Subject: Issues 293, 298 and 299, some of issue 290, other problems with user-defined DTIO
From: Van Snyder

1 Introduction

In addition to issues 290, 293, 298 and 299, there are several other problems with user-defined derived-type input/output.

Type-bound derived-type input/output is specified to be secondary to derived-type input/output defined by interface blocks. This usurps the authority (recently learned terminolology) of the NON_OVERRIDABLE attribute of a type-bound procedure. One should not be able to override a type-bound derived-type input/output procedure, except by extending the type, and overriding the binding – *but only if the existing binding does not specify the NON_OVERRIDABLE attribute.*

In order to respect the NON_OVERRIDABLE attribute, combine type-bound- and interface-block-specified user-defined derived-type input/output procedures for a single *dtio-generic-spec* into a single generic interface, and define overriding and generic extension in relation to this combination.

A small amount of the description of generic resolution of user-defined derived-type input/output procedures is in 9.5.4.4.3; it all belongs in section 14.1.2.4.3.

Where this paper and 00-311r1 specify edits in the same place, this paper takes precedence over paper 00-311r1.

2 Edits

Edits refer to 00-007r3. 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 for the editor are noted in the margin, or appear between [and] in the text.

[Editor: "NON_OVERRIDABLE" ⇒ "*binding-attr-list*".] 41:25

Constraint: No *proc-binding-stmt* shall specify a binding that overrides (4.5.3.2) one that is inherited (4.5.3.1) from the parent type and that has the NON_OVERRIDABLE binding attribute. 41:33+

Constraint: PASS_OBJ shall not be specified in a GENERIC *proc-binding-stmt* with a *dtio-generic-spec.* 42:33+

[Editor: "The set ... defines" ⇒ "A binding with a particular *dtio-generic spec* may override one inherited from the parent type (4.5.3.2). Otherwise it extends the generic interface for that *dtio-generic-spec*. The set of all accessible nonoverriding GENERIC bindings for a particular *dtio-generic-spec*, and the specific procedures specified in accessible interface blocks with the same *dtio-generic-spec*, define".] 48:21-24

[Editor: Delete "It is not ... type".] 54:31-32

If a generic binding in a type definition has the same *dtio-generic-spec* as one inherited from the parent, and the **dtv** argument of the procedure or abstract interface it specifies has the 55:22-23

same kind type parameters as the **dtv** argument of one inherited from the parent type, then the

[Editor: delete “type ... specified”.] 55:26

[Editor: Delete unresolved issue note 293.] 55:27-42

[Editor: Move “It is not ... parent type” from [54:31-32] to here.] 55:42+

[Editor: delete “based ... transferred,” (redundant to 14.1.2.4.3).] 191:19-21

[Editor: after “procedure” insert “or dummy procedure pointer”.] 191:23

[Editor: After “possible” insert “sets of characteristics for”.] 192:18

In the following four interfaces that specify the characteristics of user-defined procedures for derived-type input/output, the following syntax term is used: 192:19-20

dtv-type-spec	**is**	TYPE(*derived-type-spec*)
	or	CLASS(*derived-type-spec*)

Constraint: If *derived-type-spec* specifies an extensible type the CLASS keyword shall be used; otherwise, the TYPE keyword shall be used.

Constraint: All nonkind type parameters of *derived-type-spec* shall be assumed.

[Editor: “*genric*” ⇒ “*generic*”, “the interface shall be” ⇒ “the characteristics shall be the same as specified by the following interface”.] 192:21

[Editor: “`TYPE(whateveritis)`” ⇒ “*dtv-type-spec*”.] 192:31

[Editor: “*genric*” ⇒ “*generic*”, “the interface shall be” ⇒ “the characteristics shall be the same as specified by the following interface”.] 192:21

[Editor: “`TYPE(whateveritis)`” ⇒ “*dtv-type-spec*”.] 192:31

[Editor: “*genric*” ⇒ “*generic*”, “the interface shall be” ⇒ “the characteristics shall be the same as specified by the following interface”.] 192:21

[Editor: “`TYPE(whateveritis)`” ⇒ “*dtv-type-spec*”.] 192:31

[Editor: “*genric*” ⇒ “*generic*”, “the interface shall be” ⇒ “the characteristics shall be the same as specified by the following interface”.] 192:21

[Editor: “`TYPE(whateveritis)`” ⇒ “*dtv-type-spec*”.] 192:31

[Editor: delete “and the dummy argument names”.] 193:28-29

[Editor: adjoin to the paragraph at [193:28-29].] 193:30-34

In discussion here and elsewhere the dummy arguments in these interfaces are referred by the names given above; the names are, however, arbitrary.

In addition to the characteristics specified by the above interfaces, the **dtv** argument may optionally have the VOLATILE attribute.

[Editor: “list” ⇒ “effective”.] 193:41

[Editor: delete the “.” within the word “condition”.] 194:33

[Editor: “If” ⇒ “When” twice.] 195:1,4

[Editor: “usually in the middle” ⇒ “not necessarily at the beginning” (the standard shouldn’t 195:18

specify the statistical behavior of programs).]

[Editor: before "subsequent" insert "a", or "'s" ⇒ "s'" (plurality doesn't agree).] 195:29

[Editor: delete "Also ... io".] 350:29-34

When an effective item (9.5.2) that is of a derived type is encountered during a data transfer, if 350:35-43

- there is a generic interface that is specified using a *dtio-generic-spec* (12.3.2.1) that is appropriate to the direction (read or write) and method (formatted, list-directed, namelist or unformatted) of the data transfer as specified in 9.5.4.4.3,
- the **dtv** argument of a specific procedure or procedure pointer in that generic interface, or of a procedure or abstract interface specified in a generic binding in that generic interface, is type-compatible with the effective item,
- the kind type parameters of the effective item have the same values as corresponding kind type parameters of that **dtv** argument,
- in the case of an input/output statement that has a *format-specification*, if the effective item's corresponding edit descriptor is a DT edit descriptor, and
- the specific procedure or procedure pointer is specified in an interface block, or the specific procedure or abstract interface is specified by a nonoverridden (4.5.3.2) binding

then the reference is to the specific procedure or procedure pointer in the interface block that provides that interface, or the specific procedure or abstract interface specified by the binding.

If the first four of the requirements above are satisfied but the fifth is not, the reference is to the procedure or abstract interface specified by the overriding binding.

[Editor: delete unresolved issue notes 298 and 299.] 351:1-18

NOTE 14.10$\frac{1}{2}$ 351:19-21

If the reference is to an abstract interface, a dummy procedure or dummy procedure pointer that is not present, or a disassociated procedure pointer, an error condition occurs.

If any of the first four requirements are not satisfied, intrinsic input/output is used instead of user-defined derived-type input/output.

[Editor: "Unlimitted" ⇒ "Unlimited".] 413:33

[Editor: "polymorphiv" ⇒ "polymorphic".] 413:35