

Subject: Concerning deferred type parameters, including issues 78, 79, 134,  
138, 140, 141  
From: Van Snyder

## 1 Edits

Edits refer to 99-007r1. 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 the indicated line. Remarks for the editor are noted in the margin, or appear between [ and ] in the text.

### 1.1 Problems not addressed by editor's issues

The deferred parameters of the result of <code>NULL()</code> are undefined.	138:15+
Is this needed if item 3 of the proposed new section 14.8 advocated to address item 134 below is accepted?	<i>Question to J3</i>
Replace “with the same shape.” by “. If it is an array, it is allocated with the same bounds. If it has deferred type parameters, the values of type parameters of the component of <i>expr</i> are used for the values of corresponding parameters of the component of <i>variable</i> .”	157:42
If type parameters of a function result are deferred, which parameters are deferred is a characteristic.	266:29+
At the invocation of the procedure, a dummy procedure pointer becomes disassociated if it has <code>INTENT(OUT)</code> . If it does not have <code>INTENT(OUT)</code> then it receives the pointer association status of the actual argument and, if the actual argument is currently associated, the dummy procedure pointer becomes associated with the same target.	279:32+
This would not be needed if dummy procedure pointers were considered to be dummy data objects, not dummy procedures.	<i>Note to J3</i>
(7) If it has type parameters, they shall not be the subject of parameter inquiry.	281:13+

### 1.2 Issues 78 and 141

Constraint: If <i>pointer-object</i> is a data object, all deferred or assumed parameters of <i>target</i> shall correspond to deferred parameters of <i>pointer-object</i> .	159:33+
Otherwise, a run-time check is required to make sure the nondeferred parameter value of <i>pointer-object</i> is the same as the deferred or assumed parameter value of <i>target</i> .	<i>Note to J3</i>
[Editor, replace starting with “If <i>pointer-object</i> ”.]	160:12-15
The definition status of each deferred parameter of <i>pointer-object</i> is assumed from the definition status of the corresponding parameter of <i>target</i> . If parameters of <i>target</i> that correspond to deferred parameters of <i>pointer-object</i> are defined, their values are assumed by corresponding parameters of <i>pointer-object</i> .	
[Editor: delete through “related” .]	160:18-24

[Editor: delete.]

160:32:48

### 1.3 Issue 79

---

[Editor: Change the period that ends the sentence to a colon and replace “That is...” by the following:] 78:34-36

- If the actual argument is a pointer, a reference to the associated dummy data object or dummy function procedure pointer may occur if the actual argument pointer is associated with a target.
- If the actual argument is allocatable, a reference to the associated dummy data object may occur if the actual argument is allocated.
- A reference to the dummy data object may occur if the associated actual argument is defined.
- Inquiry about a type parameter of the dummy data object may occur if the corresponding type parameter of the associated actual argument is defined.
- The dummy data object, its pointer association status and deferred type parameters, or allocation status and deferred type parameters may be changed if the associated actual argument is definable.

---

[Editor: delete.] 276:39-41

---

[Editor: delete.] 277:1-16

---

[Editor: Add the following after “target.”:] 277:37

Deferred or assumed type parameters of the dummy argument become associated with corresponding type parameters of the actual argument.

### 1.4 Issue 134

---

A **deferred type parameter** is a nonkind type parameter for which an expression to calculate a value is not specified in the declaration of an object. A type parameter is indicated to be deferred by using a colon in a type declaration statement or a component definition statement. Values of deferred type parameters of an object become defined or undefined as specified in 14.8. 32:34+

---

[Editor: delete. Was in the wrong place, anyway.] 32:41-33:19

---

[Editor: delete.] 33:8-19

---

[Editor: Add a new section 14.8:] 379:11+

#### 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. Execution of a pointer assignment statement 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. This includes pointer assignments that result from intrinsic assignment of objects of derived type that have components with the `POINTER` attribute.
3. Events that cause a pointer to become undefined (14.6.2.1.3) or to become disassociated (14.6.2.1.2) except by deallocation 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` in 99-133 is accepted, add “other than a dummy argument that has the `VALUE` attribute” after the first “object” in the last item above.

*Note to J3*

## 1.5 Issue 138

---

[Editor: delete – covered by solution to issue 140.]

88:1-25

## 1.6 Issue 140

---

A deferred type parameter of a disassociated pointer, of a function procedure pointer, of a pointer with undefined association status, or of an unallocated variable shall not be the subject of a type parameter inquiry.

108:22-23

---

[Editor: delete.]

108:24-33