Subject: Comments on Section 16, Issues 319 and 336
From: Van Snyder

1 Edits – Organization

Edits refer to 01-007r3. 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 subdivision of the section according to [366:8-13] is not effective. Numerous entities are described in subclause 16.1 that are not names. Instead, classify according to whether entities are global etc.]

[Editor: Delete "If ... called".] 365:3

- A **global entity** is an entity that has a scope of a program; 365:4-6
- A **local entity** is an entity that has a scope of a scoping unit;
- A **construct entity** is an entity that has a scope of a construct;
- A **statement entity** is an entity that has a scope of a statement or part of a statement.

[Prepare the way for listing how entities are identified. Editor: "generic name ... specification" ⇒ "*generic-spec* in an *interface-stmt*".] 244:47-48

[Just cleaning up: Editor: "the keyword ... is" ⇒ "a *dtio-generic-spec* is".] 245:23-24

(1) A name (3.2.1), 365:8-13
(2) A statement label (3.2.4),
(3) An external input/output unit number (9.4),
(4) An identifier of a pending data transfer operation (9.5.1.8, 9.6),
(5) A generic identifier (12.3.2.1), or
(6) A binding label (12.5.2.7, 15.2.7.1).

[Editor: Delete unresolved issue 319.] 365:14-19

[Editor: Decapitate subclause 16.1. Raise subclauses within 16.1 one level.] 365:22-23

16.1 Scope of global entities 365:24

External input/output units and pending data transfer operations are global entities. 366:12-

---

16.2 Scope of local entities 365:24

[Editor: "identifiers" ⇒ "names" because operators and the assignment symbol apparently aren't in class (1).] 366:16

[Editor: "and namelist group names" ⇒ namelist group names, and statement labels".] 366:17

[Editor: "a ... identifier" ⇒ "an identifier of" (because we added labels to class 1).] 366:26

[Common blocks are global entities. Everything else in the paragraph is a local entity. The 367:12

subclause is part of 16.2 Scope of local entities (formerly 16.1.2 Local entities).]

16.2.1 Local entities that have the same names as common blocks

[16.1.2.4 has nothing to do with scope, association or definition. Editor: Move the entire subclause to [259:12+], making it 12.4.4.]	368:35ff
[Now covered elsewhere. Editor: Delete.]	372:34-41
[Now covered elsewhere. Editor: Delete, including unresolved issue note 336.]	373:1-13
[Editor: Move to [383:17+], to be near other stuff about allocatable.]	382:45-46

2 Edits – ordinary stuff

Edits refer to 01-007r3. 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 subclause has nothing to do with procedure reference. Editor: Change the title to "**Restrictions on generic declarations**".]	367:24
[Simplification: Editor: "name ... assignment" ⇒ "identifier".]	367:26-27
[Simplification: Editor: "local ... assignment" ⇒ "generic identifier".]	367:32-33
[Editor: Insert "derived" before "type" four times.]	371:19,22,25
[Editor: Insert "or a generic binding for which the *generic-spec* is a *generic-name*" after "name".]	371:25
A generic binding for which the *generic-spec* is not a *generic-name* has a scope that is the scoping unit containing an object of the type.	371:26+ New ¶
[Editor: Delete the comma.]	371:28
[Editor: Insert "is a scalar that" after "It".]	372:6
[Editor: Insert "; it has no other attributes" after "function".]	372:7
[Editor: "for" ⇒ "governing" (see [373:23]). Delete "the" twice (but not "The").]	373:36
[Run-on sentence. Editor: "and" ⇒ "; it".]	373:38
[Run-on sentence. Editor: second "and" ⇒ "; they".]	374:1
[Editor: "or an" ⇒ ", *procedure-declaration-stmt*, or".]	374:6
[Run-on sentence. Editor: "and" ⇒ ";" twice.]	374:7,30
[Editor: Italicize all of "*type-alias-stmt*" (the "ty" isn't currently italicized).]	374:25
[Editor: "to" ⇒ "with" twice.]	376:5,12
If the pointer has deferred type parameters or shape, their values are assumed from the target. If the pointer is polymorphic, its dynamic type is the dynamic type of the target.	376:24+ Same ¶
[The note is about pointer association status in particular, not pointer association in general. Editor: Move to [376:39+], after inserting "the association status of" after "considers" and replacing "in an undefined state" by "undefined" at [376:31].]	376:25-34

[Editor: Move "(7.5.2)" to be after "pointer-assigned" necause that's what it applies to, not "disassociated".] 377:5

[Editor: "an object" ⇒ "a nonpointer nonallocatable object" (see [383:25-27]).] 377:6

[Editor: Delete because they're part of (d).] 377:8,10

[Editor: Insert "is invoked" after "procedure"; "INTENT ... invoked" ⇒ "an actual argument corresponding to a dummy argument with INTENT(OUT)".] 377:9

[Editor: "a local" ⇒ "a nonsaved local" (see [383:25-27]).] 377:11

(5) The pointer is an ultimate component of an object of a type for which default initialization is specified for the component and the object is allocated. 377:13

[Editor: Insert ", default initialization is not specified for the component," before "and".] 377:22

[Association status is *not* determined by the target; definition and association status are that of the *target*. Editor: "are determined by its target" ⇒ "become those of the specified *data-target* or *proc-target*".] 377:41

[Self contradictory paragraph.] 379:29-31
A common block is permitted to contain a sequence of differing storage units. For a named common block, each scoping unit that accesses the common block shall specify an identical sequence of storage units. A blank common block may be declared with differing size in different scoping units, in which case the sequence of storage units of the shorter block declaration shall be identical to the initial sequence of the storage units of the longer one.

5.5.2.3 mentions "common blocks with the same name" and "blank common blocks". Then in a note it switches to singular. The text the above paragraph replaces mixes singular named common block and plural blank common blocks. The above paragraph assumes a blank named common block is the same entity no matter how many times it is declared. Which way do we want it?	*Note to J3*

[Run-on sentence. Editor: "and" ⇒ "; it".] 380:52

["either" implies that the following list encompasses all of the possibilities. Editor: Delete ", either".] 380:35

[Number doesn't agree. Editor: "that" ⇒ "those".] 380:36

[Editor: Insert "have" after "to".] 382:8

[Editor: "an unsaved" ⇒ "a".] 383:25
[Editor: Insert "that does not have the SAVE attribute" after "variable".] 383:26

[Editor: "an" ⇒ "a nonpointer nonallocatable".] 383:28

[*namelist-group* is not a defined syntax term. Editor: "*namelist-group*" ⇒ "*namelist-group-name*".] 384:19

[Editor: Make it a note and move to [382:17+], because it's just a procedure invocation, and that's adequately covered elsewhere in the list.] 384:25-26

[Editor: Insert "that has a subcomponent" after "object" (because it's the subcomponent, not the object that can have default initialization).] 384:42

[Editor: "object" ⇒ "subcomponent".] 384:43

[Editor: "the variable" ⇒ "any variables"; "the IOSTAT=" ⇒ "an IOMSG= or IOSTAT=". Delete ", if any".]	384:45-46
[Editor: "veriable" ⇒ "variable".]	385:23
[*internal-file-unit* is no longer a syntax term. Editor: "*internal-file-unit*" ⇒ "*internal-file-variable*".]	385:34

3 Don't know what to do

Do we want to move 5.5.1.1 Equivalence association and 5.5.2.3 Common association to 16.7.3? Or do we want to add something like "It may be equivalence association (5.5.1.1), common association (5.5.2.3) or the result of a function having ENTRY statements, and therefore several result names" at [378:4+]?

Do we want "dummy argument" to be "associated actual argument"?	383:29
What does "associated" mean in this context? Is it a concept analogous to pointer association but not yet defined? Are there any other events that cause associated variables of type C_PTR to become undefined?	385:19-23