Subject: Miscellaneous items
From: Van Snyder
References: 00-179

Here are several things that may or may not need attention. I don't even offer edits (well, sometimes I offer crappy ones). If they need attention, we can develop edits at the meeting, if we have time, or insert unresolved issue notes.

Given the stipulation at [342:4] that a type name is a name in class (1), and the requirement at [342:13-15] that a name in one class cannot be used to identify another entity in the same class, the "nor ... *type-name*" part of the constraint is unnecessary.	42:6-7
The *declaration-type-spec* includes "CLASS(...)". Was it the intent to allow "TYPE" aliases for classes? I think not – otherwise we could have `TYPEALIAS FOO => CLASS(BAR); TYPE(FOO)` declare a polymorphic object. The "*declaration-type-spec*" should be "*type-spec*".	57:36
Given the stipulation at [342:4] that a type alias is a name in class (1), and the requirement at [342:13-15] that a name in one class cannot be used to identify another entity in the same class, the "nor ... *type-name*" part of the constraint is unnecessary.	57:38-39
A type alias name cannot be used as the parent type name in an extended type declaration. Perhaps something should be said here., e.g. "A type alias name shall not be used to specify the parent type of an extended type."	57:42-43
Shouldn't the constraint be the following? Constraint: In a *declaration-type-spec*, every *type-param-value* that corresponds to a nonkind type parameter shall be a *specification-expr*, and every *type-param-value* that corresponds to a kind type parameter shall be an *initialization-expr*.	63:20-21
Shouldn't this paragraph be a constraint?	82:9-12
We find here "A user-defined derived-type input/output procedure is any procedure...." I think we do not intend to allow internal and dummy procedures, or procedure pointers. The sentence has other problems as well: It isn't enough for a procedure to have the appropriate interface; it needs to be specified in the appropriate interface block. The sentence doesn't contribute anything that's not said elsewhere in the section. Delete it. If not, at least make it consistent with the constraint at [246:30-31]. Also note that one of the proposals in paper 00-179 is to replace the interface-block-based derived-type input/output procedure specification by one based on type-bound procedures.	189:26
Everything in 11.1.2 is said elsewhere, frequently as a constraint. Can we delete section 11.1.2?	237:42-45
Not needed – it's covered by 14.1.2.3.	246:35-36
Do we need to add "accessible" after "entity," or was the intent to restrict IMPORT to work only for entities declared within the scoping unit containing the interface body?	246:39
The "otherwise" part is not true for abstract interface blocks.	247:3-4
We may want to point out in a comment that because argument B1 has assumed shape and argument B2 does not, a non-contiguous array section can be the actual argument associated with B1, but a non-contiguous array section cannot be the actual argument associated with B2.	250:4-5
It would be convenient to be able to use any accessible explicit interface to declare the interface for a procedure pointer. Could we add "**or** *procedure-name*" as an additional right-hand side for R1211? We would also need to replace "consists ... pointers" by "and specifies an explicit	252:19+

specific interface, the declared procedures or procedure pointers have the same explicit specific interface" at [253:7].

The phrase "an elemental intrinsic actual procedure may be associated with a dummy argument that is not elemental" leads one to believe that dummy arguments can be elemental. The part "that is not elemental" should be removed. Three possibilities for what to do next are (1) nothing, (2) add a parenthetic remark "(which cannot be elemental)", or (3) put in a note $12.27\frac{1}{2}$ to the effect that dummy arguments cannot be elemental. 260:9-10

We could get rid of "other than as the argument of the PRESENT intrinsic function" by making the argument of the PRESENT intrinsic function optional. 261:12-23

I think the reason for this condition is to provide bounds for the elemental-ness. If so, this condition is too strong (the dummy argument of the elemental procedure can't be optional), and not strong enough (the specified array doesn't necessarily provide the desired bounds). It should be "... unless an array of the same rank that is (1) not a dummy argument or is a present dummy argument, (2) not an unallocated allocatable array, and (3) not a disassociated pointer, is supplied as an actual argument of that elemental procedure." 261:15-17

There is at least one, and maybe two problems here. In the phrase "correspond by name to a dummy argument not present" does "not present" mean "not declared," or "it has the optional attribute and there is no associated actual argument?" I think it's the former, but we do have a section with the phrase "dummy arguments not present" in its title – and it refers to the latter. The wording should be revised to avoid this confusion. In the former case, it is impossible for a nonoptional dummy argument to correspond by name to a dummy argument not present. The dummy argument that is not present clearly doesn't have a name. 343:34-35

The sentence "If a generic..." conflicts with, or at least belongs in [344:25-26]. 343:42-44

Not needed, because of [344:40] and the new language in 5.1.2.10 that specifies that an interface body confers the EXTERNAL attribute. Perhaps [344:40] should be re-worded "(d) if there is an explicit specification of the EXTERNAL attribute (5.1.2.10) in that scoping unit". 344:35