J3/10-151 INTERPRETATION UPDATE PAGES

To: J3
From: Malcolm Cohen
Subject: Updates to SD-018.
Date: 2010/05/17

This document is the update document for SD-018.

SD-018 contains insertions for every interpretation edit that has been published as a corrigendum to 04-007, plus the unpublished corrigendum 5.

This document contains those pages which are additional, or which changed, between the previous version of SD-018 (09-018r1) and the latest version (10-018). People who have already inserted 09-018r1 into their copy of 04-007 should print out this document and insert its pages (occasionally replacing ones from 09-018r1) instead of printing the whole 10-018.

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

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

This version includes a minor correction to 09-018r1 as well as unpublished corrigendum 5.

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

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

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

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

Interp **F95/0098**, Status: *Corrigendum 5.*

Ref: 4.4.4.1, C417, [41:11-12]

Change "unless ... dummy function"
To "unless it is of type CHARACTER and is the name of a dummy function or the name of the result of an external function",
Making the whole constraint read:

C417 A function name shall not be declared with an asterisk *type-param-value* unless it is of type CHARACTER and is the name of a dummy function or the name of the result of an external function.

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

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

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

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

Interp **F95/0098**, Status: *Corrigendum 5.*

Ref: 4.4.4.1, last paragraph, item (4), [41:34,36]

After "invoking the function"
Insert "or passing it as an actual argument",
And change "host or use" to "argument, host, or use", making the whole item read:

(4) If used to specify the character length parameter of a function result, any scoping unit invoking the functionor passing it as an actual argument shall declare the function name with a character length parameter parameter value other than * or access such a definition by argument, host, or use association. When the function is invoked, the length of the result variable in the function is assumed from the value of this type parameter.

Interp **F03/0122**, Status: *Corrigendum 5.*

Ref: 4.5.1.3, 2nd paragraph, [47:11]

In the last sentence of the paragraph
Delete "declared to be PRIVATE or",
Making the whole paragraph read:

> Two data entities have the same type if they are declared with reference to the same derived-type definition. The definition may be accessed from a module or from a host scoping unit. Data entities in different scoping units 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 property 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 property or with the BIND attribute is not of the same type as an entity of a type ~~declared to be PRIVATE or~~ that has any components that are PRIVATE.

Interp **F03/0090**, Status: *Corrigendum 5.*

Ref: 4.7, constraint C494, [67:19+,21]

Immediately before constraint C494
Insert new constraint (shown below),
In constraint C494
Change "type and" to "declared type and",
Making the new constraint and C494 read:

C493a (R469) An *ac-value* shall not be unlimited polymorphic.

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 edits on page 68.

Interp **F03/0090**, Status: *Corrigendum 5.*

Ref: 4.7, 2^{nd} and 3^{rd} paragraphs [68:9,11,14+]

After "in this case, the"
Insert "declared",
After "If *type-spec* appears, it specifies the"
Insert "declared",
After paragraph 3 insert new paragraph,
Making the whole 2^{nd} and 3^{rd} paragraphs 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.
>
> 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 parameters of the *array-constructor* in accordance with the rules of intrinsic assignment (7.4.1.3).
>
> The dynamic type of the array constructor is the same as its declared type.

NOTE: This interp also has edits on page 67.

Interp **F03/0127**, Status: *Corrigendum 5.*

Ref: 5.1.2.7, 2nd and 3rd paragraphs, [81:15,26]

In paragraph 2
Change "during the execution"
To "during the invocation and execution",
Append new sentence to paragraph 3,
Making the whole two paragraphs read:

> The INTENT (IN) attribute for a nonpointer dummy argument specifies that it shall neither be defined nor become undefined during the invocation and execution of the procedure. The INTENT (IN) attribute for a pointer dummy argument specifies that during the execution of the procedure its association shall not be changed except that it may become undefined if the target is deallocated other than through the pointer (16.4.2.1.3).
>
> The INTENT (OUT) attribute for a nonpointer dummy argument specifies that it shall be defined before a reference to the dummy argument is made within the procedure and any actual argument that becomes associated with such a dummy argument shall be definable. On invocation of the procedure, such a dummy argument becomes undefined except for components of an object of derived type for which default initialization has been specified. The INTENT (OUT) attribute for a pointer dummy argument specifies that on invocation of the procedure the pointer association status of the dummy argument becomes undefined. Any actual argument associated with such a pointer dummy shall be a pointer variable. Any undefinition or definition implied by association of an actual argument with an INTENT (OUT) dummy argument shall not affect any other entity within the statement that invokes the procedure.

NOTE: This interp also has edits on page 275.

Interp **F03/0134**, Status: *Corrigendum 5.*

Ref: 5.3, 4th paragraph, [93:9]

After "intrinsic function,"
Insert "is not a component,",
Making the whole paragraph read:

> Any data entity that is not explicitly declared by a type declaration statement, is not an intrinsic function, is not a component, and is not made accessible by use association 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 to the host scope. The data entity is treated as if it were declared in an explicit type declaration in the outermost scoping unit in which it appears. An explicit type specification in a FUNCTION statement overrides an IMPLICIT statement for the name of the result variable of that function subprogram.

Interp **F03/0131**, Status: *Corrigendum 5.*

Ref: 5.5.1.1, [97;7+]

In "5.5.1.1 Equivalence association"
Insert new paragraph:

> If any data object in an *equivalence-set* has the SAVE attribute, all other objects in the *equivalence-set* have the SAVE attribute; this may be confirmed by explicit specification.

Interp **F03/0063**, Status: *Corrigendum 5.*

Ref: 5.5.2, R558-C590, [98:17,18,21,25]

Delete the second line of R558,
In C587, delete "or *proc-pointer-name*",
In C588, after "allocatable," insert "a procedure pointer,",
In C590, delete "or *proc-pointer-name*",
See next interp for the end results.

NOTE: This interp also has edits on pages 100, 411 and 416.

Interp **F03/0133**, Status: *Corrigendum 5.*

Ref: 5.5.2, constraint C588, [98:22]

After "BIND attribute,"
Insert "an unlimited polymorphic pointer".

Applying F03/0063 and F03/0133 make all of R558-C590 read:

R558	*common-block-object*	**is**	*variable-name* [(*explicit-shape-spec-list*)]
		~~**or**~~	~~*proc-pointer-name*~~

C587 (R558) Only one appearance of a given *variable-name* ~~or *proc-pointer-name*~~ is permitted in all *common-block-object-list*s within a scoping unit.

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

C589 (R558) If a *common-block-object* is of a derived type, it shall be a sequence type (4.5.1) or a type with the BIND attribute and it shall have no default initialization.

C590 (R558) A *variable-name* ~~or *proc-pointer-name*~~ shall not be a name made accessible by use association.

Interp **F03/0063**, Status: *Corrigendum 5.*

Ref: 5.5.2.3, penultimate paragraph, [100:12-15]

Delete the sentences "A procedure pointer ... and type parameters.", Making the whole paragraph read:

> A data pointer shall be storage associated only with data pointers of the same type and rank. Data pointers that are storage associated shall have deferred the same type parameters; corresponding nondeferred type parameters shall have the same value. ~~A procedure pointer shall be storage associated only with another procedure pointer; either both interfaces shall be explicit or both interfaces shall be implicit. If the interfaces are explicit, the characteristics shall be the same. If the interfaces are implicit, either both shall be subroutines or both shall be functions with the same type and type parameters.~~

NOTE: This interp also has edits on pages 98, 411 and 416.

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

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

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

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

Interp **F03/0024**, Status: *Corrigendum 5.*

Ref: 6.3.3.2, 2nd paragraph, [116:25]

After "by allocation."
Insert new sentence,
Making the whole paragraph read:

> If a pointer appears in a DEALLOCATE statement, it shall be associated with the whole of an object that was created by allocation. The pointer shall have the same dynamic type and type parameters as the allocated object, and if the allocated object is an array the pointer shall be an array whose elements are the same as those of the allocated object in array element order. Deallocating a pointer target causes the pointer association status of any other pointer that is associated with the target or a portion of the target to become undefined.

Interp **F03/0138**, Status: *Corrigendum 5.*

Ref: 7.4.2, constraint C727, [144:5-6]

Change “an external, module,”
To “a module”,
Change “or a procedure pointer”
To “a procedure pointer, 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”,
Making the whole constraint:

C727 (R742) A *procedure-name* shall be the name of ~~an external,~~ a module~~,~~ or dummy procedure, a specific intrinsic function listed in 13.6 and not marked with a bullet (•), ~~or~~ a procedure pointer, 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.

Interp **F03/0132**, Status: *Corrigendum 5.*

Ref: 9.5.2, 12th paragraph, [193:13-15]

In the paragraph beginning "If a derived-type"
Edit the paragraph as follows:

> If a derived-type list item is not processed by a user-defined derived-type input/output procedure and is not treated as a list of its individual components, all the subcomponents of that list item shall be accessible in the scoping unit containing the input/output statement and shall not be pointers or allocatable~~that list item's ultimate components shall not have the POINTER or ALLOCATABLE attribute unless that list item is processed by a user-defined derived-type input/output procedure~~.

Interp **F03/0141**, Status: *Corrigendum 5.*

Ref: 12.3.2.1, 6th paragraph, [260:2]

After "given scoping unit"
Insert ", except that if the interface is accessed by use association, there may be more than one local name for the procedure",
Append new sentence to the end of the paragraph,
Making the whole paragraph (which begins on page 259) read:

> If an explicit specific interface is specified by an interface body or a procedure declaration statement (12.3.2.3) for an external procedure, the characteristics shall be consistent with those specified in the procedure definition, except that the interface may specify a procedure that is not pure if the procedure is defined to be pure. An interface for a procedure named by an ENTRY statement may be specified by using the entry name as the procedure name in the interface body. An explicit specific interface may be specified by an interface body for an external procedure that does not exist in the program if the procedure is never used in any way. A procedure shall not have more than one explicit specific interface in a given scoping unit, except that if the interface is accessed by use association, there may be more than one local name for the procedure. If a procedure is accessed by use association, each access shall be to the same procedure declaration or definition.

Interp **F03/0071**, Status: *Corrigendum 5.*

Ref: 12.3.2.1, 9th paragraph, [261:3]

Append new sentence to paragraph, Making it read:

> A generic interface block specifies a **generic interface** for each of the procedures in the interface block. The PROCEDURE statement lists procedure pointers, external procedures, dummy procedures, or module procedures that have this interface. A generic interface is always specific. If a specific procedure in a generic interface has a function dummy argument, that argument shall have its type and type parameters explicitly declared in the specific interface.

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

Ref: 12.3.2.1.1, 2nd paragraph, [262:16]

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

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

NOTE: This interp also has an edit on p263.

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

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

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

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

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

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

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

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

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

Interp **F03/0112**, Status: *Corrigendum 5.*

Ref: 12.3.2.1.2, end of subclause, [263:14-]

At the end of subclause "12.3.2.1.2 Defined assignments",
Insert new note as follows:

NOTE 12.10a

> If the second argument of a procedure specified in a defined assignment interface block has the POINTER or ALLOCATABLE attribute, it cannot be accessed by defined assignment, since the right-hand side of the assignment is enclosed in parentheses before being associated as an actual argument with the second argument. This makes it an expression, which does not have the POINTER or ALLOCATABLE attribute.

Interp **F03/0137**, Status: *Corrigendum 5.*

Ref: 12.4.1.3, 5th paragraph, [271:28]

Append new sentences to the end of the paragraph,
Making the whole paragraph read:

> If the interface of the dummy argument is implicit and either the name of the dummy argument is explicitly typed or it is referenced as a function, the dummy argument shall not be referenced as a subroutine and the actual argument shall be a function, function procedure pointer, or dummy procedure. If both the actual argument and dummy argument are known to be functions, they shall have the same type and type parameters. If only the dummy argument is known to be a function, the function that would be invoked by a reference to the dummy argument shall have the same type and type parameters, except that an external function with assumed character length may be associated with a dummy argument with explicit character length.

Interp **F03/0127**, Status: *Corrigendum 5.*

Ref: 12.4.1.7, item (2), [275:2,5]

Change both occurrences of "during the execution"
To "during the invocation and execution",
Making the whole item read:

(2) If the allocation status of the entity or a subobject thereof is affected through the dummy argument, then at any time during the invocation and execution of the procedure, either before or after the allocation or deallocation, it may be referenced only through the dummy argument. If the value the entity or any subobject is affected through the dummy argument, then at any time during the invocation and execution of the procedure, either before or after the definition, it may be referenced only through that dummy argument unless

- (a) the dummy argument has the POINTER attribute or
- (b) the dummy argument has the TARGET attribute, the dummy argument does not have INTENT (IN), the dummy argument is a scalar object or an assumed-shape array, and the actual argument is a target other than an array section with a vector subscript.

NOTE: This interp also has edits on page 81.

Interp **F03/0135**, Status: *Corrigendum 5.*

Ref: 12.4.4, item (2), [276:36+]

In item (3) "A procedure name is established to be only specific..."
Insert new subitem before subitem (b):

(a2) if that scoping unit is of a subprogram that defines a procedure with that name;

NOTE: This interp also has an edit on page 278.

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

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

Append new list item:

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

Interp **F03/0135**, Status: *Corrigendum 5.*

Ref: 12.4.4.2, after item (3), [278:15+]

Insert new item

(3a) If the scoping unit is of a subprogram that defines a procedure with that name, the reference is to that procedure.

NOTE: This interp also has an edit on page 276.

Interp **F03/0127**, Status: *Corrigendum 5.*

Ref: 12.6, after constraint C1271, [286:22+]

Insert new constraint as follows:

C1271a The *designator* of a variable with the VOLATILE attribute shall not appear in a pure subprogram.

Interp **F03/0119**, Status: *Corrigendum 5.*

Ref: 12.7.1, constraint C1278, [287:17]

After "shall be scalar"
Change "and" to ",".

NOTE: See page 288 for the restated constraint.

Interp **F03/0119**, Status: *Corrigendum 5.*

Ref: 12.7.1, constraint C1278, [288:1]

After "ALLOCATABLE attribute"
Insert ", and shall not have a type parameter that is defined by an expression that is not an initialization expression",
Making the whole constraint (starting on page 287) read:

C1278 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 an initialization expression.

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

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

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

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

NOTE: This interp also has an edit on page 317.

Interp **F03/0125**, Status: *Corrigendum 5.*

Ref: 13.7.38, Arguments and Result Value paragraphs, [316:16-17,21,22]

After “of extensible”
Change “type” to “declared type or unlimited polymorphic”, twice,
After “the result is false; otherwise”
Insert “if the dynamic type of A or MOLD is extensible,”,
After “dynamic type of MOLD”
Insert “; otherwise the result is processor dependent”,
Making the whole Arguments, Result Characteristics and Result Value paragraphs read:

Arguments.

A	shall be an object of extensible declared type or unlimited polymorphic. If it is a pointer, it shall not have an undefined association status.
MOLD	shall be an object of extensible declared type or unlimited polymorphic. If it is a pointer, it shall not have an undefined association status.

Result Characteristics. Default logical scalar.

Result Value. If MOLD is unlimited polymorphic and is either a disassociated pointer or unallocated allocatable variable, the result is true; otherwise if A is unlimited polymorphic and is either a disassociated pointer or unallocated allocatable variable, the result is false; otherwise if the dynamic type of A or MOLD is extensible, the result is true if and only if the dynamic type of A is an extension type of the dynamic type of MOLD; otherwise the result is processor dependent.

NOTE: This interp also has edits on pages 347-348.

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

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

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

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

Interp **F03/0125**, Status: *Corrigendum 5.*

Ref: 13.7.101, Arguments paragraph, [347:30]

After "of extensible"
Change "type"
To "declared type or unlimited polymorphic".

NOTE: See page 348 for the restated text.

Interp **F03/0125**, Status: *Corrigendum 5.*

Ref: 13.7.101, Arguments and Result Value paragraphs, [348:1,3,4]

After "of extensible"
Change "type"
To "declared type or unlimited polymorphic",
Change "The result"
To "If the dynamic type of A or B is extensible, the result",
Append sentence to paragraph,
Making the whole Arguments, Result Characteristics and Result Value paragraphs read:

Arguments.

A	shall be an object of extensible declared type or unlimited polymorphic. If it is a pointer, it shall not have an undefined association status.
B	shall be an object of extensible declared type or unlimited polymorphic. If it is a pointer, it shall not have an undefined association status.

Result Characteristics. Default logical scalar.

Result Value. If the dynamic type of A or B is extensible, the~~The~~ result is true if and only if the dynamic type of A is the same as the dynamic type of B. If neither A nor B has extensible dynamic type, the result is processor dependent.

NOTE: The new text above includes the last 3 lines from page 347.

Interp **F03/0022**, Status: *Corrigendum 5.*

Ref: 14, 2nd paragraph, 1st sentence, [363:9-10]

After "IEEE_DIVIDE_BY_ZERO are supported in the scoping unit for all kinds of real and complex"
Insert "IEEE floating-point",
Making the whole first sentence (in this rather long paragraph, the rest of which is omitted here):

> If IEEE_EXCEPTIONS or IEEE_ARITHMETIC is accessible in a scoping unit, IEEE_OVERFLOW and IEEE_DIVIDE_BY_ZERO are supported in the scoping unit for all kinds of real and complex IEEE floating-point data.

Interp **F03/0034**, Status: *Corrigendum 5.*

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

At the end of the Result Value paragraph,
Append two new list items as follows:

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

Case (iv): If IEEE_SUPPORT_NAN(X) is true and X is a NaN, the result is a NaN.

Interp **F03/0129**, Status: *Corrigendum 5.*

Ref: 15.1.2.5, Argument paragraph, [395:8,16+]

After "have interoperable type and" insert "kind",
Insert new sentence after list,
Making the whole Argument paragraph read:

Argument. X shall either

(1) have interoperable type and kind type parameters and be
  (a) a variable that has the TARGET attribute and is interoperable,
  (b) an allocated allocatable variable that has the TARGET attribute and is not an array of zero size, or
  (c) an associated scalar pointer, or

(2) be a nonpolymorphic scalar, have no length type parameters, and be
  (a) a nonallocatable, nonpointer variable that has the TARGET attribute,
  (b) an allocated allocatable variable that has the TARGET attribute, or
  (c) an associated pointer.

X shall not be a zero-length string.

NOTE: This interp also has edits on pages 396 and 399.

Interp **F03/0129**, Status: *Corrigendum 5.*

Ref: 15.2.1, 1st paragraph, [396:5-7]

Replace "; if ... one.", Making the whole paragraph read:

> Table 15.2 shows the interoperability between Fortran intrinsic types and C types. A Fortran intrinsic type with particular type parameter values is interoperable with a C type if the type and kind type parameter value are listed in the table on the same row as that C type~~; if~~. If the type is character, ~~interoperability also requires that~~ the length type parameter ~~be omitted or be specified by an initialization expression whose~~ is interoperable if and only if its value is one. A combination of Fortran type and type parameters that is interoperable with a C type listed in the table is also interoperable with any unqualified C type that is compatible with the listed C type.

NOTE: This interp also has edits on pages 395 and 399.

Interp **F03/0129**, Status: *Corrigendum 5.*

Ref: 15.2.4p1 and 15.2.5p1, [399:2-3,7-8]

Change the first paragraphs of 15.2.4 and 15.2.5 as shown below,
Making the whole of 15.2.4 and the first paragraph of 15.2.5 read:

15.2.4 Interoperability of scalar variables

A named scalar Fortran variable is **interoperable** if and only if its type and type parameters are interoperable,~~ and~~ it has neither the pointer nor the allocatable attribute, and if it is of type character its length is not assumed or declared by an expression that is not an initialization expression.

An interoperable scalar Fortran variable is interoperable with a scalar C entity if their types and type parameters are interoperable.

15.2.5 Interoperability of array variables

A~~An array~~ Fortran variable that is a named array is **interoperable** if and only if its type and type parameters are interoperable,~~ and~~ it is of explicit shape or assumed size, and if it is of type character its length is not assumed or declared by an expression that is not an initialization expression.

NOTE: This interp also has edits on pages 395 and 396.

Interp **F03/0136**, Status: *Corrigendum 5.*

Ref: 16.2.3, 2^{nd} paragraph, [407:28]

After "**distinguishable** if"
Insert "one is a subroutine and the other is an array, or if",
Making the whole paragraph read:

> Two dummy arguments are **distinguishable** if one is a subroutine and the other is an array, or if neither is a subroutine and neither is TKR compatible (5.1.1.2) with the other.

Interp **F03/0140**, Status: *Corrigendum 5.*

Ref: 16.3, 2nd and 3rd paragraphs, [409:19,26]

Before "scoping unit that includes the DATA"
And before "scoping unit that includes the FORALL"
Insert "innermost executable construct or" (thus twice),
Making those two paragraphs read:

> The name of a *data-i-do-variable* in a DATA statement or an *ac-do-variable* in an array constructor has a scope of its *data-implied-do* or *ac-implied-do*. It is a scalar variable that has the type and type parameters that it would have if it were the name of a variable in the innermost executable construct or scoping unit that includes the DATA statement or array constructor, and this type shall be integer type; it has no other attributes. The appearance of a name as a *data-i-do-variable* of an implied-DO in a DATA statement or an *ac-do-variable* in an array constructor is not an implicit declaration of a variable whose scope is the scoping unit that contains the statement.
>
> The name of a variable that appears as an *index-name* in a FORALL statement or FORALL construct has a scope of the statement or construct. It is a scalar variable that has the type and type parameters that it would have if it were the name of a variable in the innermost executable construct or scoping unit that includes the FORALL, and this type shall be integer type; it has no other attributes. The appearance of a name as an *index-name* in a FORALL statement or FORALL construct is not an implicit declaration of a variable whose scope is the scoping unit that contains the statement or construct.

Interp **F03/0063**, Status: *Corrigendum 5.*

Ref: 16.4.1.3, 2^{nd} paragraph, item (7), [411:21]

Delete item (7) in the list of names in a scoping unit that override the same host-associated name.

NOTE: This interp also has edits on pages 98, 100 and 416.

Interp **F03/0063**, Status: *Corrigendum 5.*

Ref: 16.4.3.1, 2nd paragraph, item (8), [416:23]

Change "A pointer occupies"
To "A data pointer occupies", Making the whole paragraph read:

(8) A data pointer occupies a single unspecified storage unit that is different from that of any nonpointer object and is different for each combination of type, type parameters, and rank.

NOTE: This interp also has edits on pages 98, 100 and 411.