ISO/IEC 1539-1:2010 - TECHNICAL CORRIGENDUM 2

Introduction
Following the second sentence in the fifth item in the bulleted list (Data usage and computation), insert: “Multiple allocations are permitted in a single ALLOCATE statement with SOURCE=.”.

Subclause 1.6.2
Replace the two paragraphs added to the subclause in Technical Corrigendum 1 by the following six paragraphs:

> Fortran 2003 permitted a sequence type to have type parameters; that is not permitted by this part of ISO/IEC 1539.
>
> 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.
>
> The form produced by the G edit descriptor for some values and some I/O rounding modes differs from that specified by Fortran 2003.
>
> 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.
>
> 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.
>
> 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.

Subclause 1.6.3
Add the following item at the end of the bulleted list:

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

Subclause 1.6.4
In the fourth paragraph of the subclause, replace the full stop at the end of the third bulleted item by a semicolon and add a fourth item:

- the G edit descriptor with *d* equal to zero for some values.

Subclause 4.4.2.3
In the third paragraph of the subclause, in Note 4.8 change “can distinguish” to “distinguishes”.

Subclause 4.5.2.3
Replace constraint C436 by:

> 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.

Subclause 4.5.2.4
In the second sentence of the second paragraph of the subclause, delete “type parameters and”.

Subclause 4.5.6.1
In the second sentence of constraint C480, insert “noncoarray,” before “nonpointer”.

Subclause 4.5.6.3
To the second paragraph of the subclause (which was paragraph 1 prior to the edits of Technical Corrigendum 1), append the new sentence:

> If an error condition occurs during deallocation, it is processor dependent whether finalization occurs.

Replace the seventh paragraph of the subclause (which was paragraph 8 prior to the edits of Technical Corrigendum 1) by the following new paragraph. This supersedes the substitute paragraph given in Technical Corrigendum 1.

> 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.

Subclause 4.8
In constraint C4105 in the first paragraph of the subclause, change “all *ac-value* expressions in the *array-constructor* shall be of that derived type and” to “the declared type of each *ac-value* expression in the *array-constructor* shall be that derived type and”.

After constraint C4106, insert the following new constraint:

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

In the second paragraph of the subclause, change “each *ac-value* expression in the array constructor shall have the same length type parameters;” to “corresponding length type parameters of the declared type of each *ac-value* expression shall have the same value;”.

In the third paragraph of the subclause, after “Each value is converted to the” insert “type and”.

Subclause 5.3.7
In the first paragraph of the subclause, change “can only be argument associated with a contiguous effective argument” to “is contiguous”.

Subclause 5.3.10
In constraint C541 change “An entity” to “A dummy argument of a nonintrinsic procedure”.

Subclause 5.4.7
In the fourth paragraph of the subclause, replace constraint C566 by:

> C566 (R536) 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.

Subclause 5.5
In the final sentence of the third paragraph of the subclause, change “an internal or module procedure” to “a BLOCK construct, internal subprogram, or module subprogram”.

Subclause 5.6
In the fifth paragraph of the subclause, change what was originally "type, type parameters, and shape" but which was changed by Technical Corrigendum 1 to "type, kind type parameters, and rank" to "declared type, kind type parameters of the declared type, and rank".

Subclause 6.7.1.1
Replace constraint C633 by:

> 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 (R631) If *allocate-object* is scalar, *allocate-shape-spec-list* shall not appear.

Replace constraint C639 by:

> 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*.

Replace the fourth paragraph of the subclause by:

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

Subclause 6.7.1.2
In the seventh paragraph of the subclause, before "On successful", insert the new sentence:

> 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*.

In the sentence beginning "On successful", replace "that of *source-expr"* with "the value provided", twice.

At the end of the seventh paragraph append the new sentence:

> The *source-expr* is evaluated exactly once for each execution of an ALLOCATE statement.

Subclause 6.7.3.2
Append the following new sentence to the eighth paragraph of the subclause:

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

Subclause 7.1.12
In the first paragraph of the subclause, in item (6) of the numbered list, after "THIS_IMAGE" insert ", or TRANSFER".

After item (7) of the numbered list, insert new item:

> (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.

Subclause 7.2.2.2
In constraint C729 replace "an external ... bullet (•)" with "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, referenced in the scoping unit as a procedure, or that has the EXTERNAL attribute".

Subclause 8.5.1
In the bulleted list in the second paragraph of the subclause, add the following new item before the STOP statement item:

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

Subclause 9.12
Replace the fifth paragraph of the subclause by:

> The value of a specifier in an input/output statement shall not depend 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.

Subclause 10.7.5.2.2
Following the third paragraph of the subclause, add a new paragraph:

> If *d* is zero, *k*PE*w*.0 or *k*PE*w*.0E*e* editing is used for G*w*.0 editing or G*w*.0E*e* editing respectively.

In the original fourth paragraph of the subclause replace the second and subsequent sentences, including the two tables, by:

> 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 $10^{s-1} \le N < 10^{s}$. If $s<0$ or $s>d$, *k*PE*w*.*d* or *k*PE*w*.*d*E*e* editing is used for G*w*.*d* editing or G*w*.*d*E*e* editing respectively, where *k* is the scale factor (10.8.5). If $0 \le s \le d$, 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*E*e* editing.

Subclause 12.4.2.2
At the beginning of the subclause, replace "A procedure ... and" with "Within the scope of a procedure identifier, the procedure shall have an explicit interface if it is not a statement function and".

Subclause 12.4.3.2
Replace constraint C1209 by:

> 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.

Subclause 12.4.3.4.5
In the third paragraph of the subclause, in constraint C1214 as amended in Technical Corrigendum 1 replace "the same" by "that".

Subclause 12.5.2.3
Replace the second paragraph of the subclause by:

> 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.

Replace the third paragraph of the subclause by:

> If a present nonpointer dummy argument without the VALUE attribute corresponds to a nonpointer actual argument,
> - if the dummy argument is polymorphic it becomes argument associated with that actual argument;
> - if the dummy argument is nonpolymorphic, it becomes argument associated with the declared type part of that actual argument.

Subclause 12.5.2.4
Append to the second paragraph of the subclause the sentence:

> If the actual argument is a polymorphic assumed-size array, the dummy argument shall be polymorphic.

In the third paragraph of the subclause, add the following sentence at the start of the paragraph:

> The kind type parameter values of the actual argument shall agree with the corresponding ones of the dummy argument.

In the original first sentence of the third paragraph change “The type parameter values of the actual argument” to “The length type parameter values of a present actual argument”.

In the fourth paragraph of the subclause, before “scalar dummy argument” insert “present”.

In the second sentence of the seventeenth paragraph of the subclause, after “has INTENT (OUT),” change “the actual argument” to “the effective argument” and delete the last sentence of the paragraph (“If … undefined.”).

Subclause 12.5.2.5
Replace the first paragraph of the subclause by:

> The requirements in this subclause apply to an actual argument with the ALLOCATABLE or POINTER attribute that corresponds to a dummy argument with the same attribute.

Delete the fourth paragraph of the subclause, that is “The values of assumed type parameters … effective argument.”.

Subclause 12.5.2.6
Following the third paragraph of the subclause, add the new paragraph:

> The values of assumed type parameters of a dummy argument are assumed from the corresponding type parameters of its effective argument.

Subclause 12.5.2.7
Add the following sentence at the end of the third paragraph of the subclause:

> The values of assumed type parameters of a dummy argument are assumed from the corresponding type parameters of its effective argument.

Subclause 12.5.2.8
In the second paragraph of the subclause, add at the end of the sentence, "or an element of a simply contiguous array".

Subclause 12.6.2.6
In the eighth paragraph of the subclause append the sentence:

> 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.

In the ninth paragraph of the subclause append the sentence:

> 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.

Subclause 12.7
In the first paragraph of the subclause, insert as the second item in the bulleted list:

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

In the second paragraph of the subclause, following constraint C1276 add:

> 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.

and following constraint C1277 add:

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

Subclause 13.7.1
In the second paragraph of the subclause, replace the fourth to sixth sentences ("A program … invoked.") by:

> A program shall not invoke an intrinsic procedure under circumstances where a value to be assigned to a subroutine argument or returned as a function result is not representable by objects of the specified type and type parameters.

Add the following as the third paragraph of the subclause:

> 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 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. If no IEEE infinity or NaN is assigned or returned, these flags shall have the same status as when the intrinsic procedure was invoked.

Subclause 13.7.16
Following the fifth paragraph of the subclause, insert the following 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).

Subclause 13.7.67
In the third paragraph of the subclause, in the description of 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,".

Subclause 13.7.118
In the third paragraph of the subclause, in the description of the FROM argument, change "type and rank" to "type, rank, and corank".

In the description of the TO argument, after "same rank" insert "and corank".

Before the seventh paragraph of the subclause (**Example.**) insert the following new paragraph:

> 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.

Subclause 13.7.153
In the fifth paragraph of the subclause, in Case (iv), change "cannot distinguish" to "does not distinguish".

Subclause 13.8.2.1
Append the following sentence to the second paragraph of the subclause:

> The module procedures described in 13.8.2 are pure.

Subclause 15.3.4
In the first paragraph of the subclause, before C1505 add a new constraint:

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

Annex A Subclause A.2
After "whether and when an object is finalized ... (4.5.6.3);" insert a new bullet point:

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

After "the order ... event described in 6.7.3.2;" insert a new bullet point:

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