Subject: Miscellaneous items
From: Van Snyder
References: 00-240, 00-317, 00-318, 01-115

Here are several things that may or may not need attention. I offer edits for a few, but mostly I don't offer edits, or I offer crappy ones.

1 Edits from /Data

Edits refer to 01-007. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by immediately following text, while a page and line number followed by + (-) indicates that immediately following text is to be inserted after (before) the indicated line. Remarks are noted in the margin, or appear between [and] in the text.

[The note ought to apply specifically to the syntax term *data-component-def-stmt*, not generally to *component-def-stmt*. The note could be repaired, but there's no good reason to keep it: it says nothing that's not said better by the syntax rule at [40:1] and the constraint at [40:33-34], which are both on the same page; indeed, the constraint is on the previous line! Editor: Delete.] 40:35-37

If finalization is used for storage management, it often needs to be combined with defined assignment. [This is meant to address concerns expressed by David Moore at meeting 154.] 57:23-24

[When one gets to the end of the sentence one might reasonably ask "same as what?" Editor: "each *ac-value* expression" ⇒ "all *ac-value* expressions".] 61:9

[The style is inconsistent with the one at [64:37-38] Replace by:] 64:34

Constraint: An entity shall not have both the EXTERNAL attribute and the INTRINSIC attribute.

[Unlike the note at [40:35-37] that is the subject of a nearby previous remark, this one refers to the correct syntax term, and the syntax terms and constraints from which one can derive it are up to two pages away. Nonetheless it serves only a marginal role. Editor: Delete.] 65:24-26

[The sentence "The appearance ... *entity-decl-list*" duplicates the constraint at [64:42-43]. Saying something twice is an opportunity to get it wrong (twice). Editor: Delete this sentence.] 69:25-27

[Editor: After "to." insert "It is scalar even if *designator* is an array."] 96:19+

[`a%kind` is not a function reference. Editor: "`Equivalent to`" ⇒ "`Same value as`" twice.] 96:24,25

[Editor: "`component`" ⇒ " "`element`".] 96:26

[Editor: Fix the typo – See 00-302.] 116:3-5

[The first two sentences of the paragraphs at [116:19-22] and [117:38-41] are identical, except for "specification function" on page 116 and "initialization expression" on page 117, "entity" on page 116 and "object" on page 117, and "specification inquiry" on page 116 and "inquiry function" on page 117. The first difference is to be expected; the others are not.
Editor: "object" ⇒ "entity". 117:39
Editor: "inquiry function" ⇒ "specification inquiry".] 117:41

[Editor: "*pointer-object*, the" ⇒ "*pointer-object*, *target* shall not be a disassociated pointer and 134:15

the". This is needed because a disassociated array pointer is not defined to have zero size.]

[This is just an anemic version of [75:2-4]. Again, saying something twice is an opportunity to get it wrong (twice). In this case, it's just incomplete. Editor: Delete.] 134:28-29

[Editor: Delete OPTIONAL and INTENT because they're already prohibited in BLOCK DATA subprograms by the constraint at [64:29-30], and delete PUBLIC and PRIVATE because they're already prohibited in BLOCK DATA subprograms by the constraint at [69:44].] 238:17

[Editor: After "explicit" insert ", whether it is a pointer".] 242:24

If the dummy argument is not allocatable and the actual argument is allocatable, the actual argument shall be currently allocated. 255:32+ New ¶

[The phrase "an elemental intrinsic actual procedure may be associated with a dummy argument that is not elemental" at [258:5-7] leads one to believe that dummy arguments can be elemental. Also, "dummy argument" should be "dummy procedure". Editor: "argument that is not elemental" ⇒ "procedure (which is prohibited from being elemental)".] 258:6-7

[Editor: Insert "have" before "the same operator". It's inconsistent and confusing to say "have" in two of three reasonable places.] 343:19

[Editor: "assignments" ⇒" "assignment".] 343:20

2 Edits from /JOR

[Editor: Add "compatibility, Fortran 95" to the index.] 3:12

[The paragraph is about compatibility of Fortran 2000 to Fortran 95, but it's in the subclause about compatibility of Fortran 95 to Fortran 90. Editor: Move to [3:19+].] 3:36-39

3 Edits from /Edit

[Editor: Exdent "`END`".] 190:14

[Section 15 is more closely related to intrinsic procedures and modules (section 13) than to the material of sections 14 or 16. Section 14 has traditionally been considered to be the one that ought to be last (perhaps to make it easy to remove). Editor: Move section 14 to be after section 16.] §§13-16

4 Non-edits or crappy edits

If they need attention, we can develop edits at the meeting, if we have time, or insert unresolved issue notes.

Page and line numbers refer to 01-007.

I thought that we did something having to do with the relation between EXTERNAL and "defined by a means other than Fortran" but I can't find it. If I recollect correctly, a procedure defined by a means other than Fortran is an external procedure. 12:34-35, 403:36-37

There is no prohibition against a *type-alias-name* being defined in terms of another one, so 4.6

it's presumably allowed. There is therfore apparently a need to prohibit a circular definition. A constraint that a *type-alias-name* in a *type-spec* in a *type-alias* shall have been previously declared (which may perhaps be within the same statement) would suffice.

If we had a term for "type compatible and all the kind type parameters have the same value" the discussions of argument association and generic resolution would be simpler. 69:1-4

The specs really said "disassociated"! This would be cool, but almost certainly "disassociated" should be "undefined". 70:37

What if the dummy argument has assumed or deferred type parameters? At [96:14-15] and [311:14-15], type parameter inquiry is only proscribed for deferred parameters of disassociated pointers or unallocated allocatables, so inquiring about assumed type parameters of dummy arguments is presumably OK. In order to do this, it seems the optional argument of NULL ought to be required. This is also the topic of interpretation request 19, at least in the context of assumed character length. Make sure this is updated if interppretation request 19 changes this. One way is to replace "in" by "either in" at [114:7] and add "or corresponding to a dummy argument that has assumed or deferred type parameters" at the end of the sentence at [114:8]. 113:40

[69:28-33] and [81:4-7] are similar but not identical. The difference isn't entirely accounted for by the fact that one is in an attribute description, and the other is in a statement description. This is perhaps another illustration that saying something twice is an opportunity to get it wrong, or at least incomplete (twice). Consolidate them, removing the duplication, and move them to 7.1.7. 7.1.7 116-118

I suggest: 129:23+

Constraint: In the case of intrinsic assignment, the *variable* and *expr* shall have the same rank or the *expr* shall be a scalar.

Constraint: In the case of intrinsic assignment, the types and kind type parameters of *variable* and *expr* shall conform according to the rules in table 7.9.

Put table 7.9 here.

This paragraph's title is "Intrinsic assignment conformance rules" so we don't need "for an intrinsic assignment statement" again. The part about "rules of Table 7.9" should be a constraint (see edit proposed for [129:23+] above). Replace by "The *variable* and *expr* shall conform in shape. If *variable* is of derived type, corresponding type parameters of *variable* and *expr* shall have the same values." 130:20-22

If these changes are not accepted, at least move table 7.9 to [130:22+]. Also see 00-318. **Malcolm doesn't like** these.

Maybe we should add "or procedure references" – or maybe we should delete "or a defined assignment statement (7.5.1.6)". 134:25

There was a discussion in /data subgroup concerning whether the associate name ought to have the POINTER or ALLOCATABLE attribute if and only if the selector does. The outcome was apparently that they ought not to. It wouldn't hurt to have a note here explaining why not, because it's not obvious. On the other hand, if there's no technical reason for the associate name not to have the POINTER or ALLOCATABLE attributes, it would be useful: It would make it easier to allocate, deallocate and pointer-assign components that have complicated antecedents. /Data should re-examine this. If it can be done, should it be done in §14? Maybe it already is done at [357:8-22]. 152:29

9.5.3 could and perhaps should be incorporated into 9.9. 9.5.3

Doesn't account for user-defined derived-type input/output procedures.	184:26-27
Replacing "components ... comprise" by "effective items (9.5.2) that result from expanding" would be more precise.	187:40
9.8.1.26 is the only subclause of 9.8.1 that deals with two specifiers. Couldn't they be treated with separate subclauses? This will be a valuable change if the advice in 01-143 is followed.	9.8.1.26
Unresolved issue note 124 questions the precision of the wording of the requirement that "the value of a specifier in an input/output statement shall not depend on any *input-item*, *io-implied-do-variable*,...." It should also be noted that the value of the variable associated with the IOSTAT= specifier depends, at least indirectly, on these factors.	205:26-29
This paragraph is in a subclause entitled "Character editing," and therefore doesn't apply to unformatted stream output. The slash edit descriptor is specified in 10.7.2 to create a new record during formatted stream output. Lacking an explicit indication that the description of end-of-record in 9.5.3 doesn't apply to stream input, it appears that it does. Therefore there appears to be no reason to use ACHAR(10) as an end-of-record signal during stream output. If we must have a character or character sequence that is interpreted as a new-record signal, it should be provided by a named constant from the ISO_FORTRAN_ENV module.	217:11-17
This paragraph also needs to mention dummy procedures and procedure pointers.	245:6-8
What is a "dummy argument to" a procedure? Shouldn't they be "*of*"?	272:12-13
Should "storage" be "storage or construct"?	272:14
User-defined derived-type input/output needs to be in the list.	272:25-26
Subclause 14.1.2.3 has nothing to do with generic procedure references. The title ought to be "Unambiguous generic procedure definitions" or "Restrictions on generic procedure definitions such that references are unambiguous."	343:11
The sentence "If a generic..." conflicts with, or at least belongs in [348:30-31].	343:47-2
14.1.2.4.1 takes no account of procedure pointers in generics.	345
These paragraphs do not explicitly apply to defined operations or defined assignment. They apply indirectly to defined operations by way of the phrase "it is generic in exact analogy to generic procedure names" at [248:15], but there is no parallel statement concerning defined assignment. 01-115 addresses this.	345:4-14
Either "procedure" should be "interface" at [345:8], or vice-versa at [345:14]. 01-115 does this.	345:8, 14