Subject: Unresolved issues 78 and 141
References: 99-141r1
From: Van Snyder

1 Edits

Edits refer to 99-007r2. 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 the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

[Editor: Add a new paragraph.] 82:2+

Deferred parameters of an undefined or disassociated pointer are undefined [new section] (14.8). The definition status of deferred parameters of a pointer object is the same as the definition status of the corresponding parameters of the object with which it is associated. If the deferred parameters of the object with which the pointer is associated are defined, the values of those parameters of the pointer object are the same as the values of corresponding parameters of the object with which it is associated.

The above is intended to cover both pointer assignment and argument association.	*Note to J3*

Constraint: All nondeferred type parameters of *pointer-object* shall have the same values as the corresponding type parameters of *target*. 155:33+

Constraint: If *pointer-object* is a data object, all deferred or assumed parameters of *target* shall correspond to deferred parameters of *pointer-object*.

Otherwise, a run-time check is required to make sure the nondeferred parameter value of *pointer-object* is the same as the deferred or assumed parameter value of *target*.	*Note to J3*

[Editor: Delete; see 82:2+ and constraints added at 155:33+.] 156:11-31

[Editor: Replace "when it is invoked" by:] 156:34

, if it returns an allocated allocatable result or an associated pointer result.

[Editor: delete.] 156:37-48

[Editor: Add a new section 14.8:] 379:10+

14.8 Definition and undefinition of deferred type parameters

A deferred type parameter of an object may be defined or may be undefined and its definition status may change during execution of a program. An action that causes a deferred type parameter to become undefined does not imply that it was previously defined. An action that causes a deferred type parameter to become defined does not imply that it was previously undefined.

The definition status of deferred type parameters is changed by the following events:

1. Successful execution of an ALLOCATE statement or allocation of an allocatable component during intrinsic assignment (7.5.1.5) causes deferred type parameters of the allocated object to become defined.

2. Pointer assignment causes the values of deferred parameters of *pointer-object* and its ultimate components to assume the definition status of corresponding parameters of *target* and its ultimate components.

3. Events that cause a pointer to become undefined (14.6.2.1.3) or to become disassociated (14.6.2.1.2) cause the deferred type parameters of the pointer to become undefined.

4. Deallocating (6.4.3) a pointer or an allocatable object causes the deferred type parameters of the object and its subobjects to become undefined. This includes deallocation of an allocatable component of a derived type object during intrinsic assignment (7.5.1.5).

5. Reference to a procedure causes the deferred type parameters of dummy arguments that do not have INTENT(OUT), and their subobjects, to assume the same definition status as corresponding type parameters of corresponding actual arguments and their subobjects.

6. Any change in the definition status of deferred type parameters of an object causes the same change in the corresponding deferred type parameters of an associated object of the same type.

If the change suggested for VALUE at 85:26 in 99-133 is accepted, add "other than a dummy arugment that has the VALUE attribute" after the first "object" in the last item above.	*Note to J3*