Date:	10 November 1998
To:	J3
From:	Van Snyder
Subject:	Specifications, Syntax and Edits for M.23: Access to status error messages
References:	97-159, 98-172 98-173r1 98-208r2

1 Background

Specifications and three possible syntaxes to access status error messages were proposed in paper 98-173r1. Informal discussions suggested the "intrinsic procedure" approach is the preferred approach.

This paper depends on facilities proposed in papers 98-172 *Explicitly typed allocations - Rationale, Specs, Syntax, Edits* and 98-208r2 *Edits for explicitly typed allocations.*

2 Specifications

Define an intrinsic subroutine to provide access to error messages that explain conditions that cause non-zero status values to be returned by IOSTAT= or STAT= specifiers in input/output, ALLOCATE and DEALLOCATE statements. The procedure can write the message to the usual message destination, or return it in a variable. Exploit deferred-length allocation (see 98-172 and 98-208r2) to provide exactly the right length of output variable.

3 Edits

Edits refer to 98-007r3. 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 the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

13.11.4 Access status error message [263:7+]

The subroutine STATUS_ERROR_MSG returns or displays a message explaining the cause of a non-zero value returned by an IOSTAT= specifier in an input/output statement, or a STAT= specifier in an ALLOCATE or DEALLOCATE statement.

STATUS_ERROR_MSG ([STAT] [, MESSAGE] [, STMT])	Access status error messages.	[267:9+]

13.14.107+ STATUS_ERROR_MSG ([STAT] [, MESSAGE] [, STMT]) [311:35+]

Description. Access an error message that explains a non-zero value returned by IOSTAT= or STAT= specifiers in input/output, ALLOCATE and DEALLOCATE statements. Optionally display the message. Optionally return it in an argument.

Class. Subroutine

Edits accepted at meeting 147 end here.

Additional work to be completed by e-mail, and voted by two-week e-mail ballot begins here.

Arguments.

STAT (optional)	shall be scalar and of type default integer. It is an INTENT(IN) argument.

MESSAGE (optional)	shall be of type default character with rank zero or one. It may be ALLOCATABLE or not ALLOCATABLE. If it is allocatable, it is INTENT(INOUT) and shall have deferred length. If it is not ALLOCATABLE it is INTENT(OUT).

The "default character" specification for the MESSAGE argument seems to be neither necessary nor desirable. It inhibits internationalization. If a processor provides several kinds of character, it probably isn't too much extra work to provide error messages in all of those character kinds. Straw vote: retain "default" in the specification of the MESSAGE argument, replace "default character" by "character with any kind supported by the processor," or undecided? *Straw vote*

STMT (optional)	shall be scalar and of type default character. It is an INTENT(IN) assumed-length argument.

If the STMT argument is present, it shall have the value IO or MEMORY; otherwise the result is processor dependent. The value indicates the kind of statement for which the status message is to be accessed. The value specified is without regard to case or trailing blanks.

If the STAT argument is present and the STMT argument is present and has the value IO, the STAT argument shall provide a status value returned by an IOSTAT= specifier in an input/output statement. If the STAT argument is present and the STMT argument is present and has the value MEMORY, the STAT argument shall provide a status value returned by a STAT= specifier in an ALLOCATE or DEALLOCATE statement. If the value of the STAT argument is not a status value producible by the processor as a result of executing the kind of statement indicated by the STMT argument, the result is processor dependent.

If the STAT argument is present and the STMT argument is absent, the STAT argument shall provide a status value returned by an IOSTAT= specifier in an input/output statement, or by a STAT= specifier in an ALLOCATE or DEALLOCATE statement. If the value of the STAT argument could have been produced by both an input/output statement, and an ALLOCATE or DEALLOCATE statement, or by neither, the result is processor dependent.

If the STAT argument is absent and the STMT argument is present, the status that resulted from the most recently executed statement, if any, of the kind indicated by the STMT argument is used.

If the STAT argument is absent and the STMT argument is absent, the status that resulted from the most recently executed input/output, ALLOCATE, or DEALLOCATE statement, if any, is used.

If the STAT argument is absent, a processor may produce a more informative message by consulting its internal context. The phrase "status that resulted from the most recently executed statement" is intentionally vague – it can be anything useful to the processor for the purpose of constructing an informative error message; it is not necessarily a status value that would be returned by an IOSTAT= or STAT= specifier. Note

If the STAT argument is present and the status value is zero, or the most recently executed statement of the kind indicated by the STMT argument or implied by its absence was successful, or there was no such statement, the message has zero records.

If the MESSAGE argument is present and ALLOCATABLE but not allocated, it is allocated with a number of elements equal to the number of records in the message and a length equal to the length of the longest record of the message. If the number of records is zero, the length shall also be zero. If attempting to allocate MESSAGE fails, no message is accessed.

Automatic allocation and deallocation is not considered to be performed by an ALLOCATE or DEALLOCATE statement, and therefore does not affect the status of the most recently executed ALLOCATE or DEALLOCATE statement. Automatic allocations and deallocations may include, but are not limited to allocation of the MESSAGE argument, deallocation implied by ALLOCATABLE variables ceasing to exist, deallocation of ALLOCATABLE dummy arguments with INTENT(OUT) prior to execution of the procedure of which they are dummy arguments, or creation and destruction of automatic variables. Note

Van prefers: *J3 Note*

If the MESSAGE argument is present and scalar, and the message has at least one record, the MESSAGE argument is assigned the first record of the message, else the MESSAGE argument is assigned the value blank.

If the MESSAGE argument is present and rank one, for each $1 \le k \le$ SIZE(MESSAGE) if the message has k or more records, MESSAGE(k) is assigned the k'th record of the message, else MESSAGE(k) is assigned the value blank.

Malcolm preferred something more like: *J3 Note*

If the MESSAGE argument is present, it is assigned a value in one of three ways:

Case (i): If the message has zero records, the MESSAGE argument is assigned blanks.

Case (ii): If the message has one or more records and the MESSAGE argument is scalar, the MESSAGE argument is assigned the first record of the message.

Case (iii): If the message has one or more records and the MESSAGE argument is of rank one, the first element of the MESSAGE argument is assigned the first record of the message and successive elements of the MESSAGE argument are assigned successive records of the message. Remaining elements of the MESSAGE argument are assigned blanks.

Argument from Van: *Case (iii)* is incomplete by not discussing explicitly what happens if the message has more records than the MESSAGE argument has elements; including the zero-size case of the MESSAGE argument. Do you prefer what Van wrote, what Malcolm wrote, something else (supply it), or undecided? *Straw vote*

If the MESSAGE argument is absent, the processor shall display the message, if any, where a message would have been displayed if IOSTAT= or STAT= had been absent from the input/output, ALLOCATE or DEALLOCATE statement. Otherwise, the processor shall not display a message.

A program can detect that attempting to allocate MESSAGE failed by noticing that it is not allocated after STATUS_ERROR_MSG is executed. Reasonable continuations might be to invoke STATUS_ERROR_MSG again with no MESSAGE argument (thereby causing the message to be displayed), or with a MESSAGE argument that is not ALLOCATABLE (which might result in truncating records or not accessing all of the records of the message). [Maybe this note fits better in annex C.] Note