To: J3 J3/26-184

From: Lorri Menard & Dan Bonachea & JoR & Generics
Subject: Edits for UTI 031 (PROCEDURE_NAME with TEMPLATE and GENERIC)
Date: 2026-June-17
References: 26-007r1, 26-129r1, 26-132

Introduction:

Paper 26-129r1 presented edits to support SOURCE_FILE, SOURCE_LINE and PROCEDURE_NAME intrinsic procedures.

UTI 031 is specific to PROCEDURE_NAME, and correctly pointed out that 26-129r1 did not define the behavior for TEMPLATE or GENERIC procedures. The Editor further suggests that GENERIC procedures return the generic name, and that templated procedures be qualified by the template name.

Paper 26-132 addressed UTI 034 in PROCEDURE_NAME, specifically for interface bodies, modifying lines also modified by this paper. This one uses descriptive text to confirm which Case is being referenced, to avoid reconciliation conflicts between the two papers.

Edits:

[559:8] 17.10.2.29p5 PROCEDURE_NAME

For the set of cases where CONTEXT is absent or false, update Case(i) (reference appears in a subprogram) to explicitly specify GENERIC subprograms by adding a second sentence:

“If the subprogram is a generic subprogram, the result value is the generic name.”

[559:11] change "the result value is MAIN PROGRAM." to "the result has a value equal to 'MAIN PROGRAM'."

[559:12+]
Still within the set of cases where CONTEXT is absent or false, add a new Case between Case(iii) (MAIN program) and Case(iv) (BLOCK DATA) and renumber appropriately.

“Case(iii+): Otherwise, if the reference appears in a TEMPLATE construct, the result value is the name of the template.”

[559:13] change "and the result value is BLOCK DATA." to "and the result has a value equal to 'BLOCK DATA'."

[559:15-21] Convert the paragraph on these lines to two sentences with a list, as:

“If CONTEXT is present and has the value true, the result value depends on where the reference appears. The following rules apply in the rest of this paragraph:

- Square brackets indicate components that conditionally appear if the reference is enclosed in such a unit.
- The term *subprogram-name* indicates the name of an external subprogram, module subprogram, templated subprogram, or generic subprogram.
- The term *internal-subprogram-name* indicates the name of an internal subprogram, including any subprograms internal to a template subprogram or a generic subprogram.
- Other terms in italics represent the name corresponding to the syntax term.
- The remaining characters are literal, except for blank: a blank only appears in the result if it includes an unnamed main program or block data program unit.”

Ed note: Hyperlink terms above at your discretion.

[559:23] Append a colon ":" to the word ‘format’ at the end of the line for consistency.

[559:25] Case(ii) (reference in module/submodule) add the phrase "and is not within a TEMPLATE construct," after "module or submodule" so the sentence now reads:

"Otherwise, if the reference appears in a module or submodule and is not within a TEMPLATE construct, the result value has the format:"

[559:29] Replace this line with:

“If the main program is unnamed, *program-name* is replaced by ‘MAIN PROGRAM’.

[559:31+] Within the set of cases, after Case(iv) (external subprogram) and before Case(v) (BLOCK DATA) insert the following case:

“Case(iv+): If the reference appears in a templated subprogram or in the specification part of a TEMPLATE construct, the result value has the format:

{*template-name*}[*subprogram-name*[.*internal-subprogram-name*]]

preceded by the value that PROCEDURE_NAME(.TRUE.) would have in the enclosing scope unit.”

[559:32] Append a colon ":" to the word ‘format’ at the end of the line for consistency.

[559:34] Replace this line with:

“If the block data program unit is unnamed, *block-data-name* is replaced by ‘BLOCK DATA’.”

[560:Examples]

Note to committee: The following modifies the example to include GENERIC and TEMPLATE examples

[560:16-] Add a GENERIC declaration to MODULE a:

```
GENERIC SUBROUTINE g1 (x)
USE ISO_FORTRAN_ENV
TYPE (INTEGER ([int32, int64])), INTENT(INOUT) :: x
  PRINT *, PROCEDURE_NAME()                    ! 'G1'
  PRINT *, PROCEDURE_NAME(.TRUE.)                 ! 'A:G1'
END SUBROUTINE
```

[560:28-] Add a TEMPLATE construct with another, nested TEMPLATE construct to SUBMODULE b:

```
TEMPLATE t0{}
  CHARACTER(*), PARAMETER :: name = PROCEDURE_NAME()      ! 'T0'
  CHARACTER(*), PARAMETER :: full = PROCEDURE_NAME(.TRUE.) ! 'A/B:{T0}'
  TEMPLATE t1{}
    CHARACTER(*), PARAMETER :: name = PROCEDURE_NAME()      ! 'T1'
    CHARACTER(*), PARAMETER :: full = PROCEDURE_NAME(.TRUE.) ! 'A/B:{T0}{T1}'
     CONTAINS
     TEMPLATE FUNCTION t1_f{}()
       PRINT *, PROCEDURE_NAME(.FALSE.)            ! 'T1_F'
       PRINT *, PROCEDURE_NAME(.TRUE.)             ! 'A/B:{T0}{T1}T1_F'
     END FUNCTION t1_f
  END TEMPLATE t1
  CONTAINS
   SUBROUTINE t0_s()
    TEMPLATE t3{}
     CONTAINS
      SUBROUTINE t3_s
        PRINT *, PROCEDURE_NAME(.TRUE.)       !'A/B:{T0}T0_S{T3}T3_S'
      END SUBROUTINE t3_s
     END TEMPLATE t3
   END SUBROUTINE t0_s
END TEMPLATE t0
```

**Note to editor: Please fix case of 'keywords' in the rest of this example

[560:Unresolved Technical Issue 031]
can be deleted