE: This interp also has an edit on p67.

Interp **F03/0091**, Status: *J3 consideration in progress: 07-232.*

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

After “a function result,”
Insert “a component,”.

NOTE: Result omitted as it clashes with Corrigendum 1.

Interp **F03/0099**, Status: *Passed by J3 meeting: 07-339.*

Ref: 5.1.2.16, 3rd paragraph, [85:10]

Between "association status" and "and array bounds"
Insert ", dynamic type and type parameters,",
Making the whole paragraph read:

> A pointer with the VOLATILE attribute may additionally have its association status, dynamic type and type parameters, and array bounds changed by means not specified by the program.

NOTE: This interp also has edits on pages 415, 421 and 423.

Interp **F03/0046**, Status: *J3 consideration in progress: N1622*.

Ref: 5.5.2, 2nd constraint (C588), [98:20]

Before "an allocatable variable"
Insert "a polymorphic pointer,",
Making the whole constraint read:

C588 (R558) A *common-block-object* shall not be a dummy argument, a polymorphic pointer, an allocatable variable, a derived-type object with an ultimate component that is allocatable, an automatic object, a function name, an entry name, a variable with the BIND attribute, or a result name.

Interp **F90/0145**, Status: *J3 consideration in progress: 96-m136.*

Ref: 6.1.1, last paragraph, [104:24]

Append text to paragraph,
making the whole paragraph read:

> If the parent is a variable, the substring is also a variable. A substring must not be referenced or defined before the declaration of the type and type parameters of the parent string, unless the type and type parameters are determined by the implicit typing rules of the scope.

The editor says: if we need this edit, it ought to be a new paragraph.

NOTE: This interp also contained an edit to 5.1 which has already been applied in F2003, in the 3^{rd} paragraph following C538, at [74:16-17].

NOTE: This interp also contains edits to pages 107, 126, and 127.

Interp **F90/0145**, Status: *J3 consideration in progress: 96-m136.*

Ref: 6.2.2, last paragraph, 1^{st} sentence, [107:26]

After "An array section in an array."
Insert "An array element or array section must not be referenced or defined before the declaration of the array bounds.",
Making the whole paragraph read:

> An array element is a scalar. An array section is an array. An array element or array section must not be referenced or defined before the declaration of the array bounds. If a *substring-range* is present in an *array-section*, each element is the designated substring of the corresponding element of the array section.

NOTE: This interp has other edits, see p104.

Interp **F03/0098**, Status: *Passed by J3 letter ballot: 07-279/321.*

Ref: 6.3.1.1, last paragraph, [113:21]

At the end of the sentence
Insert "unless they are defined by a SOURCE= specifier",
Making the whole paragraph read:

> When an object of derived type is created by an ALLOCATE statement, any allocatable ultimate components have an allocation status of unallocated unless they are defined by a SOURCE= specifier.

NOTE: This interp also has edits on pages 421 and 422.

Interp **F90/0145**, Status: *J3 consideration in progress: 96-m136*.

Ref: 7.1.6, last paragraph, 1st sentence, [126:15-16]

Change "specified in the same *specification-part*"
To "declared in the same scoping unit",
Change "prior specification of the *specification-part*"
To "specification prior to the specification expression",
Making the whole sentence read:

> If a specification expression includes a specification inquiry that depends on a type parameter or an array bound of an entity ~~specified in the same *specification-part*~~ declared in the same scoping unit, the type parameter or array bound shall be specified in a ~~prior specification of the *specification-part*~~ specification prior to the specification expression.

Interp **F90/0145**, Status: *J3 consideration in progress: 96-m136.*

Ref: 7.1.7, last paragraph, 1st sentence, [127:31-32]

Change "specified in the same *specification-part*"
To "specified in the same scoping unit",
Change "prior specification of the *specification-part*"
To "specification prior to the initialization expression",
Making the whole sentence read:

> If an initialization expression includes a specification inquiry that depends on a type parameter or an array bound of an entity specified in the same ~~*specification-part*~~ scoping unit, the type parameter or array bound shall be specified in a ~~prior specification of the *specification-part*~~ specification prior to the initialization expression.

NOTE: This interp contains other edits, see p104.

Interp **F03/0102**, Status: *Passed by J3 meeting: 07-297r2.*

Ref: 7.1.8, 3rd paragraph, [128:6]

After "undefinition"
Insert ", or changes the pointer association status or allocation status,",
Making the whole paragraph read:

> The evaluation of a function reference shall neither affect nor be affected by the evaluation of any other entity within the statement. If a function reference causes definition or undefinition, or changes the pointer association status or allocation status, of an actual argument of the function, that argument or any associated entities shall not appear elsewhere in the same statement. However, execution of a function reference in the logical expression in an IF statement (8.1.2.4), the mask expression in a WHERE statement (7.4.3.1), or the subscripts and strides in a FORALL statement (7.4.4) is permitted to define variables in the statement that is conditionally executed.

Interp **F03/0093**, Status: *Passed by J3 letter ballot: 07-279/321.*

Ref: 7.4.1.3, 3^{rd} 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 be an array of 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 parameter~~s~~ 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*).

NOTE: The editor changed the edit to avoid two consecutive full stops.

Interp **F03/0092**, Status: *Passed by J3 letter ballot: 07-279/321.*

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 **F95/0102**, Status: *J3 consideration in progress: 05-180.*

Ref: 7.4.3.2, 1st paragraph, last sentence, [147:1]

Change “only once”
To “at most once”,
Making the whole sentence read:

> The *mask-expr* is evaluated ~~only once~~ at most once.

NOTE: This interpretation also contains a similar edit to F95 that has already been applied to F2003, in the 4th paragraph of 7.4.3.2, at [147:7].

Interp **F03/0021**, Status: *J3 consideration in progress: 05-146.*

Ref: 8.4, BNF R850, after C834, and 1st paragraph, [170:23,24+,27]

Replace "*digit* [*digit* [*digit* [*digit* [*digit*]]]]"
By "*int-literal-constant*",
Making the whole R850 read:

R850	*stop-code*	**is**	*scalar-char-constant*
		~~**or**~~	~~*digit* [*digit* [*digit* [*digit* [*digit*]]]]~~
		or	*int-literal-constant*

Insert new constraint,

C834a (R850) The *int-literal-constant* shall not have a *kind-param* and shall not have more than 5 *digits*.

In the 3rd sentence of the paragraph, after "significant",
Insert "and all stop codes are permissible even if not representable in the default integer type",
Making the whole sentence read:

Leading zero digits in the stop code are not significant and all stop codes are permissible even if not representable in the default integer type.

Interp **F03/0105**, Status: *Passed by J3 meeting: 07-302r1.*

Ref: 9.5.1, 2^{nd} paragraph, [188:8]

NOTE from the editor: The position info in this interp is **completely** wrong – wrong paragraph and wrong line number. Corrected above.

Paragraph begins "A SIZE= specifier";
Append new sentence to paragraph,
Making the whole paragraph:

> A SIZE= specifier may appear only in an input statement that contains an ADVANCE= specifier with the value NO. A SIZE= specifier shall not appear in a parent input statement if the user-defined derived-type input procedure to be invoked performs either list-directed or namelist input on the same unit.

Interp **F03/0050**, Status: *Passed by J3 letter ballot: 07-272.*

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

The 7th paragraph begins "If an internal file"; After that paragraph insert two 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.

Interp **F03/0048**, Status: *Passed by J3 meeting: 07-250r1/272.*

Ref: 9.5.3.7.2, 7th and 8th paragraphs after Note 9.46, [202:34,36]

The 7th paragraph begins "Because a child data transfer statement does not position the file prior to data transfer,";
After "more recently processed effective list item or"
Delete "record positioning",
At the beginning of the 8th paragraph,
Replace "A record positioning edit descriptor, such as TL and TR,"
With "The edit descriptors T and TL,"

Making the whole of both paragraphs:

> 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 list item or ~~record positioning~~ edit descriptor. This is not necessarily at the beginning of a record.
>
> ~~A record positioning edit descriptor, such as TL and TR~~ The edit descriptors T and TL, used on `unit` by a child data transfer statement shall not cause the record position to be positioned before the record position at the time the user-defined derived-type input/output procedure was invoked.

NOTE: This interp also has edits on pages 430 and 463.

Interp **F03/0106**, Status: *Passed by J3 meeting: 07-309r1.*

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

NOTE from the editor: 9.9.1.9 needs to give more context ("file" appears multiple times in this subclause!).

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

Edit subclauses 9.9.1.8, 9.9.1.9, 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 9.1

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.

Interp **F03/0106**, Status: *Passed by J3 meeting: 07-309r1.*

Ref: 9.9.1.16, 9.9.1.17 [213:16,21+]

NOTE from the editor: The edit to 9.9.1.17 does not specify whether this is a new paragraph or not, but a **position indicator of 21+ usually means a new paragraph AFTER line 21**, not appending a new sentence; therefore that is what is done here.

Edit subclauses 9.9.1.16 and 9.9.1.17 as shown below:

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 or if the unit specified by UNIT= is not connected to a file, 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, 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.

If the unit specified by UNIT= is not connected to a file, the value is the unit specified by UNIT=.

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

Interp **F03/0106**, Status: *Passed by J3 meeting: 07-309r1*.

Ref: 9.9.1.21, [214:20]

Edit subclause 9.9.1.21 as shown below:

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.

NOTE from the editor: The form *A* **or** *B* **or** *C* is not acceptable.

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

Interp **F03/0106**, Status: *Passed by J3 meeting: 07-309r1.*

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 READWRITE 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 SEQUENTIAL 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: *Passed by J3 meeting: 07-309r1.*

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 UNFORMATTED 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.

Interp **F03/0096**, Status: *J3 consideration in progress: 07-266r1.*

Ref: 9.11, 6th paragraph, [219:16]

Between “The value of a” and “specifier”
Insert “FMT=, ID=, IOMSG=, IOSTAT= or SIZE=”,
Making the whole paragraph read:

> The value of a FMT=, ID=, IOMSG=, IOSTAT= or SIZE= 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.

Interp **F03/0079**, Status: *Passed by J3 meeting: 07-281r2.*

Ref: 10.6.1, after item (6), [227:15+]

Insert new list item:

(7) On output of a real zero value, the digits in the exponent field shall all be zero, whether or not a nonzero scale factor is in effect.

Interp **F03/0100**, Status: *Passed by J3 meeting: 07-340r1.*

Ref: 10.6.1.2.1, antepenultimate paragraph, [228:36-37]

Paragraph begins “For an internal value that is an IEEE infinity”;
Replace the last sentence “If ... produced.”
Making the whole paragraph read:

> 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 less than the mimimum required but large enough to produce the form 'Inf' then the form 'Inf' is output.

NOTE from the editor: the edit said to use double-quotes but the rest of the paragraph used single quotes so that is what I did.

NOTE: This interp also has an edit on page 229.

Interp **F03/0100**, Status: *Passed by J3 meeting: 07-340r1.*

Ref: 10.6.1.2.1, penultimate paragraph, [229:2]

Replace the last sentence of the paragraph,
Making the whole paragraph (which begins on the previous page) read:

> 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.

Interp **F03/0051**, Status: *Passed by J3 meeting: 07-250r1/272.*

Ref: 10.9.1, after Note 10.28, [239:18-]

Insert new paragraph:

> When the first value of an r*c constant is transferred to a list item by a list-directed input statement, that input statement shall transfer all r values of that r*c constant to list items before causing any child input statement to be invoked. If that list-directed input statement is itself a child input statement, it shall transfer all r values of that r*c constant to list items before terminating.

Interp **F03/0049**, Status: *Passed by J3 letter ballot: 07-272.*

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

Append new sentence to paragraph:

> Two undelimited character sequences are considered adjacent when both were written using list-directed input/output, no intervening data transfer or input/output file positioning operations occurred, and both were written either by a single data transfer statement, or during the execution of a parent data transfer statement along with its child data transfer statements.

Interp **F03/0101**, Status: *Passed by J3 letter ballot: 07-279/321.*

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

Append new sentences to paragraph:

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

NOTE: THIS EDIT CONFLICTS WITH INTERP F03/0049.

NOTE: This interp also has edits on page 246.

Interp **F03/0097**, Status: *Passed by J3 letter ballot: 07-279/321.*

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 **F03/0059**, Status: *J3 consideration in progress: 06-133.*

Ref: 10.10.1.1, 1st paragraph, [243:24-27]

Replace "If the namelist group object name is the name of a variable of derived type, the name in the input record may be either the name of the variable or the designator of one of its components, indicated by qualifying the variable name with the appropriate component name."
With "If the namelist group object is a variable of derived type, the name in the input record may be the name of the variable. If the variable would not be processed by a user-defined derived-type input/output procedure, the name in the input record may also be the designator of one of its components, using the syntax of object designators.",
Making the whole paragraph read:

> Within the input data, each name shall correspond to a particular namelist group object name. Subscripts, strides, and substring range expressions used to qualify group object names shall be optionally signed integer literal constants with no kind type parameters specified. If a namelist group object is an array, the input record corresponding to it may contain either the array name or the designator of a subobject of that array, using the syntax of object designators (R603). If the namelist group object ~~name~~ is ~~the name of~~ a variable of derived type, the name in the input record may be ~~either~~ the name of the variable ~~or the designator of one of its components, indicated by qualifying the variable name with the appropriate component name~~. If the variable would not be processed by a user-defined derived-type input/output procedure, the name in the input record may also be the designator of one of its components, using the syntax of object designators. Successive qualifications may be applied as appropriate to the shape and type of the variable represented.

Interp **F03/0101**, Status: *Passed by J3 letter ballot: 07-279/321.*

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. It need not be compatible with namelist input.

NOTE from the editor: This edit and the previous one have incorrect hyphenation.

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

Interp **F03/0063**, Status: *J3 consideration in progress: N1658.*

Ref: 11.3, C1117, [263:12,14]

After "derived-type definitions"
Insert ", abstract interface blocks,",
Before "and type declaration"
Insert "procedure declaration statements,",
Making the whole constraint read:

C1117 (R1116) A *block-data specification-part* shall contain only derived-type definitions, abstract interface blocks, and ASYNCHRONOUS, BIND, COMMON, DATA, DIMENSION, EQUIVALENCE, IMPLICIT, INTRINSIC, PARAMETER, POINTER, SAVE, TARGET, USE, VOLATILE, procedure declaration statements, and type declaration statements.

NOTE: This interp also has an edit on p254.

Interp **F03/0063**, Status: *J3 consideration in progress: N1658.*

Ref: 11.3, C1118, [264:3]

After "specifiers"
Insert "if it declares a data object",
Append new sentence "A procedure declaration statement shall not declare an external procedure.",
Making the whole constraint read:

C1118 (R1116) A type declaration statement in a *block-data specification-part* shall not contain ALLOCATABLE, EXTERNAL, or BIND attribute specifiers if it declares a data object. A procedure declaration statement shall not declare an external procedure.

Interp **F03/0019**, Status: *J3 consideration in progress: N1658.*

Ref: 12.3.2.1, antepenultimate constraint before the 1st paragraph, [259:18-19]

Delete constraint C1209, i.e.

~~C1209~~ ~~(R1206) A *procedure-name* shall not specify a procedure that is specified previously in any *procedure-stmt* in any accessible interface with the same generic identifier.~~

Interp **F03/0088**, Status: *Passed by J3 letter ballot: 07-272.*

Ref: 12.3.2.1.1, 2^{nd} paragraph, [262:16]

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 **F03/0088**, Status: *Passed by J3 letter ballot: 07-272.*

Ref: 12.3.2.1.2, 2^{nd} 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.

Interp **F03/0064**, Status: *J3 consideration in progress: 06-133.*

Ref: 12.3.2.3, C1212, [264:22]

Before "it shall be previously declared."
Insert "or *interface-body*",
Making the whole constraint:

C1212 (R1215) The *name* shall be the name of an abstract interface or of a procedure that has an explicit interface. If *name* is declared by a *procedure-declaration-stmt* or *interface-body* it shall be previously declared. If *name* denotes an intrinsic procedure it shall be one that is listed in 13.6 and not marked with a bullet (•).

Interp **F03/0003**, Status: *Passed by J3 meeting: 07-280r1.*

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

Insert new paragraph:

> The *data-ref* shall not be an undefined pointer, a disassociated pointer, or an unallocated allocatable variable.

Interp **F03/0004**, Status: *Passed by J3 meeting: 07-337.*

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

Same edit as for F03/0003.

Interp **F03/0109**, Status: *Passed by J3 meeting: 07-338.*

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

Insert new paragraph:

> The *data-ref* shall not be an undefined pointer, a disassociated pointer, an unallocated allocatable variable, or a dummy data object that is not present (12.4.1.6).

NOTE: This edit conflicts with the edits for F03/0003 and F03/0004.

Interp **F03/0073**, Status: *J3 consideration in progress: 06-105.*

Ref: 12.4.1.2, 2nd and 3rd paragraphs, [269:3,5]

Replace both occurrences of "of type default character"
With "of type default character, of type character with the C character kind (15.1),",
Making the whole two paragraphs read:

> The type parameter values of the actual argument shall agree with the corresponding ones of the dummy argument that are not assumed or deferred, except for the case of the character length parameter of an actual argument of type default character, of type character with the C character kind (15.1), associated with a dummy argument that is not assumed shape.
>
> If a scalar dummy argument is of type default character, of type character with the C character kind (15.1),, 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 of type default character and is not assumed shape, it becomes associated with the leftmost characters of the actual argument element sequence (12.4.1.5) and it shall not extend beyond the end of that sequence.

Interp **F03/0103**, Status: *Passed by J3 meeting: 07-298r2.*

Ref: 12.4.1.2, 2nd paragraph, [269:1-4]

Replace the paragraph in its entirety,
Making it read:

The type parameter values of the actual argument shall agree with the corresponding ones of the dummy argument that are not assumed or deferred~~, except for the case of the character length parameter of an~~ unless

- the dummy argument is not present (12.4.1.6), or
- the actual argument is of type default character ~~associated with a~~ and the dummy argument ~~that~~ is not assumed shape.

NOTE: THIS EDIT CONFLICTS WITH THE ONE FOR INTERP F03/0073.

Interp **F03/0017**, Status: *J3 consideration in progress: 05-146.*

Ref: 12.4.1.3, 2nd paragraph, [271:16,19+]

After "or intrinsic procedure,"
Replace "an associated"
With "a",
Append sentence to paragraph "Except in references to intrinsic inquiry functions, if the actual argument is a pointer it shall be associated.",
Making the whole paragraph read:

> If a dummy argument is a dummy procedure without the POINTER attribute, the associated actual argument shall be the specific name of an external, module, dummy, or intrinsic procedure, ~~an associated~~ a procedure pointer, or a reference to a function that returns an associated procedure pointer. The only intrinsic procedures permitted are those listed in 13.6 and not marked with a bullet (•). If the specific name is also a generic name, only the specific procedure is associated with the dummy argument. Except in references to intrinsic inquiry functions, if the actual argument is a pointer it shall be associated.

Interp **F03/0086**, Status: *Passed by J3 letter ballot: 07-272.*

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

Replace C1242 in its entirety with:

C1242 An ELEMENTAL procedure shall not have the BIND attribute.

Interp **F03/0087**, Status: *Passed by J3 meeting: 07-250r1/272.*

Ref: 12.5.2.4, C1255, [283:10]

After “Within”
Insert “the scoping unit of”,
Making the whole constraint read:

C1255 (R1235) Within the scoping unit of the subprogram containing the *entry-stmt*, the *entry-name* shall not appear as a dummy argument in the FUNCTION or SUBROUTINE statement or in another ENTRY statement nor shall it appear in an EXTERNAL, INTRINSIC, or PROCEDURE statement.

Interp **F03/0082**, Status: *J3 consideration in progress: 06-153.*

Ref: 12.6, C1266 and C1267, [286:12,13-14]

At the end of C1266,
After "INTENT(IN)"
Insert "or the VALUE attribute",
Making the whole constraint:

C1266 The *specification-part* of a pure function subprogram shall specify that all its nonpointer dummy data objects have INTENT(IN) or the VALUE attribute.

Replace C1267 in its entirety with:

C1267 Within the *specification-part* of a pure subroutine subprogram, for each non-pointer dummy data object, either its intent shall be explicitly specified or it shall have the VALUE attribute.

Interp **F03/0047**, Status: *J3 consideration in progress: N1622.*

Ref: new subclause 13.2.3, [292:18+]

NOTE: The editor, noting that the edit as written doesn't make sense (you don't add a new 13.2 immediately after 13.2 has finished already) and that the topic of the proposed new subclause falls into the remit of the existing 13.2, has chosen to cast this as a new subclause 13.2.3 despite what the interp says. Various other minor edits have been made to make it make sense.

Insert new subclause as follows:

13.2.4 Polymorphic intrinsic function arguments and results

The following table specifies the intrinsic functions that are allowed to have polymorphic arguments, and the arguments that are allowed to be polymorphic.

Function name	Arguments permitted to be polymorphic
ALLOCATED	ARRAY, SCALAR
ASSOCIATED	POINTER, TARGET
CSHIFT	ARRAY
EOSHIFT	ARRAY, BOUNDARY
EXTENDS_TYPE_OF	A, MOLD
LBOUND	ARRAY
MERGE	TSOURCE, FSOURCE
MOVE_ALLOC	FROM, TO
PACK	ARRAY, VECTOR
RESHAPE	SOURCE, PAD
SAME_TYPE_AS	A, B
SHAPE	SOURCE
SIZE	ARRAY
SPREAD	SOURCE
TRANSFER	SOURCE
TRANSPOSE	MATRIX
UBOUND	ARRAY
UNPACK	VECTOR, FIELD

The intrinsic functions shown in the following table have a polymorphic result if and only if the specified argument is polymorphic. Where the result is specified to have the same type and type parameters as the argument specified in the following table, the result has the same dynamic type as the specified argument. If the specified argument is unlimited polymorphic the result is unlimited polymorphic; otherwise it has the same declared type as the specified argument. If another argument is required to have the same type as the specified argument, it shall be polymorphic if and only if the specified argument is polymorphic, and have the same dynamic type as the specified argument. If the specified argument is unlimited polymorphic, the other argument shall also be unlimited polymorphic; otherwise, it shall have the same declared type as the specified argument.

Function name	Argument that determines result characteristics
CSHIFT	ARRAY
EOSHIFT	ARRAY
MERGE	TSOURCE
PACK	ARRAY
RESHAPE	SOURCE
SPREAD	SOURCE
TRANSPOSE	MATRIX
UNPACK	VECTOR

Interp **F03/0042**, Status: *J3 consideration in progress: 05-121r1.*

Ref: 13.7.0, penultimate sentence, [300:13-14]

After "If"
Insert "the values of all input arguments are normal or denormal and",
After "signal; if"
Insert "the values of all input arguments are normal or denormal and",
Making the whole sentence read:

> If the values of all input arguments are normal or denormal and an infinite result is returned, the flag IEEE_OVERFLOW or IEEE_DIVIDE_BY_ZERO shall signal; if the values of all input arguments are normal or denormal and a NaN result is returned, the flag IEEE_INVALID shall signal.

Interp **F03/0030**, Status: *J3 consideration in progress: N1622.*

Ref: 14.2, 1^{st} paragraph, 1^{st} and 2^{nd} bullet points, [365:13,15,18]

In the 1^{st} bullet point of 14.2,
After both occurrences of “operation or assignment”
Insert “with finite operands”,
In the 2^{nd} bullet point,
Before “nonzero numerator”
Insert “finite”,
Making the whole first two bullet points (with the opening line):

The exceptions are

- IEEE_OVERFLOW
  This exception occurs when the result for an intrinsic real operation or assignment with finite operands has an absolute value greater than a processor-dependent limit, or the real or imaginary part of the result for an intrinsic complex operation or assignment with finite operands has an absolute value greater than a processor-dependent limit.
- IEEE_DIVIDE_BY_ZERO
  This exception occurs when a real or complex division has a finite nonzero numerator and a zero denominator.

Interp **F03/0107**, Status: *Passed by J3 meeting: 07-312r2.*

Ref: 14.8, at the end, [369:28+]

Insert new note:

NOTE 14.8a

The standard requires that code such as

```
        if (IEEE_SUPPORT_DATATYPE(x)) then
               x = IEEE_SCALB(x,2)
        else
               x = x*4
        endif
```

be executable. The elemental functions in the IEEE_ARITHMETIC module (14.9.2) must exist for all real kinds supported by the processor, even if IEEE_SUPPORT_DATATYPE returns false for some kinds. However, if IEEE_SUPPORT_DATATYPE returns false for a particular kind, these functions must not be invoked with arguments of that kind. This allows a careful programmer to write programs that work on processors that do not support IEEE arithmetic for all real kinds.

The processor might provide stub routines which allow the program to link and execute, but which will abort if they are invoked.

NOTE: This interp also contains an edit to page 370.

Interp **F03/0107**, Status: *Passed by J3 meeting: 07-312r2.*

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~~:

NOTE: This interp also contains an edit on page 369.

Interp **F03/0034**, Status: *J3 consideration in progress: N1622.*

Ref: 14.10.12, Result Value paragraph, [376:15,17+]

In case (i), after "Note:"
Insert "if X is normal,",
After case (ii)
Insert new case(iii) "If IEEE_SUPPORT_INF(X) is true and X is infinite, the result is +infinity.",
Making the whole Result Value paragraph:

Result Value.

Case (i): If the value of X is neither zero, infinity, nor NaN, the result has the value of the unbiased exponent of X. Note: if X is normal this value is equal to EXPONENT (X)−1.

Case (ii): If X==0, the result is −infinity if IEEE_SUPPORT_INF (X) is true and −HUGE (X) otherwise; IEEE_DIVIDE_BY_ZERO signals.

Case (iii): If IEEE_SUPPORT_INF (X) is true and X is infinite, the result is +infinity.

Interp **F03/0039**, Status: *J3 consideration in progress: N1622.*

Ref: 14.11, Note 14.17, last paragraph, [389:12,16+]

After "This will work almost every time, but if an"
Insert "overflow or underflow",
Append new sentence to the paragraph "This HYPOT function does not handle infinite arguments in the same way that the hypot function in the C International Standard does.",
Making the whole paragraph read:

> An attempt is made to evaluate this function directly in the fastest possible way. This will work almost every time, but if an overflow or underflow exception occurs during this fast computation, a safe but slower way evaluates the function. This slower evaluation might involve scaling and unscaling, and in (very rare) extreme cases this unscaling can cause overflow (after all, the true result might overflow if X and Y are both near the overflow limit). If the IEEE_OVERFLOW or IEEE_UNDERFLOW flag is signaling on entry, it is reset on return by the processor, so that earlier exceptions are not lost. This HYPOT function does not handle infinite arguments in the same way that the hypot function in the C International Standard does.

Interp **F03/0053**, Status: *J3 consideration in progress.*

Ref: 15.2.2, 1st paragraph, [397:3]

At the end of the 1st paragraph
Insert new sentence "Each has the BIND attribute but is not interoperable with any C struct type.",
Making the whole paragraph read:

> C_PTR and C_FUNPTR shall be derived types with pointer components. C_PTR is interoperable with any C object pointer type. C_FUNPTR is interoperable with any C function pointer type. Each has the BIND attribute but is not interoperable with any C struct type.

Interp **F03/0075**, Status: *J3 consideration in progress: 06-106.*

Ref: 15.2.3, C1505, [398:8]

Append new sentence to constraint "If the component is an array, the corresponding C component shall be an array and the shapes of the two arrays shall be as prescribed in 15.2.5.",
Making the whole constraint:

C1505 (R429) Each component of a derived type with the BIND attribute shall be a nonpointer, nonallocatable data component with interoperable type and type parameters. If the component is an array, the corresponding C component shall be an array and the shapes of the two arrays shall be as prescribed in 15.2.5.

Interp **F03/0089**, Status: *Passed by J3 letter ballot: 07-272.*

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.

Interp **F03/0073**, Status: *J3 consideration in progress: 06-102.*

Ref: 15.2.6, numbered list it, item (5), [400:14-17]

NOTE: The editor has modified the edit as it was editorially wrong.

After “any dummy argument”
Replace “without the VALUE attribute”
With “that is not a procedure and does not have the VALUE attribute”,
Before “pointer type”
Insert “object”,
Delete “; and”,
After item (5)
Insert new item “(5a) any dummy procedure argument corresponds to a formal parameter of the prototype that is of function pointer type, and the dummy procedure is interoperable with a function of the referenced type of the formal parameter; and”

Making the whole of items (5) and (5a) read:

(5) any dummy argument ~~without the VALUE attribute~~ that is not a procedure and does not have the VALUE attribute corresponds to a formal parameter of the prototype that is of a pointer type, and the dummy argument is interoperable with an entity of the referenced type (C standard, 6.2.5, 7.17, and 7.18.1) of the formal parameter; ~~and~~

(5a) any dummy procedure argument corresponds to a formal parameter of the prototype that is of function pointer type, and the dummy procedure is interoperable with a function of the referenced type of the formal parameter; and

Interp **F03/0099**, Status: *Passed by J3 meeting: 07-339.*

Ref: 16.4.2.1.4, after last paragraph, [415:27+]

Insert new paragraph:

> The association status of a pointer object with the VOLATILE attribute may change by means not specified by the program. If the association status of such an object changes, its array bounds or deferred type parameters may change. If in addition it is polymorphic, its dynamic type and additional type parameters not specified in its declaration may also change.

NOTE: This interp also has edits on pages 85, 421 and 423.

Interp **F03/0098**, Status: *Passed by J3 letter ballot: 07-279/321.*

Ref: 16.5.5, list item (19), [421:27-28+]

After "Allocation of an object"
Insert "except by an ALLOCATE statement with a SOURCE= specifier",
And insert new list item,
Making item (19) and the new item read:

(19) Allocation of an object except by an ALLOCATE statement with a SOURCE= specifier that has a nonpointer default-initialized subcomponent causes that subcomponent to become defined.

(19a) Successful execution of an ALLOCATE statement with a SOURCE= specifier causes a subobject of the allocated object to become defined if the corresponding subobject of the SOURCE= expression is defined.

NOTE: This interp also has edits on pages 113 and 422.

Interp **F03/0099**, Status: *Passed by J3 meeting: 07-339.*

Ref: 16.5.5, item (26), [421:42-43]

Delete item (26) and insert normal text paragraph as follows:

~~(26)~~ ~~An object with the VOLATILE attribute that is changed by a means not specified by the program becomes defined (see 5.1.2.16).~~

In addition, an object with the VOLATILE attribute (5.1.2.16) might become defined by means not specified by the program.

NOTE: This interp also has edits on pages 85, 415 and 423.

Interp **F03/0098**, Status: *Passed by J3 letter ballot: 07-279/321.*

Ref: 16.5.6, list item (11), [422:41,43+]

After "ALLOCATE statement"
Insert "with no SOURCE= specifier",
And insert new list item,
Making item (11) and the new item read:

(11) Successful execution of an ALLOCATE statement with no SOURCE= specifier for a nonzero-sized object that has a subcomponent for which default initialization has not been specified causes the subcomponent to become undefined.

(11a) Successful execution of an ALLOCATE statement with a SOURCE= specifier causes a subobject of the allocated object to become undefined if the corresponding subobject of the SOURCE= expression is undefined.

NOTE: This interp also has edits on pages 113 and 421.

Interp **F03/0099**, Status: *Passed by J3 meeting: 07-339.*

Ref: 16.5.6, after item (18), [423:28+]

Between list item (18) and Note 16.19
Insert new textual paragraph as follows:

> In addition, an object with the VOLATILE attribute (5.1.2.16) might become undefined by means not specified by the program.

NOTE: This interp also has edits on pages 85, 415 and 421.

Interp **F03/0048**, Status: *Passed by J3 meeting: 07-250r1/272.*

Ref: Annex A, alphabetically, [430:4+]

Add the following definition.

> **file position** (9.2.3) : A connected unit has a file position. A unit's file position typically affects where the next data transfer operation will begin transferring data into or out of the file. The file position is usually just before a record, just after a record, within a record, just before the first file storage unit in the file, just after the last file storage unit in the file, or between two adjacent file storage units.

NOTE: This interp also has edits on pages 202 and 463.

Interp **F04/0048**, Status: *Passed by J3 meeting: 07-250r1/272.*

Ref: C.6.2, 1st paragraph, [463:4]

Before "ADVANCE= specifier"
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 standard 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.

NOTE: This interp also has edits on pages 202 and 430.