Date: 9 October 1998
To: J3
From: Van Snyder
Subject: Comments on Richard Maine's comments in 98-204 and 98-205 concerning 98-184r2

This is an attempt to remedy, at least in part, some of the unresolved issues mentioned in 98-204 and 98-205. Unless otherwise specified, references are to 98-007r3.

General question for the editor: Should syntax terms be set in italic type when they appear in the text? They are italic in section 14.1.3, but not in 8.1.4.2.

1 Issues in 98-204

1.1 14.1.2 – concerning edits to 98-007r2 at 303:34

The *associate-name* must be defined not to be a "local entity of the scoping unit," because the block in a select type construct isn't a scoping unit. If *associate-name* is not defined not to be a "local entity of the scoping unit," it "leaks out" of the block to the entirety of the scoping unit in which the select type construct is contained. The alternative is to define blocks in select type constructs to be scoping units, but this allows declarations therein. (BTW, I think *all* blocks ought to be scoping units, but this got voted down during requirements analysis.)

2 Issues in 98-205

2.1 Unresolved issue 53 – page 51

The question of more ubiquitous use of genealogical terminology, e.g. use "child type of..." or "descendant type of..." instead of "extension type of..." would seem also to beg to replace the intrinsic procedure `EXTENDS_TYPE_OF` by `DESCENDANT_TYPE_OF`. Such a change, even if only to the terminology in the text, ought to be undertaken only after a debate, at least within the data subgroup, followed by a straw vote.

2.2 Unresolved issue 54 – page 142

At 142:1 and 142:4-8 change *end-select-stmt* to *end-select-case-stmt* and change *case-construct-name* to *select-construct-name*. Edit

Replace the sentence that begins at the end of 142:8 by "If a *case-stmt* specifies a *select-construct-name*, the corresponding *select-case-stmt* shall be identified by the same *select-construct-name*." Edit

At 144:35 change *end-select-stmt* to *end-select-type-stmt*. Edit

At 145:5+ add Edit

R818a *end-select-type-stmt* **is** END SELECT [*case-construct-name*]

Constraint: If the *select-type-stmt* of a *select-type-construct* is identified by a *select-construct-name*, the corresponding *end-select-type-stmt* shall specify the same *select-construct-name*. If the *select-type-stmt* of a *select-type-construct* is not identified by a *select-construct-name*, the corresponding *end-select-type-stmt* shall not specify a *select-construct-name*. If a *type-guard-stmt* specifies a *select-construct-name*, the corresponding *select-type-stmt* shall be identified by the same *select-construct-name*.

2.3 Unresolved issue 55 – page 145

If the expression is *not* evaluated, its dynamic type isn't known. I think it's *necessary* to evaluate the expression.

2.4 Unresolved issue 56 – page 145

At 145:30, "extents" should be "bounds." Edit

At 145:30, "assigned" should be "appears in a variable definition context (14.7.7)." In parallel Edit
with this change, the first sentence of 14.7.7 at 335:26-27 should be re-worded to mention Edit
constraints instead of prohibitions. E.g. "The appearance of a variable in some contexts that would imply definition or undefinition of the variable (5.1.2.3, 8.1.4.2, 12.6) implies additional constraints." At 145:31, "assignable" could be changed to "definable," which is apparently used Edit
with the same meaning at 242:20, but with an apparently different meaning at 335:37 (perhaps the latter should be changed). Neither of these usages appears to correspond to 2.5.4, which is referenced from the glossary ("definable" isn't in the index). The final issue is addressed in the final paragraph of 14.1.3. If it is necessary to add something about the scope of the *associate-name*, it should be a reference to 14.1.3.

2.5 Unresolved issue 57 – page 145

The desire for something like Pascal's `WITH` construct is only one part of the motivation for allowing non-extensible types in `SELECT TYPE` constructs. The other part is that it seems easier to allow than to prohibit. Every time I read a constraint that says "You can't do that" I wonder what would be the consequences of allowing it. If they're neutral, or, as in this case, useful, I see no reason to prohibit it. One possibility is to allow non-extensible types, but with no `TYPE ...` blocks. This seems, however, to be equally as irregular as prohibiting the use of non-extensible types.

2.6 Unresolved issue 58 – page 146

At 146:1-3, replace "type guard statement" by "*type-guard-stmt*" everywhere it appears. Edit

Replace the paragraph at 146:4-10 by "If the dynamic type of the expression is not the same Edit
as the type named in any TYPE IS type-guard-stmt, the dynamic type of the expression is an extension type of the type named in a TYPE IN type-guard-stmt, and the dynamic type of the expression is not an extension type of the type named in another TYPE IN type-guard-stmt for which the type named in the second TYPE IN type-guard-stmt is an extension type of the type named in the first TYPE IN type-guard-stmt, then the block following the TYPE IN type-guard-stmt is executed. Within the block, the associate-name is a polymorphic variable (5.1.1.8) of the class named in the TYPE IN type-guard-stmt."

2.7 Unresolved issue 59 – page 146

Maybe this is an improvement: Replace the sentence at 146:24-25 by "Within the block, the associate-name is a non-polymorphic variable that has the type that the expression would have if all of the objects of which it is composed were non-polymorphic." Edit

2.8 Unresolved issue 60 – page 324

Maybe this is an improvement: Replace the paragraph at 324:21-22 by "A name that appears as an *associate-name* in a SELECT TYPE statement has a separate scope for each block in the *select-type-construct*. Within each block, it is a variable having the type or class, type parameters and bounds specified in 8.1.4.2." Edit