Subject: Comments on section 5
From: Van Snyder
References: 01-138r1, 01-166

1 Edits

Edits refer to 01-007r1. 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.

[PARAMETER and VALUE are the only right-hand sides that are not in alphabetical order. Editor: Alphabetize the right-hand sides.] 65:35, 66:7

[Simplification:]
[Editor: Insert "ALLOCATABLE," before "TARGET".] 66:34
[Editor: Delete.] 67:1-2

[Doesn't account for a SAVE statement without a *saved-entity-list*. Editor: Delete "or" at [68:20] and insert "or by the presence of a SAVE statement without a *saved-entity-list* in the same scoping unit" after "(5.2.11)".] 68:20-21

[This note is anachronistic noise. Editor: Delete it.] 68:35-44

[The word "would" is incorrect if IMPLICIT NONE is specified. Editor: "would" ⇒ "could".] 69:4

[Editor: "the function" ⇒ "a function; after "association" insert ", or the derived type is defined within an interface body or is accessible there by use association or as a consequence of an IMPORT statement".] 71:2

[Editor: "effector" ⇒ "affector".] 72:13

[Editor: After "argument" insert ", is not a structure constructor".] 73:38

[The sentence "If an explicit-shape ... expressions" is the definition of the term "automatic array" in the previous paragraph. We might as well use the term. Editor: "If an explicit-shape ... expressions, the" ⇒ "The"; before "are" insert "of an automatic array".] 73:39, 40

[The following note would reduce the occurrence of some questions and mistakes:] 74:15+

NOTE 5.12$\frac{1}{2}$

The lower bound is not taken from the associated actual argument.

[Editor: Delete "The ALLOCATABLE ... (5.2.2)." because it's redundant.] 74:20-21

[Editor: Delete "in a type ... definition statement." because it's redundant.] 74:22-25

[Editor: Delete "The POINTER ... (5.2.10)." because it's redundant.] 74:28-30

[Editor: Delete "An array ... definition statement." because it's redundant.] 74:31-32

[Simplification:]
Editor: Insert "or a disassociated array pointer" after "array". 74:35
Editor: Start a new paragraph with "The lower..." 74:37
Editor: Delete "The size ... 13.1." 74:39-41

[The bounds ... are unaffected by ... the bounds? Editor: At [75:2] "bounds" ⇒ "bounds' specification expressions".]	75:1-2
[Editor: After "name" insert "that is not the name of a block data program unit"; Delete ", or ... procedure" because it has nothing to do with the EXTERNAL attribute, which is the topic of this subclause.]	76:6
[Editor: "the" ⇒ "a".]	76:16
[Editor: Delete ", 12.4.1.4" because alternate returns are not germane to the present discussion.]	77:9
[A dummy argument is not a type, derived or otherwise. Editor: After "type" insert "object".]	77:18
Notice that if a structure is an actual argument that is associated with a dummy argument that has INTENT(OUT), its components become undefined upon invocation of the procedure. Therefore, its components cannot be used as actual arguments associated with other dummy arguments.	77:47+
[Editor: Delete. See [65:5-6].]	78:22
[Pointers don't "point to". Accessing a target doesn't "end up". Editor: "point only to" ⇒ "only be associated with"; "end up ... target" ⇒ "access an object that is neither an explicitly specified target nor an allocated object".]	79:39-40
[This sentence implies that appearing in a DATA statement is enough to cause implicit typing. There is no leeway for IMPLICIT NONE. Replace by wording similar to [85:7-8].] If a variable that appears in a DATA statement is typed by the implicit typing rules, its appearance in any subsequent specification of the *specification-part* shall confirm this implied type and the values of any implied type parameters. An array name,	82:4-5
[Syntax rules are by-and-large in depth-first order. Editor: Move [83:14-15] to here.]	83:9+
[Simplification:] The *data-stmt-constant* shall be NULL() if and only if the corresponding *data-stmt-object* has the POINTER attribute.	 83:33-34
[Editor: Delete.]	83:40-41
[Where else would the initialization expression appear? Editor: Delete "that appears ... equals".]	85:11
[Duplicates [78:39]. Editor: Delete.]	85:18
Constraint: A *declaration-type-spec* in an *implicit-spec* shall not use the CLASS keyword.	87:12+
[Editor: After "same" insert "kind".]	90:45
[Editor: Insert a space between "[" and "*common...*".]	92:38
[The phrase "use association or" contradicts the constraint at [93:7]. Delete it.]	93:40
[Editor: Before "type parameters" insert "kind" thrice.]	94:17,18,21

2 Potential problems with no edits offered

The assertion that "*All* of a [data object's] attributes may be included in a type declaration statement..." will not be true if the answer to interpretation 90 that is described in 01-138r1	65:5-6

stands.	
There appears to be no reason for the "that has a *language-binding-spec*" part. I don't see why VALUE wouldn't work just fine for Fortran subprograms.	67:27
"If the kind ... default integer" duplicates [34:1-2].	69:7-8
"If the kind ... default real" duplicates [36:1-2].	69:11-12
"The kind ... (0.0D0)" duplicates [36:4].	69:15
"If the kind ... default complex" duplicates [37:2-3].	69:21-22
"If the kind ... default character" duplicates [38:1-2].	70:28-30
Duplicates [40:1-3].	70:28-30
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.	71:14-17
Why is "base object" here? If it needs to be here, insert "a" before "variable".	72:9
Where else might a *bind-spec-list* appear?	72:39-40
"Shape" should be "bounds".	73:10
"*explicit-shape*" and "*deferred-shape*" should be "*explicit-bounds*" and "*deferred-bounds*" here, everywhere else these syntax terms appear, and everywhere the non-syntax terms similar to them appear.	73:16,18
Is the concept of "defined" defined for anything other than a variable or a pointer association status?	74:35,39
The specs really said "disassociated"! This would be cool, but almost certainly "disassociated" should be "undefined". Evidence for this appears at [257:27] and [354:5].	76:43
The essence of note 5.16 supports the answer to interpretation 31 proposed in paper 01-166.	77:17-29
This only says when a pointer can't be referenced. Do we assume the contrapositive to be true? If so, this supports the answer to interpretation 31 proposed in paper 01-166.	79:2-3
If the advice implied by the remark for 67:27 above is accepted, insert "and the procedure has a *language-binding-spec*" after the first "argument".	80:5
The difference between the effect of VOLATILE on allocatable entities and their allocation status should be described.	80:24+
Do we need to say anything about deferred or assumed type parameters?	87:12+
The term "base object" appears to be defined only for structures. If that's true, what does the constraint mean?	90:21
Is it really possible to put a host-associated object into a common block? How could that possibly work?	93:39-40
Can pointers with deferred type parameters be in common? If so, can a pointer with deferred type parameters be "common associated" with a pointer that has nondeferred type parameters.	94:18-19