Subject: PASS_OBJ is ugly syntax, and semantics could be better, too
From: Van Snyder

1 Introduction

I have two objections to PASS_OBJ.

First, it's an ugly keyword, that is unlikely to mean anything to people reading a code who aren't experts in the nuances of Fortran.

Second, where a type-bound or object-bound procedure is invoked, if it has a dummy argument of the type to which it is bound, it is likely that the programmer will want to pass the object by which the procedure is invoked to that argument. So the default is upside-down.

This paper proposes to replace PASS_OBJ with OBJECT and NO_OBJECT, and with more sensible defaults. There are two possibilities concerning the interface of the bound procedure:

(1) If the interface has no argument of the type to which it is bound, there is no passed-object dummy argument. It is permitted to confirm this by specifying a NO_OBJECT attribute. It is not permitted to specify the OBJECT attribute.

(2) If the interface has an argument of the type to which it is bound, then

  (a) It is permitted to specify that there is nonetheless no passed-object dummy argument by specifying the NO_OBJECT attribute.

  (b) If neither OBJECT nor NO_OBJECT is specified, the first argument that has the same type as the type to which the interface is bound is the passed-object dummy argument.

  (c) If OBJECT is specified, it confirms that the first argument that has the same type as the type to which the interface is bound is the passed-object dummy argument.

  (d) If OBJECT(*argname*) is specified, *argname* is required to be the name of an argument that has the same type as the type to which the interface is bound, and it is the passed-object dummy argument.

Three other problems repaired are that C438 [45:32] and C453 [47:1-5] use "dummy variable" where they ought to use "dummy argument" and C458 [47:18-20] largely duplicates C453 [47:2-5].]

2 Straw vote

If neither OBJECT nor NO_OBJECT is specified, or OBJECT is specified without (*argname*), should the first argument be required to have the same type as the type to which the interface is bound?

The edits below assume the answer is "no".

3 Edits

Edits refer to 02-007r1. 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 associated text, while a page and line number followed by + (-) indicates that associated text is to be inserted after (before) the indicated line. Remarks are noted in the margin, or appear between [and] in the text.

		or	OBJECT [(*argname*)]	45:26
		or	NO_OBJECT	
R437$\frac{1}{2}$	*argname*	**is**	*name*	45:27+

C438 (R436) If OBJECT appears without (*argname*), the procedure component shall have an explicit interface that has a scalar, nonpointer, nonallocatable dummy argument of type *type-name*. The first such dummy argument shall be polymorphic if and only if *type-name* is extensible. 45:31-33

C438$\frac{1}{3}$ (R436) If OBJECT(*argname*) appears, the procedure component shall have an explicit interface that has a scalar, nonpointer, nonallocatable dummy argument named *argname* and having type *type-name*. This dummy argument shall be polymorphic if and only if *type-name* is extensible.

C438$\frac{2}{3}$ (R436) If OBJECT appears NO_OBJECT shall not appear.

R444	*binding-attr*	**is**	OBJECT [(*argname*)]	46:38
		or	NO_OBJECT	

C453 (R440) If OBJECT appears without (*argname*), the interface specified by *interface-name* or the procedure specified by *binding* shall have an explicit interface that has a scalar, nonpointer, nonallocatable dummy argument of type *type-name*. The first such dummy argument shall be polymorphic if and only if *type-name* is extensible. 47:2-5

C453$\frac{1}{3}$ (R440) If OBJECT(*argname*) appears, the interface specified by *interface-name* or the procedure specified by *binding* shall have an explicit interface that has a scalar, nonpointer, nonallocatable dummy argument named *argname* and having type *type-name*. This dummy argument shall be polymorphic if and only if *type-name* is extensible.

Note to J3, not an edit:
The use above of *interface-name* instead of *abstract-interface-name* assumes that 02-166 passes.

C453$\frac{2}{3}$ (R440) If OBJECT appears NO_OBJECT shall not appear.

[Editor: "PASS_OBJ" ⇒ "OBJECT" twice.] 47:6,9

[Editor: Delete.] 47:7-8

[Editor: Delete "PASS_OBJ ... extensible" because it's covered by C453 and C453$\frac{1}{3}$.] 47:18-20

[Editor: "PASS_OBJ" ⇒ "OBJECT".] 52:5+6

[Editor: Replace "If ... **argument**." by the following:] 53:2-4

If a procedure pointer or type-bound procedure has a dummy argument that has the same declared type as the derived type being defined, NO_OBJECT is not specified, and OBJECT is either not specified or specified without (*argname*), the first such dummy argument is called the **passed-object dummy argument**. If OBJECT(*argname*) is specified, the dummy argument having the name given by *argname* is the **passed-object dummy argument**. If there is no such argument or NO_OBJECT is specified, there is no passed-object dummy argument.

[Editor: "specify PASS_OBJ" ⇒ "have a passed-object dummy argument".] 58:6

[Editor: "PASS_OBJ" ⇒ "OBJECT".] 58:16+6

[Editor: “PASS_OBJ is applicable” ⇒ “there is a passed-object dummy argument (4.5.1.6)”.] 266:11

[Editor: “4.5.1” ⇒ 4.5.1.6”.] 266:12

12.4.1.1 The passed-object dummy argument and argument association 266:16

[Editor: “with the PASS_OBJ attribute” ⇒ “that has a passed-object dummy argument (4.5.1.6)” twice.] 266:17,20