Date: 29 October 1998
To: J3
From: Van Snyder
Subject: Edits for explicitly typed allocations
References: 98-146, 98-160, 98-172r3, 98-193r1

1 Background

There are change bars in the margin to indicate the differences between 98-208r1 and 98-208r2.

In Fortran 95, character length cannot be deferred until allocation. In Fortran 2000, without change, one will in addition not be able to defer specifying parameters of parameterized derived types until allocation, nor will one be able to allocate a polymorphic object with a type extended from its declared type. This paper addresses those deficiencies.

Specifications and syntax were proposed in 98-172r2, and approved with minor amendments. A post-meeting 98-172r3 included the amendments. 98-193r1 attempted to provide edits, but failed to address the need to allow more general expressions than specification expressions in explicitly typed allocations, and it was therefore withdrawn.

2 Edits

Edits refer to 98-007r3. 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 the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

[Editor: Change "...or assumed" to "...or be assumed or deferred".] [30:28]

[Editor: Start a new section after note 4.4. Add "deferred" to the index.] [30:33+]

4.2.1 Deferred type parameters

A **deferred type parameter** is a nonkind type parameter of an entity for which specifying the value is deferred until the entity is allocated (6.3.1) or associated with a target (7.5.2). Deferred type parameters of a dummy argument are assumed from the corresponding type parameters of the actual argument. A deferred type parameter of an unallocated variable, disassociated pointer, or pointer with undefined association status, shall not be referenced. A deferred type parameter may be specified in a type declaration statement or a component definition statement. A deferred type parameter may be specified only if the entity has the POINTER or ALLOCATABLE attribute. A deferred type parameter is indicated by a colon. An entity that is declared with a deferred type parameter is a **deferred-parameter entity**.

[Editor: Replace clause beginning "it is required..." with the following:] [38:11-12]

it is required to specify values explicitly for all kind parameters of a derived type entity, and either to defer or specify values explicitly for all nonkind parameters of a derived type entity.

[Editor: Replace *type-spec* by *restricted-type-spec*] [39:45]

---

or : [53:38+]

Constraint: An asterisk or colon may be used as a *type-param-value* only for a nonkind type parameter. [54:2-3]

Constraint: A colon may be used as a *type-param-value* only if the declared entity has the POINTER or ALLOCATABLE attribute. [54:5+]

[Editor: Start a new paragraph. Add "deferred" to the index.] [56:9+]

A nonkind parameter of a derived type that is specified by a colon is a deferred type parameter (4.2.1). Specification of the values of deferred parameters is deferred until execution of an ALLOCATE statement (6.3.1), pointer assignment (7.5.2), or argument association (12.4.1).

[Editor: Replace *type-spec* by *restricted-type-spec*] [59:18]

R501a *restricted-type-spec* **is** *type-spec*
or CLASS (*extensible-type-name*)

Constraint: The *extensible-type-name* shall be the name of an extensible type.

Using *extensible-type-name* prevents parametric polymorphic entities. Should *extensible-type-name* be *derived-type-spec* with a constraint that the type named in the *derived-type-spec* shall be the the name of an extensible type?	*J3 unresolved issue*

Every *type-param-value* within a *restricted-type-spec* that is an integer expression shall be a specification expression. Note: A *restricted-type-spec* is used within a specification statement; a *type-spec* may also be used in ALLOCATE statements and array constructors.

[Editor: Delete – it's been moved to *restricted-type-spec*.] [59:26-27]

[Editor: Delete the constraint. Replace "allocatable array" by "allocatable variable" everywhere it appears.] [60:7]

R510 *char-len-param-value* **is** *scalar-int-expr* [63:2]

or : [63:3+]

Constraint: If *char-len-param-value* is colon the declared entities shall also have either the POINTER or the ALLOCATABLE attribute. [63:8+]

[Editor: Start a new paragraph. Add "deferred" to the index.] [63:18+]

A **deferred character length parameter** is a deferred nonkind type parameter ([new section] 4.2.1) of a character entity; it is specified by a colon *char-len-param-value*. Specification of the length of the character entity is deferred until execution of an ALLOCATE statement (6.3.1), pointer assignment (7.5.2), or argument association (12.4.1).

in a type declaration statement, a component definition statement, or an ALLOCATE statement. Nonkind type parameters may be deferred. [68:40]

type and type parameters may be specified in a type declaration statement, a component definition statement, or an ALLOCATE statement. Nonkind type parameters may be deferred. [69:1-2]

R623 *allocate-stmt* **is** ALLOCATE ([*type-spec* ::] *allocation-list* ■
■ [, STAT = *stat-variable*]) [96:13]

[Editor: Change "array" to "variable". [96:21, 23]

Constraint: If a *type-spec* is specified, the intrinsic type or *type-name* shall be the same type as the declared type of allocate objects that are not polymorphic, or the name of an extension type (4.5.3) of the declared type of allocate objects that are polymorphic (5.1.1.8). [96:23+]

Constraint: If the shape of an allocate object is deferred, *allocate-shape-spec-list* shall be specified.

[Editor: Start a new paragraph.] [96:25+]

When an ALLOCATE statement is executed type parameters may be specified by the values of expressions given in the *type-spec* in the allocate statement. A value shall be specified for every deferred type parameter. If a value is specified for a nondeferred type parameter, it shall be the same as the value specified in the object's declaration. Otherwise, an error condition

exists.

Type parameter values are associated with type parameter names as specified in 4.5.5. If a type parameter value is associated by position with a type parameter name, a type parameter value shall be associated by position with all type parameter names that appear earlier in the type parameter name list (including type parameter names inherited from an extensible type's parent type).

Subsequent redefinition or undefinition of any entity within any expression that provides a value for a deferred type parameter does not affect the value of the parameter.

If a type is specified, allocation of a polymorphic object (5.1.1.8) allocates an object with the specified dynamic type; otherwise it allocates an object with the declared type of the object.

If an *allocate-shape-spec-list* is specified for an array that is not a deferred-shape array, the bounds specified shall be the same as the bounds for the object; otherwise, an error condition exists.

[Editor: Change "bound" to "bound or type parameter value".] [96:26]

[Editor: Change "At the time" to "When" and change "array" to "deferred-shape array".] [96:34]

[Editor: Delete (it was moved to 96:25+).] [97:1-2]

[Editor: add "type parameter" to the list – replace "or substring ending point" by "substring ending point, or type parameter".] [111:38]

[Editor: Start a new paragraph] [130:5+]

If *pointer-object* has deferred nonkind parameters, the values of those parameters are assumed from the values of the corresponding parameters of *target*. All other type parameters of *target* shall have the same values as corresponding parameters of *pointer-object*.

[Editor: Change "character length is assumed" to "character length is assumed or deferred"] [230:18]

[Editor: Change "or a target" to ", a target, or has a deferred type parameter".] [231:22]

[Editor: Replace "type parameters" by "non-deferred type parameters".] [241:9-10]

[Editor: Replace "type parameters" by "non-deferred type parameters".] [242:9]

[Editor: Add new paragraph after Note 12.21] [242:42+]

If a dummy argument is a deferred-parameter entity and does not have INTENT(OUT) the values of the deferred type parameters are assumed from the corresponding type parameters of the actual argument.

The corresponding actual argument shall have deferred the same type parameters as the dummy argument.

---

deferred type parameter (4.2.1) A derived type parameter specified by a colon. [361:34+]

deferred-parameter entity (4.2.1) A derived type or character entity having a deferred type parameter.