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".] [Editor: Delete.]	 66:34 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 appearence 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: "effector" ⇒ "affector".]	72:13
[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
[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 "The ... definition statement." because it's redundant.]	74:30-32
[Simplification:] Editor: Insert "or a disassociated array pointer" after "array". Editor: Delete "lower and upper" then start a new paragraph with "The bounds...." After "dimension" insert "of an allocatable array" Editor: Delete "The size ... 13.1." [Editor: Delete "They are specified" twice, because that's what the previous sentence says.	 74:35 74:37 74:39-41 74:43-44
[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: "An" ⇒ "The"; "the" ⇒ "a".]	76:16
[A dummy argument is not a type, derived or otherwise. Editor: After "type" insert "object".]	77:18
[Editor: Delete. See [65:5-6].]	78:22
[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. The initial association status of a pointer *data-stmt-object* is disassociated. [Editor: Delete.]	83:33-34 83:40-41
[Where else would the initialization expression appear? Editor: Delete "that appears ... equals".]	85:11
[Editor: Insert a space between "]" and "*common...*".]	92:38
[The phrase "use association or" contradicts the constraint at [93:7]; the rest of the sentence is wrong, too. Delete the sentence.]	93:39-40

2 Potential problems with no edits offered

Only needed if the argument has a *language-binding-spec* 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.	67:24-25 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
[Editor: Either after "same" insert "kind", or deconstraintify.]	90:45
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