z/OS

IBM

Language Environment Run-Time Messages

SA22-7566-08

z/OS

Language Environment Run-Time Messages

SA22-7566-08

Contents

Chapter 1. Language Environment Run-Time Messages

The messages in this topic pertain to Language Environment. Each message is followed by an explanation describing the condition that caused the message, a programmer response suggesting how you might prevent the message from occurring again, and a system action indicating how the system responds to the condition that caused the message.

The messages also contain a symbolic feedback code, which represents the first 8 bytes of a 12-byte condition token. You can think of the symbolic feedback code as the nickname for a condition. As such, the symbolic feedback code can be used in user-written condition handlers to screen for a given condition, even if it occurs at different locations in an application.

The messages in this topic contain alphabetic suffixes that have the following meaning:

I	Informational message
W	Warning message
E	Error message
S	Severe error message
C	Critical error message

CEE0000I The service completed successfully.

Explanation: The service completed successfully.

Programmer response: None

System action: None

Symbolic Feedback Code: CEE000

CEE0102S An unrecognized condition token was passed to *routine* and could not be used.

Explanation: The condition token passed to *routine* contained fields that were not within the range of accepted values.

Programmer response: Verify that the condition token passed to *routine* does not contain invalid fields.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE036

CEE0110S For data conversion from character form to internal floating-point form, an invalid character was specified in the input character string *character_string*.

Programmer response: Ensure the input character string specified for conversion contains only numerical characters. Signs, decimal points, commas, exponents are not allowed in the string. If the feedback token was omitted on the call to the conversion routine, then the condition is signaled. Otherwise, examine the feedback token upon return and take appropriate action.

System action: The output value from the conversion routine is undefined.

Symbolic Feedback Code: CEE03E

CEE0111S For data conversion from internal floating-point form to character form, the number of fraction digits specified was either negative or greater than the value specified for the length of the character string.

Programmer response: Ensure the input value specified for fraction digits is non-negative and less than the value specified for the length of the character string. If the feedback token was omitted on the call to the conversion routine, then the condition is signaled. Otherwise, examine the feedback token upon return and take appropriate action.

System action: The output value from the conversion routine is undefined.

Symbolic Feedback Code: CEE03F

CEE0112S For data conversion from internal floating-point form to character form, the value specified for the length of the output character string is outside the acceptable range. The valid range for E-format conversion is 1 to 35, and for F-format conversion is 2 to 36.

Programmer response: Ensure the input value specified for the length of the output character string is within limit. If the feedback token was omitted on the call to the conversion routine, then the condition is signaled. Otherwise, examine the feedback token upon return and take appropriate action.

System action: The output value from the conversion routine is undefined.

Symbolic Feedback Code: CEE03G

CEE0113S For data conversion from character form to internal floating-point form, the value specified for the length of the input character string is outside the acceptable range. The valid range is 1 to 35.

Programmer response: Ensure the input value specified for the length of the input character string is within limit. If the feedback token was omitted on the call to the conversion routine, then the condition is signaled. Otherwise, examine the feedback token upon return and take appropriate action.

System action: The output value from the conversion routine is undefined.

Symbolic Feedback Code: CEE03H

CEE0198S The termination of a thread was signaled due to an unhandled condition.

Explanation: Termination imminent due to an unhandled condition was signaled or was the target of a promote.

Programmer response: Call CEEITOK from a user-written condition handler to determine what condition was unhandled. With that information, you can either recover appropriately or allow termination to continue.

System action: If this condition is signaled, or is the target of a promote, and it remains unhandled at stack frame zero, the thread will terminate without re-raising this condition. If this condition was signaled with CEESGL specifying a feedback code, the feedback code is returned to the caller of CEESGL and control is returned to the next sequential instruction following the call to CEESGL.

Symbolic Feedback Code: CEE066

CEE0199W The termination of a thread was signaled due to a STOP statement.

Explanation: The termination of a thread was signaled.

Programmer response: No response is required. A thread is terminating normally.

System action: The thread is terminated in a normal manner.

Symbolic Feedback Code: CEE067

CEE0201I An unhandled condition was returned in a feedback code.

Explanation: No language run-time component event handler or CEEHDL routine handled the condition.

Programmer response: See the original condition.

System action: Language Environment returns to the point at which the original condition was signaled.

Symbolic Feedback Code: CEE069

CEE0250S An unrecognized label variable was detected. The stack frame address could not be associated with an active stack frame.

Explanation: A call to CEEGOTO was made with a bad label variable. The label variable should be a valid code point that is subject to a current save area.

Programmer response: The label variable applies to a program that is no longer active, or the label variable was not initialized. Make sure that the program is active.

System action: The thread is terminated.

Symbolic Feedback Code: CEE07Q

CEE0252W CEEHDLU was unable to find the requested user-written condition handler routine.

Explanation: A call to CEEHDLU was made to unregister a user-written condition handler that was not registered.

Programmer response: Ensure that the user-written condition handler you are trying to free is registered.

System action: No user-written condition handlers are removed.

Symbolic Feedback Code: CEE07S

CEE0253W A user-written condition handler was unregistered. Additional registration remain in the queue.

Explanation: A call to CEEHDLU was made to unregister a user-written condition handler. The user-written condition handler had been registered a multiple number of times.

Programmer response: No programmer action is required.

System action: The first occurrence of the user-written condition handler is removed from the queue. Other registrations remain on the queue.

Symbolic Feedback Code: CEE07T

CEE0254W The first parameter passed to CEEMRCR was not 0 or 1.

Explanation: The first parameter passed to CEEMRCR was neither 0 nor 1.

Programmer response: Change the first parameter (type_of_move) passed to CEEMRCR to a valid value (0 or 1).

System action: The resume cursor is not moved.

Symbolic Feedback Code: CEE07U

CEE0255S The first parameter passed to CEEMRCE was an unrecognized label.

Explanation: A move resume cursor must be made to a valid label pointed to by CEEMRCE.

Programmer response: Change the position parameter pointed to by CEEMRCE to a valid label.

System action: The thread is terminated.

Symbolic Feedback Code: CEE07V

CEE0256W The user-written condition handler routine specified was already registered for this stack frame. It was registered again.

Explanation: CEEHDLR provided for multiple registration of user-written condition handler routines but the registration of the same routine again for the same stack frame is considered unusual.

Programmer response: No response is required. This message is just a warning.

System action: The handler is registered.

Symbolic Feedback Code: CEE080

CEE0257S The routine specified contained an invalid entry variable.

Explanation: CEEHDLR could not validate the entry variable passed.

Programmer response: Build and pass CEEHDLR a valid entry variable.

System action: The thread is terminated.

Symbolic Feedback Code: CEE081

CEE0259S A move to stack frame zero using CEEMRCR was attempted from a MAIN routine.

Explanation: The handler for the first stack frame beyond stack frame zero attempted to do a move of the resume cursor with type_of_move = 1. The resume cursor was not moved.

Programmer response: Do not attempt to move the resume cursor to the caller of the main routine. If you want to end the thread, signal Termination Imminent.

System action: The resume cursor is not moved. The thread is terminated.

Symbolic Feedback Code: CEE083

CEE0260S No condition was active when a call to a condition management routine was made. The requested function was not performed.

Explanation: The condition manager had no record of an active condition.

Programmer response: No response is required. Calls to these routines should only be made within the handler routine.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE084

CEE0264S An invalid request to resume a condition was detected.

Explanation: A user-written condition handler attempted to resume for a condition for which resumption is not allowed unless the resume cursor is moved.

Note: CEE088 might not be handled and resumed without moving the resume cursor. If resumption is requested without moving the resume cursor, the environment is terminated with ABEND 4091-12.

Programmer response: Move the resume cursor as part of handling the condition.

System action: The resume request that triggered this condition is ignored.

Symbolic Feedback Code: CEE088

CEE0277W CEEMRCR was called to perform an unnecessary move.

Explanation: A user-written condition handler attempted to move the resume cursor with type_of_move = 0 and with the handle and resume cursors pointing to the same stack frame. The handle and resume cursor might point to the same stack frame either because the handler is for the incurring frame or because the resume cursor has already been moved to the frame being handled.

Programmer response: No response is necessary.

System action: No action is taken by the Condition Manager. The resume cursor is not moved.

Symbolic Feedback Code: CEE08L

CEE0355C The user-written condition handler that was scheduled using CEEHDLR returned an unrecognized result code.

Explanation: A user written condition handler passed an invalid result code. A user-written condition handler has either returned without setting a reason code variable to a valid response code or has moved the resume cursor that caused a return to condition management without a valid response code being set.

Programmer response: Supply a valid result code.

System action: The thread is terminated.

Symbolic Feedback Code: CEE0B3

CEE0356C An internal condition handler returned an unrecognized result code.

Explanation: A language run-time component condition handler passed an invalid result code.

Programmer response: Contact your service representative.

System action: The thread is terminated.

Symbolic Feedback Code: CEE0BA

CEE0374C CONDITION = *condition-id* **TOKEN =** *condition-token* **WHILE RUNNING PROGRAM** *program-name* **WHICH STARTS AT** *program-address* **AT THE TIME OF INTERRUPT PSW** *psw* **GPR 0-3** *gpr0 gpr1 gpr2 gpr3* **GPR 4-7** *gpr4 gpr5 gpr6 gpr7* **GPR 8-B** *gpr8 gpr9 gprA gprB* **GPR C-F** *gprC gprD gprE gprF* **FLT 0-2** *flt0 flt2* **FLT 4-6** *flt4 flt6*

Explanation: An unrecoverable condition occurred while processing a previous condition. This message is issued with a WTO because Language Environment has encountered a critical error while handling a previous condition. The CONDITION indicates the message representing the condition being handled and the TOKEN is the three word Language Environment Condition Token. The *program-name*, *program-address* (starting address of program), psw, and registers are for the condition being handled when the unrecoverable condition occurred. If the CEE0374C message appears several times in sequence, the conditions appear in order of occurrence. Correcting the earliest condition may allow the application to run successfully.

Programmer response: Attempt to correct the original condition by looking up the condition-token specified in the message.

System action: The thread is terminated abnormally.

Symbolic Feedback Code: CEE0BM

CEE0398W Resume with new input.

Explanation: This condition was returned from a user-written condition handler to tell Language Environment to retry the operation with new input.

Programmer response: No programmer response is required.

System action: Language Environment attempts to retry the operation.

Symbolic Feedback Code: CEE0CE

CEE0399W Resume with new output.

Explanation: This condition was returned from a user-written condition handler to tell Language Environment to retry the operation with new output.

Programmer response: No programmer response is required.

System action: Language Environment resumes execution with new output.

Symbolic Feedback Code: CEE0CF

CEE0400E An invalid action code *action-code* **was passed to routine** *routine-name***.**

Explanation: An action code parameter passed to *routine* did not contain a valid value.

Programmer response: Provide a valid action code.

System action: No system action is performed. The output is undefined.

Symbolic Feedback Code: CEE0CG

CEE0401S An invalid case code *case-code* was passed to routine *routine-name*.

Explanation: A case code parameter must be a 2-byte integer with a value of 1 or 2.

Programmer response: Provide a valid case code.

System action: No system action is performed. The output is undefined.

Symbolic Feedback Code: CEE0CH

CEE0402S An invalid control code *control-code* was passed to routine *routine-name*.

Explanation: A control code parameter must be a 2-byte integer with a value of 0 or 1.

Programmer response: Provide a valid control code.

System action: No system action is performed.

Symbolic Feedback Code: CEE0CI

CEE0403S An invalid severity code *severity-code* was passed to routine *routine-name*.

Explanation: A severity code parameter must be a 2-byte integer with a value between 0 and 4.

Programmer response: Provide a valid severity code.

System action: No system action is performed.

Symbolic Feedback Code: CEE0CJ

CEE0404W Facility ID *facility-id* with non-alphanumeric characters was passed to routine *routine-name*.

Explanation: A facility ID parameter was passed with characters not in the range of A-Z, a-z, 0-9.

Programmer response: Verify that the facility ID passed is the correct value.

System action: No system action is performed. Processing continues.

Symbolic Feedback Code: CEE0CK

CEE0450S The message inserts for the condition token with message number *message-number* and facility ID *facility-id* could not be located.

Explanation: An insert area for the given condition token did not exist. It possibly was never allocated, or was reused by another condition.

Programmer response: Verify that the *message-number* and *Facility-ID* passed contain the correct values. If so, verify that the program was run with the MSGQ option specifying a large enough value to contain all the insert areas necessary for this program to run.

System action: No system action is performed.

Symbolic Feedback Code: CEE0E2

CEE0451S An invalid destination code *destination-code* was passed to routine *routine-name*.

Explanation: A destination code must be a 4-byte integer with a value of 2.

Programmer response: Provide a valid destination code.

System action: No system action is performed. The message is not written.

Symbolic Feedback Code: CEE0E3

CEE0452S An invalid facility ID *facility-id* was passed to routine *routine-name*.

Explanation: A facility id parameter must be a 3-alphanumeric character field.

Programmer response: Provide a facility id made up of 3-alphanumeric characters that corresponds to a product recognized by Language Environment. The IBM-supplied facility ids are IBM, CEE, IGZ, and EDC.

System action: No system action is performed. The output is undefined.

Symbolic Feedback Code: CEE0E4

CEE0454S The message number *message-number* could not be found for facility ID *facility-id*.

Explanation: The message could not be located within the source message files for *facility-id*.

Programmer response: Ensure the message number is contained within the source message file for *facility-id*.

System action: No system action is performed. The message is not written.

Symbolic Feedback Code: CEE0E6

CEE0455W The message with message number *message-number* and facility ID *facility-id* was truncated.

Explanation: The message could not fit within the message buffer supplied. Msg_index contains the index into the message returned.

Programmer response: Subsequent calls to CEEMGET with the previously returned msg_index value will retrieve the remainder of the message.

System action: The index into the message is returned in msg_ptr.

Symbolic Feedback Code: CEE0E7

CEE0457S The message file destination *ddname* could not be located.

Explanation: An error was detected trying to access the given message file *ddname*.

Programmer response: Verify that the file exists and is usable.

System action: No system action is performed. The message is not written.

Symbolic Feedback Code: CEE0E9

CEE0458S The message repository *repository-name* could not be located.

Explanation: The file containing the table of message file names could not be located. The name of the file was txxxMSGT, where t was either the letter ″I″ for an IBM-assigned facility id, or ″U″ for a user-assigned facility id. xxx was the facility id. MSGT was the letters ″MSGT″.

Programmer response: Verify that the table exists and is appropriately named.

System action: No system action is performed. The message is not written.

Symbolic Feedback Code: CEE0EA

CEE0459S Not enough storage was available to create a new instance specific information block.

Explanation: A new ISI could not be created because not enough storage was available.

Programmer response: Ensure that the REGION size is sufficient to run the application. Verify that the storage sizes specified in the HEAP run-time option is reasonable, given the region size allocated to the application.

System action: No storage is allocated.

Symbolic Feedback Code: CEE0EB

CEE0460W Multiple instances of the condition token with message number *message-number* and facility ID *facility-id* were detected.

Explanation: A message insert block for the given condition token already existed. A new message insert block was created. The two were differentiated by the I_S_info field in the condition token.

Programmer response: No response is required.

System action: A call to CEEMSG or CEEMGET will format the message associated with the instance of the message insert block indicated by the I_S_info field of the condition token.

Symbolic Feedback Code: CEE0EC

CEE0461S The maximum number of unique message insert blocks was reached. This condition token had its I_S_info field set to 1.

Explanation: The maximum number of 2,147,483,647 unique message insert blocks was reached. The condition token passed had its I_S_info field set to 1.

Programmer response: No response is required.

System action: The I_S_info field in the condition token is set to 1.

Symbolic Feedback Code: CEE0ED

CEE0462S Instance specific information for the condition token with message number *message-number* and facility ID *facility-id* could not be found.

Explanation: The ISI associated with the condition token was not located. It possibly was reused by another condition if the number specified in the MSGQ run-time option was exceeded.

Programmer response: Specify a MSGQ run-time option that is sufficient to contain all the active ISIs.

System action: No system action is performed. The message is not written.

Symbolic Feedback Code: CEE0EE

CEE0463S The maximum size for an insert data item was exceeded.

Explanation: The maximum size of 254 for the length of an insert data item was exceeded.

Programmer response: Make the insert 254 characters or less. If this is not possible, divide the insert into 2 or more inserts.

System action: No system action is performed. The insert is not created.

Symbolic Feedback Code: CEE0EF

CEE0464S Instance-specific information for the condition token with message number *message-number* and facility ID *facility-id* did not exist.

Explanation: No ISI was associated with the condition token. It is most likely that the information was never created.

Programmer response: If this condition was returned by a Language Environment service, contact your service representative. Otherwise, make sure that the correct I_S_info was identified.

System action: No system action is performed. The message is not written.

Symbolic Feedback Code: CEE0EG

CEE0502S The operational descriptor for the argument list was missing in routine *routine-name*.

Explanation: The high order bit of register 1 was off or the constant X'81C3C501' was missing from the storage location immediately preceding the argument list.

Programmer response: Contact your service representative.

System action: No system action is performed.

Symbolic Feedback Code: CEE0FM

CEE0553S An internal error was detected in creating the inserts for a condition.

Explanation: An invalid insert number was passed to the routine to format inserts.

Programmer response: Contact your service representative.

System action: No system action is performed.

Symbolic Feedback Code: CEE0H9

CEE0554W A value outside the range of 0 through 999,999 was supplied. However, the value was still used as the enclave return code.

Explanation: Language Environment prefers the user to set the enclave return code to a value of 0 through 999,999.

Programmer response: If possible, change the return code to be within the range of 0 through 999,999.

System action: The value will still be used as the enclave return code.

Symbolic Feedback Code: CEE0HA

CEE0802C Heap storage control information was damaged.

Explanation: Internal control information saved in header records within the heap was damaged.

Programmer response: Ensure that your program does not write data to an area larger than the original allocation. For example, allocating a 100 byte area and then writing 120 bytes to this area could cause damage to a storage header.

System action: No storage is allocated. A severity 4 condition is signaled and the application is terminated.

Symbolic Feedback Code: CEE0P2

CEE0803S The heap identifier in a get storage request or a discard heap request was unrecognized.

Explanation: The heap identifier supplied in a call to CEEGTST or CEEDSHP did not match any known heap identifier, or the heap had already been discarded by a call to CEEDSHP (discard heap) before the request.

Programmer response: For get heap storage requests, ensure that the value in the heap identifier parameter is either 0, indicating the default heap, or an identifier returned by the CEECRHP (create heap) service. For all other requests, ensure that the heap is not discarded before the request.

System action: No storage is allocated. The value of the address parameter is undefined.

Symbolic Feedback Code: CEE0P3

CEE0804S The initial size value supplied in a create heap (CEECRHP) request was invalid.

Explanation: The initial size value supplied to CEECRHP was a negative number.

Programmer response: Ensure that the value in the initial size parameter is either 0, indicating same as the initial heap, or a positive integer.

System action: No heap is created. The value of the heap identifier is undefined.

Symbolic Feedback Code: CEE0P4

CEE0805S The increment size value supplied in a create heap (CEECRHP) request was invalid.

Explanation: The increment size value supplied to CEECRHP was a negative number.

Programmer response: Ensure that the value in the increment size parameter is either 0, indicating same as the initial heap, or a positive integer.

System action: No heap is created. The value of the heap identifier is undefined.

Symbolic Feedback Code: CEE0P5

CEE0806S The options value supplied in a create heap (CEECRHP) request was unrecognized.

Explanation: The value of the options parameter supplied to CEECRHP was not recognized.

Programmer response: Ensure that the value in the options parameter is either 0, indicating same as the initial heap, or one of the supported options values documented in the *z/OS Language Environment Programming Guide*.

System action: No heap is created. The value of the heap identifier is undefined.

Symbolic Feedback Code: CEE0P6

CEE0807S An input supplied to a create user heap request (CEEVUHCR) was not valid.

Explanation: The value of an input parameter supplied to CEEVUHCR was not correct.

Programmer response: Ensure that all of the input parameters have been properly specified on the call to CEEVUHCR.

System action: No heap is created. The value of the heap token is undefined.

Symbolic Feedback Code: CEE0P7

CEE0808S Storage size in a get storage request (CEEGTST) or a reallocate request (CEECZST) was not a positive number.

Explanation: The size parameter supplied in a get storage request call to CEEGTST or a reallocate call to CEECZST was less than or equal to 0.

Programmer response: Ensure that the size parameter is a positive integer representing the number of bytes of storage to be obtained.

System action: No storage is allocated. The value of the address parameter is undefined.

Symbolic Feedback Code: CEE0P8

CEE0809S The maximum number of heaps was reached.

Explanation: The maximum number of heaps had already been created.

Programmer response: Modify the program to discard unneeded heaps before attempting to create a new heap or restructure the application so that it requires fewer heaps.

System action: No heap is created. The value of the heap identifier is undefined.

Symbolic Feedback Code: CEE0P9

CEE0810S The storage address in a free storage (CEEFRST) request was not recognized, or heap storage (CEECZST) control information was damaged.

Explanation: The address parameter supplied in a call to CEEFRST or CEECZST did not contain the starting address of a currently allocated area in the heap. Either the supplied address was invalid, or the area had been freed previously.

Programmer response: Ensure that the address parameter contains a value returned by a call to CEEGTST or CEECZST. Ensure that the storage area to be freed has not been freed previously.

System action: No storage is freed. The address parameter is left unchanged so that its value can be examined.

Symbolic Feedback Code: CEE0PA

CEE0812S An invalid attempt to discard the initial heap was made.

Explanation: The heap identifier supplied in a discard heap request was zero (indicating the initial heap) but the initial heap could not be discarded.

Programmer response: Ensure that the heap identifier supplied in the discard heap call is an identifier returned by the create heap (CEECRHP) service.

System action: No storage is freed. The value of the heap identifier remains unchanged.

Symbolic Feedback Code: CEE0PC

CEE0813S Insufficient storage was available to satisfy a get storage (CEECZST) request.

Explanation: There was not enough free storage available to satisfy a get storage call to CEEGTST or reallocate request call to CEECZST.

Programmer response: Ensure that the REGION size is sufficient to run the application. Ensure that the size parameter in the get storage request is not an unusually large number. Verify that the storage sizes specified in the HEAP and STACK run-time options are reasonable, given the region size allocated to the application. Verify that you are using storage options that get your storage from above the line, if you can, since you can run out of storage below the line much more easily.

System action: No storage is allocated. The value of the address parameter is undefined.

Symbolic Feedback Code: CEE0PD

CEE0814S Insufficient storage was available to extend the stack.

Explanation: During prologue processing, a new stack frame could not be obtained because there was not enough free storage available.

Programmer response: Ensure that the REGION size is sufficient to run the application.

If this is an AMODE 64 application:

- Ensure that the maximum stack size sub-option of the STACK64 run-time option is large enough to meet the user stack requirements of the application. If this is a pthread, and THREADSTACK64(ON) has been specified, then ensure that the maximum stack size sub-option of the THREADSTACK64 run-time option is large enough to meet the pthread's stack requirements. See *z/OS Language Environment Programming Reference* for more information about the STACK64 run-time and THREADSTACK64 option.
- Ensure that the MEMLIMIT setting for the application is large enough to meet the overall storage requirements of the application. See *z/OS MVS Programming: Extended Addressability Guide* for more information about MEMLIMIT.

System action: A SIGSEGV signal is raised. If the process is blocking or ignoring this signal, or is catching it but has not specified that the catcher function should run on an alternate stack, the signal will be unblocked and its action set to default (i.e., terminate the process) before the signal is raised.

Symbolic Feedback Code: CEE0PE

CEE0815E The stack soft limit set by the __set_stack_softlimit() function has been exceeded.

Explanation: During prologue processing the stack soft limit set by the __set_stack_softlimit() function has been exceeded.

Programmer response: The soft limit can be increased or decreased by using the __set_stack_softlimit() function if desired.

System action: A SIGSEGV signal with a si_code of _SEGV_SOFTLIMIT is delivered to the thread where the soft limit was exceeded.

Symbolic Feedback Code: CEE0PF

CEE1000S Language Environment internal abend. ABCODE = ***abcode*** **REASON =** ***rsncode***

Explanation: This message was issued to the operators console in CICS to indicate that Language Environment had abended, with the abend code and reason code as specified in the message.

Programmer response: Refer to the Language Environment Abend Codes topic in this information for details on the cause of the abend.

System action: The transaction is terminated abnormally with the abend code stated in this message.

Symbolic Feedback Code: CEE0V8

CEE1001E A cross program branching was attempted as a result of a CICS HANDLE command with the LABEL options. This was not supported by the language used by program *program-name*.

Explanation: The HLL did not support transferring control to specified LABEL.

Programmer response: This is a language-specific restriction. See *z/OS Language Environment Programming Guide* for information on EXEC CICS.

System action: No system action is performed.

Symbolic Feedback Code: CEE0V9

CEE2001E For an exponentiation operation (RS) where R and S are real values, R was less than zero in math routine *routine-name*.**

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UH

CEE2002E The argument value was too close to one of the singularities (plus or minus pi/2, plus or minus 3pi/2, for the tangent; or plus or minus pi, plus or minus 2pi, for the cotangent) in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UI

CEE2003E For an exponentiation operation (IJ) where I and J are integers, I was equal to zero and J was less than or equal to zero in math routine *routine-name*.**

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UJ

CEE2004E For an exponentiation operation (RI) where R is real and I is an integer, R was equal to zero and I was less than or equal to zero in math routine *routine-name*.**

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UK

CEE2005E The value of the argument was outside the valid range ***range*** **in math routine** ***routine-name*****.**

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UL

CEE2006E For an exponentiation operation (RS) where R and S are real values, R was equal to zero and S was less than or equal to zero in math routine** ***routine-name*****.**

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UM

CEE2007E The exponent exceeded ***limit*** **in math routine** ***routine-name*****.**

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UN

CEE2008E For an exponentiation operation (ZP) where the complex base Z equals zero, the real part of the complex exponent P, or the integer exponent P was less than or equal to zero in math routine** ***routine-name*****.**

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UO

CEE2009E The value of the real part of the argument was greater than ***limit*** **in math routine** ***routine-name*****.**

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UP

CEE2010E The argument was less than *limit* in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine, or a hardware square root exception occurred outside of a math routine.

Programmer response: Ensure the arguments are valid to the math routine or the input is valid for the square root instruction. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled. For square root exceptions, the condition is always signaled.) If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine or square root instruction is undefined.

Symbolic Feedback Code: CEE1UQ

CEE2011E The argument was greater than *limit* in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UR

CEE2012E The argument was less than or equal to *limit* in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1US

CEE2013E The absolute value of the imaginary part of the argument was greater than *limit* in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UT

CEE2014E Both arguments were equal to *limit* in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UU

CEE2015E The absolute value of the imaginary part of the argument was greater than or equal to *limit* in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1UV

CEE2016E The absolute value of the argument was greater than *limit* in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1V0

CEE2017E The absolute value of the argument was greater than or equal to *limit* in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1V1

CEE2018E The real and imaginary parts of the argument were equal to *limit* in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1V2

CEE2019E The absolute value of the real part of the argument was greater than or equal to *limit* in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1V3

CEE2020E For an exponentiation operation (RS) where R and S are real values, either R is equal to zero and S is negative, or R is negative and S is not an integer whose absolute value is less than or equal to *limit* in math routine *routine-name*.**

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1V4

CEE2021E For an exponentiation operation (XY), the argument combination of Y*log2(X) generated a number greater than or equal to *limit* in math routine *routine-name*.**

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1V5

CEE2022E The value of the argument was plus or minus *limit* in math routine *routine-name*.

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1V6

CEE2024E An overflow has occurred in math routine *routine-name*.

Explanation: An overflow had occurred in calculating the results in the scalar math routine.

Programmer response: Ensure the input arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1V8

CEE2025W An underflow has occurred in math routine *routine-name*.

Explanation: An underflow had occurred in calculating the results in the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1V9

CEE2028E The value of the second argument was outside the valid range ***range*** **in math routine** ***routine-name*****.**

Explanation: Invalid arguments were specified to the scalar math routine.

Programmer response: Ensure the arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1VC

CEE2029E The value of the argument was equal to ***limit*** **in math routine** ***routine-name*****.**

Explanation: Invalid input arguments were specified to the scalar math routine.

Programmer response: Ensure the input arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1VD

CEE2030E The value of the second argument was equal to ***limit*** **in math routine** ***routine-name*****.**

Explanation: Invalid input arguments were specified to the scalar math routine.

Programmer response: Ensure the input arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1VE

CEE2031E The value of the argument was a non-positive whole number in math routine ***routine-name*****.**

Explanation: Invalid input arguments were specified to the scalar math routine.

Programmer response: Ensure the input arguments are valid to the math routine. You might want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1VF

CEE2040E The value of the third argument was outside the valid range ***range*** **in math routine** ***routine-name*****.**

Explanation: Invalid input arguments were specified to the scalar math routine.

Programmer response: Ensure the input arguments are valid to the math routine. You may want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1VO

CEE2041E The absolute value of the second argument was greater than either the value of the third argument or the number of bits in the first argument in math routine ***routine-name*****.**

Explanation: Invalid input arguments were specified to the scalar math routine.

Programmer response: Ensure the input arguments are valid to the math routine. You may want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1VP

CEE2042E The sum of the second and the third arguments was greater than the number of bits in the first argument in math routine ***routine-name*****.**

Explanation: Invalid input arguments were specified to the scalar math routine.

Programmer response: Ensure the input arguments are valid to the math routine. You may want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1VQ

CEE2043E The value of the second or third argument was less than 0 in math routine ***routine-name*****.**

Explanation: Invalid input arguments were specified to the scalar math routine.

Programmer response: Ensure the input arguments are valid to the math routine. You may want to register a user handler that will gain control if this condition is signaled (if the feedback token was omitted on the call to the math routine, then the condition is signaled). If you specify the feedback token on the call to the math routine, examine the feedback token upon return from the math routine and take appropriate action.

System action: The output value from the math routine is undefined.

Symbolic Feedback Code: CEE1VR

CEE2050S The length of the first argument was less than 0 or greater than 32767 in routine ***routine-name*****.**

Explanation: Invalid length of input argument was specified.

Programmer response: Ensure the length of input argument is valid.

System action: The output value is undefined.

Symbolic Feedback Code: CEE202

CEE2051S The length of the second argument was less than 0 or greater than 32767 in routine ***routine-name*****.**

Explanation: Invalid length of input argument was specified.

Programmer response: Ensure the length of input argument is valid.

System action: The output value is undefined.

Symbolic Feedback Code: CEE203

CEE2052S The length of the result was less than 0 or greater than 32767 in routine ***routine-name*****.**

Explanation: Invalid length of result was specified.

Programmer response: Ensure the length of result is valid.

System action: The output value is undefined.

Symbolic Feedback Code: CEE204

CEE2053S The value of the second argument was not positive in routine *routine-name*.

Explanation: Invalid input argument was specified.

Programmer response: Ensure the input argument is valid.

System action: The output value is undefined.

Symbolic Feedback Code: CEE205

CEE2502S The UTC/GMT was not available from the system.

Explanation: A call to CEEUTC or CEEGMT failed because the system clock was in an invalid state. The current time could not be determined.

Programmer response: Notify systems support personnel that the system clock is in an invalid state.

System action: All output values are set to 0.

Symbolic Feedback Code: CEE2E6

CEE2503S The offset from UTC/GMT to local time was not available from the system.

Explanation: A call to CEEGMTO failed because either (1) the current operating system could not be determined, or (2) the time zone field in the operating system control block appears to contain invalid data.

Programmer response: Notify systems support personnel that the local time offset stored in the operating system appears to contain invalid data.

System action: All output values are set to 0.

Symbolic Feedback Code: CEE2E7

CEE2505S The input_seconds value in a call to CEEDATM or CEESECI was not within the supported range.

Explanation: The *input_seconds* value passed in a call to CEEDATM or CEESECI was not a floating-point number between 86,400.0 and 265,621,679,999.999. The input parameter should represent the number of seconds elapsed since 00:00:00 on 14 October 1582, with 00:00:00.000 15 October 1582 being the first supported time/date, and 23:59:59.999 31 December 9999 being the last supported time/date.

Programmer response: Verify that input parameter contains a floating-point value between 86,400.0 and 265,621,679,999.999.

System action: For CEEDATM, the output value is set to blanks. For CEESECI, all output parameters are set to 0.

Symbolic Feedback Code: CEE2E9

CEE2506S An era (<JJJJ>, <CCCC> or <CCCCCCCC>) was used in a picture string passed to CEEDATM, but the input number-of-seconds value was not within the supported range. The era could not be determined.

Explanation: In a CEEDATM call, the picture string indicated that the input value was to be converted to an era; however, the input value that was specified lies outside the range of supported eras.

Programmer response: Verify that the input value contains a valid number-of-seconds value within the range of supported eras.

System action: The output value is set to blanks.

Symbolic Feedback Code: CEE2EA

CEE2507S Insufficient data was passed to CEEDAYS or CEESECS. The Lilian value was not calculated.

Explanation: The picture string passed in a CEEDAYS or CEESECS call did not contain enough information. For example, it is an error to use the picture string MM/DD (month and day only) in a call to CEEDAYS or CEESECS, because the year value is missing. The minimum information required to calculate a Lilian value is either (1) month, day and year, or (2) year and Julian day.

Programmer response: Verify that the picture string specified in a call to CEEDAYS or CEESECS specifies, as a minimum, the location in the input string of either (1) the year, month, and day, or (2) the year and Julian day.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EB

CEE2508S The date value passed to CEEDAYS or CEESECS was invalid.

Explanation: In a CEEDAYS or CEESECS call, the value in the DD or DDD field was not valid for the given year and/or month. For example, MM/DD/YY with 02/29/90, or YYYY.DDD with 1990.366 are invalid because 1990 is not a leap year. This code can also be returned for any nonexistent date value such as June 31st or January 0.

Programmer response: Verify that the format of the input data matches the picture string specification and that input data contains a valid date.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EC

CEE2509S The era passed to CEEDAYS or CEESECS was not recognized.

Explanation: The value in the era field passed in a call to CEEDAYS or CEESECS did not contain a supported era name.

Programmer response: Verify that the format of the input data matches the picture string specification and that the spelling of the era name is correct. Note that the era name must be a proper DBCS string, that is, the '<' position must contain a shift-out character (X'0E') and the '>' position must contain a shift-in character (X'0F').

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2ED

CEE2510S The hours value in a call to CEEISEC or CEESECS was not recognized.

Explanation: (1) In a CEEISEC call, the hours parameter did not contain a number between 0 and 23, or (2) in a CEESECS call, the value in the HH (hours) field did not contain a number between 0 and 23, or the AP (a.m./p.m.) field was present and the HH field did not contain a number between 1 and 12.

Programmer response: For CEEISEC, verify that the hours parameter contains an integer between 0 and 23. For CEESECS, verify that the format of the input data matches the picture string specification, and that the hours field contains a value between 0 and 23, (or 1 and 12 if the AP field is used).

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EE

CEE2511S The day parameter passed in a CEEISEC call was invalid for year and month specified.

Explanation: The day parameter passed in a CEEISEC call did not contain a valid day number. The combination of year, month, and day formed an invalid date value. Examples: year=1990, month=2, day=29; or month=6, day=31; or day=0.

Programmer response: Verify that the day parameter contains an integer between 1 and 31, and that the combination of year, month, and day represents a valid date.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EF

CEE2512S The Lilian date value passed in a call to CEEDATE or CEEDYWK was not within the supported range.

Explanation: The Lilian day number passed in a call to CEEDATE or CEEDYWK was not a number between 1 and 3,074,324.

Programmer response: Verify that the input parameter contains an integer between 1 and 3,074,324.

System action: The output value is set to blanks.

Symbolic Feedback Code: CEE2EG

CEE2513S The input date passed in a CEEISEC, CEEDAYS, or CEESECS call was not within the supported range.

Explanation: The input date passed in a CEEISEC, CEEDAYS, or CEESECS call was earlier than 15 October 1582, or later than 31 December 9999.

Programmer response: For CEEISEC, verify that the year, month, and day parameters form a date greater than or equal to 15 October 1582. For CEEDAYS and CEESECS, verify that the format of the input date matches the picture string specification, and that the input date is within the supported range.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EH

CEE2514S The year value passed in a CEEISEC call was not within the supported range.

Explanation: The year parameter passed in a CEEISEC call did not contain a number between 1582 and 9999.

Programmer response: Verify that the year parameter contains valid data, and that the year parameter includes the century. For example, you must specify the year as 1990, not as 90.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EI

CEE2515S The milliseconds value in a CEEISEC call was not recognized.

Explanation: In a CEEISEC call, the milliseconds parameter (*input_milliseconds*) did not contain a number between 0 and 999.

Programmer response: Verify that the milliseconds parameter contains an integer between 0 and 999.

Symbolic Feedback Code: CEE2EJ

System action: The output value is set to 0.

CEE2516S The minutes value in a CEEISEC call was not recognized.

Explanation: (1) In a CEEISEC call, the minutes parameter (*input_minutes*) did not contain a number between 0 and 59, or (2) in a CEESECS call, the value in the MI (minutes) field did not contain a number between 0 and 59.

Programmer response: For CEEISEC, verify that the minutes parameter (*input_minutes*) contains an integer between 0 and 59. For CEESECS, verify that the format of the input data matches the picture string specification, and that the minutes field contains a number between 0 and 59.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EK

CEE2517S The month value in a CEEISEC call was not recognized.

Explanation: (1) In a CEEISEC call, the month parameter (*input_month*) did not contain a number between 1 and 12, or (2) in a CEEDAYS or CEESECS call, the value in the MM field did not contain a number between 1 and 12, or the value in the MMM, MMMM, etc. field did not contain a correctly spelled month name or month abbreviation in the currently active National Language.

Programmer response: For CEEISEC, verify that the month parameter (*input_month*) contains an integer between 1 and 12. For CEEDAYS and CEESECS, verify that the format of the input data matches the picture string specification. For the MM field, verify that the input value is between 1 and 12. For spelled-out month names (MMM, MMMM, etc.), verify that the spelling or abbreviation of the month name is correct in the currently active National Language.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EL

CEE2518S An invalid picture string was specified in a call to a date/time service.

Explanation: The picture string supplied in a call to one of the date/time services was invalid. Only one era character string can be specified. The picture string contained an invalid DBCS string or contains more than one era descriptor.

Programmer response: Verify that the picture string contains valid data. Only one era character string can be specified. If the picture string contains the X'0E' (shift-out) character, this indicates the presence of DBCS data. Therefore, (1) the DBCS data must be terminated by a X'0F' (shift-in) character, (2) there must be an even number of characters between the shift-out and shift-in, and (3) these characters must all be valid DBCS characters.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EM

CEE2519S The seconds value in a CEEISEC call was not recognized.

Explanation: (1) In a CEEISEC call, the seconds parameter (*input_seconds*) did not contain a number between 0 and 59, or (2) in a CEESECS call, the value in the SS (seconds) field did not contain a number between 0 and 59.

Programmer response: For CEEISEC, verify that the seconds parameter (*input_seconds*) contains an integer between 0 and 59. For CEESECS, verify that the format of the input data matches the picture string specification, and that the seconds field contains a number between 0 and 59.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EN

CEE2520S CEEDAYS detected non-numeric data in a numeric field, or the date string did not match the picture string.

Explanation: The input value passed in a CEEDAYS call did not appear to be in the format described by the picture specification. For example, non-numeric characters appear where only numeric characters are expected.

Programmer response: Verify that the format of the input data matches the picture string specification and that numeric fields contain only numeric data.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EO

CEE2521S The Japanese (<JJJJ>) or Chinese (<CCCC>) year-within-Era value passed to CEEDAYS or CEESECS was zero.

Explanation: In a CEEDAYS or CEESECS call, if the YY or ZYY picture token was specified, and if the picture string contained one of the era tokens such as <CCCC> or <JJJJ>, then the year value must be greater than or equal to 1. In this context, the YY or ZYY field means year within Era.

Programmer response: Verify that the format of the input data matches the picture string specification and that the input data is valid.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2EP

CEE2522S An era (<JJJJ>, <CCCC> or <CCCCCCCC>) was used in a picture string passed to CEEDATE, but the Lilian date value was not within the supported range. The era could not be determined.

Explanation: In a CEEDATE call, the picture string indicated that the Lilian date was to be converted to an era, but the Lilian date lies outside the range of supported eras.

Programmer response: Verify that the input value contains a valid Lilian day number within the range of supported eras.

System action: The output value is set to blanks.

Symbolic Feedback Code: CEE2EQ

CEE2523W The system time was not available when CEERAN0 was called. A seed value of 1 was used to generate a random number and a new seed value.

Explanation: A seed value of 0 was specified in a CEERAN0 call, indicating that the current system time should be used as a seed value. Because the system time was not available, a seed value of 1 was used to generate a new seed value.

Programmer response: If seed=1 is acceptable, no action is required. Otherwise, code an appropriate non-zero seed, or refer to message CEE2502.

System action: A seed value of 1 is assumed. CEERAN0 returns both a random number and a new seed value.

Symbolic Feedback Code: CEE2ER

CEE2524S An invalid seed value was passed to CEERAN0. The random number was set to -1.

Explanation: CEERAN0 was called with a seed value that was out of range.

Programmer response: Code a seed value between 0 and 2147483646, inclusive, for the CEERAN0 call.

System action: The random number output was set to -1, and the seed value input was not changed.

Symbolic Feedback Code: CEE2ES

CEE2525S CEESECS detected non-numeric data in a numeric field, or the timestamp string did not match the picture string.

Explanation: The input value passed in a CEESECS call did not appear to be in the format described by the picture specification. For example, non-numeric characters appear where only numeric characters are expected, or the a.m./p.m. field did not contain the strings AM or PM.

Programmer response: Verify that the format of the input data matches the picture string specification and that numeric fields contain only numeric data.

System action: The output value is set to 0.

Symbolic Feedback Code: CEE2ET

CEE2526E The date string returned by CEEDATE was truncated.

Explanation: In a CEEDATE call, the output string was not large enough to contain the formatted date value.

Programmer response: Verify that the output string variable is large enough to contain the entire formatted date. Ensure that the output parameter is at least as long as the picture string parameter.

System action: The output value is truncated to the length of the output parameter.

Symbolic Feedback Code: CEE2EU

CEE2527E The timestamp string returned by CEEDATM was truncated.

Explanation: In a CEEDATM call, the output string was not large enough to contain the formatted timestamp value.

Programmer response: Verify that the output string variable is large enough to contain the entire formatted timestamp. Ensure that the output parameter is at least as long as the picture string parameter.

System action: The output value is truncated to the length of the output parameter.

Symbolic Feedback Code: CEE2EV

CEE2529S A debug tool has terminated the enclave.

Explanation: The debug tool terminated the enclave at the user's request. Under VM, abend code 4094, reason code X'28' is issued. Under MVS, return code 3000 is issued.

Programmer response: No programmer response is necessary.

System action: The enclave is terminated.

Symbolic Feedback Code: CEE2F1

CEE2530S A debug tool was not available.

Explanation: Either the debug environment was corrupted or could not load the debug event handler.

Programmer response: Make sure the debug tool is installed with the loadable name CEEEVDBG.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE2F2

CEE2531S The local time was not available from the system.

Explanation: A call to CEELOCT failed because the system clock was in an invalid state. The current time could not be determined.

Programmer response: Notify systems support personnel that the system clock is in an invalid state.

System action: All output values are set to 0.

Symbolic Feedback Code: CEE2F3

CEE2533S The value passed to CEESCEN was not between 0 and 100.

Explanation: The *century_start* value passed in a CEESCEN call was not between 0 and 100, inclusive.

Programmer response: Ensure that the input parameter is within range.

System action: The 100-year window assumed for all 2-digit years is unchanged.

Symbolic Feedback Code: CEE2F5

CEE2534W Insufficient field width was specified for a month or weekday name in a call to CEEDATE or CEEDATM. Output set to blanks.

Explanation: The CEEDATE or CEEDATM callable services issued this message whenever: (1) the picture string contained MMM, MMMMMZ, WWW, Wwww, etc., requesting a spelled out month name or weekday name, (2) the national language currently in effect was a DBCS (Double Byte Character Set) language such as NATLANG(JPN), and (3) the month name currently being formatted contained more characters than can fit in the indicated field.

Programmer response: Increase the field width to contain the longest month or weekday name being formatted, including two bytes for the SO/SI characters. For Japanese, eight characters are sufficient (3 DBCS + SO/SI), so one should specify MMMMMMMM or MMMMMMMZ, WWWWWWWW or WWWWWWWWZ in the picture string.

System action: The month name and weekday name fields that are of insufficient width are set to blanks. The rest of the output string is unaffected. Processing continues.

Symbolic Feedback Code: CEE2F6

CEE2535S Profiler loaded, Debug Tool unavailable.

Explanation: Profiler and Debug Tool cannot run concurrently.

Programmer response: To dynamically invoke Debug Tool, set PROFILE run-time option OFF.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE2F7

CEE2600I Success with zero result.

Explanation: The floating-point input value was a true zero, and the caller is to provide the appropriate formatting.

Programmer response: No programmer response is required.

System action: Program continues.

Symbolic Feedback Code: CEE2H8

CEE2601I Success with positive result.

Explanation: The conversion has been completed successfully, and the result is strictly greater than zero.

Programmer response: No programmer response is required.

System action: Program continues.

Symbolic Feedback Code: CEE2H9

CEE2602I Success with negative result.

Explanation: The conversion has been completed successfully, and the result is strictly less than zero.

Programmer response: No programmer response is required.

System action: Program continues.

Symbolic Feedback Code: CEE2HA

CEE2603I Success with plus-rounded-to-zero result.

Explanation: The conversion has been completed successfully, and the result contains a zero result that was created by a strictly positive input value that rounded to zero.

Programmer response: No programmer response is required.

System action: Program continues.

Symbolic Feedback Code: CEE2HB

CEE2604I Success with minus-rounded-to-zero result.

Explanation: The conversion has been completed successfully, and the result contains a zero result that was created by a strictly negative input value that rounded to zero.

Programmer response: No programmer response is required.

System action: Program continues.

Symbolic Feedback Code: CEE2HC

CEE2606E Result overflows output field.

Explanation: The floating-point input value is too large or the output character string is too small to contain the fixed-point representation of the input argument.

Programmer response: Ensure the input floating-point value is properly specified and the length of the output character string is big enough to contain the fixed-point representation of the input argument.

System action: The result value is undefined.

Symbolic Feedback Code: CEE2HE

CEE2607E Result has underflowed.

Explanation: The conversion would have resulted in a number smaller than the underflow threshold for the floating point representation.

Programmer response: Ensure the input character value to be converted is specified correctly. If the feedback token was omitted on the call to the conversion routine, then the condition is signaled. Otherwise, examine the feedback token upon return and take appropriate action.

System action: A true floating point zero result has been returned.

Symbolic Feedback Code: CEE2HF

CEE2608E Result has overflowed.

Explanation: The conversion would have resulted in a number larger than the overflow threshold for the floating point representation.

Programmer response: Ensure the input character value to be converted is specified correctly. If the feedback token was omitted on the call to the conversion routine, then the condition is signaled. Otherwise, examine the feedback token upon return and take appropriate action.

System action: The maximum positive floating point magnitude has been returned.

Symbolic Feedback Code: CEE2HG

CEE2701S An invalid category parameter was passed to a locale function.

Explanation: An invalid category parameter was passed to a locale function. Valid categories are: LC_ALL, LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, and LC_TIME.

Programmer response: Supply a valid category to the function.

System action: The thread is terminated.

Symbolic Feedback Code: CEE2KD

CEE2702S An invalid locale name parameter was passed to a locale function.

Explanation: An invalid locale name parameter was passed to a locale function. Locale name must be one provided with the product or constructed using the LOCALEDEF utility.

Programmer response: Supply a valid locale name to the function.

System action: The thread is terminated.

Symbolic Feedback Code: CEE2KE

CEE2999C An internal logic error was detected in a date/time routine.

Explanation: An internal logic error was detected in one of the date/time services. Internal date/time control blocks might have been damaged.

Programmer response: Verify that the program doesn't inadvertently overlay areas of storage reserved for library use.

System action: The requested action is not completed. The application is terminated.

Symbolic Feedback Code: CEE2TN

CEE3098S The user routine traceback could not be completed.

Explanation: The user routine traceback could not be completed due to an error detected in tracing back through the DSA chain.

Programmer response: Attempt to perform problem determination through the use of a dump.

System action: The user routine traceback is not completed.

Symbolic Feedback Code: CEE30Q

| **CEE3100E The title or option string passed was longer than 120 bytes.**

| **Explanation:** The maximum character string length for a dump is larger than allowed.

| **Programmer response:** Specify a string with 120 characters or less.

| **System action:** The title string is truncated to 120 bytes.

| **Symbolic Feedback Code:** CEE30S

| **CEE3101E The title or option string passed to CEE3DMP is longer than 132 bytes.**

| **Explanation:** The maximum character string length for a dump title is 132 bytes.

| **Programmer response:** Specify a dump title string with 132 characters or less.

| **System action:** The title string is truncated to 132 bytes.

| **Symbolic Feedback Code:** CEE30T

CEE3102E Invalid CEE3DMP options or suboptions were found and ignored.

Explanation: Invalid options or suboptions were found in the options parameter to CEE3DMP.

Programmer response: Check the options string passed to CEE3DMP. Make sure it has correct syntax and values for options and suboptions as specified in the *z/OS Language Environment Programming Reference*. Also, make sure the options string is 255 characters long.

System action: The invalid options or suboptions are ignored, valid options and suboptions are processed, and a dump is performed.

Symbolic Feedback Code: CEE30U

CEE3103S An error occurred in writing messages to the dump file.

Explanation: An error occurred in trying to write information to the dump file, whose file name was specified with the FNAME option to CEE3DMP. The default file name was CEEDUMP, or CESE transient data queue under CICS.

Programmer response: Make sure the file name is correct as specified in the options to CEE3DMP. Also, make sure there is enough room in the file to contain the dump.

System action: Dump processing is terminated at the point where the file error is detected.

Symbolic Feedback Code: CEE30V

CEE3104S Information could not be successfully extracted for this DSA.

Explanation: Some information associated with the DSA or save area passed to CEETRCB could not be determined.

Programmer response: If no information could be extracted by CEETRCB, it is likely that the DSAPTR parameter does not point to an actual DSA or save area.

System action: All information that could be extracted is returned by CEETRCB. Any information that could not be extracted is zero or blank (depending on parameter type).

Symbolic Feedback Code: CEE310

CEE3105S The language dump exit was unsuccessful.

Explanation: A language component of Language Environment returned this condition to the common component of Language Environment when an error had occurred in the language component's dump event handler that was not covered in the conditions returned by the dump CWI services.

Programmer response: Not applicable. This is an internal condition within Language Environment, and is never seen by the application programmer.

System action: The common component of Language Environment ignores this condition and continues dump processing.

Symbolic Feedback Code: CEE311

CEE3106S An invalid parameter value was specified in a call to the CEEVDMP CWI service.

Explanation: The CEEVDMP CWI service was called with an invalid value for one of the parameters.

Programmer response: Check to make sure the parameters on the call to CEEVDMP are correct. In particular, check the lengths of the strings passed as parameters.

System action: The CEEVDMP service returns to the caller without adding any information to the dump.

Symbolic Feedback Code: CEE312

CEE3107E The CEEHDMP or CEEBDMP CWI service encountered inaccessible storage during dump processing.

Explanation: The CEEHDMP CWI service encountered inaccessible storage while dumping a storage area, or the CEEBDMP CWI service encountered inaccessible storage while dumping a control block.

Programmer response: Make sure the address and length of the storage area are correct for CEEHDMP. Make sure the address and offset are correct for CEEBDMP, and that CEEBDMP is dumping the control block with a correct mapping for the control block.

System action: The message `Inaccessible storage` is printed in the dump at the point of the encounter. The storage or control block dumping terminates, and CEEHDMP or CEEBDMP returns to the calling routine.

Symbolic Feedback Code: CEE313

CEE3108E An invalid option, suboption, or delimiter was found in the dump option string.

Explanation: An invalid option, suboption, or delimiter was found in the dump option string.

Programmer response: Correct the error location as indicated in the position parameter.

System action: No system action is taken.

Symbolic Feedback Code: CEE314

CEE3110E The cmd parameter is not a valid command for __le_traceback().

Explanation: The *cmd* parameter was not a valid command for __le__traceback(). No processing was performed.

Programmer response: Provide a valid *cmd* parameter. Refer to the description of __le_traceback() in *z/OS XL C/C++ Run-Time Library Reference* for a list of valid commands.

System action: The request has failed. The application continues to run.

Symbolic Feedback Code: CEE316

| **CEE3111I CEEDUMP was defined as a dummy data set. No Language Environment dump processing was performed.**

| **Explanation:** This message is issued when a Language Environment dump report DD name is defined to DUMMY in a job step and CEE3DMP is called with a NULL feedback token code (fc).

| **Programmer response:** Verify if a CEEDUMP DD card should be defined to another data set.

| **System action:** No Language Environment dump processing is performed.

| **Symbolic Feedback Code:** CEE317

CEE3186E A field type parameter of the CEEBDMP CWI service contained an invalid value.

Explanation: The *field_ids*, *field_length*, or *field_types* parameter of the CEEBDMP CWI service contained an invalid value.

Programmer response: Check to make sure the mapping of the control block specified to CEEBDMP through these three parameters is correct.

System action: CEEBDMP returns to its caller. Information might have been written to the dump.

Symbolic Feedback Code: CEE33I

CEE3191E An attempt was made to initialize an AMODE24 application without using the ALL31(OFF) and STACK(,,BELOW) run-time options.

Explanation: During initialization it was detected that a program began in AMODE 24, yet the options required for completely safe execution in AMODE 24 were not fully specified.

Programmer response: Specify run-time options ALL31(OFF) and STACK(,,BELOW) for AMODE 24 operation.

System action: Program initialization continues.

Symbolic Feedback Code: CEE33N

CEE3192C The Language Environment anchor support was not installed or was not supported on the operating system.

Explanation: The Language Environment anchor was the address of the Language Environment main control block, the CAA. The underlying operating system must provide the Language Environment anchor support for Language Environment to get its main control block, the CAA. Because the anchor was not installed, the application was not able to run properly.

Programmer response: Report the error to your systems programmer. Check whether Language Environment anchor support is installed properly on the underlying operating system.

System action: The application is terminated.

Symbolic Feedback Code: CEE33O

CEE3193I The invocation command parameter string contained an unmatched quote character.

Explanation: The invocation command parameter string contained a beginning quote (either single quote or double quote) but a matching end quote was not found.

Programmer response: Correct the string.

System action: The entire string is treated as user parameters.

Symbolic Feedback Code: CEE33P

CEE3194E An attempt was made to initialize an AMODE24 program when the XPLINK(ON) run-time option was in effect. AMODE24 programs are not supported in an XPLINK environment.

Explanation: During initialization it was detected that a program began in AMODE 24 and the XPLINK(ON) run-time option was in effect. The XPLINK(ON) run-time option may be in effect because it was specified or because XPLINK-compiled routines were found in the initial program of the application.

Programmer response: If the application does not invoke any XPLINK-compiled routines, specify run-time options XPLINK(OFF), ALL31(OFF) and STACK(,,BELOW) for AMODE 24 operation. Otherwise, the application must be modified to eliminate either the AMODE24 routines or the XPLINK routines.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE33Q

CEE3195W The SNAP dump file could not be opened.

Explanation: The SNAP dump file could not be opened.

Programmer response: If a SNAP dump was desired, determine the reason the file could not be opened and correct the problem.

System action: No SNAP dump was taken.

Symbolic Feedback Code: CEE33R

CEE3196W The id number was not in the allowed range.

Explanation: The id number was not in the required range of 0 to 255.

Programmer response: This is an internal problem. Contact your service representative.

System action: The id number MOD 256 was used.

Symbolic Feedback Code: CEE33S

CEE3197W An invalid value for reserved was passed.

Explanation: An invalid value for the reserved parameter was passed to the SNAP dump service.

Programmer response: This is an internal problem. Contact your service representative.

System action: The invalid value is ignored.

Symbolic Feedback Code: CEE33T

CEE3198S A SNAP dump was requested on an unsupported system.

Explanation: The SNAP dump service was called to produce a SNAP dump on an unsupported system.

Programmer response: This is an internal problem. Contact your service representative.

System action: The SNAP dump was not produced.

Symbolic Feedback Code: CEE33U

CEE3199S An error was returned from the SNAP system function.

Explanation: The SNAP dump service was called. It invoked the SNAP system service that failed.

Programmer response: This is an internal problem. Contact your service representative.

System action: The SNAP dump was not produced.

Symbolic Feedback Code: CEE33V

CEE3201S The system detected an operation exception (System Completion Code=0C1).

Explanation: The program attempted to execute an instruction with an invalid operation code. The operation code may be unassigned or the instruction with that operation code cannot be installed on this platform. See a Principles of Operation manual for a full list of operation exceptions.

Programmer response: Examine the contents of registers 14 and 15. If register 15 has a value of 0, then the cause was probably a routine didn't exist and a branch was made to location 0. This would indicate a link-edit failure. Examine the contents of register 14 to determine the point at which the branch was made. Also examine the linkage editor map for any unresolved references reported by the linkage editor.

Another possible cause is a routine branched to some unintended location, such as a conflict in addressing mode between the calling and the called routine, or any other program error that branched to the wrong location.

System action: The thread is terminated.

Symbolic Feedback Code: CEE341

CEE3202S The system detected a privileged-operation exception (System Completion Code=0C2).

Explanation: Attempted to execute a privileged operation code while the machine was in a problem state. See a Principles of Operation manual for a full list of privileged-operation exceptions.

Programmer response: Examine the contents of registers 14 and 15. If register 15 has a value of 0, then the probable cause is that a routine doesn't exist and a branch was made to location 0. This would indicate a link-edit failure. Examine the contents of register 14 to determine the point at which the branch was made. Also examine the linkage editor map for any unresolved references reported by the linkage editor.

Another possible cause is a routine branched to some unintended location, such as a conflict in addressing mode between the calling and the called routine, or any other program error that branched to the wrong location.

System action: The thread is terminated.

Symbolic Feedback Code: CEE342

CEE3203S The system detected an execute exception (System Completion Code=0C3).

Explanation: Your program attempted to execute an EXECUTE instruction where the target of the first EXECUTE instruction was another EXECUTE instruction. See a Principles of Operation manual for a full list of execute exceptions.

Programmer response: Check your application for errors in the EXECUTE instructions. See a Principles of Operation manual for a full list of execute exceptions.

System action: The thread is terminated.

Symbolic Feedback Code: CEE343

CEE3204S The system detected a protection exception (System Completion Code=0C4).

Explanation: Your program attempted to access a storage location to which it was not authorized.

Programmer response: Check your application for these common errors:
- Using the wrong AMODE to reference storage
- Trying to use a pointer that has not been set
- Trying to store data into storage reserved for the system
- Using an invalid index to an array

See a Principles of Operation manual for a full list of protection exceptions.

System action: The thread is terminated.

Symbolic Feedback Code: CEE344

CEE3205S The system detected an addressing exception (System Completion Code=0C5).

Explanation: Your program attempted to reference a main-storage location that was not available in the configuration. See a Principles of Operation manual for a full list of addressing exceptions.

Programmer response: Check your application for these common errors:
- Using the wrong AMODE to reference storage
- Trying to use a pointer that has not been set
- Trying to store data into storage reserved for the system
- Using an invalid index to an array

See a Principles of Operation manual for a full list of addressing exceptions.

System action: The thread is terminated.

Symbolic Feedback Code: CEE345

CEE3206S The system detected a specification exception (System Completion Code=0C6).

Explanation: Your program attempted an invalid operation such as incorrect use of registers. The register used for an operation was invalid. Examples include using an odd register number when an even register number was required, using a bad number for floating point registers, or having data that was not correctly aligned.

Programmer response: If the program is being produced by a compiler, then you might be able to specify a different optimization level to by-pass the problem. See a Principles of Operation manual for a full list of specification exceptions.

System action: The thread is terminated.

Symbolic Feedback Code: CEE346

CEE3207S The system detected a data exception (System Completion Code=0C7).

Explanation: Your program attempted to use a decimal instruction incorrectly. See a Principles of Operation manual for a full list of data exceptions.

Programmer response: Check the variables associated with the failing statement to make sure that they have been initialized correctly.

System action: The thread is terminated.

Symbolic Feedback Code: CEE347

CEE3208S The system detected a fixed-point overflow exception (System Completion Code=0C8).

Explanation: Your program attempted to use signed binary arithmetic or signed left-shift operations and an overflow occurred. See a Principles of Operation manual for a full list of fixed-point overflow exceptions.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The thread is terminated.

Symbolic Feedback Code: CEE348

CEE3209S The system detected a fixed-point divide exception (System Completion Code=0C9).

Explanation: Your program attempted to perform a signed binary division and the divisor is zero. See a Principles of Operation manual for a full list of fixed-point divide exceptions.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The thread is terminated.

Symbolic Feedback Code: CEE349

CEE3210S The system detected a decimal-overflow exception (System Completion Code=0CA).

Explanation: Your program attempted to perform a mathematical operation and one or more nonzero digits were lost because the destination field in a decimal operation was too short to contain the results. See a Principles of Operation manual for a full list of decimal-overflow exceptions.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The thread is terminated.

Symbolic Feedback Code: CEE34A

CEE3211S The system detected a decimal-divide exception (System Completion Code=0CB).

Explanation: Your program attempted to perform a mathematical operation where, in decimal division, the divisor is zero or the quotient exceeds the specified data-field size. See a Principles of Operation manual for a full list of decimal-divide exceptions.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The thread is terminated.

Symbolic Feedback Code: CEE34B

CEE3212S The system detected an exponent-overflow exception (System Completion Code=0CC).

Explanation: Your program attempted a floating-point operation and the result characteristic exceeded 127 and the result fraction was not zero. See a Principles of Operation manual for a full list of exponent-overflow exceptions.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The thread is terminated.

Symbolic Feedback Code: CEE34C

CEE3213S The system detected an exponent-underflow exception (System Completion Code=0CD).

Explanation: Your program attempted a floating-point operation and the result characteristic is less than zero and the result fraction was not zero. See a Principles of Operation manual for a full list of exponent-underflow exceptions.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The thread is terminated.

Symbolic Feedback Code: CEE34D

CEE3214S The system detected a significance exception (System Completion Code=0CE).

Explanation: Your program attempted a floating-point addition or subtraction and the resulting fraction was zero. See a Principles of Operation manual for a full list of significance exceptions.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The thread is terminated.

Symbolic Feedback Code: CEE34E

CEE3215S The system detected a floating-point divide exception (System Completion Code=0CF).

Explanation: Your program attempted a do a floating-point divide and the divisor had a zero fraction. See a Principles of Operation manual for a full list of floating-point divide exceptions.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The thread is terminated.

Symbolic Feedback Code: CEE34F

CEE3216S The system detected an IEEE inexact exception. The result was truncated.

Explanation: An IEEE-inexact condition is recognized when the rounded result of an operation differs in value from the intermediate result computed as if exponent range and precision were unbounded.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The process is terminated.

Symbolic Feedback Code: CEE34G

CEE3217S The system detected an IEEE inexact exception. The result was truncated.

Explanation: An IEEE-inexact condition is recognized when the rounded result of an operation differs in value from the intermediate result computed as if exponent range and precision were unbounded.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The process is terminated.

Symbolic Feedback Code: CEE34H

CEE3218S The system detected an IEEE exponent-underflow exception. The result was truncated.

Explanation: An IEEE-underflow condition is recognized when the exponent of the exact result of an operation would be less than the minimum exponent of the target format.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The process is terminated.

Symbolic Feedback Code: CEE34I

CEE3219S The system detected an IEEE exponent-underflow exception. The result was inexact and truncated.

Explanation: An IEEE-underflow condition is recognized when the exponent of the exact result of an operation would be less than the minimum exponent of the target format.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The process is terminated.

Symbolic Feedback Code: CEE34J

CEE3220S The system detected an IEEE exponent-underflow exception. The result was inexact and incremented.

Explanation: An IEEE-underflow condition is recognized when the exponent of the exact result of an operation would be less than the minimum exponent of the target format.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The process is terminated.

Symbolic Feedback Code: CEE34K

CEE3221S The system detected an IEEE exponent-overflow exception.

Explanation: An IEEE-overflow condition is recognized when the exponent of the rounded result of an operation would be greater than the maximum exponent of the target format if the exponent range were unbounded.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The process is terminated.

Symbolic Feedback Code: CEE34L

CEE3222S The system detected an IEEE exponent-overflow exception. The result was inexact and truncated.

Explanation: An IEEE-overflow condition is recognized when the exponent of the rounded result of an operation would be greater than the maximum exponent of the target format if the exponent range were unbounded.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The process is terminated.

Symbolic Feedback Code: CEE34M

CEE3223S The system detected an IEEE exponent-overflow exception. The result was inexact and incremented.

Explanation: An IEEE-overflow condition is recognized when the exponent of the rounded result of an operation would be greater than the maximum exponent of the target format if the exponent range were unbounded.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The process is terminated.

Symbolic Feedback Code: CEE34N

CEE3224S The system detected an IEEE division—by—zero exception.

Explanation: An IEEE-division—by—zero condition is recognized when in BFP division the divisor is zero and the dividend is a finite nonzero number.

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The process is terminated.

Symbolic Feedback Code: CEE34O

CEE3225S The system detected an IEEE invalid operation exception.

Explanation: An IEEE-invalid-operation condition is recognized when any of the following occur:

1. An SNaN is encountered in any arithmetic or comparison operation.
2. A QNaN is encountered in a comparison by COMPARE AND SIGNAL.
3. A difference is undefined (addition of infinities of opposite sign, or subtraction of infinities of like sign).
4. A product is undefined (zero times infinity).
5. A quotient is undefined (DIVIDE instruction with both operands zero or both operands infinity).
6. A remainder is undefined (DIVIDE TO INTEGER with a dividend of infinity or a divisor of zero).
7. A square root is undefined (negative nonzero operand).

Programmer response: You can use a condition handling routine to correct the data values and resume the application.

System action: The process is terminated.

Symbolic Feedback Code: CEE34P

| **CEE3226S An IEEE inexact exception was reported.**

| **Explanation:** IEEE interruption simulation (IIS) event instructions were used to report an IEEE inexact exception.
| Floating point software might detect that a rounded result of an operation differed in value from the result that could
| have been computed if precision and exponent range were unbounded.

| **Programmer response:** You can use a condition handling routine to recover from the error and resume the
| application.

System action: The thread/process is terminated.

Symbolic Feedback Code: CEE34Q

CEE3227S An IEEE underflow exception was reported.

Explanation: IEEE interruption simulation (IIS) event instructions were used to report an IEEE underflow exception. Floating point software might detect that the exact result of an operation was nonzero and less in magnitude than the smallest normal number in the target format, but the result could still be represented exactly.

Programmer response: Use a condition handling routine to recover the error and resume the application.

System action: The thread/process is terminated.

Symbolic Feedback Code: CEE34R

CEE3228S An IEEE underflow and inexact exception was reported.

Explanation: IEEE interruption simulation (IIS) event instructions were used to report an IEEE underflow and inexact exception. Floating point software might detect that the exact result of an operation was nonzero and less in magnitude than the smallest normal number in the target format, and the result could not be represented exactly.

Programmer response: Use a condition handling routine to recover the error and resume the application.

System action: The thread/process is terminated.

Symbolic Feedback Code: CEE34S

CEE3229S An IEEE overflow exception was reported.

Explanation: IEEE interruption simulation (IIS) event instructions were used to report an IEEE overflow exception. Floating point software might detect that the rounded result of an operation was larger in magnitude than the largest finite number in the target format, but the scaled result could still be represented exactly.

Programmer response: Use a condition handling routine to recover the error and resume the application.

System action: The thread/process is terminated.

Symbolic Feedback Code: CEE34T

CEE3230E Vector unnormalized operand exception occurred.

Explanation: The parameters to the vector instruction were floating-point numbers that are unnormalized.

Programmer response: The data to be processed by the vector instructions must be normalized before it is to be handled in a vector instruction. Normalize the input value by adding floating-point zero (0.0) to the qdata item.

System action: The user program is terminated unless the condition is handled.

Symbolic Feedback Code: CEE34U

CEE3231S An IEEE overflow and inexact exception was reported.

Explanation: IEEE interruption simulation (IIS) event instructions were used to report an IEEE overflow and inexact exception. Floating point software might detect that the rounded result of an operation was larger in magnitude than the largest finite number in the target format, and the scaled result could not be represented exactly.

Programmer response: Use a condition handling routine to recover the error and resume the application.

System action: The thread/process is terminated.

Symbolic Feedback Code: CEE34V

| **CEE3232S An IEEE division-by-zero exception was reported.**

| **Explanation:** IEEE interruption simulation (IIS) event instructions were used to report an IEEE division-by-zero
| exception. Floating point software might detect that a finite non-zero floating point number was divided by zero.

| **Programmer response:** Use a condition handling routine to recover the error and resume the application.

| **System action:** The thread/process is terminated.

| **Symbolic Feedback Code:** CEE350

| **CEE3233S An IEEE invalid-operation exception was reported.**

| **Explanation:** IEEE interruption simulation (IIS) event instructions were used to report an IEEE invalid-operation
| exception. Floating point software might detect one of the following situations:

| • A NaN was encountered.
| • The result of an arithmetic operation was undefined (for example: zero divided by zero).
| • The result of a QUANTIZE, REROUND, conversion, or other operations was undefined.

| **Programmer response:** Use a condition handling routine to recover the error and resume the application.

| **System action:** The thread/process is terminated.

| **Symbolic Feedback Code:** CEE351

CEE3250C The system or user abend ***abend-code*** **was issued.**

Explanation: A system or user abend has occurred.

Programmer response: Look in the messages and codes or system codes manual for the particular platform to resolve the system-described problem.

System action: The program is terminated abnormally.

Symbolic Feedback Code: CEE35I

CEE3251I An ATTENTION condition occurred.

Explanation: An ATTENTION condition was signaled after polling code was invoked.

Programmer response: Do whatever is appropriate for the user to do, after the user hits the ″attention″ key.

System action: The program is resumed after the point where the condition was signaled.

Symbolic Feedback Code: CEE35J

CEE3253C A critical condition occurred during the sort operation.

Explanation: An unrecoverable error prevented SORT from completing.

Programmer response: Take the appropriate action defined by the SORT messages.

System action: The thread is terminated.

Symbolic Feedback Code: CEE35L

CEE3254C An incorrect DFSORT Plist was passed to CEE3SRT.

Explanation: The parameter list for CEE3SRT must be the 31 bit list specified by DFSORT.

Programmer response: Correct the parameter list for CEE3SRT.

System action: The thread is terminated.

Symbolic Feedback Code: CEE35M

CEE3255C An attempt to call CEE3SRT was made from within a DFSORT exit routine.

Explanation: Only one sort can be active at a time. A program called during the execution of SORT must have attempted to invoke sort again.

Programmer response: Do not attempt a sort from within a sort exit.

System action: The thread is terminated.

Symbolic Feedback Code: CEE35N

CEE3260W No condition was active when a call to a condition management routine was made.

Explanation: The condition manager had no record of an active condition.

Programmer response: No response is required. Calls to condition management routines should only be made within the handler routine.

System action: No system action is performed.

Symbolic Feedback Code: CEE35S

CEE3261W *service-name* **is not supported.**

Explanation: The service was no longer supported. It was provided for migration and compatibility with previous releases of Language Environment.

Programmer response: Migrate an application to a supported function.

System action: The service did not take any action.

Symbolic Feedback Code: CEE35T

CEE3262W An invalid condition token was passed. The condition token did not represent an active condition.

Explanation: The condition token passed to CEE3CIB did not represent a condition that is currently active.

Programmer response: No programmer response required.

System action: No system action is taken.

Symbolic Feedback Code: CEE35U

CEE3263C The condition handler's condition information block was damaged. The requested function was not performed.

Explanation: The condition manager did not have a valid CIB chain.

Programmer response: This is an internal problem. Contact you service representative.

System action: The requested function is not performed.

Symbolic Feedback Code: CEE35V

CEE3264S No machine state block found in association with the current stack frame.

Explanation: Your program has not established a valid machine state block (via CEE3SRP) associated with the current stack frame.

Programmer response: Make sure CEE3SRP is issued before calling CEE3GMB.

System action: The thread is terminated.

Symbolic Feedback Code: CEE360

CEE3292W The language run-time component id was already registered. No action was taken.

Explanation: The CEE3DHDL CWI was invoked previously with the same language run-time component id.

Programmer response: No programmer response required.

System action: If this condition is not handled, execution continues at the instruction after the CEE3DHDL invocation.

Symbolic Feedback Code: CEE36S

CEE3293C The Language Environment was corrupted. The save area chain was broken.

Explanation: The save area chain was not intact.

Programmer response: Language Environment always expects the save area to be valid, and usually abends when it is not. Ensure that the save chain is valid.

System action: The function was not completed, and did not schedule the routine.

Symbolic Feedback Code: CEE36T

CEE3294E The cancel request could not be performed, since the routine was not previously scheduled.

Explanation: A request was made to cancel a routine, but that routine could not be found on the active chain. The routine that was requested to cancel either was never scheduled or was previously deleted.

Programmer response: Ensure that routines are scheduled via CEEHDLR before an attempt is made to delete them.

System action: No routine is released.

Symbolic Feedback Code: CEE36U

CEE3295E The condition string from CEE3SPM did not contain all of the settings, because the returned string was truncated.

Explanation: The QUERY option of the CEE3SPM service needed a larger character string to represent the conditions.

Programmer response: Try increasing the character string length.

System action: Some items might have been filled in.

Symbolic Feedback Code: CEE36V

CEE3296E Some of the data in the condition string from CEE3SPM could not be recognized.

Explanation: The data encountered in the string could not be interpreted.

Programmer response: Correct the character representation for the condition(s) and ensure that the string is padded with blanks.

System action: Only conditions that could be recognized were set.

Symbolic Feedback Code: CEE370

CEE3297E The service completed successfully for recognized condition(s), unsuccessfully for unrecognized (invalid) condition(s).

Explanation: The data encountered in the string could not be interpreted.

Programmer response: Correct the character representation for the conditions.

System action: Only conditions that could be recognized were set.

Symbolic Feedback Code: CEE371

CEE3298E CEE3SPM attempted to PUSH settings onto a full stack.

Explanation: There was not enough storage for the CEE3SPM PUSH service to save all of the conditions.

Programmer response: Increase the size of the storage.

System action: No settings were changed.

Symbolic Feedback Code: CEE372

CEE3299E CEE3SPM attempted to POP settings off an empty stack.

Explanation: A call to CEE3SPM was made to POP the stack. There were no elements on the stack to POP.

Programmer response: Ensure that something is on the stack before you attempt to POP it.

System action: No settings are changed.

Symbolic Feedback Code: CEE373

CEE3300E The action parameter in CEE3SPM was not one of the digits 1 to 5.

Explanation: A call to CEE3SPM was made with an invalid action.

Programmer response: Use an action value parameter between 1 and 5 when invoking CEE3SPM.

Symbolic Feedback Code: CEE374

System action: No settings were changed.

CEE3301E The first parameter was not one of the digits expected.

Explanation: A call was made to a condition management subroutine that did not have a valid parameter for the action parameter.

Programmer response: This is an internal error. The internal routine was called with an improper parameter. Contact your service representative.

System action: No system action is performed.

Symbolic Feedback Code: CEE375

CEE3350S Unable to find the event handler.

Explanation: An internal error occurred when Language Environment attempted to load a required language run-time component module.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE38M

CEE3351S Unable to properly initialize the event handler.

Explanation: An internal error occurred when Language Environment attempted to initialize a required language run-time component module.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE38N

CEE3352E The enclave terminated with a non-zero return code.

Explanation: An internal error occurred while attempting to terminate an enclave.

Programmer response: Contact your service representative.

System action: No system action is performed.

Symbolic Feedback Code: CEE38O

CEE3353S The parameter manipulation service was called, but not during the create enclave event, or not by a language run-time component corresponding to the MAIN program.

Explanation: The parameter manipulation service was used during enclave initialization by the language in which the main program was written. It was used in an illegal manner.

Programmer response: This is an internal problem. Contact your service representative.

System action: The requested parameter manipulation is not performed and the main parameter list might not be correct.

Symbolic Feedback Code: CEE38P

CEE3354S The parameter list manipulation service was called in a CICS environment.

Explanation: The parameter manipulation service was used during enclave initialization by the language in which the main program was written. It cannot be used in a CICS environment.

Programmer response: This is an internal problem. Contact your service representative.

System action: The requested parameter manipulation is not performed and the main parameter list might not be correct.

Symbolic Feedback Code: CEE38Q

CEE3355S A language run-time component initialization has failed.

Explanation: An internal error occurred while attempting to establish a minimum environment for a language run-time component.

Programmer response: This is an internal problem. Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE38R

CEE3356S The rc_modifier must be in the range of 1 through 4. The return code modifier was not changed.

Explanation: The rc_modifier was not in the range of 1 through 4. The return code modifier that was first established by the enclave termination services or by the condition handling was kept.

Programmer response: Provide a valid rc_modifier.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE38S

CEE3357S The service was invoked outside of the language run-time component enclave termination. No action was taken.

Explanation: CEESRCM was to be called during the language run-time component enclave termination. It was invoked outside of the language run-time component enclave termination.

Programmer response: Ensure that the routine is called during the enclave termination.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE38T

CEE3358E The service was invoked outside of the member enclave initialization. No action was taken.

Explanation: This CWI service can only be invoked from within member language enclave initialization.

Programmer response: Move the use of this service to within enclave initialization event handling, or, determine the proper event to be using at the point where you are trying to invoke this event.

System action: The service returns, without performing the function of the service.

Symbolic Feedback Code: CEE38U

CEE3359E The module or language list is not supported in this environment.

Explanation: The module or language list is not supported in this environment.

Programmer response: Use a supported module or language list for this service when running on this version of the operating system.

System action: The function is not performed.

Symbolic Feedback Code: CEE38V

CEE3360S The stack frame was not found on the call chain.

Explanation: The stack frame parameter passed to the CEE3SMS CWI did not point to a valid stack frame on the call chain.

Programmer response: This is an internal problem. Contact your service representative.

System action: The CEE3SMS CWI returns without allocating a machine state control block.

Symbolic Feedback Code: CEE390

CEE3361W A nested enclave completed with an unhandled condition of severity two or greater.

Explanation: If a nested enclave is created due to an SVC-assisted linkage (LINK on VM or MVS, CMSCALL on VM), and it subsequently abends or program checks, or it software-signals a condition of severity two or greater, then condition token CEE391 was signaled in the creator of the nested enclave.

Programmer response: Check condition token CEE391.

System action: If the signal of the CEE391 condition is not handled, execution continues at the instruction after the LINK or CMSCALL.

Symbolic Feedback Code: CEE391

CEE3362S No main or fetchable procedure or function was present within the load module.

Explanation: The load module contained neither a main procedure/function nor a fetchable procedure/function.

Programmer response: Correct the load module.

System action: The application is terminated.

Symbolic Feedback Code: CEE392

CEE3363S A second main procedure or function was entered without crossing a nested enclave boundary.

Explanation: A direct call was made to a main procedure. The program should have been loaded and/or called using a defined language construct like `fetch()` or `system()`.

Programmer response: Correct the load module.

System action: The application is terminated.

Symbolic Feedback Code: CEE393

CEE3364W The enclave name was truncated by the enclave naming service during initialization.

Explanation: The enclave naming service was used by the language in which the main program was written during enclave initialization. It was passed a name longer than 32 characters.

Programmer response: This is an internal problem. Contact your service representative.

System action: The truncated name is used as the enclave name.

Symbolic Feedback Code: CEE394

CEE3365S The enclave naming service was called, but not during enclave initialization, or not by a language run-time component corresponding to the MAIN program.

Explanation: The enclave naming service was used by the language in which the main program was written during enclave initialization. It was used in an illegal manner.

Programmer response: This is an internal problem. Contact your service representative.

System action: The application is terminated.

Symbolic Feedback Code: CEE395

CEE3370W The program invocation name could not be found, and the returned name was blank.

Explanation: CEEBGIN could not determine the name under which the program was invoked.

Programmer response: No response is required.

System action: No system action is performed.

Symbolic Feedback Code: CEE39A

CEE3380W The target load module was not recognized by Language Environment.

Explanation: The language list could not be returned because the target load module was not recognized. A value of zero was returned.

Programmer response: No response is required.

System action: Processing continues. No system action is performed.

Symbolic Feedback Code: CEE39K

CEE3400W The condition name was not recognized and the value of the condition token was undefined.

Explanation: CEEQFBC was passed a condition name that could not be translated into a corresponding Language Environment condition token.

Programmer response: No programmer action is required.

System action: No system action is taken.

Symbolic Feedback Code: CEE3A8

CEE3401W The condition token was not recognized and the value of the condition name was undefined.

Explanation: CEEBFBC was passed a condition token that could not be translated into a corresponding condition name.

Programmer response: No programmer action is required.

System action: No system action is taken.

Symbolic Feedback Code: CEE3A9

CEE3402E The condition token passed was invalid and the value of the condition name was undefined.

Explanation: CEEBFBC was passed a condition token that was determined to be invalid and could not to be translated into a corresponding condition name.

Programmer response: No programmer action is required.

System action: No system action is taken.

Symbolic Feedback Code: CEE3AA

CEE3424S CEE3SMO was called from outside a user-written condition handler.

Explanation: CEE3SMO can only be called from within a user-written condition handler.

Programmer response: Only code calls to CEE3SMO from within user-written condition handlers.

System action: The application is terminated.

Symbolic Feedback Code: CEE3B0

CEE3425S Severity 0 or 1 condition was signaled with CEESGLN.

Explanation: The caller of CEESGLN signaled a severity 0 or 1 condition; however, resumption is never allowed for a condition signaled from CEESGLN.

Programmer response: Do not signal severity 0 and 1 conditions from CEESGLN. Use CEESGL when signaling a severity 0 and 1 conditions.

System action: The condition is changed to CEE3B1 and, if unhandled, the enclave is terminated.

Symbolic Feedback Code: CEE3B1

CEE3426S There was an invalid request to fix-up and resume a condition.

Explanation: There was a request to fix-up and resume from a user-written condition handler and either (1) the resume cursor was moved, or (2) the condition was signaled from CEESGLN or from CEESGL without a feedback code. The resume cursor can not be moved by a user-written condition handler if fix-up and resume behavior is desired. Conditions signaled from CEESGLN or from CEESGL without a feedback code can not be resumed without moving the resume cursor. The original condition is indicated in the next message in the message file.

Programmer response: A user-written condition handler must move the resume cursor and return a result code of 10 (Resume) in order to resume a condition signaled by CEESGLN or by CEESGL without a feedback code.

System action: The condition is promoted to CEE3B2 and, if unhandled, the enclave is terminated.

Symbolic Feedback Code: CEE3B2

CEE3427S A user-written condition handler promoted a condition signaled by CEESGLN to severity 0 or 1.

Explanation: The condition handling mechanism allows condition handlers to promote their current condition and then handle new ones. If a severity 2 or above condition signaled by CEESGLN was promoted to a 0 or 1 condition, the purpose of CEESGLN would be violated - programs can never resume following a call to CEESGLN. (Language Environment allows severity 0 or 1 conditions to resume.) Note that the original condition is indicated in the next message in the message file.

Programmer response: User-written condition handlers must not promote conditions that are not allowed to resume to severity 0 or 1.

System action: The condition is promoted to CEE3B3 and if unhandled the enclave is terminated.

Symbolic Feedback Code: CEE3B3

CEE3428S Condition signaled by CEESGLN is not enabled by a language run-time component.

Explanation: Some conditions are disabled by a language run-time component. For example Fixed Point Overflow conditions in COBOL are ignored and the application is resumed. Such a condition must not be signaled by CEESGLN. Note that the original condition is indicated in the next message in the message file.

Programmer response: Do not use CEESGLN to signal a condition from a language that does not enable the condition.

System action: The condition is promoted to CEE3B4 and if unhandled the enclave is terminated.

Symbolic Feedback Code: CEE3B4

CEE3429S Move resume cursor relative is not permitted in a user-written condition handler registered with the USRHDLR run-time option.

Explanation: You can register a user-written condition handler at stack frame 0 with the USRHDLR run-time option. The move resume cursor relative (CEEMRCR) service was not permitted at stack frame 0.

Programmer response: Remove all references to CEEMRCR in user-written condition handlers registered with the USRHDLR run-time option.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3B5

CEE3449S An internal message services error occurred during termination.

Explanation: A message service was called to perform a service during termination, but the service could not be completed because certain resources were no longer available.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3BP

CEE3450E Only one language was on the stack when a POP request was made to CEE3LNG. The current language was returned in the desired language parameter.

Explanation: CEE3LNG cannot POP since the resulting stack was empty.

Programmer response: No programmer action is required.

System action: The current language is returned in the *desired_language* parameter and the stack remains unchanged.

Symbolic Feedback Code: CEE3BQ

CEE3451S The desired language *desired-language* **for the PUSH or SET function for CEE3LNG was invalid. No operation was performed.**

Explanation: The *desired_language* parameter was not a valid 3-character national language id.

Programmer response: Provide a valid *desired_language* parameter. A list of the valid national languages is provided in *z/OS Language Environment Programming Reference*.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3BR

CEE3452S The function *function* **specified for CEE3LNG was not recognized. No operation was performed.**

Explanation: The function parameter must be a fullword binary 1, 2, 3, or 4.

Programmer response: Provide a fullword binary 1, 2, 3, or 4 in the function parameter.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3BS

CEE3454S **The *function* requested in CEE3LNG failed because at least one of the high-level languages did not accept the change from the *function*.**

Explanation: An internal error prevented the requested change from being made.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3BU

CEE3455E **Only one country code was on the stack when a POP request was made to CEE3CTY. The current country code was returned in the country code parameter.**

Explanation: CEE3CTY cannot POP the stack since the resulting stack was empty.

Programmer response: No programmer action is required.

System action: The current country code is returned in the *country_code* parameter and the stack remains unchanged.

Symbolic Feedback Code: CEE3BV

CEE3456S **The country code *country-code* for the PUSH or SET function for CEE3CTY was invalid. No operation was performed.**

Explanation: The *country_code* parameter was not a valid 2-character country code.

Programmer response: Provide a valid *country_code* parameter. A list of the valid country codes is provided in *z/OS Language Environment Programming Reference*.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3C0

CEE3457S **The function *function* specified for CEE3CTY was not recognized. No operation was performed.**

Explanation: The *function* parameter must be a fullword binary 1, 2, 3, or 4.

Programmer response: Provide a fullword binary 1, 2, 3, or 4 in the *function* parameter.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3C1

CEE3458E **The country code *country_code* was invalid for CEE3MC2. The default currency symbol *currency_symbol* was returned. No international currency symbol was returned.**

Explanation: The *country_code* parameter was not a valid 2-character country code. The default currency symbol was returned. No international currency symbol was returned.

Programmer response: Provide a valid country_code parameter. A list of the valid country codes is provided in the *z/OS Language Environment Programming Reference*.

System action: The default currency symbol is returned. No international currency symbol is returned.

Symbolic Feedback Code: CEE3C2

CEE3459S **The *function* requested in CEE3CTY failed because at least one of the high-level languages did not accept the change from the *function*.**

Explanation: The function requested failed because one of the high-level languages did not accept the change.

Programmer response: An internal error prevented the requested change from being made. Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3C3

CEE3460E The decimal separator '*decimal_separator*' was truncated and was not defined in CEE3MDS.

Explanation: The *decimal_separator* parameter must be a 2-character field. The resulting decimal separator might not be valid. The decimal separator was left-justified and padded on the right with a blank if necessary.

Programmer response: Provide a 2-character *decimal_separator* parameter.

System action: The decimal separator is truncated and placed into the given parameter.

Symbolic Feedback Code: CEE3C4

CEE3461E The country code *country_code* was invalid for CEE3MDS. The default decimal separator '*decimal_separator*' was returned.

Explanation: The *country_code* parameter was not a valid 2-character country code. The default decimal separator was returned.

Programmer response: Provide a valid *country_code* parameter. A list of the valid country codes is provided in *z/OS Language Environment Programming Reference*.

System action: The default decimal separator is returned.

Symbolic Feedback Code: CEE3C5

CEE3462E The currency symbol '*currency_symbol*' was truncated and was not defined in CEE3MCS.

Explanation: The *currency_symbol* parameter must be a 2-character field. The resulting currency symbol might not be valid. The currency symbol was left-justified and padded on the right with a blank if necessary.

Programmer response: Provide a 2-character *currency_symbol* parameter.

System action: The currency symbol is truncated and placed into the given parameter.

Symbolic Feedback Code: CEE3C6

CEE3463E The country code *country_code* was invalid for CEE3MCS. The default currency symbol '*currency_symbol*' was returned.

Explanation: The *country_code* parameter was not a valid 2-character country code. The default currency symbol was returned.

Programmer response: Provide a valid *country_code* parameter. A list of the valid country codes is provided in *z/OS Language Environment Programming Reference*.

System action: The default currency symbol is returned.

Symbolic Feedback Code: CEE3C7

CEE3464E The thousands separator '*thousands_separator*' was truncated and was not defined in CEE3MTS.

Explanation: The *thousands_separator* parameter must be a 2-character field. The resulting thousands separator might not be valid. The thousands separator was left-justified and padded on the right with a blank if necessary.

Programmer response: Provide a 2-character *thousands_separator* parameter.

System action: The thousands separator is truncated and placed into the given parameter.

Symbolic Feedback Code: CEE3C8

CEE3465E The country code *country_code* was invalid for CEE3MTS. The default thousands separator '*thousands_separator*' was returned.

Explanation: The *country_code* parameter was not a valid 2-character country code. The default thousands separator was returned.

Programmer response: Provide a valid *country_code* parameter. A list of the valid country codes is provided in *z/OS Language Environment Programming Reference*.

System action: The default thousands separator is returned.

Symbolic Feedback Code: CEE3C9

CEE3466E The date picture string *date_pic_string* was truncated and was not defined in CEEFMDA.

Explanation: The *date_pic_string* parameter must be an 80-character field. The resulting *date_pic_string* might not be valid. The *date_pic_string* was left-justified and padded on the right with a blank if necessary.

Programmer response: Provide an 80-character *date_pic_string* parameter.

System action: The *date_pic_string* is truncated and placed into the given parameter.

Symbolic Feedback Code: CEE3CA

CEE3467E The country code *country_code* was invalid for CEEFMDA. The default date picture string *date_pic_string* was returned.

Explanation: The *country_code* parameter was not a valid 2-character country code. The default date picture string was returned.

Programmer response: Provide a valid *country_code* parameter. A list of the valid country codes is provided in *z/OS Language Environment Programming Reference*.

System action: The default date picture string is returned.

Symbolic Feedback Code: CEE3CB

CEE3468E The time picture string *time_pic_string* was truncated and was not defined in CEEFMTM.

Explanation: The *time_pic_string* parameter must be an 80-character field. The resulting *time_pic_string* might not be valid. The *time_pic_string* was left-justified and padded on the right with a blank if necessary.

Programmer response: Provide an 80-character *time_pic_string* parameter.

System action: The *time_pic_string* is truncated and placed into the given parameter.

Symbolic Feedback Code: CEE3CC

CEE3469E The country code *country_code* was invalid for CEEFMTM. The default time picture string *time_pic_string* was returned.

Explanation: The *country_code* parameter was not a valid 2-character country code. The default time picture string was returned.

Programmer response: Provide a valid *country_code* parameter. A list of the valid country codes is provided in *z/OS Language Environment Programming Reference*.

System action: The default time picture string is returned.

Symbolic Feedback Code: CEE3CD

CEE3470E The date and time string *datetime_str* was truncated and was not defined in CEEFMDT.

Explanation: The *datetime_str* parameter must be an 80-character field. The resulting *datetime_str* might not be valid. The *datetime_str* was left-justified and padded on the right with a blank if necessary.

Programmer response: Provide an 80-character *datetime_str* parameter.

System action: The *datetime_str* is truncated and placed into the given parameter.

Symbolic Feedback Code: CEE3CE

CEE3471E The country code *country_code* was invalid for CEEFMDT. The default date and time picture string *datetime_str* was returned.

Explanation: The *country_code* parameter was not a valid 2-character country code. The default date and time string was returned.

Programmer response: Provide a valid *country_code* parameter. A list of the valid country codes is provided in *z/OS Language Environment Programming Reference*.

System action: The default date and time string is returned.

Symbolic Feedback Code: CEE3CF

CEE3472S An internal message services error occurred while getting storage for the message inserts.

Explanation: Insufficient heap storage was available to complete message services.

Programmer response: If possible, free unneeded heap storage or contact your service representative.

System action: No message insert area is created.

Symbolic Feedback Code: CEE3CG

CEE3473S An internal message services error occurred while processing the inserts for this message.

Explanation: Corrupted storage was encountered when attempting to initialize a message insert block.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3CH

CEE3475S An internal message services error occurred while freeing the insert area.

Explanation: Corrupted storage was encountered when attempting to free a message insert block.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3CJ

CEE3476S An internal message services error occurred while freeing storage for the message inserts.

Explanation: Message services detected a heap storage freeing failure.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3CK

CEE3480S An internal message services error occurred while processing the inserts for a message.

Explanation: Message services detected an insert error while formatting a message.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3CO

CEE3481S An internal message services error occurred while processing the inserts for a message.

Explanation: Corrupted storage was encountered when attempting to process a message insert block.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3CP

CEE3482S An internal message services error occurred while processing the inserts for a message.

Explanation: An invalid insert was encountered when attempting to process a message insert block.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3CQ

CEE3484E A message could not be written to *ddname* **because the message length of** *message_length* **exceeded the allowable maximum of** *max-message-length***.**

Explanation: The message could not be written to *ddname* data set because message services will not process a message whose length is greater than *max-message-length*.

Programmer response: Reduce the size of the message or divide it into sections of acceptable length.

System action: The message is not written.

Symbolic Feedback Code: CEE3CS

CEE3485S An internal message services error occurred while locating the message number within a message file.

Explanation: The message library for the given message number was located and loaded, but the message number could not be found within the library.

Programmer response: Contact your service representative.

System action: The given message library is loaded, but no other action is performed.

Symbolic Feedback Code: CEE3CT

CEE3486S An internal message services error occurred while formatting a message.

Explanation: Corrupted storage was encountered when attempting to process a message insert block.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3CU

CEE3487S An internal message services error occurred while locating a message number within the ranges specified in the repository.

Explanation: The message number could not be found within the ranges in the message_library_table.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3CV

CEE3488S An internal message services error occurred while formatting a message.

Explanation: An invalid internal message buffer length was detected while formatting a message.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3D0

CEE3489S An internal message services error occurred while getting storage necessary to format a message.

Explanation: No heap storage was available to get storage needed to complete the formatting of a message.

Programmer response: If possible, free unneeded heap storage or contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3D1

CEE3490S An internal message services error occurred while attempting to write a message.

Explanation: The given ddname or destination was not valid or was not available.

Programmer response: Contact your service representative.

System action: The message is not written.

Symbolic Feedback Code: CEE3D2

CEE3491S An internal message services error occurred while getting storage.

Explanation: No heap storage was available to get storage needed to write out a message.

Programmer response: If possible, free unneeded heap storage or contact your service representative.

System action: The message is not written.

Symbolic Feedback Code: CEE3D3

CEE3492S An internal message services error occurred while attempting to write a message.

Explanation: An error was detected while trying to OPEN, WRITE, or CLOSE a given ddname or destination.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3D4

CEE3493W An internal message services error occurred while attempting to close the message file.

Explanation: Language Environment could not close the specified ddname, because Language Environment either did not own it, or the file was not currently open.

Programmer response: Contact your service representative.

System action: No system action is performed.

Symbolic Feedback Code: CEE3D5

CEE3494S An internal message services error occurred while attempting to close the message file.

Explanation: An error was detected while trying to CLOSE the given ddname.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3D6

CEE3495S An internal message services error occurred while formatting a message.

Explanation: An error preventing the completion of message formatting was detected.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3D7

CEE3496I An internal message services error occurred while formatting a message.

Explanation: An internal error was detected while locating the inserts for a message.

Programmer response: Contact your service representative.

System action: No system action is performed.

Symbolic Feedback Code: CEE3D8

CEE3497E The message file was discovered to have an insufficient LRECL of ***too-small*****.**

Explanation: The message file cannot have an LRECL less than 14. This was to allow for the message number and 4 characters of text per line.

Programmer response: Specify an LRECL of 14 or greater.

System action: The message file LRECL is forced to the default LRECL value.

Symbolic Feedback Code: CEE3D9

CEE3498I The message file was already open.

Explanation: A request to open the message file via CEEOPMF could not be completed because it was already open.

Programmer response: No programmer response is necessary.

System action: No system action is taken.

Symbolic Feedback Code: CEE3DA

CEE3499E The message file was unable to be opened.

Explanation: A request to open the message file via CEEOPMF could not be completed.

Programmer response: Ensure that the ddname to used for the message file is a valid name, and the data set is usable.

System action: No system action is taken.

Symbolic Feedback Code: CEE3DB

CEE3500S Not enough storage was available to load ***module-name*****.**

Explanation: Not enough storage was available to load the requested module into virtual memory.

| **Programmer response:** Ensure that the region size is sufficient to run the application. If necessary, delete the
| modules that the application does not need or delete free unused storage, and retry the load request. The module
| name specified in the message might be truncated for display purposes.

System action: Module is not loaded. The application might abend.

| If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the
| error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing
| ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the
| _EDC_DLL_DIAG environment variable.

If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

If this was a DLL load or open request, a CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3DC

CEE3501S The module *module-name* was not found.

Explanation: The system could not find the load module whose name was specified on the parameter list to the Language Environment load service, in the indicated library (job library or link library).

Programmer response: Make sure the requesting program name was not incorrectly modified. Make sure that the indicated library is correct in the job step. Correct the error, and execute the job step again. The module name specified in the message might be truncated for display purposes.

System action: Module is not loaded. The application might abend.

If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

If this was a DLL load or open request, a CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3DD

CEE3502S The module name *module-name* was too long.

Explanation: The module name length was greater than the name length supported by the underlying operating system.

Programmer response: Correct the module name length and execute the job step again. The module name specified in the message was truncated for display purposes.

System action: Name length is truncated to the name length supported by the underlying operating system. The requested module might or might not be loaded.

If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

If this was a DLL load or open request, a CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3DE

CEE3503S The load request for module *module-name* was unsuccessful.

Explanation: The system could not load the load module.

Programmer response: Check the original abend from the operating system and refer to the underlying operating system message manual for explanation and programer's response. The module name specified in the message might be truncated for display purposes.

System action: Module was not loaded. The application might abend.

If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message wass sent to the Language Environment message file. Other DLL diagnostic options, such as issuing

| ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the
| _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to
| the C dlerror() function.

| If this was a DLL load or open request, a CEEDLLF DLL failure control block was populated with further DLL error
| diagnostics.

Symbolic Feedback Code: CEE3DF

CEE3504S The delete request for module *module-name* **was unsuccessful.**

Explanation: The load module might already have been deleted or was never loaded.

Programmer response: Make sure the requesting module name is not incorrectly modified.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3DG

CEE3505S The library vector table (LIBVEC) descriptor module *module-name* **could not be loaded.**

Explanation: During LIBVEC initialization the library vector table descriptor module could not be found.

Programmer response: Make sure you passed the name of the library vector table (LIBVEC) descriptor rather than the entry address. Make sure the name was not incorrectly modified and the indicated library (job library or link library) is correct.

System action: LIBVEC initialization is not performed.

Symbolic Feedback Code: CEE3DH

CEE3506S The library packaged subroutine module *module-name* **could not be loaded.**

Explanation: The system could not find the load module whose name was specified in the library vector table (LIBVEC) descriptor module in the indicated library (job library or link library).

Programmer response: Make sure the requesting program name was not incorrectly modified. Make sure that the indicated library is correct in the job step. Correct the error, and execute the job step again.

System action: Module is not loaded and LIBVEC initialization is not performed.

Symbolic Feedback Code: CEE3DI

CEE3507S Not enough storage was available for the library vector table (LIBVEC) *table-name***.**

Explanation: Insufficient storage was available to build the LIBVEC.

Programmer response: If necessary, free available storage and retry LIBVEC initialization.

System action: No storage is allocated for the LIBVEC. LIBVEC initialization did not complete.

Symbolic Feedback Code: CEE3DJ

CEE3508S The number of library packaged subroutines specified in the descriptor module *module-name* **exceeded the maximum of 256 library packages. No library packages were loaded.**

Explanation: The maximum number of library packages supported is 256.

Programmer response: Repackage the library routines so that the number of library packages does not exceed the maximum supported.

System action: LIBVEC initialization did not complete.

Symbolic Feedback Code: CEE3DK

CEE3509S The number of library vector slots specified in the descriptor module *module-name* either exceeded 1024 or was less than 1.

Explanation: A minimum of 1 library routine entry name was required to build the library vector table. A maximum of 1024 library routines entry names is allowed in a LIBVEC.

Programmer response: If library vector slots exceed the maximum, you might have to build more than one LIBVEC.

System action: LIBVEC initialization did not complete.

Symbolic Feedback Code: CEE3DL

CEE3510S The module *module-name* is a member of the library packaged subroutine *module-name* and could not be deleted.

Explanation: Library package subroutines are not allowed to be deleted.

Programmer response: Correct your program so that it does not request a library package subroutine be deleted.

System action: Module not deleted.

Symbolic Feedback Code: CEE3DM

CEE3511S The function code *function-code* was invalid.

Explanation: Valid functions codes for the verify library vector subroutine are delete and load.

Programmer response: Make sure the function code passed to the verify library vector subroutine is delete/load. Correct the program and execute job step again.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3DN

| **CEE3512S An z/OS UNIX file system load of module *module-name* failed. The system return code was *return-code*; the reason code was *reason-code*.**

| **Explanation:** The callable service BPX1LOD failed while attempting to load module *module-name* from the z/OS UNIX file system. The system return and reason codes were returned.

| **Programmer response:** See *z/OS UNIX System Services Messages and Codes* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary. The module name specified in the message may have been truncated for display purposes.

System action: Unless the condition is handled, the default action is to terminate the enclave.

| If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| If this was a DLL load or open request, a CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3DO

CEE3513S The library vector table (LIBVEC) *table-name* could not be terminated.

Explanation: LIBVEC termination failed due to an inability to delete library subroutines or free the storage obtained for the LIBVEC.

Programmer response: Contact your service representative.

System action: LIBVEC termination did not complete.

Symbolic Feedback Code: CEE3DP

CEE3514C An internal error, Unknown Operating System, was detected.

Explanation: The underlying operating system was unsupported in Language Environment.

Programmer response: Language Environment runs under the control of, or in conjunction with, the following operating systems/subsystems: MVS/ESA, CICS/ESA, IMS/ESA, DB2, SQL/DS, DFSORT, ISPF, and TSO/E.

System action: The application is terminated.

Symbolic Feedback Code: CEE3DQ

CEE3515I No modules were loaded.

Explanation: No application load modules have been loaded via Language Environment load service in the current application.

Programmer response: No programmer response is necessary.

System action: No system action is taken.

Symbolic Feedback Code: CEE3DR

CEE3517S The dynamic allocation of the msgfile was not successful. The reason code was *reason-code*.

Explanation: Language Environment attempted to dynamically allocate a ddname for the msgfile. However, the allocation was not successful.

Programmer response: Use the error reason code to determine why the dynalloc service did not complete successfully.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3DT

CEE3518S The module *module-name* was not found in an authorized library

Explanation: An authorized program requested the load of a module that could not be found in an authorized library or concatenation of libraries.

| **Programmer response:** If the requested module can not be found, make sure the module exists in a system or
| user-defined authorized library. Correct the error, and run the job step again. The module name specified in the
| message might be truncated for display purposes.

System action: Module is not loaded. The application might abend.

| If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the
| error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing
| ctrace(), signaling a condition, and the turning off _EDC_DLL_DIAG diagnostics, were available through the
| _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to
| the C dlerror() function.

| If this was a DLL load or open request, a CEEDLLF DLL failure control block was populated with further DLL error
| diagnostics.

Symbolic Feedback Code: CEE3DU

CEE3530S The service was invoked for a load module.

Explanation: The CEEPPOS service was invoked for a load module. Only program objects are supported. No action was taken.

Programmer response: Contact your service representative.

System action: Unless the condition is handled the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3EA

CEE3531S The entry point was not recognized by Language Environment.

Explanation: The CEEPPOS service was invoked and Language Environment was not able to recognize the entry point style. Only Language Environment enabled entry point styles are supported for program objects. No action was taken.

Programmer response: Contact your service representative.

System action: Unless the condition is handled the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3EB

CEE3532S The requested class does not exist in the program object.

Explanation: The CEEPPOS service was invoked. If this is an OBTAIN for class C_WSA then this indicates that the program object does not have writable static. If this is a LOCATE for class C_@@DLLI, C_@@STINIT or C_@@PPA2 then this indicates that the program object does not contain the class. No action was taken.

Programmer response: Contact your service representative.

System action: Unless the condition is handled the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3EC

CEE3533S The service invoked a system function which was unsuccessful. The system return code was *return_code* and the system reason code was *reason_code*.

Explanation: The CEEPPOS service invoked program management system service for a program object. The system return code and reason code were returned.

Programmer response: Contact your service representative.

System action: Unless the condition is handled the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3ED

CEE3534S The requested function is not supported.

Explanation: The CEEPPOS service was invoked with a function that is not recognized. No action was taken.

Programmer response: Contact your service representative.

System action: Unless the condition is handled the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3EE

CEE3535S The requested class_name is not supported, or a required input value for the specified class_name was not correctly specified.

Explanation: The CEEPPOS service was invoked with class_name that is not recognized, or the required input for the function with class_name CEE_ALL is incorrect. No action was taken.

Programmer response: If your application is involving CEEPPOS, then check that the class_name specified is correct for the function being used. If you are using a function that accepts the class_name CEE_ALL, make sure you specify the class_address and class_size correctly. Otherwise, contact your service representative.

System action: Unless the condition is handled the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3EF

CEE3536S Not enough storage was available for the WSA.

Explanation: The CEEPPOS service was invoked to OBTAIN the WSA and storage was not available to load the WSA into virtual memory. No action was taken.

Programmer response: Contact your service representative.

System action: Unless the condition is handled the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3EG

CEE3537S The request to release the WSA was unsuccessful.

Explanation: The CEEPPOS service was invoked to RELEASE the WSA and the system could not release the WSA because the class_address was not valid.

Programmer response: Contact your service representative.

System action: Unless the condition is handled the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3EH

CEE3538S The request to refresh the WSA was unsuccessful.

Explanation: The CEEPPOS service was invoked to REFRESH the WSA and the system could not refresh the WSA because the class_address was not valid.

Programmer response: Contact your service representative.

System action: Unless the condition is handled the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3EI

CEE3539S The load request for program object *module-name* **was unsuccessful for the current level of CICS.**

Explanation: The load request for module, *module-name* resulted in loading a program object. The load service does not support loading a program object for the current level of CICS.

Programmer response: Rebuild the module using the Language Environment Prelinker Utility and reexecute.

System action: The module is not loaded. Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3EJ

CEE3540S The load request for program object *module-name* **was unsuccessful.**

Explanation: The load request for module, *module-name* resulted in loading a program object. The load service does not support loading a program object.

Programmer response: Rebuild the module using the Language Environment Prelinker Utility and reexecute.

System action: The module is not loaded. Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3EK

CEE3541S A Writeable Static Area (WSA) associated with the entry point was not found.

Explanation: The CEEPFWSA service was invoked and Language Environment was not able to find an executable module containing the specified entry point. A search is made of the executable module containing main (if present), any fetched, dynamically-called, PIPI-loaded modules, CEEFETCHed modules and any loaded DLLs.

Programmer response: Verify that the entry point passed to CEEPFWSA is a valid C/370 or Language Environment style entry point contained within a currently loaded executable module. Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3EL

CEE3542S Unable to find a valid Entry Point or PPA1 or PPA2 for this DSA.

Explanation: DSA pointer, DSA format and Entry Point address are the required parameters for the CEEYEPAF callable service. They have to be passed to the CEEYEPAF CWI. PPA1 and PPA2 pointers are optional parameters. But if the PPA2 pointer is passed to the CEEYEPAF CWI, then the PPA1 pointer has to be passed to the CEEYEPAF CWI as well.

Programmer response: Pass the correct parameters to the CEEYEPAF callable service. PPA1 pointer has to be passed to the CEEYEPAF CWI, if PPA2 pointer is passed to it.

System action: The request was unsuccessful. The address of the Entry Point or PPA1 or PPA2 is not established.

Symbolic Feedback Code: CEE3EM

CEE3543E Requested optional field not found in the passed PPA1.

Explanation: The optional field parameter passed to the CEEYPPAF callable service doesn't exist in the passed PPA1 structure.

Programmer response: Pass the correct optional name parameter to the CEEYPPAF CWI.

System action: The request was unsuccessful. The address of the optional field is evaluated to 0.

Symbolic Feedback Code: CEE3EN

CEE3544E Optional field requested is not valid (1 - 9).

Explanation: The valid value for the optional field parameter passed to the CEEYPPAF callable service is from number 1 to number 9.

Programmer response: Pass the valid number as the optional field parameter to the CEEYPPAF callable service.

System action: The request was unsuccessful. The address of the optional field is evaluated to 0.

Symbolic Feedback Code: CEE3EO

CEE3545E Unable to verify the passed PPA1 as valid for XPLINK.

Explanation: The PPA1 parameter passed to the CEEYPPAF callable service is not a valid XPLINK PPA1.

Programmer response: Pass the XPLINK PPA1 parameter to the CEEYPPAF callable service.

System action: The request was unsuccessful. The address of the optional field is not established.

Symbolic Feedback Code: CEE3EP

CEE3546E An error occurred while attempting to find the previous DSA.

Explanation: A program check occurred in the unwind process.

Programmer response: Make sure the input DSA is valid. Verify the DSA format is correct.

System action: The request was unsuccessful. A feedback is returned. If no feedback is requested and the raised condition is unhandled, the enclave terminates.

Symbolic Feedback Code: CEE3EQ

CEE3547E The DSA physical callee was requested and the physical callee format was not.

Explanation: Callable service CEEYDSAF requires the physical callee format when the physical callee parameter is passed.

Programmer response: Make sure the physical callee format is passed if physical callee is requested.

System action: The request was unsuccessful. A feedback is returned. If no feedback is requested and the raised condition is unhandled, the enclave terminates.

Symbolic Feedback Code: CEE3ER

CEE3548E The callable service was passed a DSA format of -1 and was unable to determine the format of the passed DSA.

Explanation: Unable to determine whether the DSA is up or down format.

Programmer response: Verify the DSA is valid.

System action: The callable service will return feedback. If feedback is not requested, a severity 1 condition will be raised. If the condition remains unhandled, processing continues.

Symbolic Feedback Code: CEE3ES

CEE3549S The service was invoked for a program object that contains some combination XPLINK and NOXPLINK-compiled parts.

Explanation: A Language Environment service found a program object that contains a combination of AMODE 64, XPLINK, and NOXPLINK-compiled parts. You cannot mix any combination of AMODE 64, XPLINK and NOXPLINK-compiled parts in the same program object.

Programmer response: Rebind the program object after recompiling the parts so that they are all compiled either AMODE 64, XPLINK or NOXPLINK. Alternatively, you can split the program object into separate DLLs, one containing AMODE 64 compiled parts, one containing XPLINK-compiled parts and the other containing NOXPLINK compiled parts. Then each of these DLLs would be bound with the DLL side deck of the other. Refer to *z/OS XL C/C++ Programming Guide* for more details on DLLs.

System action: If the Language Environment service that detected the condition was passed a feedback code parameter, then the feedback code representing this message is returned. Otherwise the enclave is terminated.

Symbolic Feedback Code: CEE3ET

CEE3550S DLL *dll-name* does not contain a CEESTART CSECT.

Explanation: The application is attempting to load DLL *dll-name* implicitly or explicitly, but the CEESTART CSECT cannot be located within it.

Programmer response: Make sure that when you generate the DLL, it contains a CEESTART CSECT.

System action: If this was an implicit DLL reference, the condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

| If this was an explicit DLL Load (CEEPLDE) or DLL Open (CEEPOPDL) request, the feedback code is returned to the caller.

| If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3EU

CEE3551S DLL *dll-name* does not contain any C functions.

Explanation: DLL *dll-name* does not contain any C functions.

Programmer response: Make sure that you are loading the correct module, and that the DLL is built correctly.

System action: The condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, the error message is sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3EV

CEE3552S DLL *dll-name* does not export any variables or functions.

Explanation: DLL *dll-name* does not export any variables or functions. Either the definition side-deck supplied to your application is incorrect, or the DLL is generated incorrectly.

Programmer response: Ensure that the DLL was built properly.

1. Specify #pragma export in your source or compile with EXPORTALL compiler option.
2. Compile with DLL, RENT, and LONGNAME compiler options.
3. Ensure that the DLL was built properly.

System action: The condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, the error message is sent to the Language
| Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning
| off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3F0

CEE3553S DLL *dll-name* is part of a circular list.

Explanation: A deadlock condition was discovered while processing a DLL load request for DLL *dll-name.* The deadlock condition exists because the DLLs that are being loaded depend on each other. The following situation illustrates a deadlock condition. DLL A has static constructors that require objects from DLL B. DLL B has static constructors that require objects from DLL A. When DLL A is loaded, its static constructors require objects from DLL B. This forces DLL B to be loaded, requiring objects from DLL A. Since the loading of DLL A has not completed, a deadlock condition exists.

Programmer response: Remove the circular list dependency from the DLLs.

System action: The condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3F1

CEE3554S There is not enough storage to load DLL *dll-name*.

Explanation: There is insufficient storage to satisfy the DLL Load or DLL Open request for DLL *dll-name.*.

Programmer response: Increase the region size.

System action: If this was an implicit DLL reference, the condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

If this was an explicit DLL Load (CEEPLDE) or DLL Open (CEEPOPDL) request, the feedback code is returned to the caller.

| If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the
| error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing
| ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the
| _EDC_DLL_DIAG environment variable.

If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3F2

CEE3555S A call was made from a NOXPLINK-compiled application to an XPLINK-compiled exported function in DLL *dll-name* and the XPLINK(ON) runtime option was not specified.

Explanation: During Language Environment initialization, XPLINK resources need to be allocated if any XPLINK-compiled functions are going to be called during the execution of the application. Language Environment tries to detect this by inspecting the attributes of the initial program. If the initial program consists of NOXPLINK-compiled functions that may at some point call an XPLINK-compiled function, then the XPLINK(ON) runtime option must be used to indicate to Language Environment initialization that XPLINK resources should be allocated.

Programmer response: Specify the XPLINK(ON) runtime option.

System action: If this was an implicit DLL reference, the condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

If this was an explicit DLL Load (CEEPLDE) or DLL Open (CEEPOPDL) request, the feedback code is returned to the caller.

| If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the
| error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing
| ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the
| _EDC_DLL_DIAG environment variable.

If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3F3

CEE3556S An internal error was detected, no WSA could be found associated with entry point *entry-pt*.

Explanation: This is an internal error. Language Environment could not find the WSA associated with the caller of a DLL function (the entry point in the message is the address of the caller).

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3F4

CEE3557S An internal error was detected by Language Environment during DLL load.

Explanation: This is an internal error. Language Environment could not complete DLL load processing because an unexpected condition was encountered in the format of either the DLL or the DLL application.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3F5

CEE3558S DLL *dll-name* does not export any variables.

Explanation: The application made an implicit reference to DLL *dll-name*. During the load of the DLL, it was determined that the application references external variables from the DLL. However, the DLL that was loaded does not contain any exported variables.

Programmer response: Ensure that the DLL indicated in the job library or link library is the correct version, and that it contains the external variable.

System action: The condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, the error message is sent to the Language
| Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning
| off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3F6

CEE3559S External variable *variable-name* was not found in DLL *dll-name*.

Explanation: The application is attempting to refer to external variable, *variable-name*. However, this variable is not defined in DLL, *dll-name*.

Programmer response: Ensure that the DLL indicated in the job library or link library is the correct version, and that it contains the external variable.

System action: The condition is signaled. If the condition is not handled the default action is to terminate the enclave.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, the error message is sent to the Language
| Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning
| off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3F7

CEE3560S DLL *dll-name* does not export any functions.

Explanation: The application made an implicit reference to DLL, *dll-name*. During the load of the DLL it was determined that the application references functions from the DLL. However, the DLL that was loaded does not contain any exported functions.

Programmer response: Ensure that the DLL indicated in the job library or link library is the correct version, and that it contains the external function.

System action: The condition is signaled. If the condition is not handled the default action is to terminate the enclave.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, the error message is sent to the Language
| Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning
| off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3F8

CEE3561S External function *function-name* was not found in DLL *dll-name*.

Explanation: The application is attempting to refer to an external function, *function-name* that is not defined in the DLL, *dll-name*.

Programmer response: Ensure that the DLL indicated in the job library or link library is the correct version, and that it contains the external function.

System action: The condition is signaled. If the condition is not handled the default action is to terminate the enclave.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, the error message is sent to the Language
| Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning
| off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3F9

CEE3562S There is not enough storage to obtain a function pointer for external function *function-name* in DLL *dll-name*.

Explanation: There is insufficient heap storage to satisfy a Query DLL Function request for function *function-name* in DLL *dll-name*.

Programmer response: Increase the region size.

| **System action:** If this was an explicit DLL Query Function (CEEPQDF) request, the feedback code is returned to the
| caller.

If this was a C dllqueryfn() or dlsym() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlsym() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FA

CEE3563S Attempted to load DLL *dll-name* while running C++ destructors.

Explanation: The application is attempting to load DLL *dll-name* while running C++ destructors.

Programmer response: Make sure that you are not referring to DLL variables or functions from your C++ destructors.

System action: If this was an implicit DLL reference, the condition is signaled. If the condition was not handled, the default action is to terminate the enclave.

If this was an explicit DLL Load (CEEPLDE) or DLL Open (CEEPOPDL) request, the feedback code is returned to the caller.

If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and the turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FB

CEE3564S DLL constructors or destructors did not complete, so DLL *dll-name* cannot be used.

Explanation: DLL *dll-name*, which was being loaded or deleted, was in the process of running static constructors or destructors. However, the process did not complete (probably because the thread was abnormally terminated). The DLL is left in an indeterminate state. This error was detected by a thread that was attempting to load or delete the same DLL, and was waiting for the constructors or destructors to complete.

Programmer response: Determine the cause of the incomplete constructor or destructor process. Ensure that the constructors or destructors are not the cause of the thread termination that lead to this condition.

System action: The condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

If this was a C dllload(), dlopen(), dllqueryfn(), dllqueryvar(), dlsym(), dllfree(), or dlclose() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlopen(), dlsym(), or dlclose() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FC

CEE3565I The input dll-token was NULL.

Explanation: The dll-token supplied to the DLL Free request is not valid.

Programmer response: You must request a DLL Load or call the C dlopen() function to initialize a dll-token properly before attempting to delete a DLL.

System action: The request is ignored.

| If this was a C dllfree() or dlclose() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlclose() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FD

CEE3566I There are no DLLs to delete.

Explanation: An attempt was made to delete a DLL, but no DLLs are loaded, or the dll-token passed is inactive.

Programmer response: .Ensure that the DLL Free or C dlclose() request is invoked after the DLL Load or C dlopen() request, respectively, has completed successfully, and that you have no extra DLL Free or C dlclose() requests using this dll-token in your application.

System action: The request is ignored.

| If this was a C dllfree() or dlclose() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlclose() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FE

CEE3567I A logical delete was performed for DLL *dll-name* , but the DLL was not physically deleted.

| **Explanation:** The DLL Free or C dlclose() request completed successfully. DLL *dll-name* is not physically deleted because either there was an implicit DLL Load performed against this DLL by the application, or multiple DLL Load or C dlopen() requests were made for the DLL.

Programmer response: If the DLL was loaded implicitly by referring to an external variable or an external function, it will be physically deleted by Language Environment at enclave termination. Otherwise, to free or close the DLL, issue a DLL Free or C dlclose() request using the proper dll-token.

System action: Execution continues.

| If this was a C dllfree() or dlclose() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlclose() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FF

CEE3568I No DLL could be found which matched the input dll-token.

Explanation: The dll-token supplied to the DLL Free or C dlclose() request could not be matched to a DLL loaded by this application.

Programmer response: Ensure that the dll-token supplied to the DLL Free or C dlclose() request is the same as the one returned from the DLL Load or C dlopen() request, respectively, and that it has not been overwritten.

System action: The request is ignored.

If this was a C dllfree() or dlclose() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlclose() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FG

CEE3569E The DLL function was not allowed because destructors are running for the DLL.

Explanation: A DLL Free, Query Variable, Query Function, C dlclose(), or C dlsym() request was made for a DLL that is currently running destructors. Since destructors are running, the DLL is about to be freed. Further function requests using this DLL are not allowed.

Programmer response: Do not issue DLL function requests from one thread while the DLL is being freed from another thread.

System action: If this was an explicit DLL Free (CEEPFDE), Query Variable (CEEPQDV), Query Function (CEEPQDF), Close (CEEPCLDL), or Symbol (CEEPSYDL) request, the feedback code was returned to the caller.

If this was a C dllqueryfn(), dllqueryvar(), dlsym(), dllfree(), or dlclose() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlclose() or dlsym() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FH

CEE3570S DLL name *dll-name* was not valid.

Explanation: Either the DLL name provided as input was null, or the length of the DLL name was negative.

Programmer response: If this was an implicit DLL reference, make sure that the DLL was built correctly. If this was an explicit DLL Load or C dlopen() request, verify that the DLL name was specified correctly.

System action: If this was an implicit DLL reference, the condition is signaled. If the condition is not handled, the default action is to terminate the enclave. If this was an explicit DLL Load (CEEPLDE) or DLL Open (CEEPOPDL) request, the feedback code is returned to the caller.

If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FI

CEE3571S Storage for writeable static was not available for DLL *dll-name.*

Explanation: Not enough storage was available for allocation of writeable static for DLL *dll-name*.

Programmer response: Ensure that the REGION size is sufficient to run the application. Verify that the storage sizes specified in the HEAP and STACK run-time options are reasonable, given the region size allocated to the application. Verify that you are using storage options that get your storage from above the line, if you can, since you can run out of storage below the line much more easily.

System action: The request is ignored. The load module is deleted from storage.

| If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FJ

CEE3572I The input dll-token was not available for use.

Explanation: The dll-token supplied to a DLL Query Function, DLL Query Variable, or C dlsym() request could not be used because one of the following was true:
- the dll-token was null
- the dll-token was not valid
- the dll-token had been marked inactive as a result of an explicit DLL Free or C dlclose() request.

Programmer response: Ensure that the proper dll-token is supplied to the DLL request, and that the subject DLL is not freed prematurely.

System action: The request is ignored.

| If this was a C dllqueryfn(), dllqueryvar(), or dlsym() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlsym() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FK

CEE3573I Dll *dll-name* does not export any functions.

Explanation: An attempt was made to query an external function, but DLL *dll-name* does not contain any exported functions.

Programmer response: Ensure that the DLL indicated in the job library or link library is the correct version, and that it contains the external function.

System action: The request is ignored.

| If this was a C dllqueryfn() or dlsym() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlsym() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FL

CEE3574I External function *function-name* was not found in DLL *dll-name.*

Explanation: An attempt was made to query an external function, but function *function-name* was not found in the export section of the DLL *dll-name*.

Programmer response: Ensure that the function name specified on the DLL Query Function request is correct, that the DLL indicated in the job library or link library is the correct version, and that it contains the external function.

System action: The request is ignored.

If this was a C dllqueryfn() or dlsym() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlsym() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FM

CEE3575I DLL *dll-name* does not export any variables.

Explanation: An attempt was made to query an external variable, but DLL *dll-name* does not contain any exported variables.

Programmer response: Ensure that the DLL indicated in the job library or link library is the correct version, and that it contains the external variable.

System action: The request is ignored.

If this was a C dllqueryvar() or dlsym() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlsym() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FN

CEE3576I External variable *variable-name* was not found in DLL *dll-name.*

Explanation: An attempt was made to query an external variable, but *variable-name* was not found in the export section of DLL *dll-name*.

Programmer response: Ensure that the variable name specified on the DLL Query Variable request is correct, that the DLL indicated in the job library or link library is the correct version, and that it contains the external variable.

System action: The request is ignored.

If this was a C dllqueryvar() or dlsym() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlsym() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FO

CEE3577I The external function was not found in DLL *dll-name.*

Explanation: An attempt was made to query an external function in DLL *dll-name*, but either the function name was null, or the length of the function name was negative.

Programmer response: Ensure that the function name and length are specified correctly on the DLL Query Function request.

System action: The request is ignored.

If this was a C dllqueryfn() or dlsym() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlsym() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FP

CEE3578I The external variable was not found in DLL *dll-name.*

Explanation: An attempt was made to query an external variable in DLL *dll-name*, but either the variable name was null, or the length of the variable name was negative.

Programmer response: Ensure that the variable name and length are specified correctly on the DLL Query Variable request.

System action: The request is ignored.

If this was a C dllqueryvar() or dlsym() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlsym() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FQ

CEE3579S Attempted to delete DLL *dll-name* **while running C++ destructors.**

Explanation: The application is attempting to delete DLL *dll-name* while running C++ destructors.

Programmer response: Make sure that you are not deleting this DLL from your C++ destructors.

System action: The feedback code is returned to the caller.

If this was an explicit DLL Free (CEEPFDE) or DLL Close (CEEPCLDL) request, the feedback code is returned to the caller.

If this was a C dllfree() or dlclose() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

If this was an explicit C dlclose() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FR

CEE3580S External variable *variable-name* **was not found in DLL** *dll-name***.**

Explanation: An attempt was made to reference an external variable, but *variable-name* is not supported as an exported variable from DLL *dll-name*.

Programmer response: Verify that the side deck representing the DLL is correct. Since the DLL does not export this variable, it should not be present in the side deck. Contact the supplier of the DLL.

System action: The condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

If the _EDC_DLL_DIAG environment variable is not set to QUIET, the error message is sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FS

CEE3581S An internal error was detected by Language Environment during the load of DLL *dll-name*.

Explanation: This is an internal error. Language Environment could not complete DLL load processing because an unexpected condition was encountered in the format of either the DLL or the DLL application.

Programmer response: Contact your service representative.

System action: Unless the condition is handled, the default action is to terminate the enclave.

| If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the
| error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing
| ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the
| _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to
| the C dlerror() function.

| If this was an implicit DLL load, or an explicit DLL Load or Open, a CEEDLLF DLL failure control block was populated
| with further DLL error diagnostics.

Symbolic Feedback Code: CEE3FT

CEE3582S An attempt was made to load a new module containing XPLINK-compiled functions and the XPLINK(ON) runtime option was not specified.

Explanation: During Language Environment initialization, XPLINK resources need to be allocated if any XPLINK-compiled functions are going to be called during the execution of the application. Language Environment tries to detect this by inspecting the attributes of the initial program. If the initial program consists of NOXPLINK-compiled functions that may at some point call an XPLINK-compiled function, then the XPLINK(ON) runtime option must be used to indicate to Language Environment initialization that XPLINK resources should be allocated.

Programmer response: Specify the XPLINK(ON) runtime option.

System action: If no feedback code was provided, the condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

If a feedback code was provided, the feedback code is returned to the caller.

Symbolic Feedback Code: CEE3FU

CEE3583S The transition from standard Language Environment linkage conventions to XPLINK linkage conventions could not be performed. The transition routine CEEVROND could not locate the information required to perform the transition or the information was not valid.

Explanation: CEEVROND gains control before the routine actually being called. It uses information from the PPA1 of the called routine to perform parameter list and return value mapping between the two linkage conventions. On input, it expects register 0 to be the address of a function descriptor for a routine with an Language Environment conforming XPLINK entry point.

Programmer response: Depending on the TERMTHDACT run-time option, a CEEDUMP may be available for additional diagnosis. Use the traceback information in the CEEDUMP to determine the two routines involved.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3FV

CEE3584E The transition from standard Language Environment linkage conventions to XPLINK linkage conventions could not be performed. The transition routine CEEVROND could not determine the length of the parameter list being passed to the called routine.

Explanation: CEEVROND gains control before the routine actually being called. It uses information from the PPA1 of the called routine to perform parameter list and return value mapping between the two linkage conventions. The PPA1 indicated that the called routine expects a variable length parameter list. The parameter list was not located within a stack frame where its length could not be approximated.

Programmer response: Depending on the TERMTHDACT run-time option, a CEEDUMP may be available for additional diagnosis. Use the traceback information in the CEEDUMP to determine the two routines involved.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3G0

CEE3585E The transition from XPLINK linkage conventions to Language Environment standard linkage conventions could not be performed. The transition routine CEEVRONU could not locate the information required to perform the transition or the information was not valid.

Explanation: CEEVRONU gains control before the routine actually being called. It uses information from the PPA1 of the called routine and from the return address of the calling routine to perform parameter list and return value mapping between the two linkage conventions. On input, it expects register 5 to be the address of a function descriptor for or the entry point of a NOXPLINK-compiled routine. This routine must have an Language Environment conforming entry point.

Programmer response: Depending on the TERMTHDACT run-time option, a CEEDUMP may be available for additional diagnosis. Use the traceback information in the CEEDUMP to determine the two routines involved.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3G1

CEE3586S An attempt was made to resolve DLL references from a NORENT compiled program that was loaded into read only storage.

Explanation: When a program makes an implicit reference to a DLL, that reference is resolved at run-time by Language Environment. When the program is non-reentrant or naturally-reentrant (i.e. NORENT compiled), it doesn't have a writeable static area (WSA) so the reference resides within the executable program itself. This requires that the program be loaded into read-write storage, which is normally the case. However, if the program resides in LPA, or was link-edited with the RENT option, then it may get loaded into read-only storage.

Note that NOXPLINK compiled programs that call C RTL functions have those calls resolved statically via stubs in the SCEELKED data set, and therefore programs that call just C RTL (and other non-DLL) functions and are compiled NOXPLINK would never see this message.

XPLINK compiled programs have their C RTL references resolved dynamically through the C RTL side deck (CELHS003 in SCEELIB), and as such XPLINK compiled programs call the C RTL using DLL call mechanisms.

Programmer response: The traceback in the CEEDUMP will show the load of a program in process. Ensure that this program resides in read-write storage. This can be done by link-editing the program NORENT, or STEPLIBing to it in an unauthorized data set. Do not put the program into LPA.

Optionally, the program can be compiled and link-edited with the RENT option to provide constructed reentrancy, and all writeable references will reside in WSA.

System action: Unless the condition is handled, the default action is to terminate the enclave.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, the error message is sent to the Language
| Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning
| off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3G2

CEE3587S A call was made to a function in the AMODE 31 DLL *dll-name* from an AMODE 64 caller.

Explanation: There is no support for mixing 31-bit Addressing Mode (AMODE 31) and 64-bit Addressing Mode (AMODE 64) applications across DLL calls.

Programmer response: Make sure that the AMODE of the calling function matches the AMODE of the DLL. You may need to verify with the DLL provider that they support running in the AMODE you are calling from, and that you are using the correct DLL.

If the DLL provider does not provide this DLL built AMODE 64, then you will not be able to use it unless you rebuild your application as AMODE 31.

If the DLL provider does support this DLL in both AMODE 31 and AMODE 64, they would ship two copies of the same DLL which are built with different AMODEs. They would ship them using one of the following techniques:

- Ship two DLLs with different names. In this case you will need to make sure your application references the correct DLL name for the AMODE in which you're running, either explicitly on the dllload() or dlopen() call, or implicitly by choosing the correct side deck at linkedit time.
- Ship two DLLs with the same name but installed in different paths or data sets. In this case you will need to specify the correct DLL location at run-time using either the LIBPATH environment variable or STEPLIB, respectively.

| **System action:** If this was an implicit DLL reference, the condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

| If this was an explicit DLL Load (CEEPLDE) or DLL Open (CEEPOPDL) request, the feedback code is returned to the caller.

| If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3G3

CEE3588S A call was made to a function in the AMODE 64 DLL *dll-name* from an AMODE 31 caller.

Explanation: There is no support for mixing 31-bit Addressing Mode (AMODE 31) and 64-bit Addressing Mode (AMODE 64) applications across DLL calls.

Programmer response: Make sure that the AMODE of the calling function matches the AMODE of the DLL. You may need to verify with the DLL provider that they support running in the AMODE you are calling from, and that you are using the correct DLL.

If the DLL provider does not provide this DLL built AMODE 31, then you will not be able to use it unless you rebuild your application as AMODE 64.

If the DLL provider does support this DLL in both AMODE 31 and AMODE 64, they would ship two copies of the same DLL which are built with different AMODEs. They would ship them using one of the following techniques:

- Ship two DLLs with different names. In this case you will need to make sure your application references the correct DLL name for the AMODE in which you're running, either explicitly on the dllload() or dlopen() call, or implicitly by choosing the correct side deck at linkedit time.
- Ship two DLLs with the same name but installed in different paths or data sets. In this case you will need to specify the correct DLL location at run-time using either the LIBPATH environment variable or STEPLIB, respectively.

| **System action:** If this was an implicit DLL reference, the condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

| If this was an explicit DLL Load (CEEPLDE) or DLL Open (CEEPOPDL) request, the feedback code is returned to the caller.

| If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3G4

CEE3589S The service was invoked for a program object that contains WSA located above the 2GB bar and below the bar.

Explanation: A Language Environmentservice found a program object that contains a WSA located above and below the 2GB bar. You cannot mix any WSA above and below the bar parts in the same program object.

Programmer response: Rebind the program object after recompiling the parts so that they are all compiled either above the bar or below the bar.

System action: If the Language Environment service that detected the condition was passed a feedback code parameter, then the feedback code representing this message is returned. Otherwise the enclave is terminated.

Symbolic Feedback Code: CEE3G5

CEE3590I Input dll-token not permitted for this DLL function.

Explanation: The DLL handle supplied to this DLL function call could not be matched to a DLL handle returned from a previous C dlopen() request.

Programmer response: Ensure that the DLL handle supplied to a C dlclose() or dlsym() request is the same as one returned from a dlopen() request. You cannot share a DLL handle between the older explicit DLL services (dllload, dllqueryvar, dllqueryfn, dllfree) and the newer services (dlopen, dlsym, dlclose).

| **System action:** If this was a C dlclose() or dlsym() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was a C dlclose() or dlsym() request, this error could also be returned to the caller through a subsequent C dlerror() request.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3G6

CEE3591I DLL *dll-name* does not export any symbols.

Explanation: An attempt was made to locate an external DLL symbol, but DLL *dll-name* does not contain any exported symbols.

Programmer response: The request is ignored. Ensure that the DLL indicated in the job library or link library is the correct version, and that it contains the external symbol.

| **System action:** If this was a C dlsym() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the _EDC_DLL_DIAG environment variable.

| If this was a C dlsym() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3G7

CEE3592I The external symbol was not found in DLL *dll-name*.

Explanation: An attempt was made to locate an external symbol in DLL *dll-name*, but either the symbol name was null, or the length of the symbol name was negative.

Programmer response: The request is ignored. Ensure that the symbol name and length are specified correctly on the C dlsym() request.

| **System action:** If the _EDC_DLL_DIAG environment variable is not set to QUIET, the error message is sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| This error can also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3G8

CEE3593I External symbol *symbol-name* **was not found in DLL** *dll-name*

Explanation: An attempt was made to locate an external symbol, but *symbol-name* was not found in the export section of DLL *dll-name*, or any of its dependent DLLs that were also loaded at the time of the C dlopen() request.

Programmer response: The request is ignored. Ensure that the symbol name specified on the C dlsym() request is correct, that the DLL indicated in the job library or link library is the correct version, and that it contains the external variable. If the symbol is expected to be found in one of the dependent DLLs, then you may need to make the C dlopen() request with the RTLD_NOW flag to ensure all dependent DLLs are preloaded and available for subsequent C dlsym() requests.

| **System action:** If this was a C dlsym() request and the _EDC_DLL_DIAG environment variable was not set to
| QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such
| as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the
| _EDC_DLL_DIAG environment variable.

| If this was a C dlsym() request, this error could also be returned to the caller through a subsequent call to the C
| dlerror() function.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3G9

CEE3594I External symbol *symbol-name* **was not found as a result of a search of the global symbol object.**

Explanation: An attempt was made to locate an external symbol in the global symbol object, but *symbol-name* was not found in the export section of any of the ″non-LOCAL″ DLLs currently loaded.

Programmer response: The request is ignored. Ensure that the symbol name specified on the C dlsym() request is correct, and that the DLL that is expected to contain the symbol was opened at least once with the RTLD_GLOBAL flag.

| **System action:** If this was a C dlsym() request and the _EDC_DLL_DIAG environment variable was not set to
| QUIET, the error message was sent to the Language Environment message file. Other DLL diagnostic options, such
| as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the
| _EDC_DLL_DIAG environment variable.

| If this was a C dlsym() request, this error could also be returned to the caller through a subsequent call to the C
| dlerror() function.

| A CEEDLLF DLL failure control block was populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3GA

CEE3595S DLL *dll-name* **does not contain a CELQSTRT CSECT.**

Explanation: The application is attempting to load DLL *dll-name* implicitly or explicitly, but the CELQSTRT CSECT cannot be located within it.

Programmer response: Make sure that when you generate the DLL, it contains a CELQSTRT CSECT.

System action: If this was an implicit DLL reference, the condition is signaled. If the condition is not handled, the default action is to terminate the enclave.

If this was an explicit DLL Load (dllload()) or DLL Open (dlopen()) request, the feedback code is returned to the caller.

| If this was a C dllload() or dlopen() request and the _EDC_DLL_DIAG environment variable was not set to QUIET, the
| error message was sent to the Language Environment message file. Other DLL diagnostic options, such as issuing
| ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, were available through the
| _EDC_DLL_DIAG environment variable.

If this was an explicit C dlopen() request, this error could also be returned to the caller through a subsequent call to the C dlerror() function.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: CEE3GB

CEE3601I The string '*string*' was found where a delimiter was expected following a quoted suboption for the run-time option *option*.

Explanation: A quoted suboption must be followed by either a comma, right parenthesis, or space.

Programmer response: Correct the run-time options string.

System action: The characters following *suboption* up to the next comma, space, or parenthesis are ignored.

Symbolic Feedback Code: CEE3GH

CEE3602I An end quote delimiter did not occur before the end of the run-time option string.

Explanation: Quotes, either single or double, must be in pairs.

Programmer response: Correct the run-time options string.

System action: The end quote is assumed.

Symbolic Feedback Code: CEE3GI

CEE3603I The character '*character*' is not a valid run-time option delimiter.

Explanation: Options must be separated by either a space or a comma.

Programmer response: Correct the run-time options string.

System action: *character* is ignored.

Symbolic Feedback Code: CEE3GJ

CEE3604I The character '*character*' is not a valid suboption delimiter for run-time options.

Explanation: Suboptions must be separated by a comma.

Programmer response: Correct the run-time options string.

System action: The separator is assumed to be a comma.

Symbolic Feedback Code: CEE3GK

CEE3605I The string '*string*' was found where a delimiter was expected following the suboptions for the run-time option *option*.

Explanation: Suboptions that are enclosed within parentheses must be followed by either a space or a comma.

Programmer response: Correct the run-time options string.

System action: The characters following the right parenthesis up to the next comma or space are ignored.

Symbolic Feedback Code: CEE3GL

CEE3606I The string '*string*' was too long and was ignored.

Explanation: The maximum string length for an option or suboption was exceeded.

Programmer response: Correct the run-time options string.

System action: *string* is ignored.

Symbolic Feedback Code: CEE3GM

CEE3607I The end of the suboption string did not contain a right parenthesis.

Explanation: A left parenthesis did not have a matching right parenthesis.

Programmer response: Correct the run-time options string.

System action: The right parenthesis is assumed.

Symbolic Feedback Code: CEE3GN

CEE3608I The following messages pertain to the invocation command run-time options.

Explanation: The messages after this one up to the next message of this type with a different source, pertain to the invocation command.

Programmer response: No programmer response is required.

System action: No system action is performed.

Symbolic Feedback Code: CEE3GO

CEE3609I The run-time option *option* **is not supported.**

Explanation: *option* was an option from a previous release that was not supported or mapped by Language Environment.

Programmer response: Consult the appropriate migration guide for a list of options supported for the language. Correct the run-time options string.

System action: *option* is ignored.

Symbolic Feedback Code: CEE3GP

CEE3610I The run-time option *old-option* **was mapped to the run-time option** *le-option***.**

Explanation: *old-option* was an option from a previous release that was supported by Language Environment for compatibility.

Programmer response: Consult the appropriate migration guide for a list of options supported for the language. Change the run-time options string to use the *le-option* instead.

System action: *old-option* is mapped to *le-option*.

Symbolic Feedback Code: CEE3GQ

CEE3611I The run-time option *option* **was an invalid run-time option or is not supported in this release of Language Environment.**

Explanation: *option* was not a valid option. Either the option is not valid for this release of Language Environment, or was recognized by Language Environment for previous release language campatibility.

Programmer response: Consult *z/OS Language Environment Programming Reference* or the appropriate migration guide for a list of options supported on the release of Language Environment being used. Change the run-time options string, or execute this application on a level of Language Environment that supports this option.

System action: The option *option* is ignored.

Symbolic Feedback Code: CEE3GR

CEE3612I Too many suboptions were specified for the run-time option *option***.**

Explanation: The number of suboptions specified for *option* exceeded that defined for the option.

Programmer response: A list of valid run-time options is provided in *z/OS Language Environment Programming Reference*. Correct the run-time options string.

System action: The extra suboptions are ignored.

Symbolic Feedback Code: CEE3GS

CEE3613I The run-time option *old-option* appeared in the options string.

Explanation: *old-option* (SPIE, NOSPIE, STAE, or NOSTAE) was an option from a previous release that was supported by Language Environment for compatibility, but ignored if TRAP was specified. See *z/OS Language Environment Programming Reference* for the interactions between TRAP and SPIE, NOSPIE, STAE, or NOSTAE.

Programmer response: Change the run-time options string to use the TRAP option instead of SPIE, NOSPIE, STAE, or NOSTAE.

System action: *old-option* is ignored if TRAP is specified, otherwise it is mapped to TRAP.

Symbolic Feedback Code: CEE3GT

CEE3614I An invalid character occurred in the numeric string '*string*' of the run-time option *option*.

Explanation: *string* did not contain all decimal numeric characters.

Programmer response: Correct the run-time options string to contain all numeric characters.

System action: The *string* is ignored.

Symbolic Feedback Code: CEE3GU

CEE3615I The installation default for the run-time option *option* could not be overridden.

Explanation: *option* was defined as non-overridable at installation time.

Programmer response: Correct the run-time options to not specify this *option*.

System action: The option is ignored.

Symbolic Feedback Code: CEE3GV

CEE3616I The string '*string*' was not a valid or supported suboption of the run-time option *option* in this release.

Explanation: *string* was not in the set of recognized values or not supported in this release of Language Environment.

Programmer response: Consult *z/OS Language Environment Programming Reference* or the appropriate migration guide for a list of suboptions for option *option* supported on the release of Language Environment being used. Change the invalid suboption *string* for the run-time option *option* or execute this application on a level of Language Environment that supports this suboption. Remove the invalid suboption *string* from the run-time.

System action: The suboption is ignored.

Symbolic Feedback Code: CEE3H0

CEE3617I The number *number* of the run-time option *option* exceeded the range of -2147483648 to 2147483647.

Explanation: *number* exceeded the range of -2147483648 to 2147483647.

Programmer response: Correct the run-time options string to be within the acceptable range of -2147483647 to 2147483647.

System action: The *number* is ignored.

Symbolic Feedback Code: CEE3H1

CEE3618I The run-time option *option* was not valid from the invocation command.

Explanation: *option* was not valid from the invocation command.

Programmer response: Remove the *option* run-time option from the invocation command.

System action: *option* is ignored.

Symbolic Feedback Code: CEE3H2

CEE3619I The value *value* was not a valid MSGQ number.

Explanation: *value* must be greater than zero.

Programmer response: Correct the value in the run-time options string to be greater than zero.

System action: *value* is ignored.

Symbolic Feedback Code: CEE3H3

CEE3620I The following messages pertain to the assembler user exit run-time options.

Explanation: The messages after this one up to the next message of this type with a different source pertain to the assembler user exit.

Programmer response: No response is required.

System action: No system action is performed.

Symbolic Feedback Code: CEE3H4

CEE3621I The run-time option *option* was not valid from the assembler user exit.

Explanation: *option* was not valid from the assembler user exit.

Programmer response: Remove the *option* from the run-time options specified in the assembler user exit.

System action: *option* is ignored.

Symbolic Feedback Code: CEE3H5

CEE3622I The STORAGE option quoted suboption string '*string*' was not one character long.

Explanation: The only acceptable length for STORAGE suboptions within quotes is one.

Programmer response: Correct the STORAGE run-time option quoted suboption string to be one character long.

System action: The suboption is ignored.

Symbolic Feedback Code: CEE3H6

CEE3623I The UPSI option suboption string '*string*' was not eight characters long.

Explanation: The only acceptable length for the UPSI suboption is eight.

Programmer response: Correct the UPSI run-time option suboption string to be eight characters long.

System action: The suboption is ignored.

Symbolic Feedback Code: CEE3H7

CEE3624I One or more error messages pertaining to the run-time options included in the invocation command were lost.

Explanation: The run-time options error table (ROET) overflowed.

Programmer response: Correct the reported errors so the discarded errors fit into the error table.

System action: The errors that are detected after the table overflowed are discarded.

Symbolic Feedback Code: CEE3H8

CEE3625I One or more error messages pertaining to the run-time options returned by the assembler user exit were lost.

Explanation: The run-time options error table (ROET) overflowed.

Programmer response: Correct the reported errors so the discarded errors fit into the error table.

System action: The errors that are detected after the table overflowed are discarded.

Symbolic Feedback Code: CEE3H9

CEE3626I One or more error messages pertaining to the run-time options contained within the programmer defaults were lost.

Explanation: The run-time options error table (ROET) overflowed.

Programmer response: Correct the reported errors so the discarded errors fit into the error table.

System action: The errors that are detected after the table overflowed are discarded.

Symbolic Feedback Code: CEE3HA

CEE3627I The following messages pertain to the programmer default run-time options.

Explanation: The messages after this one up to the next message of this type with a different source, pertain to the programmer default options.

Programmer response: No response is required.

System action: No system action is performed.

Symbolic Feedback Code: CEE3HB

CEE3628I The run-time option *option* **was not valid from the programmer defaults.**

Explanation: *option* was not valid from the programmer defaults.

Programmer response: Correct the run-time options string by removing the *option* run-time option.

System action: The option is ignored.

Symbolic Feedback Code: CEE3HC

CEE3629I The run-time option *old-option* **was partially mapped to the run-time option** *le-option***.**

Explanation: *old-option* was an old language option that was being supported by Language Environment for compatibility. The user should use the Language Environment option *le-option* instead.

Programmer response: Change the run-time options string to use the Language Environment option instead.

System action: *old-option* is partially mapped to its Language Environment equivalent.

Symbolic Feedback Code: CEE3HD

CEE3630I One or more settings of the run-time options STAE or SPIE were ignored.

Explanation: STAE, SPIE, NOSTAE, and NOSPIE (options from a previous release) were ignored when the TRAP option was specified. See *z/OS Language Environment Programming Reference* for the interactions between TRAP and SPIE, NOSPIE, STAE, or NOSTAE.

Programmer response: Remove the STAE, SPIE, NOSTAE, or NOSPIE run-time options if the TRAP run-time option is specified.

System action: STAE, SPIE, NOSTAE, or NOSPIE are ignored.

Symbolic Feedback Code: CEE3HE

CEE3631I One or more settings of the run-time options STAE or SPIE were mapped to TRAP.

Explanation: STAE, SPIE, NOSTAE, and NOSPIE are options from a previous release that are supported by Language Environment for compatibility. See *z/OS Language Environment Programming Reference* for the interactions between TRAP and SPIE, NOSPIE, STAE, or NOSTAE.

Programmer response: Change the run-time options string to use the TRAP option instead of SPIE, STAE, NOSPIE, or NOSTAE.

System action: STAE, SPIE, NOSTAE, or NOSPIE are mapped to TRAP.

Symbolic Feedback Code: CEE3HF

CEE3632I POSIX(ON) run-time option specified and the UNIX System Services feature is not available on the underlying operating system.

Explanation: The POSIX(ON) option was specified but the UNIX System Services feature was not available on the underlying operating system.

Programmer response: Check with your system programmer to ensure that UNIX System Services feature is available. Remove the POSIX(ON) run-time option if the UNIX System Services feature is not available.

System action: POSIX(ON) is ignored.

Symbolic Feedback Code: CEE3HG

CEE3633W The total length of the combined ENVAR strings exceeded 250 characters.

Explanation: The total length of the combined ENVAR strings exceeded the maximum limit of 250 characters.

Programmer response: Reduce the total length of the ENVAR strings to less than the 250 character maximum.

System action: The ENVAR string is ignored.

Symbolic Feedback Code: CEE3HH

CEE3634I The number *number* of the run-time option *option* exceeded the range of -32768 to 32767.

Explanation: *number* exceeded the range of -32768 to 32767.

Programmer response: Correct the run-time options string to be within the range of -32768 to 32767.

System action: The *number* is ignored.

Symbolic Feedback Code: CEE3HI

CEE3635I The string *string* was not a valid RECFM suboption specification for run-time option MSGFILE.

Explanation: *string* for RECFM suboption must be one of the following: F, FA, FB, FBA, FBS, FBSA, U, UA, V, VA, VB, or VBA.

Programmer response: Specify a valid RECFM suboption string of F, FA, FB, FBA, FBS, FBSA, U, UA, V, VA, VB, or VBA.

System action: *string* is ignored.

Symbolic Feedback Code: CEE3HJ

CEE3636I The value *value* exceeded the maximum allowable LRECL or BLKSIZE of 32760 bytes.

Explanation: *value* cannot be greater than 32760.

Programmer response: Correct the LRECL or BLKSIZE suboption value to be less than or equal to 32760 bytes.

System action: *value* is ignored.

Symbolic Feedback Code: CEE3HK

CEE3637I The number *number* specified in the *suboption* suboption of the run-time option *option* is not a valid hexadecimal number in the range 0 to FFFFFFFF.

Explanation: An invalid hexadecimal numeral was specified or the range of the *number* exceeds 0 to FFFFFFFF.

Programmer response: Correct the run-time options string to be a valid hexadecimal number in the range of 0 to FFFFFFFF.

System action: The *number* is ignored.

Symbolic Feedback Code: CEE3HL

CEE3638I The table size of *size*, specified in the TRACE run-time option, exceeds the maximum allowed value of 16777215.

Explanation: *size* exceeded the maximum allowed value of 16777215.

Programmer response: Correct the TRACE run-time options to not exceed the maximum of 16777215.

System action: The *size* is ignored.

Symbolic Feedback Code: CEE3HM

CEE3639I The ID suboption *suboption* of the TRACE run-time option must consist of the keyword 'ID' followed by a one or two digit number in the range 0 to *number*

Explanation: The format of the ID suboption of the TRACE run-time option is *IDxx=nnnnnnnn* where *xx* is a one or two digit decimal number with no blanks between it and either the ID or the equal-sign.

Programmer response: Correct the ID suboption of the TRACE run-time option to be the correct format where *xx* is a one or two digit decimal number with no blanks between it and either the ID or the equal-sign.

System action: The *suboption* is ignored.

Symbolic Feedback Code: CEE3HN

CEE3640W Multithreading function is being used in your application but the UNIX System Services feature is not available on the underlying operating system.

Explanation: Multithreading function is being used in your application and that function is not supported. The underlying operating system must have the UNIX System Services feature installed and active when you run this application.

Programmer response: If your application uses multithreading ensure that the underlying operating system has the UNIX System Services option installed and that it is active when your application is running.

System action: Execution continues.

Symbolic Feedback Code: CEE3HO

CEE3641I The *number* of the run-time option *option* exceeded the range of 0 to 2147483647.

Explanation: *number* exceeded the range of 0 to 2147483647.

Programmer response: Correct the run-time option string to be within the range of 0 to 2147483647.

System action: The *number* is ignored.

Symbolic Feedback Code: CEE3HP

CEE3642I The cell pool size *number* of the run-time option *option* is not valid.

Explanation: *number* is either not a multiple of 8 or not in the range from 8 to 2048.

Programmer response: Correct the run-time options string so that the *number* is in the range from 8 to 2048.

System action: The *number* is ignored.

Symbolic Feedback Code: CEE3HQ

CEE3643I The cell pool percentage *number* of the run-time option *option* exceeded the range of 1 to 90.

Explanation: *number* exceeded the range of 1 to 90.

Programmer response: Correct the run-time options string so that the *number* is in the range from 1 to 90.

System action: The *number* is ignored.

Symbolic Feedback Code: CEE3HR

CEE3644I TEST option negates PROFILE option setting.

Explanation: The TEST and PROFILE run-time options cannot be active at the same time. If TEST and PROFILE ON are specified together, Language Environment will not load the profiler tool.

Programmer response: To specify a PROFILE option, ensure the NOTEST run-time option is specified for your application or as your system default. The NOTEST option should be first when specifying NOTEST and PROFILE together via a compiler #pragma runopts directive or on application invocation.

System action: The PROFILE option is ignored.

Symbolic Feedback Code: CEE3HS

CEE3645I Profiler not loaded; module *profiler-name* not accessible.

Explanation: The PROFILE ON run-time option has been specified but Language Environment has not loaded the profiler tool.

Programmer response: Ensure that the profile module CEEEVPRF exists and is accessible to Language Environment. When calling the profiler application, Language Environment must be able to locate and access the CEEEVPRF module.

System action: The PROFILE option is ignored.

Symbolic Feedback Code: CEE3HT

CEE3646I The following messages pertain to the region default run-time options.

Explanation: The messages after this one, and up to the next message of this type with a different source, pertain to the region default options.

Programmer response: No response is required.

System action: No system action is performed.

Symbolic Feedback Code: CEE3HU

CEE3647I The region default for the run-time option *option* could not be overridden.

Explanation: *option* was defined as non-overridable at region initialization time.

Programmer response: Correct the run-time options to not specify this *option*.

System action: The option is ignored.

Symbolic Feedback Code: CEE3HV

CEE3648S POSIX(ON) run-time option in a nested enclave *enclave-name* is not supported.

Explanation: In Language Environment, a process can have only one enclave that is running with POSIX(ON), and that enclave must be the first enclave. All nested enclaves must be running with POSIX(OFF).

Programmer response: Specify the POSIX(ON) run-time option for only the first enclave. Make sure all nested enclaves specify POSIX(OFF).

System action: The application will be terminated.

Symbolic Feedback Code: CEE3I0

CEE3649W The parameter string returned from CEE3PRM exceeded the maximum length of 80 bytes and was truncated.

Explanation: The user parameters exceed 80 characters. The first 80 bytes are returned and the remainder of the user parameters are truncated.

Programmer response: Reduce the user parameter string to less than 80 bytes.

System action: The user parameter string is truncated to 80 bytes. The truncated value is returned to the caller in the character string parameter.

Symbolic Feedback Code: CEE3I1

CEE3700I The storage and options report heading replaced a previous heading.

Explanation: The specified report heading has replaced a heading set by an earlier call to CEERPTH.

Programmer response: None required.

System action: The report heading is replaced by the new heading.

Symbolic Feedback Code: CEE3JK

CEE3701W Heap damage found by HEAPCHK run-time option.

Explanation: This is a title message for the heap check section of the message file.

Programmer response: View the messages following this one to see where damage was found.

System action: No system action is performed.

Symbolic Feedback Code: CEE3JL

CEE3702S Program terminating due to heap damage.

Explanation: This is the last message in the heap check section of the message file.

Programmer response: The message file will contain the address, expected and actual data for each damaged area found.

System action: The application is terminated.

Symbolic Feedback Code: CEE3JM

CEE3703I In *controlblock* Control Block, the *fieldname* is damaged.

Explanation: A Language Environment control block *controlblock* has damage in the *fieldname* area.

Programmer response: The message following this one in the message file will identify the address of the damage and the expected data. If you did not use the HEAPCHK run-time option, re-run the application with HEAPCHK(ON) to help locate the cause of the problem. If you used the HEAPCHK run-time option and are unable to locate the cause of the problem, contact your service representative.

System action: No system action is performed.

Symbolic Feedback Code: CEE3JN

CEE3704I Expected data at address *address*:*data*.

Explanation: Provides the address *address* and expected data *data* when heap damage is found.

Programmer response: The data area following this message provides the actual data found at the damaged location.

System action: No system action is performed.

Symbolic Feedback Code: CEE3JO

CEE3705I Pointer at *address* should point to a valid *controlblock*.

Explanation: The pointer at location *address* should point to a Language Environment control block with a *controlblock* eye catcher.

Programmer response: The data area following this message provides the actual data found at the damaged location.

System action: No system action is performed.

Symbolic Feedback Code: CEE3JP

CEE3706I The contents of the free tree node at *address1* in the heap segment beginning at *address2* do not match the STORAGE run-time option heap_free_value.

Explanation: The contents of storage at *address1* should match the heap_free_value specified with the STORAGE run-time option. The first 16 bytes at *address1* provide header information and are not expected to match the heap_free_value.

Programmer response: The data area following this message provides the actual data found at the damaged location.

System action: No system action is performed.

Symbolic Feedback Code: CEE3JQ

CEE3707I *branch* pointer is bad in the free tree at *address1* in the heap segment beginning at *address2*.

Explanation: The pointer to the *branch* branch of the free tree node at address *address1* does not point to another free tree node.

Programmer response: The data area following this message provides the actual data found at the damaged location.

System action: No system action is performed.

Symbolic Feedback Code: CEE3JR

CEE3708I *branch* length is bad in the free tree at *address1* in the heap segment beginning at *address2*.

Explanation: The length of the *branch* branch of the free tree node at address *address1* is damaged.

Programmer response: The data area following this message provides the actual data found at the damaged location.

System action: No system action is performed.

Symbolic Feedback Code: CEE3JS

CEE3709I Either the *branch* pointer or length is damaged in the free tree at *address1* in the heap segment beginning at *address2*.

Explanation: The pointer to the *branch* branch of the free tree node at address *address1* plus it's length does not match a heap storage element.

Programmer response: The data area following this message provides the actual data found at the damaged location.

System action: No system action is performed.

Symbolic Feedback Code: CEE3JT

CEE3710I Heap element at *address* is damaged; expected data is *word1*:*word2*.

Explanation: The header of the heap storage element at *address* does not match the expected data *word1* and *word2*.

Programmer response: The data area following this message provides the actual data found at the damaged location.

System action: No system action is performed.

Symbolic Feedback Code: CEE3JU

CEE3711I Processing of the HEAPCHK run-time option has been terminated due to a previous error. Heaps are no longer being checked for damage.

Explanation: A message preceding this one in the message file will provide the actual error which caused the HEAPCHK processing to be terminated.

Programmer response: Locate the message that describes the actual error and take appropriate action to correct the problem. If you are unable to locate the cause of the problem, contact your service representative.

System action: No system action is performed.

Symbolic Feedback Code: CEE3JV

CEE3712I The number *number* of the run-time option *option* must end with an M to indicate the number of Megs of storage.

Explanation: The number number of the run-time option option does not end in M.

Programmer response: Correct the run-time options string so that the value ends with M.

System action: The number is ignored

Symbolic Feedback Code: CEE3K0

CEE3713I The cell pool count *number* of the run-time option *option* must be at least 4.

Explanation: The cell pool count *number* of the run-time option *option* was less than 4.

Programmer response: Correct the run-time options string so that the number is more than 4.

System action: The number is ignored.

Symbolic Feedback Code: CEE3K1

CEE3714I The heap pool contains a cell to be freed at *address* but it has already been freed.

Explanation: The cell at *address* is being freed twice by the application.

Programmer response: Review the heap pool trace data for the heap pool to learn the application offset that is freeing this cell.

System action: No system action is performed.

Symbolic Feedback Code: CEE3K2

CEE3715W The parameter string returned from CEE3PR2 was truncated due to insufficient storage space for the string provided by the caller.

Explanation: The user parameters exceeded the space provided by the caller and the remainder of the user parameters are truncated.

Programmer response: Increase the storage space to the length returned by the service.

System action: The user parameter string is truncated to the length of the space provided by the caller.

Symbolic Feedback Code: CEE3K3

CEE3728S The use of a function, which is not supported by this release of Language Environment was detected

Explanation: The application has exploited a new function not available on this release of Language Environment.

Programmer response: Remove the usage of the unsupported function from the application or execute this application on a release of Language Environment that supports this function. Depending on the TERMTHDACT run-time option, a CEEDUMP may be available for additional diagnosis. One can use the traceback information in the CEEDUMP to locate the specific Language Environment function used, that caused this message.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE3KG

CEE3730I One or more error messages pertaining to the system run-time options were lost.

Explanation: The run-time options error table (ROET) overflowed.

Programmer response: None

System action: The errors that are detected after the table overflowed are discarded.

Symbolic Feedback Code: CEE3KI

CEE3731I The following messages pertain to the system default run-time options in the *groupname* in *membername*.

Explanation: The messages after this one up to the next message of this type with a different source, pertain to the system default options. These defaults are found in options group *groupname*, in Parmlib member *membername* or in the SET CEE command.

Programmer response: None

System action: None

Symbolic Feedback Code: CEE3KJ

CEE3732I An end of comment delimiter is missing on line *line* of parmlib member *membername*.

Explanation: Line number line in the parmlib member *membername* is missing an ending delimiter for a comment. All comments must end on the same line on which they begin.

Programmer response: Correct the options in the Parmlib member.

System action: None.

Symbolic Feedback Code: CEE3KK

CEE3733I The eof for parmlib member *membername* was reached before finding a closing option group delimiter for the options group *groupname*.

Explanation: The delimiter that ends option group *groupname* is missing. Each group must begin and end with a parenthesis.

Programmer response: Correct the Parmlib member.

System action: None

Symbolic Feedback Code: CEE3KL

CEE3734I The EOF for parmlib member *membername* was reached before finding a closing quote delimiter for the options group *groupname*.

Explanation: Each quoted string must end with a quote character that matches the one used to start the string.

Programmer response: Correct the Parmlib member.

System action: None

Symbolic Feedback Code: CEE3KM

CEE3735I Text on line *line* starting at column *column* is outside of an options group in parmlib member *membername*.

Explanation: All text must be in a comment or an options group.

Programmer response: Correct the options in the Parmlib member.

System action: None

Symbolic Feedback Code: CEE3KN

CEE3736I Error code *code* on line number *line* of parmlib member *membername*.

Explanation: An I/O error occurred while reading the parmlib member.

Programmer response: Correct the Parmlib member.

System action: None.

Symbolic Feedback Code: CEE3KO

CEE3737I Contents of parmlib member *membername*.

Explanation: This is the heading of the listing of the contents of a parmlib member specified by CEE= at IPL. One or more lines may follow with the contents of the parmlib member.

Programmer response: None

System action: None

Symbolic Feedback Code: CEE3KP

CEE3738I A CEE= parmlib member was not found or is in error.

Explanation: z/OS CEE parmlib parsing has encountered a syntax error in one of the specified parmlib members or the specified parmlib member does not exist. Once the system has completed it's IPL, check the hardcopy log for the errors.

Programmer response: None

System action: The system prompts for a new CEE= parmlib specification.

Symbolic Feedback Code: CEE3KQ

CEE3739I Language Environment initialization complete.

Explanation: Language Environment has completed the reading and processing of parmlib members.

Programmer response: None

System action: The processing of parmlib members has completed.

Symbolic Feedback Code: CEE3KR

CEE3740I The following messages pertain to the system default run-time options.

Explanation: The messages after this one up to the next message of this type with a different source, pertain to the system default options.

Programmer response: None

System action: The processing of parmlib members has completed.

Symbolic Feedback Code: CEE3KS

CEE3741I The run-time option *option* was not valid from the system default options.

Explanation: *option* was not valid from the System Default options.

Programmer response: Remove the *option* run-time option from the Parmlib member or from the SET CEE command.

System action: *option* is ignored.

Symbolic Feedback Code: CEE3KT

CEE3742I The SET CEE command has completed.

Explanation: The SET CEE command has completed processing the parmlib members. Errors may have been reported while parsing the run-time options in the member(s).

Programmer response: None

System action: The run-time options in the Parmlib member specified with the SET CEE command have been saved.

Symbolic Feedback Code: CEE3KU

CEE3743I The SETCEE command has completed.

Explanation: The SETCEE command was successful has completed processing the run-tim options. Errors may have been reported while parsing the run-time options in the member(s)..

Programmer response: None

System action: The run-time options specified with the SETCEE command have been saved.

Symbolic Feedback Code: CEE3KV

CEE3744I

hh.mm.ss DISPLAY
CEE=(*members*) or NO MEMBERS SPECIFIED

Explanation: The following material is part of the message text:

hh.mm.ss The time in hours (00--23), minutes (00--59), and seconds (00--59) for the DISPLAY CEE command.

CEE=(*members*) The parmlib member name list specified on the SET CEE command or in the IEASYS member used at IPL.

NO MEMBERS SPECIFIED is displayed when there have been no CEEPRM members.

Programmer response: None

System action: None

Symbolic Feedback Code: CEE3L0

CEE3745I

hh.mm.ss DISPLAY
***groupname* or NO MEMBERS SPECIFIED**
Where Set Options
Text

Explanation: The following material is part of the message text:

hh.mm.ss
The time in hours (00--23), minutes (00--59), and seconds (00--59) for the DISPLAY CEE command.

groupname
The name of the group for the options listed in the text portion of this message. Either CEECOPT, CEEDOPT, or CELQDOPT.

CEE=(members)
The parmlib member name list specified on the SET CEE command or in the IEASYS member used at IPL.

NO MEMBERS SPECIFIED
Displayed when there have been no CEEPRM members.

text A list of run-time options that have been set by either a CEEPRM or the SETCEE command. Only options that have been set in a member or by SETCEE will be displayed in this list. CEEPRM

Programmer response: None

System action: None

Symbolic Feedback Code: CEE3L1

CEE3746I No option group *groupname* exists.

Explanation: The group name *groupname* requested on the Display CEE or SETCEE command is invalid.

Programmer response: Reissue the command with a valid group name.

System action: None

Symbolic Feedback Code: CEE3L2

CEE3750F Note: An alternative heap manager is in use.

Explanation: Issued as part of a CEE3DMP report.

Symbolic Feedback Code: CEE3L6

CEE3751I The alternative heap manager does not support XPLINK.

Explanation: The user or application attempted to use an alternative heap manager for an XPLINK application, but alternative heap manager indicated that it does not support the XPLINK environment.

Programmer response: Select an alternative heap manager that supports XPLINK, or do not use one. Contact the owner of the alternative heap manager if necessary.

System action: The alternative heap manager is ignored. Processing continues with the normal heap manager.

Symbolic Feedback Code: CEE3L7

CEE3752I The alternative heap manager did not supply all of the replacement routine addresses.

Explanation: A NULL or incorrect value for one of the replacement routines was set by the alternative heap manager.

Programmer response: Contact the owner of the alternative heap manager.

System action: The alternative heap manager is ignored. Processing continues with the normal heap manager.

Symbolic Feedback Code: CEE3L8

CEE3775W A conflict was detected between the TERMTHDACT suboption CICSDDS and &level. The TERMTHDACT level setting has been set to TRACE.

Explanation: The thread termination dump output level setting conflicts with the destination setting of CICSDDS. A level of TRACE or less must be used with CICSDDS.

Programmer response: Change the value of the detail level to TRACE or less.

System action: The level setting was reset to TRACE.

Symbolic Feedback Code: CEE3LV

CEE3781I The value of the reg_stor_amount sub-option of the TERMTHDACT option was not in the valid range 0–256.

Explanation: The value of the reg_stor_amount sub-option was not a member in the range of 0–256.

Programmer response: Change the value of the reg_stor_amount sub-option to be in the valid range 0–256.

System action: The value is ignored. The default value of 96 is used.

Symbolic Feedback Code: CEE3M5

CEE3782E The value of the REGSTOR option of CEE3DMP was not in the valid range 0–256.

Explanation: The value of the REGSTOR option of CEE3DMP was not a number in the range of 0–256.

Programmer response: Change the value of the REGSTOR option to be in the valid range 0–256.

System action: The value is ignored. The default value of 96 is used.

Symbolic Feedback Code: CEE3M6

| **CEE3783I The page_len suboption of the CEEDUMP run-time option is not valid.**

| **Explanation:** The page_len suboption supplied for the CEEDUMP run-time option is not valid. Page_len must adhere
| to the following rules:

| • It must be a 1 to 9 digit decimal whole number.

| • The value must be either 0 or greater than 9.

| **Programmer response:** Verify that page_len complies with the previous rules.

| **System action:** The suboption value is ignored. Processing continues.

| **Symbolic Feedback Code:** CEE3M7

| **CEE3784I The input string <string> is not a valid SYSOUT class.**

| **Explanation:** An invalid sysout class was encountered during CEEDUMP run-time options parsing.

| The sysout class provided must adhere to the following rules:

| • It must be one character long.

| • The class can equal ’A’ through ’Z’ or ’0’ through ’9’ or ’*’ as JCL rules allowed.

| • The class can not be specified in quotation marks.

| **Programmer response:** Verify that the sysout class complies with the previous rules.

| **System action:** The suboption value is ignored. Option processing stops.

| **Symbolic Feedback Code:** CEE3M8

| **CEE3785I The dynamical allocation of a CEEDUMP DD statement using the specified sysout class has
| failed.**

| **Explanation:** The CEEDUMP run-time option specified the use of a sysout class for defining a CEEDUMP DD card.
| However, the dynamic allocation of the CEEDUMP data set failed during the use of the given sysout class <class>. A
| new CEEDUMP data set was allocated to SYSOUT=* instead.

| **Programmer response:** Verify that the sysout class specified in the CEEDUMP run-time option is a valid class in
| your installation.

| **System action:** A CEEDUMP data set is allocated to SYSOUT=*.

| **Symbolic Feedback Code:** CEE3M9

| **CEE3786I The input string <string> is not a valid sysout form-name.**

| **Explanation:** An invalid form-name was encountered during CEEDUMP run-time options parsing. The form-name
| provided must consist of one through four alphanumeric or national (#,@,$) characters according to JCL rules.

| **Programmer response:** Verify that the form-name specified is valid.

| **System action:** The suboption value is ignored. Option processing stops.

| **Symbolic Feedback Code:** CEE3MA

| **CEE3787I The <option> run-time option did not contain a required left parenthesis.**

| **Explanation:** A required left parenthesis was not found while processing a suboption of the above mentioned
| run-time option.

| **Programmer response:** Correct the run-time option string.

| **System action:** The run-time option string processed is ignored. Option processing stops.

| **Symbolic Feedback Code:** CEE3MB

CEE3790I The CEEOPTS data set specifies an unsupported data set type.

Explanation: Language Environment was unable to read the data set specified as the DD:CEEOPTS run-time options file due to it being in an unsupported format.

Programmer response: Make sure that the dataset specified is in a supported format. Refer to the Language Environment Programming Guide for details on supported dataset types.

System action: The invalid DD:CEEOPTS dataset is ignored, the options specified within it are not read in, and execution continues as normal.

Symbolic Feedback Code: CEE3ME

CEE3791I An I/O error occurred accessing the CEEOPTS data set.

Explanation: Language Environment was unable to read the data set specified as the DD:CEEOPTS run-time options file due to a open/read error.

Programmer response: Make sure that the data set specified exists and the program has appropriate permissions to access it.

System action: The invalid DD:CEEOPTS dataset is ignored, the options specified within it are not read in, and execution continues as normal.

Symbolic Feedback Code: CEE3MF

CEE3792I The following messages pertain to the CEEOPTS data set run-time options.

Explanation: The messages after this one up to the next message of this type with a different source, pertain to the CEEOPTS data set.

Programmer response: No programmer response is required.

System action: No system action is performed.

Symbolic Feedback Code: CEE3MG

CEE3793I One or more error messages pertaining to the DD:CEEOPTS run-time options were lost.

Explanation: The run-time options error table (ROET) overflowed.

Programmer response: Correct the reported errors so the discarded errors fit into the error table.

System action: The errors that are detected after the table overflowed are discarded.

Symbolic Feedback Code: CEE3MH

CEE3794I The run-time option *option* was not valid from the DD:CEEOPTS dataset.

Explanation: *option* was not valid from the DD:CEEOPTS dataset.

Programmer response: Remove the *option* run-time option from the DD:CEEOPTS dataset.

System action: *option* is ignored.

Symbolic Feedback Code: CEE3MI

CEE3795I The dataset specified by DD:CEEOPTS contained more data than is allowed. The data was truncated starting on line *line* at column *column*.

Explanation: The dataset specified by DD:CEEOPTS contained greater than 3072 bytes of data. Language Environment limits the amount of data within this dataset to 3072 bytes. Data beyond this limit indicated by line and column was truncated.

Programmer response: Make sure that the dataset specified does not contain more than 3072 bytes of data (including blanks). Shorten the dataset and re-run application.

System action: The excess data is truncated. The first 3072 bytes of the DD:CEEOPTS dataset are processed, and execution continues as normal.

Symbolic Feedback Code: CEE3MJ

CEE3796I An attempt to dynamically take a dump was not successful. The error return code was *return* and the reason code was *reason*.

Explanation: Language Environment invoked IEATDUMP to dynamically allocate a dump data set. However, the allocation was not successful. *Return* and *reason* are returned by the IEATDUMP system service or by RACF. IEATDUMP or RACF might issue other messages to the console or SYSLOG. See message CEE3798I for the name of the data set.

Programmer response: See *z/OS MVS Programming: Assembler Services Reference IAR-XCT* for information about IEATDUMP return and reason codes.

System action: No data set is allocated, no dump is created.

CEE3797I Language Environment has dynamically created a dump.

Explanation: Language Environment was able to obtain a dump. See message CEE3798I for the name of the data set.

Programmer response: View the dump by using IPCS to determine the cause of the ABEND.

System action: None.

CEE3798I Attempting to take a dump for ABEND *abend_code* to data set: *dumpname*

Explanation: Language Environment has determined that a dump for ABEND code *abend_code* needs to be created. A data set *dumpname* will be allocated.

Programmer response: This message should be followed by either CEE3797I or CEE3796I.

System action: None

CEE3800S The address passed to the stack segment routine was not within any Language Environmentt stack segment.

Explanation: The address passed to the stack segment routine was not within any currently allocated Language Environment stack segment.

Programmer response: Contact your service representative. This is an internal error.

System action: The bounds, segment type, and chain are undefined.

Symbolic Feedback Code: CEE3MO

CEE3817E The member event handler did not return a useable function pointer.

Explanation: The member language which compiled the input load module either does not support the CEEPGFD CWI, or encountered an unrecoverable error.

Programmer response: Ensure that the input load module is compiled from a language which supports the CEEPGFD CWI.

System action: CEEPGFD returns an unusable function pointer.

Symbolic Feedback Code: CEE3N9

CEE3818E The member event handler encountered an error.

Explanation: The member language which compiled the input load module either does not support the CEEPRFD CWI, or encountered an unrecoverable error.

Programmer response: Ensure that the input load module is compiled from a language which supports the CEEPRFD CWI, and that the function pointer is a valid pointer obtained from the CEEPGFD CWI.

System action: No function pointer is released.

Symbolic Feedback Code: CEE3NA

CEE3819I An invalid string *string* was found in the run-time option ENVAR.

Explanation: The string does not contain an equal sign. The input beginning at the string will be ignored.

Programmer response: ENVAR strings must be in the form of 'name=value'. The string may be missing or have misplaced quotation marks. You can specify multiple environment variables, separating the name=value pairs with commas. Quotation marks are required when specifying multiple variables.

System action: The invalid ENVAR string is ignored.

Symbolic Feedback Code: CEE3NB

CEE3821I The run-time option *old-option* appeared in the options string and is ignored when the THREADSTACK option is specified.

Explanation: *old-option* (NONIPTSTACK or NONONIPSTACK) is an option from a previous release that is supported by Language Environment for compatibility, but ignored if THREADSTACK is specified.

Programmer response: Change the run-time options string to use the new option THREADSTACK instead of NONIPTSTACK or NONONIPTSTACK.

System action: *old-option* is ignored if THREADSTACK is specified.

Symbolic Feedback Code: CEE3ND

CEE3825I The run-time option NONONIPTSTACK or NONIPTSTACK appeared in the options string. NONONIPTSTACK or NONIPTSTACK were mapped to THREADSTACK.

Explanation: NONIPTSTACK and NONONIPSTACK are options from a previous release that are supported by Language Environment for compatibility.

Programmer response: Change the run-time options string to use the THREADSTACK option instead of NONONIPTSTACK or NONIPTSTACK.

System action: NONONIPTSTACK or NONIPTSTACK is mapped to THREADSTACK.

Symbolic Feedback Code: CEE3NH

CEE3836I A statement number is not available for this DSA. DWARF data in the load module is corrupted.

Explanation: This is an internal error. Language Environment could not determine a statement number from the load module associated to this DSA. The required Compile Debug Architecture information found in this load module is corrupted. This error can take place when the respective load module was originally produced as a UNIX file name but was then incorrectly copied to a data set (some contents of the load module have an ASCII format). If this is the case, recompile the source associated to the load module and use the c89 utility -o option to generate a load module in a data set, or compile the source directly in a MVS environment.

Programmer response: If the problem persists contact your system representative.

System action: The statement number for this DSA is not displayed in the CEEDUMP. Processing continues.

Symbolic Feedback Code: CEE3NS

CEE3837I Statement numbers are not available. The explicit DLL load of DLL CDAEQED failed with feedback code <fc>.

Explanation: This is an internal error. Language Environment could not load to storage a CDA DLL required for obtaining statement numbers. Refer to the CEEPLDE feedback code documentation for further information about the cause of this failure. Verify SCEERUN2 member CDAEQED is found in the system's search order.

Programmer response: None

System action: Statement numbers are not displayed in the CEEDUMP. Processing continues.

Symbolic Feedback Code: CEE3NT

CEE3838I Statement numbers are not available. The explicit DLL load of DLL CDAEQDPI failed with feedback code <fc>.

Explanation: This is an internal error. Language Environment could not load to storage a CDA DLL required for obtaining statement numbers. Refer to the CEEPLDE feedback code documentation for further information about the cause of this failure. Verify SCEERUN2 member CDAEQDPI is found in the system's search order.

Programmer response: None

System action: Statement numbers are not displayed in the CEEDUMP. Processing continues.

Symbolic Feedback Code: CEE3NU

CEE3839I Statement numbers are not available. The explicit DLL load of DLL CELQDSNF failed with feedback code <fc>.

Explanation: This is an internal error. Language Environment could not load to storage a DLL required for obtaining statement numbers. Refer to the CEEPLDE feedback code documentation for further information about the cause of this failure. Verify SCEERUN2 member CELQDSNF is found in the system's search order.

Programmer response: None.

System action: Statement numbers are not displayed in the CEEDUMP. Processing continues.

Symbolic Feedback Code: CEE3NV

CEE3840I Statement numbers are not available. dllqueryfn() failed for a function in the DLL CELQDSNF.

Explanation: This is an internal error. Language Environment could not locate the function that determines the statement number.

Programmer response: Contact your service representative.

System action: Statement numbers are not displayed in the CEEDUMP. Processing continues.

Symbolic Feedback Code: CEE3O0

CEE3841I A statement number is not available for this DSA. An internal routine failed with return code <rc> and reason code <rs>.

Explanation: This is an internal error. Language Environment could not determine a statement number for this DSA.

Programmer response: Contact your service representative.

System action: A statement number is not displayed in the CEEDUMP. Processing continues.

Symbolic Feedback Code: CEE3O1

CEE3842I Additional Fully Qualified Name information is suppressed.

Explanation: Additional lines corresponding to other DSA entries are not shown in the traceback report. Excluding information about the main entry for C/C++ programs, up to 51 lines can be dumped in the Fully Qualified Names section of the traceback. Only complete entries will be output.

Programmer response: None.

System action: Some entries are not displayed in the traceback's Fully Qualified Names section. CEEDUMP processing continues.

Symbolic Feedback Code: CEE3O2

CEE3843I The program unit name is too long to be displayed. See the Fully Qualified Names section for the complete name.

Explanation: A program unit name longer than 60 characters cannot be displayed in the Condition Information section of the CEEDUMP. However, it is listed in the Fully Qualified Names section of the traceback for the entry name shown above this message.

Programmer response: None.

System action: Processing continues.

Symbolic Feedback Code: CEE3O3

| **CEE3845I CEEDUMP processing started.**

| **Explanation:** This is a message that will be issued in the dump report after the header in page 1. If the Language
| Environment dump report successfully completes then message CEE3846I will be issued at the end of the report.

| **Programmer response:** None.

| **System action:** Language Environment dump processing continues.

| **Symbolic Feedback Code:** CEE3O5

| **CEE3846I CEEDUMP processing completed.**

| **Explanation:** Processing for the Language Environment dump report has completed.

| **Programmer response:** None.

| **System action:** None.

| **Symbolic Feedback Code:** CEE3O6

| **CEE3854I Additional Fully Qualified Name information is suppressed.**

| **Explanation:** Additional lines corresponding to other DSA entries are not shown in the Fully Qualified Names section
| of the traceback. Only up to 72 lines can be displayed in this section. Only complete entries in the Fully Qualified
| Names section are shown.

| **Programmer response:** None

| **System action:** Some entries are not displayed in the traceback's Fully Qualified Names section. Language
| Environment dump processing continues.

| **Symbolic Feedback Code:** CEE3OE

| **CEE3858I Possible Bad Branch: Statement: <statement num> Offset: <offset>**

| **Explanation:** This message is issued by a signaled condition or CEE3DMP to help determine the error location when
| an exception occurs due to an invalid branch. If this message is issued by a signaled condition, then this message will
| be displayed along with the rest of the condition message. If it is displayed as part of a CEEDUMP report, it will
| appear in the condition information section for the active routine. If the statement number is not available, then only
| the offset will be listed in this message. The message number is not displayed along with the message text.

| **Programmer response:** Check the statement number or offset or both provided in this message for an invalid
| branch in your routine. For further information on how to use a statement number, entry address or entry offset to
| locate a problem in a routine, see *z/OS Language Environment Debugging Guide*.

| **System action:** Processing continues.

| **Symbolic Feedback Code:** CEE3OI

CEE3900S The function code passed to CEE3USR was not 1 or 2.

Explanation: The *function_code* specified in a CEE3USR call was invalid

Programmer response: Invoke CEE3USR with *function_code* 1 (for SET) or 2 (for QUERY).

System action: Neither SET nor QUERY is performed.

Symbolic Feedback Code: CEE3PS

CEE3901S The field number passed to CEE3USR was not 1 or 2.

Explanation: The field number specified in a CEE3USR call was invalid.

Programmer response: Invoke CEE3USR with *field_number* 1 or 2.

System action: Neither SET nor QUERY is performed.

Symbolic Feedback Code: CEE3PT

CEE3910S The CEECENQ function was invoked in a non CICS environment.

Explanation: The CEECENQ function is only available when running in a CICS environment. The issuing application must be running under the CICS subsystem in order to use this function.

Programmer response: Contact your service representative. This is an internal error.

System action: The function has not been performed.

Symbolic Feedback Code: CEE3Q6

CEE3911S The CEECENQ function is not supported at this level of CICS.

Explanation: The CEECENQ function is only available when running in a CICS environment. The issuing application must be running under a CICS level that supports this function.

Programmer response: Upgrade to a supported CICS level. This is an internal error.

System action: The function has not been performed.

Symbolic Feedback Code: CEE3Q7

CEE3912S The CEECENQ function has failed during an EXEC CICS ADDRESS EIB command.

Explanation: The CEECENQ function has failed while trying to obtain a CICS EIB address via an EXEC CICS ADDRESS EIB command.

Programmer response: Contact your service representative. This is an internal error.

System action: The function has not been performed.

Symbolic Feedback Code: CEE3Q8

CEE3913S The CEECENQ function has failed during an EXEC CICS ENQ command.

Explanation: The CEECENQ function has failed while issuing an EXEC CICS ENQ command.

Programmer response: Contact your service representative. This is an internal error.

System action: The function has not been performed.

Symbolic Feedback Code: CEE3Q9

CEE3914I The CEECENQ function is unable to enqueue on the specified resource.

Explanation: The CEECENQ function has received a response from an EXEC CICS ENQ command indicating that the resource is unavailable.

Programmer response: The resource is not available at this time.

System action: The function has not been performed.

Symbolic Feedback Code: CEE3QA

CEE3915S The CEECDEQ function is not supported at this level of CICS.

Explanation: The CEECDEQ function is only available when running in a CICS environment. The issuing application must be running under a CICS level that supports this function.

Programmer response: Upgrade to a supported CICS level. This is an internal error.

System action: The function has not been performed.

Symbolic Feedback Code: CEE3QB

CEE3916S The CEECDEQ function has failed during an EXEC CICS ADDRESS EIB command.

Explanation: The CEECDEQ function has failed while trying to obtain a CICS EIB address via an EXEC CICS ADDRESS EIB command.

Programmer response: Contact your service representative. This is an internal error.

System action: The function has not been performed.

Symbolic Feedback Code: CEE3QC

CEE3917S The CEECDEQ function has failed during an EXEC CICS DEQ command.

Explanation: The CEECDEQ function has failed while issuing an EXEC CICS DEQ command.

Programmer response: Contact your service representative. This is an internal error.

System action: The function has not been performed.

Symbolic Feedback Code: CEE3QD

CEE3918I The CEECDEQ function is unable to dequeue using the specified duration value.

Explanation: The CEECDEQ function has been invoked with a duration period that does not match the duration period that was specified on the corresponding enqueue for the named resource.

Programmer response: Contact your service representative. This is an internal error.

System action: The function has not been performed.

Symbolic Feedback Code: CEE3QE

CEE3919S The CEECDEQ function was invoked in a non CICS environment.

Explanation: The CEECDEQ function is only available when running in a CICS environment. The issuing application must be running under the CICS subsystem in order to use this function.

Programmer response: Contact your service representative. This is an internal error.

System action: The function has not been performed.

Symbolic Feedback Code: CEE3QF

| **CEE3930W The input value** *input_value* **in a call to the callable service** *callable_service_name* **was not within**
| **the valid range.**

| **Explanation:** The input value specified lies outside the valid range.

| **Programmer response:** Check the documentation to determine the range of values.

| **System action:** The request is ignored. Processing continues.

| **Symbolic Feedback Code:** CEE3QQ

| **CEE3931W CEEDLYM was invoked in a CICS environment.**

| **Explanation:** The CEEDLYM callable service cannot be used in a CICS environment.

| **Programmer response:** No programmer response required.

| **System action:** The request is ignored. Processing continues.

| **Symbolic Feedback Code:** CEE3QR

| **CEE3932W The system service *system_service* failed with return code *return_code* and reason code**
| ***reason_code*.**

| **Explanation:** The call to the system service failed. The return code and reason code (if the service provided a
| reason code) were returned.

| **Programmer response:** Check the documentation for the system service to determine the appropriate action to take
| for this return code and reason code. Consult with your system support personnel if necessary.

| **System action:** The request is ignored. Processing continues.

| **Symbolic Feedback Code:** CEE3QS

CEE4001S General Failure: Service could not be completed.

Explanation: An error was encountered attempting to complete a C/370 locale function.

Programmer response: Contact your service representative.

System action: The thread is terminated.

Symbolic Feedback Code: CEE3T1

CEE4015S Input Error: The number of characters to be transformed must be greater than zero.

Explanation: An error was encountered attempting to complete a C/370 locale function. The CEESTXF callable service was called with a number parameter that was non-positive.

Programmer response: Make sure the parameter is positive.

System action: The thread is terminated.

Symbolic Feedback Code: CEE3TF

CEE4086S Input Error: The number of characters to be formatted must be greater than zero.

Explanation: An error was encountered attempting to complete a C/370 locale function. The CEEFMON or CEEFTDS callable service was called with a maxsize parameter that was non-positive.

Programmer response: Make sure the parameter is positive.

System action: The thread is terminated.

Symbolic Feedback Code: CEE3VM

CEE4087S Input Error: The version_info control block contains a number that is not valid.

Explanation: An error was encountered attempting to complete a C/370 locale function. The CEELCNV or CEEQDTC callable service was called with a parameter that pointed to a an invalid version number.

Programmer response: Make sure the parameter points to a version_info control block that contains a correct version number.

System action: The thread is terminated.

Symbolic Feedback Code: CEE3VN

CEE5001S POSIX function was not available. POSIX(ON) run-time option must be in effect and UNIX System Services started.

Explanation: The requested POSIX service failed because the POSIX(ON) run-time option was not in effect and/or UNIX System Services were not started.

Programmer response: Specify the POSIX(ON) run-time option and/or contact your system programmer to start the UNIX System Services.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE4S9

CEE5002S POSIX function was not available. UNIX System Services were not started.

Explanation: The requested POSIX service failed because the UNIX System Services were not started.

Programmer response: Contact your system programmer to start the UNIX System Services.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE4SA

CEE5101C During initialization, the callable service BPX1MSS failed. The system return code was *return_code***; the reason code was** *reason_code***. The application will be terminated.**

Explanation: The callable service BPX1MSS failed with return code *return_code* and reason code *reason_code* because the application was not authorized to use UNIX System Services.

Programmer response: Contact your system administrator to have the id registered with UNIX System Services to use these services. See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your UNIX System Services support personnel if necessary.

System action: The application is terminated.

Symbolic Feedback Code: CEE4VD

CEE5102E A request to dump process and enclave information could not be guaranteed to be consistent due to system constraints.

Explanation: A call to dump services was made in a multithread environment, but the *QUIESCE_FREEZE* feature of the callable service BPX1PTQ was not available on the release that was running.

Programmer response: Check that the level of UNIX System Services you are running supports the *QUIESCE_FREEZE* feature of the UNIX System Services callable service BPX1PTQ. *QUIESCE_FREEZE* is an UNIX System Services Release 2 feature.

System action: A dump is taken with unpredictable results.

Symbolic Feedback Code: CEE4VE

CEE5103W The dump service was busy.

Explanation: A call to dump services was made in a multithread environment while another thread had requested that all threads be frozen.

Programmer response: To dump your active thread, put the call to the dump service in a loop that iterates until the dump is successful. However, dump information for your thread might already be in the dump report due to another thread requesting a dump of all threads in the process.

System action: No dump is taken. The thread is not terminated.

Symbolic Feedback Code: CEE4VF

CEE5104S The callable service BPX1PTQ failed. The system return code was *return_code*, the reason code was *reason_code*.

Explanation: The callable service, BPX1PTQ, is called by Language Environment to freeze and unfreeze threads. If this service fails, Language Environment will return the *return_code* and *reason_code*.

Programmer response: Look up the return code and reason code in *z/OS UNIX System Services Programming: Assembler Callable Services Reference* and take the appropriate action. It is possible that the service failed due to the fact that another thread had already given a freeze request. Consult with your system support personnel if necessary.

System action: The thread is terminated.

Symbolic Feedback Code: CEE4VG

CEE5105S A call was made to *callable_service_name* without the thread being frozen.

Explanation: A call was made to the CWI *callable_service_name* with a caaptr parameter which pointed to a CAA pointer of a thread that was not frozen.

Programmer response: Consult with your system support personnel.

System action: The thread is terminated.

Symbolic Feedback Code: CEE4VH

CEE5106S A call was made to *callable_service_name* with an invalid caaptr parameter.

Explanation: A call was made to the CWI *callable_service_name* with a caaptr parameter which didn't point to a valid CAA.

Programmer response: Consult with your system support personnel.

System action: The thread is terminated.

Symbolic Feedback Code: CEE4VI

CEE5151S The POSIX *fork()* function could not operate on the Language Environment member ID number *member_id*.

Explanation: The Language Environment member cannot be the object of a *fork()* or *vfork()* function. This message is issued from within a single-thread environment. In a multithread environment, message CEE5154S is issued.

Programmer response: Make sure that the member can be the object of a *fork()* or *vfork()* function before issuing the function.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE50V

CEE5152S The callable service BPX1FRK for the *fork* function was unsuccessful. The system return code was *return_code*, the reason code was *reason_code*.

Explanation: The callable service BPX1FRK for the *fork()* or *vfork()* function failed. The system return code and reason code were returned. The *return_code* and *reason_code* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE510

CEE5154S The requested *fork()* service failed because it was invoked from a multithread environment.

Explanation: The *vfork()* and *exec()* services can be invoked only from within a single-thread environment. The *fork()* service is not permitted from within a multithread environment where the application contains a high level language other than C/C++. A multithread *fork()* is not allowed in a Language Environment preinitialization (CEEPIPI) environment.

Programmer response: Try the requested service in a single-thread environment, use supported high level languages, and/or avoid using *fork()* in a multithread CEEPIPI environment.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE512

CEE5155S The callable service BPX1SPN for the *spawn* function was unsuccessful. The system return code was *return_code*, the reason code was *reason_code*.

Explanation: The callable service BPX1SPN for the *spawn() or spawnp()* function failed. The system return code and reason code were returned. The *return_code* and *reason_code* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE513

CEE5161S The callable service BPX1EXC for the exec() family function was unsuccessful. The system return code was *return_code*, the reason code was *reason_code*.

Explanation: The callable service BPX1EXC for the *exec()* family function failed. The system return code and reason code were returned.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE519

CEE5162S The environment variable, _CEE_RUNOPTS, was too long.

Explanation: During *exec()* processing, Language Environment propagated run-time options from the program issuing the *exec()* by concatenating all run-time options that were specified on invocation of this program with those specified in the environment variable *_CEE_RUNOPTS*. The size of the work area used to perform this concatenation was insufficient.

Programmer response: Verify that the value of the *_CEE_RUNOPTS* environment variable does not contain superfluous blanks or invalid run-time options.

System action: *exec()* family function failed.

Symbolic Feedback Code: CEE51A

CEE5176S There was insufficient storage to process an environment variable.

Explanation: The environment variable processing service was called, however, there was insufficient system storage available to process the request.

Programmer response: Additional system storage is required before more environment variables can be processed. Either free some heap storage or rerun the application with a larger region size.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE51O

CEE5177S A bad input character was detected for the environment variable name.

Explanation: The environment variable processing service was called, however, the name specified contained an invalid ″=″ character.

Programmer response: Correct the environment variable name and retry.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE51P

CEE5178S The environment variable anchor or array contained an invalid address.

Explanation: The environment variable processing service was called, however, an invalid anchor or array address was encountered.

Programmer response: For C/C++ programmers: If `**environ` is used to set or clear an environment variable, make sure that the address is correct.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE51Q

CEE5179S A parameter to the environment variable processing routine contained an invalid value.

Explanation: The environment variable processing service was called, however, one of the parameters specified contained an invalid value. For instance, the name length or value length was negative, or the function code was invalid.

Programmer response: Correct the parameter and retry.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE51R

CEE5180I The specified environment variable name already exists.

Explanation: An attempt was made to set an environment variable whose name already exists, but overwrite was not specified.

Programmer response: If you want to modify the variable, specify function code 5.

System action: The original variable is not modified.

Symbolic Feedback Code: CEE51S

CEE5201S The signal SIGFPE was received.

Explanation: A signal indicating an erroneous arithmetic operation was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the POSIX process and produce a traceback or dump, depending on how the TERMTHDACT run-time option is set. The return code is set to 3000 and the signal number for the process termination is set to 8.

Symbolic Feedback Code: CEE52H

CEE5202S The signal SIGILL was received.

Explanation: A signal indicating an invalid hardware instruction was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the POSIX process and produce a traceback or dump, depending on how the TERMTHDACT run-time option is set. The return code is set to 3000 and the signal number for the process termination is set to 4.

Symbolic Feedback Code: CEE52I

CEE5203S The signal SIGSEGV was received.

Explanation: A signal indicating an invalid memory reference was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the POSIX process and produce a traceback or dump, depending on how the TERMTHDACT run-time option is set. The return code is set to 3000 and the signal number for the process termination is set to 11.

Symbolic Feedback Code: CEE52J

CEE5204S The signal SIGABND was received.

Explanation: A signal indicating a user or system initiated abend was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the POSIX process and produce a traceback or dump, depending on how the TERMTHDACT run-time option is set. The return code is set to 3000 and the signal number for the process termination is set to 18.

Symbolic Feedback Code: CEE52K

CEE5205S The signal SIGTERM was received.

Explanation: A signal indicating a termination signal was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the POSIX process and produce a traceback or dump, depending on how the TERMTHDACT run-time option is set. The return code is set to 3000 and the signal number for the process termination is set to 15.

Symbolic Feedback Code: CEE52L

CEE5206S The signal SIGINT was received.

Explanation: A signal indicating an interruptive attention signal was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the POSIX process and produce a traceback or dump, depending on how the TERMTHDACT run-time option is set. The return code is set to 3000 and the signal number for the process termination is set to 2.

Symbolic Feedback Code: CEE52M

CEE5207E The signal SIGABRT was received.

Explanation: A signal indicating an abnormal termination signal was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the enclave with a return code of 2000 and the signal number for the process termination is set to 3.

Symbolic Feedback Code: CEE52N

CEE5208S The signal SIGUSR1 was received.

Explanation: A signal indicating an application-defined signal 1 was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the enclave with a return code of 3000 and the signal number for the process termination is set to 16.

Symbolic Feedback Code: CEE52O

CEE5209S The signal SIGUSR2 was received.

Explanation: A signal indicating an application-defined signal 2 was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the POSIX process and produce a traceback or dump, depending on how the TERMTHDACT run-time option is set. The return code is set to 3000 and the signal number for the process termination is set to 17.

Symbolic Feedback Code: CEE52P

CEE5210S The signal SIGHUP was received.

Explanation: A signal indicating a hangup on the controlling terminal or the termination of the controlling process was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the POSIX process and produce a traceback or dump, depending on how the TERMTHDACT run-time option is set. The return code is set to 3000 and the signal number for the process termination is set to 1.

Symbolic Feedback Code: CEE52Q

CEE5211S The signal SIGSTOP was received.

Explanation: A signal indicating a 'STOP' was raised. This signal cannot be caught or ignored.

Programmer response: None.

System action: The process is stopped.

Symbolic Feedback Code: CEE52R

CEE5212C The signal SIGKILL was received.

Explanation: A signal indicating a termination signal was raised. This signal cannot be caught or ignored.

Programmer response: None.

System action: The system abnormally terminates the process.

Symbolic Feedback Code: CEE52S

CEE5213S The signal SIGPIPE was received.

Explanation: A signal indicating a write to a pipe with no readers was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the POSIX process and produce a traceback or dump, depending on how the TERMTHDACT run-time option is set. The return code is set to 3000 and the signal number for the process termination is set to 13.

Symbolic Feedback Code: CEE52T

CEE5214S The signal SIGALRM was received.

Explanation: A signal was raised, indicating a timeout condition such as initiated by the *alarm()* function.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the POSIX process and produce a traceback or dump, depending on how the TERMTHDACT run-time option is set. The return code is set to 3000 and the signal number for the process termination is set to 14.

Symbolic Feedback Code: CEE52U

CEE5215W The signal SIGCONT was received.

Explanation: A signal indicating a 'continue if stopped' signal was raised.

Programmer response: None.

System action: If the default action is SIG_DFL, all stopped threads in the process are continued.

Symbolic Feedback Code: CEE52V

CEE5216W The signal SIGCHLD was received.

Explanation: A signal indicating a terminated child process or stopped condition was raised.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE530

CEE5217S The signal SIGTTIN was received.

Explanation: A signal was raised, indicating a read from a control terminal attempted by a language run-time component of a background process group condition.

Programmer response: None.

System action: If the default action is SIG_DFL, all threads in the process are stopped.

Symbolic Feedback Code: CEE531

CEE5218S The signal SIGTTOU was received.

Explanation: A signal was raised, indicating a write to a control terminal attempted by a language run-time component of a background process group condition.

Programmer response: None.

System action: If the default action is SIG_DFL, all threads in the process are stopped.

Symbolic Feedback Code: CEE532

CEE5219W The signal SIGIO was received.

Explanation: A signal indicating the completion of an input or output operation was raised.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE533

CEE5220S The signal SIGQUIT was received.

Explanation: A signal indicating an interruptive terminal signal requesting the process termination was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the enclave with a return code of 3000 and the signal number for the process termination set to 24.

Symbolic Feedback Code: CEE534

CEE5221S The signal SIGTSTP was received.

Explanation: A signal was raised, indicating an interruptive stop signal by a language run-time component of a background process group condition.

Programmer response: None.

System action: If the default action is SIG_DFL, all threads in the process are stopped.

Symbolic Feedback Code: CEE535

CEE5222S The signal SIGTRAP was received.

Explanation: A signal indicating a trap condition was raised.

Programmer response: None.

System action: If the signal is unhandled, the default action is to terminate the enclave with a return code of 3000 and the signal number for the process termination set to 26.

Symbolic Feedback Code: CEE536

CEE5223W The signal SIGIOERR was received.

Explanation: A signal indicating an I/O error was raised.

Programmer response: See *z/OS XL C/C++ Programming Guide* for information on SIGIOERR and how to respond to this error.

System action: No system action is taken.

Symbolic Feedback Code: CEE537

CEE5224W The signal SIGDCE was received.

Explanation: The SIGDCE signal was generated as a result of a `MODIFY DCEKERN,DEBUG pid=` command. It communicates to a DCE-enabled process a desire to enable DCE run-time debug messages. If the target process is not a DCE process, the target process does not know how to handle SIGDCE.

Programmer response: None.

System action: No system action taken.

Symbolic Feedback Code: CEE538

CEE5225S The signal SIGPOLL was received.

Explanation: This signal indicates that a pollable event has occurred. If the signal is unhandled, the following default action will be applied: The program (enclave) is terminated and a traceback or dump is issued depending on the TERMTHDACT run-time option. The return code is set to 3000 and the signal number for the process termination is set to 5.

Programmer response: None.

System action: No system action taken.

Symbolic Feedback Code: CEE539

CEE5226W The signal SIGURG was received.

Explanation: This signal indicates that high bandwidth data is available at a socket.

Programmer response: None.

System action: No system action taken.

Symbolic Feedback Code: CEE53A

CEE5227S The signal SIGBUS was received.

Explanation: This signal indicates that a bus error has occurred. If the signal is unhandled, the following default action will be applied: The program (enclave) is terminated and a traceback or dump is issued depending on the TERMTHDACT run-time option. The return code is set to 3000 and the signal number for the process termination is set to 10.

Programmer response: None.

System action: No system action taken.

Symbolic Feedback Code: CEE53B

CEE5228S The signal SIGSYS was received.

Explanation: This signal indicates that a bad system call was detected. If the signal is unhandled, the following default action will be applied: The program (enclave) is terminated and a traceback or dump is issued depending on the TERMTHDACT run-time option. The return code is set to 3000 and the signal number for the process termination is set to 12.

Programmer response: None.

System action: No system action taken.

Symbolic Feedback Code: CEE53C

CEE5229W The signal SIGWINCH was received.

Explanation: This signal indicates that the window size has changed.

Programmer response: None.

System action: No system action taken.

Symbolic Feedback Code: CEE53D

CEE5230S The signal SIGXCPU was received.

Explanation: This signal indicates that the CPU time limit has been exceeded. If the signal is unhandled, the following default action will be applied: The program (enclave) is terminated and a traceback or dump is issued depending on the TERMTHDACT run-time option. The return code is set to 3000 and the signal number for the process termination is set to 29.

Programmer response: None.

System action: No system action taken.

Symbolic Feedback Code: CEE53E

CEE5231S The signal SIGXFSZ was received.

Explanation: This signal indicates that the file size limit has been exceeded. If the signal is unhandled, the following default action will be applied: The program (enclave) is terminated and a traceback or dump is issued depending on the TERMTHDACT run-time option. The return code is set to 3000 and the signal number for the process termination is set to 30.

Programmer response: None.

System action: No system action taken.

Symbolic Feedback Code: CEE53F

CEE5232S The signal SIGVTALRM was received.

Explanation: This signal indicates that a virtual timer has expired. If the signal is unhandled, the following default action will be applied: The program (enclave) is terminated and a traceback or dump is issued depending on the TERMTHDACT run-time option. The return code is set to 3000 and the signal number for the process termination is set to 31.

Programmer response: None.

System action: No system action taken.

Symbolic Feedback Code: CEE53G

CEE5233S The signal SIGPROF was received.

Explanation: This signal indicates that a profiling timer has expired. If the signal is unhandled, the following default action will be applied: The program (enclave) is terminated and a traceback or dump is issued depending on the TERMTHDACT run-time option. The return code is set to 3000 and the signal number for the process termination is set to 32.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE53H

CEE5234I The signal SIGDUMP was received.

Explanation: A signal indicating a dump signal was raised. This signal cannot be caught or ignored.

Programmer response: None.

System action: The system will obtain a user address space dump.

Symbolic Feedback Code: CEE53I

CEE5235I The signal SIGDANGER was received.

Explanation: A signal indicating OMVS subsystem shutdown in progress was received.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE53J

CEE5236I The signal SIGTHSTOP was received.

Explanation: A signal indicating that a thread should be stopped was raised. This signal cannot be caught or ignored.

Programmer response: None.

System action: The specified thread will be stopped.

Symbolic Feedback Code: CEE53K

CEE5237I The signal SIGTHCONT was received.

Explanation: A signal indicating a thread should be resumed was raised. This signal cannot be caught or ignored.

Programmer response: None.

System action: The specified thread will be resumed.

Symbolic Feedback Code: CEE53L

CEE5301S An invalid message number was received by the internal signal handling routine.

Explanation: The message number specified in the condition token passed to CEEOKILL was invalid. Only those message numbers that correspond to valid POSIX signals (5201 through 5224) are allowed.

Programmer response: None.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE55L

CEE5302S **A signal could not be raised due to a system-detected error, with return code *error-code* and reason code *reason-code*.**

Explanation: The Language Environment library routine called the callable services BPX1KIL (for kill or raise) or BPX1PTK (all others including *pthread_kill* and CEESGL) and the call was not successful. The system return code and reason code were returned. The *error-code* is a decimal number, and the *reason-code* is hexadecimal.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE55M

CEE5401S **The function code *func_code* to CEEMPMSG was invalid.**

Explanation: A call to CEEMPMSG, the message handler, had an invalid function code.

Programmer response: Report the error to your system support personnel.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE58P

CEE5402I **The callable service BPX1OPN, when invoked to open the message file, was unsuccessful. The system return code was *posix_rc*, the reason code was *posix_rsn*.**

Explanation: The callable service BPX1OPN, when invoked to open the message file, failed. The system return code and reason code were returned. The *posix_rc* and *posix_rsn* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: No system action is taken.

Symbolic Feedback Code: CEE58Q

CEE5403I **The callable service BPX1WRT, when invoked to write to the message file, was unsuccessful. The system return code was *posix_rc*; the reason code was *posix_rsn*.**

Explanation: The callable service BPX1WRT, when invoked to write to the message file, failed. The system return code and reason code were returned. The *posix_rc* and *posix_rsn* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: No system action is taken.

Symbolic Feedback Code: CEE58R

CEE5404I **The callable service BPX1CLO, when invoked to close the message file, was unsuccessful. The system return code was *posix_rc*; the reason code was *posix_rsn*.**

Explanation: The callable service BPX1CLO, when invoked to close the message file, failed. The system return code and reason code were returned. The *posix_rc* and *posix_rsn* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: No system action is taken.

Symbolic Feedback Code: CEE58S

CEE5405I The callable service BPX1GCW, when invoked to determine the current working directory, was unsuccessful. The system return code was *posix_rc*; the reason code was *posix_rsn*.

Explanation: The callable service BPX1GCW, when invoked to determine the current working directory, failed. The system return code and reason code were returned. The *posix_rc* and *posix_rsn* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the appropriate action to take for this return code and reason code.

System action: No Language Environment dump taken.

Symbolic Feedback Code: CEE58T

CEE5526S There was not enough storage to create the new key.

Explanation: There was not enough storage to create the new key. Keys are allocated in heap storage.

Programmer response: Increase the storage allocation available to the execution of the program by either freeing some heap storage or rerunning the application with a larger region size.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5CM

CEE5527S The key namespace was exhausted.

Explanation: The maximum number of keys allowed have been created.

Programmer response: Use the `sysconf()` function to determine the maximum number of keys that can be created within an enclave. Do not exceed this limit.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5CN

CEE5528S Termination was in progress. Key creates were not allowed.

Explanation: Key creates (CEEOPKC) calls were not allowed during thread termination after all destructor routines have been executed.

Programmer response: None.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5CO

CEE5529S There was no storage to bind value to the key.

Explanation: There was not enough storage available in the address space to acquire sufficient heap storage to satisfy the CEEOPSS call.

Programmer response: Either free some heap storage or rerun the application with a larger region size.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5CP

CEE5530S The specified key ID was invalid.

Explanation: The key identifier passed on the call to internal services CEEOPSS or CEEOPGS did not refer to a valid key.

Programmer response: Before setting or getting the value associated with a key, the key must be created using the internal CEEOPKC service.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5CQ

CEE5531S The key set was not allowed.

Explanation: Key set operation (CEEOPSS) calls were not allowed during thread termination after all destructor routines had been executed.

Programmer response: None.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5CR

CEE5532S The key get was not allowed.

Explanation: Key gets (CEEOPGS) calls were not allowed during thread termination after all destructor routines had been executed.

Programmer response: None.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5CS

CEE5533S An invalid key pointer was passed in key create operation.

Explanation: The pointer to the storage location for which the newly created key was to be placed was invalid.

Programmer response: None.

System action: The key create is not performed.

Symbolic Feedback Code: CEE5CT

CEE5551S The array entry described by the slot parameter was already set.

Explanation: Conditional invocation of CEEOSETE internal service failed because the array entry described by the slot parameter was already set. The *slot* parameter described an array entry that was already in use. The entry can be set by using UNCONDITIONAL form of this CWI.

Programmer response: None.

System action: The entry is not set.

Symbolic Feedback Code: CEE5DF

CEE5552S The array entry described by the slot parameter was invalid or was reserved.

Explanation: Invocation of the CEESETE internal service failed because the array entry described by the slot parameter was invalid or was reserved. The *slot* parameter described an array entry that was unavailable for use. The value of *slot* was invalid.

Programmer response: Choose a value for *slot* that is not reserved.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5DG

CEE5553S The callable service BPX1IPT (run a program on the IPT task) was unsuccessful. The system return code was *return_code*; the reason code was *reason_code*.

Explanation: The callable service BPX1IPT (run a program on the IPT task) failed. The system return code and reason code were returned. The *return_code* and *reason_code* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5DH

CEE5601S The attributes object parameter did not contain a valid initialized attributes object (POSIX PTAT).

Explanation: Each routine for which the thread attributes object was a parameter checks certain fields in that object to verify that they contain valid values. If any of the fields that were checked by the routine contained an invalid value, this condition was raised.

Programmer response: Modify the calling program to pass a valid parameter object.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5F1

CEE5602S The detachstate parameter did not contain a valid value.

Explanation: The *detachstate* parameter must be a fullword containing binary 0 for undetached or 1 for detached.

Programmer response: Modify the calling program to pass a valid parameter value.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5F2

CEE5603S The threadweight parameter did not contain a valid value.

Explanation: The *threadweight* parameter must be a fullword containing binary 0 for heavy-weight, or 1 for medium-weight.

Programmer response: Modify the calling program to pass a valid parameter value.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5F3

CEE5604S A new thread could not be created due to a system-detected error, with error code *error-code* and reason code *reason-code*.

Explanation: The Language Environment library routine called UNIX System Services and failed. The error code and reason code were returned. The *error-code* and *reason-code* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* or *z/VM: OpenExtensions Callable Services Reference* for the appropriate action to take for this error code and reason code. Consult with your system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5F4

CEE5605S A new thread could not be created due to an insufficient storage condition.

Explanation: Storage resource was insufficient for a new thread to be created.

Programmer response: Use a larger region size or release some heap storage, and retry the application.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5F5

CEE5606S The callable service BPX1PTJ failed due to an invalid thread ID on a join request. The system return code was *return_code*; the reason code was *reason_code*.

Explanation: The callable service BPX1PTJ was called by the Language Environment internal join service CEEOPJ to wait for a thread to terminate. However, BPX1PTJ returned without waiting because the ID of the target thread specified on the join request was invalid. The *return_code* and *reason_code* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* or *z/VM: OpenExtensions Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5F6

CEE5607S **The callable service BPX1PTJ failed due to an invalid thread ID on a join request. The system return code was** *return_code***; the reason code was** *reason_code***.**

Explanation: The callable service BPX1PTJ was called by the Language Environment internal join service CEEOPJ to wait for a thread to terminate. However, BPX1PTJ returned without waiting because the ID of the target thread specified on the join request was the same as the ID of the calling thread that would result in a deadlock condition. The *return_code* and *reason_code* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* or *z/VM: OpenExtensions Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5F7

CEE5608S **The callable service BPX1PTJ failed during a thread join request. The system return code was** *return_code***; the reason code was** *reason_code***.**

Explanation: The callable service BPX1PTJ was called by the Language Environment internal join service CEEOPJ to wait for a thread to terminate. However, BPX1PTJ returned without waiting. The *return_code* and *reason_code* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* or *z/VM: OpenExtensions Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5F8

CEE5609S **The callable service BPX1PTJ failed during a thread join request. The system return code was** *return_code***; the reason code was** *reason_code***.**

Explanation: The callable service BPX1PTJ was called by the Language Environment internal join service CEEOPJ to wait for a thread to terminate. However, BPX1PTJ returned without waiting indicating the target thread was not in an undetached state and could not be joined. The *return_code* and *reason_code* fields are decimal numbers.

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* or *z/VM: OpenExtensions Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5F9

CEE5612S **The stacksize parameter did not contain a valid value.**

Explanation: The *stacksize* parameter must be a fullword containing a binary number that is greater than or equal to zero. If positive, it specifies the number of bytes to be used for the stack. If 0, it specifies that 1/2 of all available storage should be used for the stack.

Programmer response: Modify the calling program to pass a valid parameter value.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5FC

CEE5613S The synctype parameter did not contain a valid value.

Explanation: The *synctype* parameter must be a fullword containing binary 0 for synchronous.

Programmer response: Modify the calling program to pass a valid parameter value.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5FD

CEE5626S There was insufficient storage for cleanup push operation.

Explanation: The internal service CEEOPCPU failed while trying to acquire heap storage for the cleanup routine registration.

Programmer response: Either free some heap storage or rerun the application with a larger region size.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5FQ

CEE5627S Cleanup push was not allowed during thread termination.

Explanation: Cleanup routine push operations (CEEOPCPU) were not allowed during thread termination after all cleanup routines have been executed.

Programmer response: None.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5FR

CEE5628S The cleanup pop was not allowed during thread termination.

Explanation: Cleanup routine pop operations (CEEOPCPO) were not allowed during thread termination after all cleanup routines have been executed.

Programmer response: None.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5FS

CEE5629S The cleanup stack was empty.

Explanation: The internal service CEEOPCPO failed because there were no cleanup routines available. The cleanup stack was empty.

Programmer response: None.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5FT

CEE5651S The oncecontrol parameter did not contain a valid value.

Explanation: The *oncecontrol* parameter must be a fullword containing binary 0 for initial value, or one of the other defined but not externalized values set by the *pthread_once* routine.

Programmer response: Modify the calling program to pass a valid parameter value. *oncecontrol* should be initialized by the caller to *PTHREAD_ONCE_INIT(0)*. After initialization, it should not be modified directly but only by calling the *pthread_once* routine. It should not be tested directly by the caller, but only implicitly by calling the *pthread_once* routine.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5GJ

CEE5701S The mutex object was not initialized.

Explanation: The mutex-related service that was invoked requires the mutex object specified as a parameter be initialized.

Programmer response: Use the internal service CEEOPMI to initialize the mutex object before invoking the service that failed.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5I5

CEE5702S The mutex was already owned.

Explanation: A thread invoked the mutex lock internal service CEEOPML specifying a nonrecursive mutex that had already been locked by the thread. Only a mutex that had been given the attribute RECURSIVE can be locked multiple times by the same thread.

Programmer response: None.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5I6

CEE5703S An addressing exception occurred referencing a mutex object or mutex attribute object.

Explanation: The address of a mutex object or mutex attribute object passed as an parameter on a mutex related service call was invalid. An addressing exception program interrupt occurred when the called service referenced this address.

Programmer response: Specify the correct mutex object or mutex attribute object when passing parameters to the mutex-related service call.

System action: The application is terminated.

Symbolic Feedback Code: CEE5I7

CEE5704C An addressing exception occurred referencing system storage allocated for mutexes.

Explanation: An addressing exception program interrupt occurred when a mutex-related service referenced system storage allocated for mutexes.

Programmer response: Make sure that the application has not written over system storage before issuing the service call.

System action: The thread is terminated.

Symbolic Feedback Code: CEE5I8

CEE5705S A mutex object has been changed since it was initialized.

Explanation: The mutex destroy internal service CEEOPMD detected that a mutex object (specified as a parameter) had changed since it was initialized by the mutex initialization internal service CEEOPMI. The mutex was destroyed, but internal service CEEOPMD did not alter the storage associated with the mutex object. However, if internal service CEEOPMI was invoked, the storage was altered.

Programmer response: Make sure the application is not incorrectly reusing storage associated with the mutex object after initializing it.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5I9

CEE5706S The mutex was not owned by thread.

Explanation: A thread called the mutex unlock internal service CEEOPMU to unlock the mutex but the mutex was not owned by the thread. A thread acquired a mutex with the mutex lock internal service CEEOPML or mutex trylock internal service CEEOPMT and was said to own the lock.

Programmer response: Structure the application so that the thread that locks the mutex also unlocks the mutex.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5IA

CEE5707I The mutex was busy.

Explanation: The mutex trylock internal service CEEOPMT was invoked to lock a nonrecursive mutex that was already locked.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE5IB

CEE5708S The mutex object was already initialized.

Explanation: The mutex initialization internal service CEEOPMI was called to initialize a mutex object that had already been initialized.

Programmer response: Call the mutex destroy internal service CEEOPMD to destroy an initialized mutex object before initializing it again.

System action: The request is rejected.

Symbolic Feedback Code: CEE5IC

CEE5709S The mutex attribute object was not initialized.

Explanation: The mutex attribute-related services required that the mutex attribute object specified as an parameter be initialized.

Programmer response: Use internal service CEEOPXI to initialize the mutex attribute object before invoking the service that failed.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5ID

CEE5710S There was insufficient storage to initialize a mutex object.

Explanation: The mutex initialization internal service CEEOPMI was called to initialize a mutex object. However, there was insufficient system storage available.

Programmer response: Get additional system storage before initializing more mutex objects.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5IE

CEE5711S A mutex attribute object has been changed since it was initialized.

Explanation: The mutex attribute destroy internal service CEEOPXD detected that a mutex attribute object specified as a parameter was changed since it was initialized by the mutex attribute initialization internal service CEEOPXI. The mutex attribute object was destroyed, but internal service CEEOPXD did not alter the storage associated with the mutex attribute object. However, if internal service CEEOPXI was invoked, the storage was altered.

Programmer response: Make sure the application is not incorrectly reusing storage associated with the mutex attribute object after initializing it.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5IF

CEE5712S The mutex attribute object was already initialized.

Explanation: The mutex attribute initialization internal service CEEOPXI was called to initialize a mutex attribute object that had already been initialized.

Programmer response: Call the mutex attribute destroy internal service CEEOPXD to destroy an initialized mutex attribute object before initialized it again.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5IG

CEE5713S There was insufficient storage to initialize a mutex attribute object.

Explanation: The mutex attribute initialization internal service CEEOPXI was called to initialize a mutex attribute object. However, there was insufficient system storage available.

Programmer response: Acquire additional system storage before initializing more mutex attribute objects.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5IH

CEE5714S The mutex was busy.

Explanation: The mutex destroy internal service CEEOPMD was invoked to destroy a mutex that was in use. A mutex that was locked or associated with a condition wait or timed wait cannot be destroyed.

Programmer response: Verify that your applications owns the mutex before calling CEEOPMD to destroy the mutex.

System action: The request is rejected.

Symbolic Feedback Code: CEE5II

CEE5715S An addressing exception occurred while referencing attribute kind storage.

Explanation: The address where to return attribute kind that was passed as an parameter on a *getkind_np* service request, was invalid. An addressing exception occurred when the *getkind_np* service CEEOPX attempted to store the attribute kind value at this address.

Programmer response: Specify the correct attribute kind address parameter on the *getkind_np* service call.

System action: The application is terminated.

Symbolic Feedback Code: CEE5IJ

CEE5716C System mutex storage could not be freed.

Explanation: The destroy mutex internal service CEEOPMD was unable to free storage allocated for a mutex by the mutex initialization service CEEOPMI.

Programmer response: Check if the application might have written over system storage. Report this problem to the storage administrator.

System action: Thread is terminated.

Symbolic Feedback Code: CEE5IK

CEE5717C System mutex attribute storage could not be freed.

Explanation: The destroy mutex attribute internal service CEEOPXD was unable to free storage allocated for a mutex attribute by the mutex attribute initialization service CEEOPXI.

Programmer response: Check if the application might have written over system storage. Report this problem to the system administrator.

System action: Thread is terminated.

Symbolic Feedback Code: CEE5IL

CEE5718C There was invalid mutex attribute storage.

Explanation: The mutex attribute *getkind_np* internal service CEEOPXG found an invalid value in system storage for mutex attributes.

Programmer response: Check if the application might have written over system storage. Report this problem to the system administrator.

System action: Thread is terminated.

Symbolic Feedback Code: CEE5IM

CEE5719S There was an invalid attribute value.

Explanation: The attribute kind parameter in a call to internal service CEEOPXS specified an invalid attribute kind value. Valid values for *setkind_np* are NONRECURSIVE (0), RECURSIVE (1), NONRECURSIVE + NODEBUG (2), and RECURSIVE + NODEBUG (3).

Programmer response: Specify a correct attribute kind value.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5IN

CEE5720C A thread waiting for a mutex was forced to terminate.

Explanation: An event, such as the initial thread terminating, forced all threads to terminate including threads waiting for a mutex.

Programmer response: Check that all threads exit correctly.

System action: The thread is terminated.

Symbolic Feedback Code: CEE5IO

CEE5721C There was insufficient resource to initialize another mutex.

Explanation: The mutex init internal service, CEEOPMI, was invoked to initialize a mutex, but not enough resource was available to initialize another mutex.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE5IP

CEE5722I There was insufficient privilege to initialize the mutex.

Explanation: The mutex init internal service, CEEOPMI, was invoked to initialize a mutex, but not enough resource was available to initialize another mutex.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE5IP

CEE5724I There was insufficient resource to obtain the mutex.

Explanation: The mutex lock or trylock internal services CEEOPML or CEEOPMT was invoked to lock a recursive mutex, but not enough resource was available to obtain this recursive mutex another time.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE5IS

CEE5726S The condition object was not initialized.

Explanation: The condition-related service required that the condition object (specified as an parameter) be initialized.

Programmer response: Use internal service CEEOPCI to initialize the condition object before invoking the service that failed.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5IU

CEE5727S The condition signal service (CEEOPCS) failed due to a system detected error with error code *return-code* **and reason code** *reason-code***.**

Explanation: The condition signal internal service CEEOPCS called the callable service BPX1CPO to signal another thread waiting on the condition. BPX1CPO returned an unexpected error code and reason code.

Programmer response: Report this failure to your system administrator.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5IV

CEE5728C An addressing exception occurred referencing system storage related to conditions variables.

Explanation: An addressing exception program interrupt occurred when a condition variable related service referenced system storage allocated for a condition variable.

Programmer response: Make sure that the application has not written over system storage before issuing the service call.

System action: The thread on which the addressing exception occurred is terminated.

Symbolic Feedback Code: CEE5J0

CEE5729S The mutex specified on a condition wait or timed wait request was a recursive mutex. The request was rejected.

Explanation: The condition wait internal service CEEOPCW and condition timed wait internal service CEEOPCT did not accept a request since the mutex associated with the request was recursive.

Programmer response: Do not specify a mutex with the recursive attribute on a condition wait or timed wait request.

System action: The request is rejected.

Symbolic Feedback Code: CEE5J1

CEE5730S The mutex specified on a condition wait or timed wait request was a different mutex than the one already associated with the condition variable. The request was rejected.

Explanation: Different threads called the condition wait internal service CEEOPCW or condition timed wait internal service CEEOPCT and specified the same condition object but different mutexes on the requests.

Programmer response: Make sure that all threads waiting on a particular condition variable specify the same mutex in their condition wait or timed wait requests.

System action: The request is rejected.

Symbolic Feedback Code: CEE5J2

CEE5731S The condition wait service or timed wait service failed due to a system detected error with error code *return-code* **and reason code** *reason-code***.**

Explanation: The condition wait internal service CEEOPCW or timed wait internal service CEEOPCT called the callable service BPX1CSE to set up for a condition wait. BPX1CSE returned an unexpected error code and reason code.

Programmer response: Report this failure to your system administrator.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5J3

CEE5732S The condition wait service failed due to a system detected error with error code *return-code* **and reason code** *reason-code***.**

Explanation: The condition wait internal service CEEOPCW called the callable service BPX1CWA to block a thread. BPX1CWA returned an unexpected error code and reason code.

Programmer response: Report this failure to your system administrator.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5J4

CEE5733S The value specified for number of seconds to wait was invalid. The condition wait request was rejected.

Explanation: The value for number of seconds to wait that was passed to the condition timed wait internal service CEEOPCT must be a non-negative number of seconds since midnight, January 1, 1970. This value cannot exceed 2,147,483,647 seconds.

Programmer response: Time to wait should be specified as current calendar in seconds since midnight, January 1, 1970. Be sure the service you are using to get current calendar time returns seconds since midnight, January 1, 1970, or that your program is correctly converting the value obtained.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5J5

CEE5734S The value specified for number of nanoseconds to wait was invalid. The condition wait request was rejected.

Explanation: The value for number of nanoseconds to wait that was passed to the condition timed wait internal service CEEOPCT must be a non-negative number that does not exceed 1,000,000,000 (1,000 million).

Programmer response: Be sure to initialize the nanosecond parameter to a value in the range 0 to 1,000,000,000 before invoking the condition wait service.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5J6

CEE5735S The value for current calendar time was invalid. The condition wait request was rejected.

Explanation: The system time of day (TOD) clock was not properly initialized or had overflowed. The value for current calendar time returned to the condition timed wait internal service CEEOPCT by the store clock (STCK) instruction was invalid.

Programmer response: Report this problem to your system administrator.

System action: The condition wait is rejected.

Symbolic Feedback Code: CEE5J7

CEE5736I The time to wait specified on a condition timed wait request has elapsed.

Explanation: Current calendar time in seconds since midnight, January 1, 1970, was equal to or greater than the time to wait. The time to wait was specified to the condition timed wait internal service CEEOPCT in seconds plus nanoseconds. Internal service CEEOPCT returned to allow the thread to continue processing.

Programmer response: Be sure you are specifying seconds to wait as current calendar time in seconds since midnight, January 1, 1970, plus some additional number of seconds.

System action: No system action is taken.

Symbolic Feedback Code: CEE5J8

CEE5737S The condition timed wait service failed due to a system error with error code *return-code* and reason code *reason-code*.

Explanation: The condition timed wait internal service CEEOPCT called the callable service CPX1CTW to block a thread. CPX1CTW returned an unexpected error code and reason code.

Programmer response: Report this failure to your system administrator.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5J9

CEE5738S The condition attribute object was already initialized.

Explanation: The condition attribute initialization internal service CEEOPDI was called to initialize a condition attribute object that had already been initialized.

Programmer response: Call the condition attribute destroy internal service CEEOPDD to destroy an initialized condition attribute object before calling internal service CEEOPDI to initializing it.

System action: The request is rejected.

Symbolic Feedback Code: CEE5JA

CEE5739S There was insufficient storage to initialize a condition attribute object.

Explanation: The condition attribute initialization internal service CEEOPDI was called to initialize a condition attribute object. However, there was insufficient system storage available to do so.

Programmer response: Additional system storage is required before attempting to initialize more condition attribute objects.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5JB

CEE5740S An addressing exception occurred referencing condition object or condition attribute object.

Explanation: The address of a condition object or condition attribute object passed as an parameter on a condition variable related service call was invalid. An addressing exception program interrupt occurred when the called service referenced this address.

Programmer response: Make sure the application correctly specifies the condition object or condition attribute object parameter in the service call.

System action: The application is terminated.

Symbolic Feedback Code: CEE5JC

CEE5741C The system condition attribute storage could not be freed.

Explanation: The destroy condition attribute internal service CEEOPDD was unable to free storage allocated for a condition attribute by the condition attribute initialization internal service CEEOPDI.

Programmer response: Check if the application might have written over system storage. Report this problem to your system administrator.

System action: Thread is terminated.

Symbolic Feedback Code: CEE5JD

CEE5742S The condition object was already initialized.

Explanation: The condition initialization internal service CEEOPCI was called to initialize a condition object that had already been initialized.

Programmer response: Call the condition destroy internal service CEEOPCD to destroy an initialized condition object before initializing it again.

System action: The request is rejected.

Symbolic Feedback Code: CEE5JE

CEE5743S The condition attribute object was not initialized.

Explanation: An uninitialized condition attribute object was specified as an parameter on a call to the condition initialization internal service CEEOPCI.

Programmer response: Call internal service CEEOPDI to initialize the condition attribute object before invoking internal service CEEOPCI.

System action: The request is rejected.

Symbolic Feedback Code: CEE5JF

CEE5744S There was insufficient storage to initialize a condition object.

Explanation: The condition initialization internal service CEEOPCI was called to initialize a condition object. However, there was insufficient system storage available to do so.

Programmer response: Get additional system storage before initializing more condition objects.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5JG

CEE5745C The system condition variable storage could not be freed.

Explanation: The destroy condition internal service CEEOPCD was unable to free storage allocated for a condition variable by the condition initialization internal service CEEOPCI.

Programmer response: Check if the application might have written over system storage. Report this problem to your system administrator.

System action: Thread is terminated.

Symbolic Feedback Code: CEE5JH

CEE5746S A condition attribute object had been changed since it was initialized.

Explanation: The condition attribute destroy internal service CEEOPDD detected that a condition attribute object (specified as a parameter) had changed since it was initialized by the condition attribute initialization internal service CEEOPDI. The condition attribute object was destroyed, but internal service CEEOPDD did not alter the storage associated with the condition attribute object. However, if internal service CEEOPDI was invoked, the storage was altered.

Programmer response: Check that the application is correctly reusing storage associated with the condition attribute object after initializing it.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5JI

CEE5747S The condition variable was busy.

Explanation: The condition destroy internal service CEEOPDD was invoked to destroy a condition variable that was in use. A condition variable that was in use by one or more threads for a condition wait or timed wait cannot be destroyed.

Programmer response: Retry the request or determine why the condition variable is in use.

System action: The request is rejected.

Symbolic Feedback Code: CEE5JJ

CEE5748S A condition object had been changed since it was initialized.

Explanation: The condition destroy service CEEOPCD detected that a condition object (specified as a parameter) was changed since it was initialized by the condition initialization internal service CEEOPCI. The condition variable was destroyed, but internal service CEEOPDD did not alter the storage associated with the condition object. However, if internal service CEEOPCI was invoked, the storage was altered.

Programmer response: Check that the application is correctly reusing storage associated with the condition object after initializing it.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5JK

CEE5749S An invalid attribute kind value was passed.

Explanation: The attribute kind parameter *setkind_np* that was passed to internal service CEEOPDS specified an invalid value. Valid values for *setkind_np* are NODEBUG (2).

Programmer response: Specify a valid attribute kind value.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5JL

CEE5750S An addressing exception occurred referencing attribute kind storage.

Explanation: The address where to return attribute kind that was passed as a parameter on a *getkind_np* service request, was invalid. An addressing exception occurred when the internal service CEEOPDG attempted to store the attribute kind value at this address.

Programmer response: Check that the application is correctly specifying the attribute kind address parameter on the *getkind_np* service call.

System action: The application is terminated.

Symbolic Feedback Code: CEE5JM

CEE5751C Invalid condition attribute storage detected.

Explanation: Internal service CEEOPDG found an invalid value in system storage for condition attribute *getkind_np*.

Programmer response: Check if the application might have written over system storage. Report this problem to your system administrator.

System action: Thread is terminated.

Symbolic Feedback Code: CEE5JN

CEE5761C Latch services were not available. The application will be terminated.

Explanation: A Language Environment function invoked internal Language Environment latch services when these services were not available. Latch services are initialized only when the POSIX(ON) run-time option was in effect.

Programmer response: Report problem to your system administrator.

System action: The application is terminated.

Symbolic Feedback Code: CEE5K1

CEE5762C The latch was already owned. The application will be terminated.

Explanation: A Language Environment function invoked internal Language Environment latch services to request a latch. The thread from which the request was made already holds the latch.

Programmer response: Report problem to your system administrator.

System action: The application is terminated.

Symbolic Feedback Code: CEE5K2

CEE5763C The latch was not owned. The application will be terminated.

Explanation: A Language Environment function invoked internal Language Environment latch services to release a latch. The thread from which the request was made did not hold the latch.

Programmer response: Report problem to your system administrator.

System action: The application is terminated.

Symbolic Feedback Code: CEE5K3

CEE5764S The lock object was not initialized.

Explanation: The mutex or read-write lock related service that was invoked requires the lock object specified as a parameter be initialized.

Programmer response: Use the internal service CEEOPMI to initialize the mutex or read-write lock object before invoking the service that failed.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5K4

CEE5765S The read-write lock was already held for writing.

Explanation: A thread invoked the read-write rdlock internal service CEEOPRL specifying a read-write lock that had already been locked by the thread for writing. A read-write lock can only be locked for reading multiple times by the same thread.

Programmer response: None

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5K5

CEE5766S An addressing exception occurred referencing a lock object or lock attribute object.

Explanation: The address of a mutex or read-write lock object, or mutex or read-write lock attribute object passed as a parameter on a lock related service call was invalid. An addressing exception program interrupt occurred when the called service referenced this address.

Programmer response: Specify the correct lock object or lock attribute object when passing parameters to the lock related service call.

System action: The application is terminated.

Symbolic Feedback Code: CEE5K6

CEE5767S An addressing exception occurred referencing system storage allocated for locks.

Explanation: An addressing exception program interrupt occurred when a mutex or read-write lock-related service referenced system storage allocated for locks.

Programmer response: Make sure that the application has not written over system storage before issuing the service call.

System action: The thread is terminated.

Symbolic Feedback Code: CEE5K7

CEE5768S A lock object has been changed since it was initialized.

Explanation: The mutex or read-write lock destroy internal service CEEOPMD detected that a lock object (specified as a parameter) had changed since it was initialized by the lock initialization internal service CEEOPMI. The lock was destroyed, but internal service CEEOPMD did not alter the storage associated with the read-write lock object. However, if internal service CEEOPMI was invoked, the storage was altered.

Programmer response: Make sure the application is not incorrectly reusing storage associated with the read-write lock object after initializing it.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5K8

CEE5769S The read-write lock was not held by thread.

Explanation: A thread called the read-write lock unlock internal service CEEOPRU to unlock the read-write lock but the read-write lock was not held by the thread. A thread acquired a read-write lock with one of the following read-write lock internal services:
- rdlock internal service CEEOPRL
- tryrdlock internal service CEEOPRT
- wrlock internal service CEEOPWL
- trywrlock internal service CEEOPWT

and was said to have held the lock.

Programmer response: Structure the application so that the thread that locks the read lock also unlocks the read lock.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5K9

CEE5770S The read-write lock was already held.

Explanation: A thread invoked the read-write wrlock internal service CEEOPWL specifying a read-write lock that had already been locked by the thread. A read-write lock can only be locked for reading multiple times by the same thread.

Programmer response: Structure the application so that the thread that locks the read-write lock also unlocks the read-write lock.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5KA

CEE5771S The read-write lock was already initialized.

Explanation: The read-write lock initialization internal service CEEOPMI was called to initialize a read-write lock object that had already been initialized.

Programmer response: Call the real-write lock destroy internal service CEEOPMD to destroy an initialized read-write lock object before initializing it again.

System action: The request is rejected.

Symbolic Feedback Code: CEE5KC

CEE5772S The lock attribute object was not initialized.

Explanation: The mutex or read-write lock attribute related services required that the lock attribute object specified as a parameter be initialized.

Programmer response: Use internal service CEEOPXI to initialize the mutex or read-write lock attribute object before invoking the service that failed.

System action: No system action is taken.

Symbolic Feedback Code: CEE5KC

CEE5773S There was insufficient storage to initialize a read-write lock object.

Explanation: The read-write lock initialization internal service CEEOPMI was called to initialize a read-write lock object. However, there was insufficient system storage available.

Programmer response: Get additional system storage before initializing more read-write lock objects.

System action: The request is rejected.

Symbolic Feedback Code: CEE5KD

CEE5774S A read-write lock attribute object has been changed since it was initialized.

Explanation: The read-write lock attribute destroy internal service, CEEOPXD, detected that a read-write lock attribute object specified as a parameter was changed since it was initialized by the read-write lock attribute initialization internal service, CEEOPXI. The read-write lock attribute object was destroyed, but internal service, CEEOPXD, did not alter the storage associated with the read-write lock attribute object. However, if internal service, CEEOPXI, was invoked, the storage was altered.

Programmer response: Make sure the application is not incorrectly reusing storage associated with the read-write lock attribute object after initializing it.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5KE

CEE5775S The read-write lock attribute object was already initialized.

Explanation: The read-write lock attribute initialization internal service, CEEOPXI, was called to initialize a read-write lock attribute object that had already been initialized.

Programmer response: Call the read-write lock attribute destroy internal service, CEEOPXD, to destroy an initialized read-write lock attribute object before initializing it again.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5KF

CEE5776S There was insufficient storage to initialize a read-write lock attribute object.

Explanation: The read-write lock attribute initialization internal service, CEEOPXI, was called to initialize a read-write lock attribute object, however, there was insufficient system storage available.

Programmer response: Acquire additional system storage before initializing more read-write lock attribute objects.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5KG

CEE5777S The lock was busy.

Explanation: The mutex or read-write lock destroy internal service CEEOPMD was invoked to destroy a lock that was in use. A mutex or read-write lock that is locked, or a mutex that is associated with a condition wait or timed wait, cannot be destroyed.

Programmer response: Verify that no other thread holds the lock before calling CEEOPMD to destroy the lock.

System action: The request is rejected.

Symbolic Feedback Code: CEE5KH

CEE5778S An addressing exception occurred while referencing attribute return storage.

Explanation: The address where to return attribute information that was passed as a parameter on a call to the mutex and read-write lock attribute internal service, CEEOPXG, was invalid. An addressing exception occurred when the CEEOPXG internal service attempted to store the attribute value at this return address.

Programmer response: Specify the correct attribute return address parameter on the CEEOPXG internal service call.

System action: The application is terminated.

Symbolic Feedback Code: CEE5KI

CEE5779S System lock storage could not be freed.

Explanation: The destroy mutex and read-write lock internal service, CEEOPMD, was unable to free storage allocated for a mutex or read-write lock by the lock initialization service, CEEOPMI.

Programmer response: Check if the application might have written over system storage. Report this problem to the storage administrator.

System action: The thread is terminated.

Symbolic Feedback Code: CEE5KJ

CEE5780S System lock attribute storage could not be freed.

Explanation: The destroy lock attribute internal service, CEEOPXD, was unable to free storage allocated for a mutex or read-write lock attribute by the lock attribute initialization service, CEEOPXI.

Programmer response: Check if the application might have written over system storage. Report this problem to the system administrator.

System action: The thread is terminated.

Symbolic Feedback Code: CEE5KK

CEE5781S There was invalid lock attribute storage.

Explanation: The mutex and read-write lock attribute internal service, CEEOPXG, found an invalid value in system storage for lock attributes of the specified type.

Programmer response: Check if the application might have written over system storage. Report this problem to the system administrator.

System action: The thread is terminated.

Symbolic Feedback Code: CEE5KL

CEE5782S There was an invalid lock attribute value.

Explanation: The attribute parameter in a call to internal service, CEEOPXS, specified an invalid attribute value for lock attributes of the specified type. Valid values for type *setkind_np* or *settype are*:
- NONRECURSIVE + DEBUG + ERRORCHECK (0)
- RECURSIVE + DEBUG + ERRORCHECK (1)
- NONRECURSIVE + NODEBUG + ERRORCHECK (2)
- RECURSIVE + NODEBUG + ERRORCHECK (3)
- NONRECURSIVE + DEBUG + NOERRORCHECK (4)
- RECURSIVE + DEBUG + NOERRORCHECK (5)
- NONRECURSIVE + NODEBUG + NOERRORCHECK (6)
- RECURSIVE + NODEBUG + NOERRORCHECK (7)

Valid values for type *setpshared* are:

- PRIVATE (0)
- SHARED (8)

Programmer response: Specify a correct attribute value.

System action: Unless the condition is handled, the default action it to terminate the enclave.

Symbolic Feedback Code: CEE5KM

CEE5783C A thread waiting for a read-write lock was forced to terminate.

Explanation: An event, such as the initial thread terminating, forced all threads to terminate including threads waiting for a read-write lock.

Programmer response: Check that all threads exit correctly.

System action: The thread is terminated.

Symbolic Feedback Code: CEE5KN

CEE5784I There was insufficient resource to initialize another read-write lock.

Explanation: The read-write lock init internal service, CEEOPMI, was invoked to initialize a read-write lock, but not enough resource was available to initialize another read-write lock.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE5KO

CEE5785I There as insufficient privilege to initialize the read-write lock.

Explanation: The read-write lock init internal service, CEEOPMI, was invoked to initialize a read-write lock, but not enough privilege was available to initialize the read-write lock.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE5KP

CEE5786S The z/OS UNIX callable service, BPX1SLK, failed during shared lock processing. The system return code was *return_code*, the reason code was *reason_code*. X'00'

Explanation: The z/OS UNIX callable service, BPX1SLK, was called by a Language Environment internal service for shared mutex or read-write lock processing. BPX1SLK returned without performing the specified shared lock processing (initialize, lock, unlock, or destroy).

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your z/OS UNIX system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5KQ

CEE5787I There as insufficient resource to obtain the read-write lock.

Explanation: The read-write lock rdlock or tryrdlock internal services, CEEOPRL, or CEEOPRT was invoked to lock a read-write lock, but not enough resource was available to obtain this read-write lock another time for read.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE5KR

CEE5788I The read-write lock was busy.

Explanation: The read-write lock tryrdlock internal service, CEEOPRT, was invoked to lock a read-write lock for read that was already locked for write or had an outstanding write lock request.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE5KS

CEE5789I The read-write lock was busy.

Explanation: The read-write lock trywrlock internal service, CEEOPWT, was invoked to lock a read-write lock for write that was already locked for read or write.

Programmer response: None.

System action: No system action is taken.

Symbolic Feedback Code: CEE5KT

CEE5790S There was insufficient storage to lock a read-write lock object.

Explanation: A thread attempted to lock a read-write lock by calling one of the following read-write lock internal services:
- rdlock internal service, CEEOPRL
- tryrdlock internal service, CEEOPRT
- wrlock internal service, CEEOPWL
- trywrlock internal service, CEEOPWT

However, there was insufficient system storage available.

Programmer response: Acquire additional system storage before attempting to lock more read-write lock objects.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5KU

CEE5791C System read-write lock storage could not be freed.

Explanation: The read-write lock unlock internal service, CEEOPRU was unable to free storage allocated for a read-write lock by one of the following read-write lock internal services:
- rdlock internal service, CEEOPRL
- tryrdlock internal service, CEEOPRT
- wrlock internal service, CEEOPWL
- trywrlock internal service, CEEOPWT

Programmer response: Check if the application might have written over system storage. Report this problem to the storage administrator.

System action: The thread is terminated.

Symbolic Feedback Code: CEE5KV

CEE5792I There was insufficient resource to initialize another condition object.

Explanation: The condition object init internal service CEEOPCI was invoked to initialize a condition object, but not enough resource was available to initialize another condition object.

Programmer response: None

System action: No system action is taken.

Symbolic Feedback Code: CEE5L0

CEE5793I There was insufficient privilege to initialize the condition object.

Explanation: The condition object init internal service CEEOPCI was invoked to initialize a condition object, but not enough privilege was available to initialize the condition object.

Programmer response: None

System action: No system action is taken.

Symbolic Feedback Code: CEE5L1

CEE5794S The condition attribute object was not initialized.

Explanation: The condition wait related services required that the condition attribute object specified as a parameter be initialized.

Programmer response: Use internal service CEEOPDI to initialize the condition attribute object before invoking the service that failed.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5L2

CEE5795S The callable service BPX1SMC failed during shared condition variable processing. The system return code was *return_code* **the reason code was** *reason_code***.**

Explanation: The callable service BPX1SMC was called by a Language Environment internal service for shared condition variable processing. BPX1SMC returned without performing the specified shared condition variable processing (initialize, wait, signal, or destroy).

Programmer response: See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* or *z/VM: OpenExtensions Callable Services Reference* for the appropriate action to take for this return code and reason code. Consult with your system support personnel if necessary.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5L3

CEE5796S The callable service BPX1SMC failed during shared lock processing. The system return code was *return_code* **the reason code was** *reason_code***.**

Explanation: The callable service BPX1SMC was called by a Language Environment internal service for shared mutex lock processing. BPX1SMC returned without performing the specified shared lock processing (initialize, lock, unlock, or destroy).

Programmer response: Use internal service CEEOPDI to initialize the condition attribute object before invoking the service that failed.

System action: Unless the condition is handled, the default action is to terminate the enclave.

Symbolic Feedback Code: CEE5L4

Chapter 2. C Prelinker and the C Object Library Utility Messages

This topic provides message information for the prelinker and object library utilities.

A return code is generated to indicate the degree of prelinking success. The return codes are as follows:

0	No error detected; processing completed; successful execution anticipated.
4	Possible error (warning) detected; processing completed; successful execution probable.
8	Error detected; processing might have been completed; successful execution impossible.
12	Severe error detected; processing terminated abnormally; successful execution impossible.

The messages issued by the prelinker and object library utility have the following format:

EDCnnnns text <&s>

where

nnnn Error message number

s Error severity

I	Informational message
W	Warning message
E	Error message
S	Severe error message

&s Substitution variable, such as &1

Note: For C messages less than 4000, see *z/OS XL C/C++ Compiler and Run-Time Migration Guide for the Application Programmer*.

The prelinker and object library utility can return messages that are listed in this topic.

EDC4000S Unable to open ***&1*****.**

Explanation: An error was encountered during a file open.

Programmer response: Make sure that the named file has the proper DCB requirements (for example, RECFM, BLKSIZE, LRECL).

System action: Processing terminates.

EDC4001S Unable to read ***&1*****.**

Explanation: An error was encountered during a file read.

Programmer response: Make sure that the named file has the proper DCB requirements (for example, RECFM, BLKSIZE, LRECL). If the file has been corrupted recreate it, then recompile and run the new file.

System action: Processing terminates.

EDC4002S Unable to write to *&1*.

Explanation: An error was encountered during a file write.

Programmer response: Make sure that the named file has the proper DCB requirements (for example, RECFM, BLKSIZE, LRECL). Also make sure that sufficient write space is available.

Programmer response: Ensure that sufficient disk space is available.

System action: Processing terminates.

EDC4004W Invalid options: *&1*.

Explanation: The listed prelinker options were invalid.

Programmer response: Enter the list of valid options.

System action: The invalid prelinker options are ignored and processing continues.

EDC4005E No input decks were specified.

Explanation: No input decks were specified for the prelink.

Programmer response: Input at least one deck.

System action: The prelinker will process nothing.

EDC4006S Object deck was missing TXT cards.

Explanation: A corrupted input deck was encountered during the prelink.

Programmer response: Recompile the source routine and run it again. If the problem persists, call your IBM service representative.

System action: Processing terminates.

EDC4007S Object deck had multiple initialized CSECTs.

Explanation: A corrupted input deck was encountered during the prelink.

Programmer response: Recompile the source routine and run it again. If the problem persists, call your IBM service representative.

System action: Processing terminates.

EDC4008S Invalid initialization deck (RLDs span cards).

Explanation: A corrupted input deck was encountered during the prelink.

Programmer response: Recompile the source routine and run it again. If the problem persists, call your IBM service representative.

System action: Processing terminates.

EDC4009S Invalid initialization deck (RLDs and TXTs not in sync).

Explanation: A corrupted input deck was encountered during the prelink.

Programmer response: Recompile the source routine and run it again. If the problem persists, call your IBM service representative.

System action: Processing terminates.

EDC4010W A zero length static object was found in assembler deck.

Explanation: A zero length static object was encountered in an assembler deck.

Programmer response: Redefine the object with the appropriate size.

System action: Incorrect or undefined execution could result.

EDC4011E Unresolved writable static references were detected.

Explanation: Undefined writable static objects were encountered at prelink termination.

Programmer response: Prelink with the MAP option to find the objects in question and include these objects with the prelink step.

System action: Prelinker produces object module with unresolved writable static references. Use of this object module could result in incorrect or undefined execution.

EDC4012E No input decks were found.

Explanation: Input decks were not found for the prelink.

System action: The prelinker will process nothing.

EDC4013I No map displayed as no writable static was found.

Explanation: No writable static objects were found during the prelink.

System action: If MAP option is specified, no static map is produced because no writable static objects were found.

EDC4014E Undefined writable static objects were detected:

Explanation: The listed writable static objects were undefined at prelink termination.

Programmer response: Include these objects during the prelink.

System action: Incorrect or undefined execution could result.

EDC4015W Unresolved references were detected:

Explanation: The listed objects were unresolved at prelink termination. Unresolved C library objects are not required for the prelink step, but should be resolved during the link-edit step. Unresolved writable static objects or unresolved objects referring to writable static objects are required for prelink.

Programmer response: To correct the latter, include these objects during the prelink.

System action: Prelinker only resolves writable static objects or unresolved objects referring to writable static objects.

EDC4016W Duplicate objects were detected.

Explanation: The listed objects were defined multiple times.

Programmer response: Define the objects consistently.

System action: Incorrect execution could occur unless the objects are defined consistently.

EDC4017W Duplicate object *&1* was defined with different sizes.

Explanation: An object had been defined multiple times with different sizes. The larger of the different sizes was taken. Incorrect execution could occur unless the object is defined consistently.

Programmer response: Define the object consistently.

System action: The largest size is used.

EDC4018E No member name specified and NAME card not found.

Explanation: For the ADD function, you must specify a member name, or a NAME card indicating the member name must be present in the object module.

Programmer response: Repeat the step specifying an appropriate member name.

System action: Processing terminates.

EDC4019S Invalid or missing XSD cards.

Explanation: A corrupted input deck was encountered during processing.

Programmer response: Recompile your source routine and repeat the step. If the problem persists, call your IBM service representative.

System action: Processing terminates.

EDC4020W Continuation card missing for *&1* control card.

Explanation: A control card of type *&1* was encountered with the continuation column set, but there was no next card or the next card was not a valid continuation card.

Programmer response: Add the appropriate continuation card or set continuation column 72 to blank if no continuation card is required.

System action: The card is ignored and processing continues.

EDC4021W Invalid syntax specified on *&1* control card.

Explanation: A control card with invalid syntax was encountered during processing.

Programmer response: If the card is required, correct the syntax errors and repeat the step. If the card is not required, the warning message can be removed by deleting the invalid card.

System action: If the card was an INCLUDE card, the card is processed up to the syntax error and the remainder of the card is ignored. If the card was not an INCLUDE card, the card is ignored. In either case, processing continues.

EDC4022W More than one *&1* card found in *&2*.

Explanation: More than one control card of type *&1* was encountered during the processing of *&2*.

Programmer response: No recovery is necessary unless the incorrect card was chosen or incorrect processing was performed. In this case, remove the offending card and repeat the step.

System action: If the card is a NAME card and this was encountered during the prelink step, the last NAME card is used and processing continues.

If the card is a NAME card and this was encountered during the C370LIB ADD or GEN steps, all NAME cards for *&2* are ignored and processing continues.

EDC4023W Continuation cards not allowed for *&1* card. Card ignored.

Explanation: A control card of type *&1* was found to be expecting a continuation card. Information for a card of this type must be specified on one card.

Programmer response: Correct the card if necessary, set continuation column 72 to blank, and repeat the step.

System action: The card is ignored and processing continues.

EDC4024W RENAME card cannot be used for short name *&1*.

Explanation: A RENAME card was encountered that attempted to rename a short name to another name. RENAME cards are valid only for long names for which there is no corresponding short name.

Programmer response: The warning message can be removed by deleting the invalid RENAME card.

System action: The card is ignored and processing continues.

EDC4025W Multiple RENAME cards found for *&1*. First valid one taken.

Explanation: More than one RENAME card was encountered for the name *&1*.

Programmer response: The prelinker map shows which output name was chosen. If this was not the intended name, remove the duplicate RENAME card(s) and repeat the step.

System action: The first RENAME card with a valid output name is chosen.

EDC4026W May not RENAME long name *&1* to another long name *&2*.

Explanation: A RENAME card had been encountered that attempted to rename a long name to another long name.

Programmer response: The prelinker map shows which output name was chosen. If this was not the intended name, replace the invalid RENAME card with a valid output name and repeat the step. To remove the warning message, delete the invalid RENAME card.

System action: The card is ignored and processing continues.

EDC4027W May not RENAME defined long name *&1* to defined name *&2*.

Explanation: A RENAME card had been encountered that attempted to rename the defined long name *&1* to another defined name *&2*.

Programmer response: The prelinker map shows which output name was chosen. If this was not the intended name, replace the invalid RENAME card with a valid output name and repeat the step. To remove the warning message, delete the invalid RENAME card.

System action: The card is ignored and processing continues.

EDC4028W RENAME card of *&1* to *&2* ignored since *&2* is target of another RENAME.

Explanation: Multiple RENAME cards had been encountered attempting to rename two different names the same name *&2*.

Programmer response: The prelinker map shows which name was renamed to *&2*. If the output name for *&2* was not the intended name, change the name and repeat the step. To remove the warning message, delete the extra RENAME card(s).

System action: The first valid RENAME card for *&2* is chosen.

EDC4029W *&1* and *&2* mapped to same name (*&3*) due to UPCASE option.

Explanation: A name (*&1*) that was made uppercase because of the UPCASE option collided with the output name (*&3*) of another name (*&2*).

Programmer response: If both names (*&1* and *&2*) correspond to the same object the warning can be ignored. If the names do not correspond to the same object or if the warning is to be removed, do one of the following:
- Use a RENAME card to rename one of the names to something other than *&3*.
- Change one of the names in the source routine.
- Use `#pragma` map in the source routine on one of the names.
- Do not run the step with the UPCASE option.

System action: Both names (*&1* and *&2*) are mapped to *&3*.

EDC4030E Missing command operands.

Explanation: One or more operands were missing on the invocation of object library utility command.

Programmer response: Add the proper operands and repeat the step.

System action: Processing terminates.

EDC4031W File *&1* not found.

Explanation: The specified file could not be located to perform the command.

Programmer response: Try the command again, specifying the appropriate file.

System action: If possible, processing continues ignoring the particular file.

EDC4032E Error with *&1* command. Return code=*&2*.

Explanation: In order to perform the library command, the system command *&1* was issued and resulted in a return code of *&2*.

Programmer response: Diagnose the problem using the return code and any messages generated.

System action: Processing terminates.

EDC4033E Invalid C370LIB-directory encountered in library *&1*.

Explanation: An invalid or corrupted C370LIB-directory had been encountered.

Programmer response: Use the C370LIB DIR command to recreate the C370LIB-directory and repeat the step.

System action: Processing terminates.

EDC4034E Library *&1* did not contain a C370LIB-directory.

Explanation: The library *&1* did not contain a C370LIB-directory necessary to perform the command.

Programmer response: The library was not created with the C370LIB command. Use the C370LIB DIR command to create the C370LIB-directory and repeat the step.

System action: Processing terminates.

EDC4035W Member *&1* not found in library *&2*.

Explanation: The specified member *&1* was not found in the library.

Programmer response: Use the C370LIB MAP command to display the names of library members.

System action: Processing continues.

EDC4036E Invalid command operands: *&1*.

Explanation: Invalid operands were specified on the invocation of this command.

Programmer response: Specify the correct operands and repeat the step.

System action: Processing terminates.

EDC4037E File *&1* had invalid format.

Explanation: The specified file, *&1*, did not have the proper format. The file should be fixed-format with a record length of 80.

Programmer response: Correct the file or file specification and repeat the step.

System action: The file is ignored and processing continues.

EDC4038E Library *&1* not found.

Explanation: The library, *&1*, could not be found to perform the command.

Programmer response: Try the command again, specifying the correct library.

System action: Processing terminates.

EDC4039E Library *&1* had invalid format.

Explanation: The specified library, *&1*, did not have the proper format. The library must contain object modules as members and be fixed-format with a record length of 80.

Programmer response: Correct the library or library specification and repeat the step.

System action: Processing terminates.

EDC4042S Virtual storage exceeded.

Explanation: The utility ran out of memory. This sometimes happens with large files or routines with large functions. Very large routines limit the optimization that can be done.

Programmer response: Divide the file into several smaller sections or shorten the function.

System action: Processing terminates.

EDC4043E Invalid symbol table encountered in archive library *&1*.

Explanation: The archive library from the MVS system had invalid information in its symbol table.

Programmer response: Rebuild the archive library.

System action: The invalid archive library is ignored and processing continues.

EDC4044E Archive library *&1* did not contain a symbol table.

Explanation: The symbol table for the archive library from the MVS system could not be found.

Programmer response: Rebuild the archive library.

System action: The invalid archive library is ignored and processing continues.

EDC4045E Archive library *&1* not found.

Explanation: The archive library could not be found.

Programmer response: Try the command again specifying the appropriate file.

System action: The invalid archive library is ignored and processing continues.

EDC4046E Archive library *&1* had invalid format.

Explanation: The file was found but did not have the correct information to be recognized as an archive library.

Programmer response: Rebuild the archive library.

System action: The invalid archive library is ignored and processing continues.

EDC4048W Card *&1* of *&2* is invalid.

Explanation: The card shown is not valid.

Programmer response: Prelink the DLL and generate a new, uncorrupted definition side-deck.

EDC4049E Unresolved references could not be imported.

Explanation: The same symbol was referenced in both DLL and non-DLL code. The DLL reference could have been satisfied by an IMPORT control statement which was processed, but the non-DLL reference could not.

Programmer response: You must either supply a definition for the referenced symbol during the prelink step or recompile the code containing the non-DLL reference with the `DLL` compiler option so that it becomes a DLL reference.

EDC4050W Card *&1* of *&2* is not the continuation of an IMPORT control statement.

Explanation: The object deck of `IMPORT` control statements is corrupted.

Programmer response: Prelink the exporting DLL again to generate a new object deck of control statements or get a new copy from the DLL provider.

EDC4051W Duplicate IMPORT definitions are detected.

Explanation: A name referenced in DLL code was not defined within the application but more than one IMPORT control statement was seen with that symbol name. The first one seen by the prelinker was used.

Programmer response: Check the Import Symbol Map section of the Prelinker Map to see if you are importing the symbol from the correct DLL. If not, change the input order of the IMPORT control statements.

EDC4052W Module name *&1* chosen for generated IMPORT control statements.

Explanation: The prelinker has assigned the default name TEMPNAME to the module in the Definition Side-Deck.

Programmer response: Include a NAME control statement in the prelinker input or specify the output object module of the prelinker to be a PDS member so that the prelinker will use that as the module name in generated IMPORT control statements.

EDC4057E Truncated duplicate #pragma mapped objects are detected.

Explanation: The prelinker has detected duplicate names which were longer than 8 characters in length but have been #pragma mapped and truncated; therefore, cannot be renamed by the prelinker. The duplicate names could have also been mapped for other reasons such as a CSECT compiler option or a #pragma csect statement.

Programmer response: Modify the duplicate names listed in the message to avoid the conflict.

EDC4058E Truncated duplicate #pragma mapped objects are detected.

Explanation: The prelinker has detected duplicate names which were longer than 8 characters in length but have been #pragma mapped and truncated; therefore, cannot be renamed by the prelinker. The duplicate names could have also been mapped for other reasons such as a CSECT compiler option or a #pragma csect statement.

Programmer response: Ensure the MAP option is in effect. The truncated #pragma mapped duplicates will be listed in the prelinker MAP if the MAP option is in effect. Once the truncated #pragma mapped duplicates are identified, modify the duplicate names to avoid the conflict.

Severe Error Messages

The following error messages are produced by the prelinker or the Object Library Utility if the message file is itself invalid.

EDC0090 Unable to open message file *&1*.

EDC0091 Invalid offset table in message file *&1*.

EDC0092 Message component *&1* is not found.

EDC0093 Message file *&1* corrupted.

EDC0094 Integrity check failure on msg *&1*.

EDC0095 **Bad substitution number in message** *&1*.

EDC0096 **Virtual storage exceeded.**

Chapter 3. C Utility Messages

This topic provides message information for the `localedef` utility, the `iconv` utility, and the `genxlt` utility. See *z/OS XL C/C++ Messages* for messages for the DSECT and CXXFILT utilities.

localedef Messages

This topic contains the `localedef` messages.

Return Codes

The `localedef` utility returns the following return codes:

0 No errors were detected and the locales were generated successfully .

4 Warning messages were issued and the locales were generated successfully, or error messages were issued but the BLDERR option was specified.

4 Warning or errors were detected and the locale was not generated.

Messages

The messages issued by the `localedef` utility have the following format.

Message Format: EDCnnnn ss text <%n$x>

nnnn Error message number

ss Error severity

10	Warning message
30	Error message
40	Severe error

%n$x Substitution variable

%	The start of the substitution variable
n	The number that represents the line position of the variable
$	A delimiter
x	The kind of variable (d=decimal, c=character, s=string)

Warning messages will be issued when the FLAG(E) option is not specified. The default FLAG(W), will produce warnings. When warning messages (severity of 10) are issued, the minimum return code from the `localedef` utility is 4.

When error messages (severity of 30) are issued, the locale is built only if the BLDERR option is specified. The default is NOBLDERR. If BLDERR is specified, the return code would be set to a minimum of 4. If error messages are issued and NOBLDERR was specified (or by default), the return code will be set to a minimum of 8.

When severe error messages (severity of 40) are issued, the locale is not built and the return code from the `localedef` utility is 12.

The messages that can be issued are as follows:

EDC4100 40 The symbol ’%1$s’is not the correct type.

Explanation: This message is issued while processing a locale definition file containing a reference to a symbol that does not have expected type. This error occurs commonly when the LC_CTYPE keywords are used as character references in any locale definition file category.

Programmer response: Use a symbolic name instead of a character reference.

System action: The locale has not been created.

EDC4101 40 Could not open ’%1$s’ for read.

Explanation: This message is issued if the open for read failed for any file required by the `localedef` utility.

Programmer response: Verify the file name is correct and the file/data set exists.

System action: The locale has not been created.

EDC4102 40 Internal error in file %1$s on line %2$d.

Explanation: An internal error had occurred in the `localedef` utility.

Programmer response: Examine the locale definition and charmap files for possible errors. Report error to IBM.

System action: The locale has not been created.

EDC4103 40 Syntax Error: expected %1$d arguments and received %2$d arguments.

Explanation: This message is issued in the locale definition file when a keyword was expecting a fixed number of arguments and not enough arguments were supplied.

Programmer response: Add the missing arguments to the keyword in the locale definition file.

System action: The locale has not been created.

EDC4104 40 Illegal limit in range specification.

Explanation: An error had occurred in a range in the LC_CTYPE category of the locale definition file. The locale was not created.

Programmer response: Examine the locale definition file for possible errors.

System action: The locale has not been created.

EDC4105 40 Memory allocation failure on line %1$d in module %2$s.

Explanation: The `localedef` utility was unable to allocate memory.

Programmer response: Under MVS and TSO, increase region size and rerun the `localedef` utility. Under CMS, increase the virtual machine size and rerun the `localedef` utility.

System action: The locale has not been created.

EDC4106 40 Could not open file ’%1$s’ for write.

Explanation: This message is issued if the open for write failed when the `localedef` utility attempted to generate the C program. The file name passed to `fopen()` is included in the message.

Programmer response: Under CMS, verify the A-Disk exists and is in WRITE mode.

System action: The locale has not been created.

EDC4107 40 The '%1$s' character is longer than <mb_cur_max>.

Explanation: The length of value assigned to the specified symbol in the charmap file must not be bigger than the value assigned to *mb_cur_max*. *mb_cur_max* defaults to 1 and can only have the values 1 or 4 for EBCDIC locales and 1, 2 or 3 for ASCII locales. If multibyte characters are required then the value of *mb_cur_max* must also include the shift_in and shift_out characters even though the shift_in and shift_out characters are not entered into the charmap file as part of a character definition.

Programmer response: Increase the size of *mb_cur_max* or remove the extra byte in multibyte sequence assigned to the symbol specified.

System action: The locale has not been created.

EDC4108 10 The '%1$s' symbolic name was undefined and has been ignored.

Explanation: The specified symbolic name used in the locale definition file was not defined in the charmap file. When a symbolic name that is not defined is used in the LC_CTYPE or LC_COLLATE categories, the warning is issued.

Programmer response: Define the specified symbol name in the charmap file.

System action: The character has been ignored and the locale has been created.

EDC4109 40 The '%1$s' symbolic name was undefined.

Explanation: The specified symbolic name used in the locale definition file was not defined in the charmap file. When a symbolic name that is not defined is used in categories other than LC_CTYPE or LC_COLLATE, an error message is issued.

Programmer response: Define the specified symbol name in the charmap file.

System action: The locale has not been created.

EDC4110 40 The start of the range, '%1$s', must be numerically less than the end of the range, '%2$s'.

Explanation: In the collation section of the locale definition file, the start range codepoint specified must be less than the end range codepoint specified. These codepoints were assigned values in the charmap file where the codepoints can be assigned in any order.

Programmer response: Change the collation range codepoints in the locale definition file so that the start of the range is less than the end of the range.

System action: The locale has not been created.

EDC4111 40 The symbol range containing %1$s and %2$s was incorrectly formatted.

Explanation: The symbolic names used in range definition in the charmap file should consist of zero or more nonnumeric characters, followed by an integer formed by one or more decimal digits. The characters preceding the integer should be identical in the two symbolic names, and the integer formed by the digits in the second name should be equal to or greater than the integer formed by the digits in the first name. This is interpreted as a series of symbolic names formed from the common part and each of the integers between the first and second integer, inclusive.

In the following example, the first line is valid as both names have the same prefix, followed by four digits, whereas the second example has a different prefix for the first and second name, and is invalid.

```
<ab0101>...<ab0120>  \x42\xc1
<abc0101>...<ab0120> \x42\xc1
```

Programmer response: Check the specified symbolic names to ensure compliance to the above rules.

System action: The locale has not been created.

EDC4112 40 Illegal character reference or escape sequence in '%1$s'.

Explanation: A character reference or escape sequence had been defined that was not legal.

Programmer response: Make the character reference or escape sequence legal.

System action: The locale has not been created.

EDC4113 30 The symbolic name '%1$s', had already been specified.

Explanation: The specified symbolic name in the charmap file had already been specified. A symbolic name should only be defined once.

Programmer response: Remove the duplicate symbolic name from the charmap file.

System action: The locale has not been created.

EDC4114 10 There are characters in the codeset which were unspecified in the collation order.

Explanation: There were characters defined in the charmap file that were not used in the collation category of the locale definition file. The locale was still created. The characters were added at the end of the collation sequence.

Programmer response: If required, add the missing characters from the charmap file to the collation category of the locale definition file.

System action: The locale has been created and the characters were added at the end of the collation sequence.

EDC4115 30 Illegal decimal constant '%1$s'.

Explanation: The decimal constant of type '\dnnn' specified in the charmap file was greater than decimal 255.

Programmer response: Change the decimal constant in the charmap file to a value less than or equal to 255.

System action: The locale has not been created.

EDC4116 30 Illegal octal constant '%1$s'.

Explanation: The octal constant of type '\nnn' specified in the charmap file was greater than octal 377.

Programmer response: Change the octal constant in the charmap file to a value less than or equal to octal 377.

System action: The locale has not been created.

EDC4117 30 Illegal hexadecimal constant '%1$s'.

Explanation: The hexadecimal constant of type '\xnn' specified in the charmap file was greater than hexadecimal FF.

Programmer response: Change the hexadecimal constant in the charmap file to a value less than or equal to hexadecimal FF.

System action: The locale has not been created.

EDC4118 30 Missing closing quote in string '%1$s'.

Explanation: The string specified had a opening double quote but no closing double quote. The closing quote will be added.

Programmer response: Add the closing double quote after the string.

System action: The locale has not been created. If BLDERR option is specified, the characters between the opening double quote and the end of line character will be used.

EDC4119 30 Illegal character, '%1$c', in input file.

Explanation: An illegal character had been found in the charmap or locale definition file.

Programmer response: Remove the character.

System action: The locale has not been created. If BLDERR option is specified, the character is ignored.

EDC4120 30 The character for '%1$s' statement is missing. Statement is ignored.

Explanation: When defining the escape character or comment character in the charmap or locale definition file, a character was not supplied.

Programmer response: Insert a character to be defined as the escape character or comment character in the charmap or locale definition file.

System action: The statement was ignored and the escape character or comment character was not changed. The locale has not been created. If BLDERR option is specified, the default comment or escape character is used.

EDC4121 30 '%1$c' is not a POSIX Portable Character. Character is ignored.

Explanation: When defining escape_char or comment_char in the charmap or locale definition file, the character was less than space.

Programmer response: Define the escape_char or comment_char in the charmap or locale definition file with a character greater than space.

System action: The statement was ignored and the escape character or comment character was not changed. The locale has not been created. If BLDERR option is specified, the default comment or escape character is used.

EDC4122 30 The character symbol '%1$s' is missing the closing '>'. The '>' is added.

Explanation: The character symbol specified had a less than sign at the beginning of the symbol but no closing greater than sign. The symbol was accepted.

Programmer response: Add the greater than sign after the symbol.

System action: The locale has not been created. If BLDERR option is specified, the characters between the open '<' and the end of line character is used.

EDC4123 30 Unrecognized keyword, '%1$s', statement is ignored.

Explanation: When a dot is not used in a string or as part as of an ellipses (...), the keyword is unrecognized, the statement is ignored.

Programmer response: Remove the dot which is part of the unrecognized keyword or add the missing dots to make up ellipses (...).

System action: The locale has not been created.

EDC4124 40 The encoding specified for the '%1$s' character is unsupported.

Explanation: The multibyte character was not valid, contains a shift out without a corresponding shift in or a shift in character without a corresponding shift out. The locale was not created.

Programmer response: If the string contains unmatched shift in or shift out characters, remove them.

System action: The locale has not been created.

EDC4125 30 The character, '%1$s', has already been assigned a weight. Specification is ignored.

Explanation: The specified character or symbolic name in the collation category of the locale definition file, had already been defined.

Programmer response: Remove the duplicate character or symbolic name for the collation category.

System action: The locale has not been created. If BLDERR option is specified, the second definition is ignored.

EDC4126 30 A character in range ’%1$s...%2$s’ already had a collation weight. Range is ignored.

Explanation: A character or symbolic name in the specified range in the collation category of the locale definition file, had already been defined in the collation category.

Programmer response: Remove the duplicate character or adjust the range so as not to cover duplicate characters.

System action: The locale has not been created. If BLDERR option is specified, the second definition is ignored.

EDC4127 10 No toupper section defined for this locale source file.

Explanation: The toupper keyword in the LC_CTYPE category in the locale definition file was not specified. The lowercase character ’a’ to ’z’ are mapped to the characters ’A’ to ’Z’.

Programmer response: Add the lowercase characters ’a’ to ’z’ and ’A’ to ’Z’ to the toupper section of the LC_CTYPE category in the locale definition file.

System action: The locale has been created.

EDC4128 10 The use of the ’...’ keyword assumed that the codeset was contiguous between the two range endpoints specified.

Explanation: This warning is always produced when ellipses (...) are used in defining collation sequences in the locale definition file because the locale may not be portable whenever ellipses are used.

Programmer response: Instead of using ellipses, insert all the symbol names between the two range endpoints.

System action: The locale is still created.

EDC4129 30 The symbolic name, ’%1$s’, referenced had not yet been specified in the collation order.

Explanation: Collation weights in the locale definition file must use symbolic names that have already been specified in the collation order.

Programmer response: Remove the reference to the symbolic name from the collation weights that have not yet been specified in the collation order.

System action: The locale has not been created. If BLDERR option is specified, the unspecified reference to the symbolic name is ignored.

EDC4130 30 Error in file %1$s, on line %2$d, at character %3$d.

Explanation: An error had occurred in the charmap or locale definition file on the line number supplied and at the character position supplied. The line number and character position in the message indicates the position within the file when the error was detected. This may be after the line containing the error.

Programmer response: See the message EDC4131 10 for more information.

EDC4131 10 Warning in file %1$s, on line %2$d, at character %3$d.

Explanation: A warning message had been produced for the line number supplied, at the character position supplied in the charmap or locale definition file name supplied. The line number and character position in the message indicates the position within the file when the error was detected. This may be after the line containing the error.

Programmer response: See the message EDC4132 30 for more information.

EDC4132 30 Syntax error in file %1$s, on line %2$d, at character %3$d.

Explanation: A syntax error had been found in the charmap or local definition file name supplied, on the line number supplied and at the character position supplied. The line number and character position in the message indicates the position within the file when the error was detected. This may be after the line containing the error.

Programmer response: Change the line in the charmap or locale definition file to conform to the POSIX standard format.

EDC4133 40 Specific collation weight assignment was not valid when no sort keywords have been specified.

Explanation: The number of sort rules, such as forward, backward, no-substitute or position, specified after the order_start keyword must be greater than or equal to the number of weights assigned to any one character in the collation category of the locale definition file. When no sort rules are specified, one forward sort rule is assumed.

Programmer response: Add additional sort rules to the order_start keyword.

System action: The locale has not been created.

EDC4134 10 The <mb_cur_min> keyword must be defined as 1, you have defined it as %1$d. Value is ignored.

Explanation: The <mb_cur_min> keyword in the charmap file can only be set to 1.

Programmer response: Change the value of the <mb_cur_min> keyword in the charmap file to 1.

System action: The Value was ignored and the locale has been created.

EDC4135 30 The <code_set_name> must contain only characters from the POSIX portable character set, '%1$s' is not valid.

Explanation: The <code_set_name> in the charmap file must only use graph characters. It must contain only characters from the portable character set. The character %1$s is not valid.

Programmer response: Remove the character from the <code_set_name> in the charmap file that is not in the portable character set.

System action: The locale has not been created. If BLDERR option is specified, the <code_set_name> is used anyway.

EDC4136 30 The collation directives forward and backward are mutually exclusive.

Explanation: Each sort rules of the order_start keyword of the collation category in the locale definition file can consist of one or more sort rules separated by commas. The sort rules forward and backward, cannot be used at the same time.

Programmer response: Specify only forward or backward but not both.

System action: The locale has not been created.

EDC4137 30 Received too many arguments, expected %1$d.

Explanation: This message is issued in the locale definition file when a keyword is expecting a fixed number of arguments and too many arguments are supplied.

Programmer response: Remove the unnecessary argument in the locale definition file.

System action: The locale has not been created. If BLDERR option is specified, the extra arguments are ignored.

EDC4138 30 The %1$s category had already been defined.

Explanation: The specified category in the locale definition file should only be defined once.

Programmer response: Remove the specified duplicate category.

System action: The locale has not been created. If BLDERR option is specified, the second definition of the duplicate category is ignored.

EDC4139 10 The %1$s category was empty.

Explanation: The specified category in the locale definition file did not contain any keywords.

Programmer response: Remove the empty category or add keywords to the specified category.

System action: The locale has been created.

EDC4140 30 Unrecognized category %1$s was not processed by localedef.

Explanation: User defined categories in the locale definition file were not supported. That is, categories that are not LC_CTYPE, LC_COLLATE, LC_MONETARY, LC_NUMERIC, LC_TIME, LC_MESSAGES, LC_SYNTAX or LC_TOD were not processed by the `localedef` utility.

Programmer response: Remove the unrecognized category from the locale definition file.

System action: The locale has not been created. If BLDERR option is specified, the unsupported categories are ignored.

EDC4141 10 The POSIX defined categories must appear before any unrecognized categories.

Explanation: User defined categories in the locale definition file must appear after the POSIX defined categories LC_CTYPE, LC_COLLATE, LC_MONETARY, LC_NUMERIC, LC_TIME, LC_MESSAGES, LC_SYNTAX and LC_TOD.

Programmer response: Move the unrecognized category to the end of locale definition file.

System action: The locale has been created.

EDC4142 30 The file code for the digit %1$s was not one greater than the file code for %2$s.

Explanation: The values assigned to the digit symbolic names <zero> to <nine> in the charmap file must be in sequence and be contiguous.

Programmer response: Change the value assigned to the specified digit symbolic name in the charmap file so that it is one greater than the value assigned to the preceding digit symbolic name.

System action: The locale has not been created.

EDC4143 30 The process code for the digit %1$s is not one greater than the process code for %2$s.

Explanation: The wide character values assigned to the digit symbolic names <zero> to <nine> in the charmap file must be in sequence and be contiguous.

Programmer response: Change the wide character value assigned to the specified digit symbolic name in the charmap file so that it is one greater than the wide character value assigned to the preceding digit symbolic name.

System action: The locale has not been created. If BLDERR option is specified, the values are forced to be used.

EDC4144 30 The symbol %1$s has already been defined. Ignoring definition as a collating-symbol.

Explanation: The collation symbol must be a symbolic name, enclosed between angle brackets (< and >), and should not duplicate any symbolic name in the charmap file or any other name defined in the collation definition.

Programmer response: Use another symbolic name for the collating symbol.

System action: The locale has not been created. If BLDERR option is specified, the definition as a collating-symbol is ignored.

EDC4145 10 Locale did not conform to POSIX specifications for the LC_CTYPE '%1$s' keyword.

Explanation: The specified keyword in the LC_CTYPE category in the locale definition contained characters that conflict with the POSIX definition of the category. This may be caused by the following:

- The upper keyword contained characters from the cntrl, digit, punct or space keywords.
- The lower keyword contained characters from the cntrl, digit, punct or space keywords.
- The alpha keyword contained characters from the cntrl, digit, punct or space keywords.
- The space keyword contained characters from the digit, upper, lower, alpha or xdigit keywords.
- The cntrl keyword contained characters from the digit, upper, lower, alpha, graph, punct, print or xdigit keywords.
- The punct keyword contained characters from the digit, upper, lower, alpha, cntrl, space or xdigit keywords.
- The graph keyword contained characters from the cntrl keyword.
- The print keyword contained characters from the cntrl keyword.

Programmer response: Remove the character from the specified keyword that conflicts with characters from one of the other keywords.

System action: The locale has been created.

EDC4146 10 Locale did not specify the minimum required for the LC_CTYPE '%1$s' keyword. Setting to POSIX defined defaults.

Explanation: The specified keyword in the LC_CTYPE category in the locale definition file did not contain the minimum characters required by the keyword. The minimum requirements for the keywords are as follows:

- The upper keyword does not contain the required characters 'A' to 'Z'.
- The lower keyword does not contain the required characters 'a' to 'z'.
- The digit keyword does not contain the required digits 0 through 9.
- The xdigit keyword does not contain the required digits 0 through 9,the uppercase letters 'A' through 'F' and the lowercase letters 'a' through 'f'.
- The space keyword does not contain the required characters space, form feed, newline, carriage return, horizontal tab and vertical tab.
- The blank keyword does not contain the required characters space and tab.

Programmer response: Specify the minimum requirements for the specified keyword.

System action: The locale has been created.

EDC4147 10 Locale did not specify only '0', '1', - '2', '3', '4', '5', '6', '7', '8', and '9' for LC_CTYPE digit keyword.

Explanation: The digit keyword in the LC_CTYPE category in the locale definition file can only contain the characters required, '0' to '9'.

Programmer response: Remove the character outside the '0' to '9' range in the digit keyword.

System action: The locale will still be created.

EDC4148 10 Locale did not specify only '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a' through 'f', and 'A' through 'F' for LC_CTYPE xdigit keyword.

Explanation: The xdigit keyword in the LC_CTYPE category in the locale definition file can only contain the characters required, '0' to '9' and 'A' to 'F' or 'a' to 'f'. The locale will still be created.

Programmer response: Remove the character outside the range '0' to '9' and 'A' to 'F' or 'a' to 'f' to the xdigit keyword.

System action: The locale will still be created.

EDC4149 30 The number of operands to LC_COLLATE order exceeded COLL_WEIGHTS_MAX.

Explanation: The number of sort rules, such as forward, backward, no-substitute or position, specified after the order_start keyword must not exceed COLL_WEIGHTS_MAX in the collation category of the locale definition file.

Programmer response: Reduce the number of sort rules to the order_start keyword.

System action: The locale has not been created. If BLDERR option is specified, the extra operands are ignored.

EDC4150 30 Both '%1$s' and '%2$s' symbols must be characters and not collation symbols or elements.

Explanation: When defining ranges using ellipses (...) in the collation category of the locale definition file, the endpoints of the range must be characters or symbolic names defined in the charmap file. They should not be collating-symbol operands or collating-element operands.

Programmer response: Use different characters for the range endpoints.

System action: The locale has not been created. If the BLDERR option is specified, the defined ranges will not be used.

EDC4151 10 Option %1$s is not valid and was ignored.

Explanation: The option specified in the message is not a valid `localedef` utility option or a valid option has been specified with an invalid value.

Programmer response: Rerun the `localedef` utility with the correct option.

System action: The specified option was ignored and the locale has been created.

EDC4152 10 No matching right parenthesis for %1$s option.

Explanation: The option specified had a sub option beginning with a left parenthesis but no right parenthesis was present. The option and suboption were accepted and the locale was still produced.

Programmer response: Add the right parenthesis after the sub option.

System action: The option and sub option has been accepted and the locale was still produced.

EDC4153 10 Required symbolic name %1$s not defined in the charmap file.

Explanation: The symbolic name specified is not defined in the charmap file and must be specified.

Programmer response: Define the missing symbol name in the charmap file.

System action: The locale has been produced.

EDC4154 30 Keyword 'copy' cannot be nested.

Explanation: A locale category specifies the copy keyword, and the locale from which the category is being copied from also includes copy keyword for the same category.

Programmer response: Change the name of the existing locale to be copied to the name specified in existing locale copy keyword.

System action: The locale has not been created. If BLDERR option is specified, the default is used for the category.

EDC4155 30 'copy' keyword category '%1$s' not found.

Explanation: The specified category cannot be found in the locale definition file that was included using the copy keyword. The category is not copied.

Programmer response: Change the name of the existing locale to be copied or add the specified category to the locale definition file.

System action: The category is not copied and the locale has not been created.

EDC4156 30 LC_SYNTAX '%1$s' character can only be a punctuation character.

Explanation: The specified character defined in the LC_SYNTAX category of the locale definition file, must be a punctuation character. The character was ignored.

Programmer response: Only use punctuation characters as LC_SYNTAX characters.

System action: The locale has not been created.

EDC4157 10 LC_SYNTAX '%1$s' character can only have a length of 1. Ignoring additional characters.

Explanation: The specified character defined in the LC_SYNTAX category of the locale definition file contained more than one character, or specified a multibyte character. The LC_SYNTAX characters must only be single-byte characters.

Programmer response: Only use single-byte characters as LC_SYNTAX characters.

System action: The locale is created ignoring additional characters.

EDC4158 10 LC_SYNTAX '%1$s' character could not be found in the charmap file. Assigned code page %2$s symbol is '%3$s'.

Explanation: The LC_SYNTAX category was omitted, or the character was omitted from the LC_SYNTAX category, and the `localedef` utility attempted to assign the default value. The specified symbolic name was not found in the charmap file, and the character has been assigned the code point value from the IBM-1047 code page.

Programmer response: Specify the character in the LC_SYNTAX category that exists in charmap file, or change the charmap file to include the specified symbolic name.

System action: The local is still created.

EDC4159 30 Duplicate characters for '%1$s' and '%2$s' found in LC_SYNTAX.

Explanation: The specified characters from the LC_SYNTAX category have the same code points assigned.

Programmer response: Change the characters to specify different code points for each of the LC_SYNTAX characters.

System action: The locale has not been build.

EDC4160 10 The '%1$s' keyword is not supported and was ignored.

Explanation: The specified keyword is not defined in the POSIX standard.

Programmer response: Remove the specified keyword.

System action: The undefined keyword is ignored and locale has been created.

EDC4161 10 The <mb_cur_max> keyword must be defined as 1 or 4, you have defined it as %1$d. Value is ignored.

Explanation: The <mb_cur_max> keyword can have the value of 1 for single-byte characters only, or 4 to support DBCS characters. Values of other than 1 or 4 are ignored.

Programmer response: Specify <mb_cur_max> as either 1 or 4.

System action: The value is ignored and locale has been created. If BLDERR option is specified, the value of 1 is used.

EDC4162 30 Both <shift_in> and <shift_out> must be specified or neither specified.

Explanation: Either the <shift_in> keyword or the <shift_out> keyword have been specified, but not both.

Programmer response: Specify either both or neither <shift_in> and <shift_out>.

System action: The locale has not been created.

EDC4163 30 You have exceeded the maximum number of alternate strings for alt_digits.

Explanation: Up to 100 alternate strings can be specified for the alt_digits keyword for the values from zero to 99.

Programmer response: Remove the extra alternate strings.

System action: The locale has not been created. If BLDERR option is specified, the extra strings are ignored.

EDC4164 30 The grouping string '%1$s' is invalid.

Explanation: The string specified for the LC_NUMERIC grouping keyword or LC_MONETARY mon_grouping keyword is not in the correct format. The string should consist of numbers in the range -1 and 254 separated by semicolons.

Programmer response: Correct the grouping or mon_grouping string to be in the correct format.

System action: The locale has not been created.

EDC4165 10 The grouping string '%1$s' is invalid and had been truncated to '%2$s'.

Explanation: The string specified for the LC_NUMERIC grouping keyword or LC_MONETARY mon_grouping keyword is not in the correct format. The string should consist of numbers in the range -1 and 254 separated by semicolons, with no other numbers or semicolons following the -1.

Programmer response: Remove the characters from the grouping or mon_grouping string following the -1.

System action: The characters following the -1 were ignored and the locale has been created.

EDC4166 30 The value '%1$d' for '%2$s' is invalid.

Explanation: The value %1$d specified is not a value for the specified keyword. For example, the day is not valid for the specified month, or the month is not in the range from 1 to 12.

Programmer response: Correct the value for the specified keyword to be within the correct range for that keyword.

System action: The locale has not been created. If BLDERR option is specified, localdef assign 0 to value '%1$d'.

EDC4167 30 '%1$s' specified with no '%2$s'.

Explanation: The keyword specified can only be specified if the other keyword is also specified. Either both or neither should be specified.

Programmer response: Either remove the first keyword specified, or add the other required keyword.

System action: The locale has not been created.

EDC4168 10 'daylight_name' must be specified if Daylight Saving Time information is to be used by the mktime and localtime functions.

Explanation: Keywords had been specified in the LC_TOD category, but the 'daylight_time' keyword had not been specified. The other keywords will be ignored.

Programmer response: Remove the other keywords from the LC_TOD category, or add the 'daylight_time' keyword.

System action: The locale has been build.

EDC4169 30 One-to-many mappings cannot be specified against a collating-symbol, collating-element or the UNDEFINED symbol.

Explanation: A one-to-many mapping has been specified in the LC_COLLATE category against a collating-symbol, collating-element or the UNDEFINED symbol. For example, all of the following would cause this error message:

```
collating-symbol  <HIGH>
collating-element <ch> from "<c><h>"
<HIGH>    "<A>"
<ch>      "<B>"
UNDEFINED "<C>"
```

Programmer response: Remove the one-to-many mapping from the collating-symbol, collating-element or the UNDEFINED symbol.

System action: The locale has not been build.

EDC4170 40 Write failed while writing to file %1$s.

Explanation: The write failed to the specified file.

Programmer response: Look for a basic problem, such as insufficient disk space or lack of access to the file. If you are still unable to determine the cause of the write failure, contact your system programmer.

System action: The locale has not been created.

EDC4171 30 The process code of the first character of the collating-element was greater than the maximum process code.

Explanation: The wchar_t value for the first character in a collating-element was greater than the largest character specified in the charmap file. This may occur if the charmap file specifies <mb_cur_max> of 4, but did not specify the DBCS characters, and the collating-element begins with a DBCS character.

Programmer response: Either use a charmap file that specifies <mb_cur_max> of 1, or change the collating-element to not start with a DBCS character.

System action: The locale has not been created.

EDC4172 30 Consecutive ellipses collating elements are not allowed.

Explanation: Consecutive ellipses cannot occur in a range specification.

Programmer response: Remove one of the ellipses.

System action: The locale has not been created.

EDC4173 30 The grouping integer %1$s is invalid.

Explanation: The grouping integer must be >= -1 and <255.

Programmer response: Correct the grouping integer value.

System action: The locale has not been created.

EDC4174 10 The grouping integer list is invalid and has been truncated after -1.

Explanation: No grouping integers can follow -1.

Programmer response: Correct the grouping integer list.

System action: The grouping integers following -1 were ignored and the locale has been created.

EDC4175 40 Missing output locale name.

Explanation: The name of the file in which localedef creates the locale object was not specified when localedef was invoked.

Programmer response: Specify the name of the file in which the locale object is to be created.

System action: The locale has not been created.

EDC4176 40 Could not create a temporary file.

Explanation: The temporary file for locale generation could not be created.

Programmer response: Ensure that there are sufficient system resources to generate the temporary file and contact your system programmer.

System action: The locale has not been created.

EDC4177 40 You have specified different names for the same character '%1$d'.

Explanation: This message is produced when the option -w is specified and more than one symbolic name was assigned to the same character.

Programmer response: If multiple symbolic names for the same character are intended, then no action is required; otherwise, an unintended duplicate exists and should be corrected.

System action: The value was accepted and the locale has been created.

EDC4178 30 Character for escape_char statement missing. Statement ignored.

Explanation: The escape_char statement must be followed by a graphic character that is not a space.

Programmer response: Add a character after the escape_char statement.

System action: The statement has been ignored and the escape character set to default. The locale has been created.

EDC4179 30 Character for comment_char statement missing. Statement ignored.

Explanation: The comment_char statement must be followed by a graphic character that is not a space.

Programmer response: Add a character after the comment_char statement.

System action: The statement has been ignored and comment character set to default. The locale has been created.

EDC4200 40 usage: localedef [-c][-w][-A][-X][-f charmap][-i locsrc][-L binder opts] [-m methfile] locname

Explanation: Follow the usage information on how to correctly invoke localedef.

Programmer response: Correct the invocation syntax.

System action: The locale has not been generated.

EDC4201 40 The compile was unsuccessful. Required header files localdef.h and lc_core.h may be the wrong version or may be missing.

Explanation: The compile or link of the temporary method file was unsuccessful.

Programmer response: Ensure that the C compiler and the binder are correctly installed and functional.

System action: The locale has not been created.

EDC4202 40 The methods for mbtowc, mbstowcs, wctomb, wcstombs, mblen, wcwidth, wcswidth must be specified in the <methodfile>.

Explanation: The listed methods are mandatory and must be specified in the method file.

Programmer response: Add missing methods and rebuild the locale.

System action: The locale has not been generated.

EDC4203 40 Locale can not mix private method table methods and global method table methods.

Explanation: Use either private method table methods or global method table methods.

Programmer response: Change the method file to contain only methods from the method table of the same type.

System action: The locale has not been generated.

EDC4204 40 Unable to exec /bin/sh to process intermediate files.

Explanation: This is a system problem.

Programmer response: Contact your system programmer.

System action: The locale has not been created.

EDC4206 40 Missing end of symbol in string '%1$s'.

Explanation: The symbol is missing end of symbol character '>'.

Programmer response: Correct the symbol specification and rebuild the locale.

System action: The locale has not been created.

EDC4207 30 The symbolic name '%1$s' is not defined in the character map.

Explanation: The reported symbol name was not defined in the character map.

Programmer response: Ensure that the proper character map is used.

System action: The locale generation will continue.

EDC4208 40 Unable to load locale '%1$s' for copy directive.

Explanation: The locale named in the copy directive of the locale source could not be loaded.

Programmer response: Ensure that the locale name is correct and represents a full or relative path name.

System action: The locale has not been created.

EDC4209 40 Locale name longer than PATH_MAX (%1$d).

Explanation: The locale name that appears in a copy directive is longer than the maximum path length indicated.

Programmer response: Rename the locale or put it in a directory structure with a smaller depth.

System action: The locale has not been created.

EDC4210 30 '%1$s' was not declared in a charclass statement.

Explanation: The character class listed in the message was found in the locale specification but was not declared in a charclass statement.

Programmer response: Ensure that the character class is spelled correctly and rebuild the locale. If the character class was correctly spelled, add it to the charclass statement.

System action: The value was added to the character class table, and the locale has been created.

EDC4211 30 Collating symbols such as %1$s can not have explicit weights. Specification ignored.

Explanation: Collating symbols cannot be assigned explicit weights. Their weights are derived from the relative position they occupy in the locale specification.

Programmer response: Remove the weight assignment from the affected collating symbol.

System action: The value assigned to the collating symbol was ignored, and the locale has been created.

EDC4212 30 Ellipsis on the right hand side may only be used with ellipsis or UNDEFINED symbols on the left hand side.

Explanation: An ellipsis was used to assign a collation weight to a character but this is not allowed.

Programmer response: Use the correct weight assignment syntax.

System action: The value was ignored, and the locale has been created.

EDC4213 30 Ellipsis may not be used as one of the characters in a one-to-many mapping.

Explanation: Ellipsis was used in a one to many mapping but this is not allowed.

Programmer response: Use the correct one to many mapping syntax.

System action: The value was ignored, and the locale has been created.

EDC4214 40 Stack overflow error.

Explanation: The stack used for symbols and semantic elements was exhausted.

Programmer response: Contact your IBM representative.

System action: The locale has not been created.

EDC4215 40 Required symbol name %1$s not defined in character map file.

Explanation: The listed name was used in the locale specification but was not defined in the character map file.

Programmer response: Ensure that the correct character map file is used.

System action: The locale has not been created.

EDC4216 30 The symbol %1$s is too long. It will be truncated to %2$d bytes.

Explanation: The symbol name is longer than the maximum name length supported.

Programmer response: Use a shorter symbol name.

System action: The name was truncated, and the locale has been created.

EDC4217 30 The symbol '%1$s' character is undefined. This character along with any range statements it may be in will be ignored.

Explanation: The symbolic name for a character was not defined in the character map file.

Programmer response: Ensure that the correct character map file is used.

System action: The value was ignored, and the locale has been created.

EDC4218 30 More weights were defined for character %1$s than were specified with the order_start keyword.

Explanation: The character listed in the message was assigned more weights than the order_start keyword indicates.

Programmer response: Use the correct number of weights in line with what was specified in the order_start keyword.

System action: The extra weights were ignored and the locale has been created.

iconv Utility Messages

This topic contains the `iconv` return codes and messages.

Return Codes

The `iconv` utility returns the following return codes:

0 No errors were detected and the file was successfully converted from the input codeset to the output codeset.

4 The specified conversions are not supported, the given input file cannot be read, or there is a usage-syntax error.

8 An unusable character was encountered in the input file.

>8 A severe error occurred.

Messages

The messages issued by the `iconv` utility have the following format:

Message Format: EDCnnnn s text <%n$x>

nnnn Error message number

s Error severity

- **10** Warning message
- **30** Error message

%n$x	Substitution variable	
	%	The start of the substitution variable
	n	The number that represents the line position of the variable
	$	A delimiter
	x	The kind of variable (d=decimal, c=character, s=string)

The messages that can be issued are as follows:

EDC4151 10 Option *%1$s* is not valid and was ignored.

Explanation: The option specified in the message is not a valid `iconv` utility message, or a valid option had been specified with an invalid value.

System action: The specified option has been ignored.

Programmer response: Rerun the `iconv` utility, specifying the correct option.

EDC4152 10 No matching right parenthesis for *%1$s* option.

Explanation: The option specified had a suboption beginning with a left parenthesis, but no right parenthesis was present.

Programmer response: Add the right parenthesis.

System action: The option has been accepted as entered.

EDC4180 30 FROMCODE option had not been specified.

Explanation: The FROMCODE option is required, but had not been specified.

Programmer response: Rerun the `iconv` utility, specifying the FROMCODE option.

System action: The file has not been converted.

EDC4181 30 TOCODE option has not been specified.

Explanation: The TOCODE option is required, but has not been specified.

Programmer response: Rerun the `iconv` utility, specifying the TOCODE option.

System action: The file has not been converted.

EDC4182 30 Cannot open converter from *%1$s* to *%2$s*.

Explanation: The `iconv` utility could not locate the conversion from *%1$s* to *%2$s*.

Programmer response: Check the *%1$s* and *%2$s* specified to ensure they are correct.

System action: The file has not been converted.

EDC4183 30 Cannot open input file.

Explanation: The `iconv` utility could not open the input file.

Programmer response: Verify that the correct file name has been specified and rerun the `iconv` utility.

System action: The file has not been converted.

EDC4184 30 Cannot open output file.

Explanation: The `iconv` utility could not open the output file.

Programmer response: Correct the cause of the error and rerun the `iconv` utility.

System action: The file has not been converted.

EDC4185 30 Invalid character found.

Explanation: An invalid character was detected in the input file and could not be converted.

Programmer response: Correct the invalid character in the input file, or specify a different FROMCODE and TOCODE.

System action: The file is converted up to the record in error.

EDC4186 30 Truncated character found.

Explanation: The end of the file has been reached, and a truncated multibyte character was detected.

Programmer response: Correct the invalid character in the input file, or specify a different FROMCODE and TOCODE.

System action: The last character has not been converted.

EDC4187 30 Unable to allocate enough memory.

Explanation: The `iconv` utility could not allocate buffers for use when converting the file.

Programmer response: Rerun the `iconv` utility with more memory.

System action: The file has not been converted.

EDC4188 30 I/O error on file *filename*.

Explanation: An input or output error was detected with the *filename*.

This message is issued if the record format of the output file was fixed and the output records did not have the same length as the output file, or if the record format of the output file was variable and the output records were longer than the maximum record length.

Programmer response: Correct the cause of the input/output error and rerun the `iconv` utility.

System action: The file is converted up to the record in error.

EDC4189 30 Unable to fetch messages file EDCIMSGE.

Explanation: The `iconv` utility could not fetch the message file EDCIMSGE.

Programmer response: Ensure that the `iconv` utility has sufficient storage to run. Under CMS, ensure that the GLOBAL LOADLIB command has been issued. Under MVS and TSO, ensure that the correct libraries are specified on the STEPLIB DD statement.

System action: The file has not been converted.

genxlt Utility Messages

The messages issued by the `genxlt` utility have the following format:

Message Format: EDCxxxx nn text <%n$x>

xxxx Error message number

nn Error severity
- **10** Warning message
- **30** Error message

%n$x Substitution variable
- **%** The start of the substitution variable
- **n** The number that represents the line position of the variable
- **$** A delimiter
- **x** The kind of variable (d=decimal, s=string)

The messages that can be issued are as follows:

EDC4151 10 Option *%1$s* is not valid and was ignored.

Explanation: The option specified in the message is not a valid `genxlt` utility message, or a valid option had been specified with an invalid value.

Programmer response: Rerun the `genxlt` utility, specifying the correct option.

System action: The specified option is ignored.

EDC4190 30 Unable to open data file.

Explanation: The `genxlt` utility could not open the input file.

Programmer response: Check that the correct file name has been specified and rerun the `genxlt` utility.

System action: The conversion table has not been built.

EDC4191 30 Unable to open target file.

Explanation: The `genxlt` utility could not open the output file.

Programmer response: Correct the cause of the error, and rerun the `genxlt` utility.

System action: The conversion table has not been built.

EDC4192 30 There was no assignment for index *%1$d*.

Explanation: The character *%1$d* had not been assigned a character to which it must be converted.

Programmer response: Update the input file to specify a conversion value for the character specified.

System action: The conversion table has not been built.

EDC4193 30 Unable to write for target file.

Explanation: An output error was detected with the output file.

Programmer response: Correct the cause of the output error and rerun the `genxlt` utility.

EDC4194 30 Invalid format at line *%1$d*.

Explanation: The line *%1$d* is not valid. The conversion table has not been built.

Programmer response: Correct the line in error and rerun the `genxlt` utility.

System action: The conversion table has not been built.

EDC4195 30 Unable to fetch messages file EDCGMSGE.

Explanation: The `genxlt` utility could not fetch the message file EDCGMSGE.

Programmer response: Ensure that the `genxlt` utility has sufficient storage to run. Under CMS, ensure that the GLOBAL LOADLIB command has been issued. Under MVS and TSO, ensure that the correct libraries are specified on the STEPLIB DD statement.

System action: The file has not been converted.

Chapter 4. XL C/C++ Run-Time Messages

The following run-time messages pertain to C/C++ that have an EDC prefix.. For information on the messages from the C/C++ legacy class libraries, which have a prefix of CLB, refer to *z/OS XL C/C++ Messages*. Each message is followed by an explanation describing the condition that caused the message, a programmer response suggesting how you might prevent the message from occurring again, and a system action indicating how the system responds to the condition that caused the message.

The messages also contain a symbolic feedback code, which represents the first 8 bytes of a 12-byte condition token. You can think of the symbolic feedback code as the nickname for a condition. As such, the symbolic feedback code can be used in user-written condition handlers to screen for a given condition, even if it occurs at different locations in an application.

The messages in this topic contain alphabetic suffixes that have the following meaning:

I	Informational message
W	Warning message
E	Error message
S	Severe error message
C	Critical error message

EDC5000I No error occurred.

Explanation: The value of errno is zero.

Programmer response: None.

System action: None.

Symbolic Feedback Code: EDC4S8

EDC5001I A domain error occurred.

Explanation: An input argument to one of the math functions was outside the domain over which the mathematical function is defined.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the math functions that produced this error.

System action: The math function fails.

Symbolic Feedback Code: EDC4S9

EDC5002I A range error occurred.

Explanation: The result of the math function could not be represented as a double value, or a buffer provided by the user was not large enough to contain a function's output.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function that produced this error.

System action: The function fails or returns partial results.

Symbolic Feedback Code: EDC4SA

EDC5003I Truncation of a record occurred during an I/O operation.

Explanation: Truncation occurred because: 1) the specified record length on the write operation was larger than the record buffer size; 2) an attempt to extend the record buffer for a CMS variable length file failed which caused record truncation; or 3) the record read in was larger than the record buffer size.

Programmer response: For a text stream, place the newline character earlier in the record to shorten the record size. For a file opened for record I/O, specify a smaller number of bytes for `fread()`, `fwrite()`, or `fupdate()`.

System action: The return value depends on the operation attempted. In all cases, the buffer will be read or written up to the point where truncation occurred.

Symbolic Feedback Code: EDC4SB

EDC5004I The size of the specified record was too small.

Explanation: The record length was too small because the specified record size for an `fwrite()` or `fupdate()` function was smaller than the minimum record length allowed for the file.

Programmer response: Increase the size or count parameter on the `fwrite()` or `fupdate()` function.

System action: In all cases, the write or update operation fails.

Symbolic Feedback Code: EDC4SC

EDC5005I A write operation may not immediately follow a read operation.

Explanation: If the last operation on a non-VSAM file opened for record I/O was a read, a write operation may not directly follow.

Programmer response: Invoke `fflush()`, `rewind()`, `fseek()`, or `fsetpos()` between the read and write operations on the file stream.

System action: The write operation fails.

Symbolic Feedback Code: EDC4SD

EDC5006I A read operation may not immediately follow a write operation.

Explanation: If the last operation on a file was a write, a read operation may not directly follow.

Programmer response: Invoke `fflush()`, `rewind()`, `fseek()`, or `fsetpos()` between the write and read operations on the file stream.

System action: The read operation fails.

Symbolic Feedback Code: EDC4SE

EDC5007I The I/O buffer could not be allocated.

Explanation: Memory was not available for the allocation of various buffers when invoking any of the I/O functions.

Programmer response: Run the program in a larger region.

System action: The I/O function returns NULL or EOF.

Symbolic Feedback Code: EDC4SF

EDC5008I The LRECL or BLKSIZE exceeded the maximum allowable value.

Explanation: On CMS, the resultant CMS LRECL was greater than 65535. On MVS, the specified LRECL or BLKSIZE was greater than 32760.

Programmer response: Change the open attributes specified to be within the valid limits.

System action: The `fopen()/freopen()` function returns NULL.

Symbolic Feedback Code: EDC4SG

EDC5009I An I/O operation was attempted using an invalid FILE pointer.

Explanation: The FILE pointer that was input to the I/O function was not an active FILE pointer created by `fopen()/freopen()`.

Programmer response: Ensure that the FILE pointer was created by `fopen()/freopen()`, and that no I/O operation is attempted after `fclose()`.

System action: The specified I/O operation fails.

Symbolic Feedback Code: EDC4SH

EDC5010I A read operation was attempted on a file that was not opened for reading.

Explanation: When a read operation (for example, `fgetc()`, `fread()`) is invoked, the file specified must be opened in a mode that supports reading.

Programmer response: Open the file with a mode that supports reading. The following modes do NOT support reading: 'w', 'wb', 'a', or 'ab'.

System action: The read operation fails.

Symbolic Feedback Code: EDC4SI

EDC5011I The number of ungetc() push-back characters has exceeded the maximum allowed.

Explanation: An `ungetc()` was attempted but could not be honored because there were already pushed-back characters waiting to be read, and the number of pushed-back characters already equaled the maximum allowed.

Programmer response: Either read a pushed-back character, or reposition the file before attempting to push back another character.

System action: The `ungetc()` function returns EOF.

Symbolic Feedback Code: EDC4SJ

EDC5012I File positioning is not allowed for this data set.

Explanation: An attempt was made to either acquire the file position or reposition the file using an I/O function that is not supported by the file.

Programmer response: For non-VSAM files, do not open the file with the NOSEEK option. For VSAM PATH data sets, position the FILE pointer to the beginning of the data set, or do not issue the `ftell()` or `fseek()` functions. If the data set resides on a device that does not support repositioning, either move the data set or do not use the positioning functions.

System action: The function fails.

Symbolic Feedback Code: EDC4SK

EDC5013I No hiperspace blocks are available for expansion.

Explanation: A hiperspace has been filled to its maximum allowed size. An attempt has been made to add more data to the hiperspace.

Programmer response: Check with your system programmer to see if there is currently a shortage of resources for hiperspaces.

System action: The write operation fails.

Symbolic Feedback Code: EDC4SL

EDC5014I An attempt was made to acquire a position that is before the start of the file.

Explanation: The `ftell()` and `fgetpos()` functions cannot return a file position when characters are pushed back using `ungetc()` at the start of the file. This is because the resultant position ends up being before the start of the file, and this is not a valid position.

Programmer response: Do not call `ftell()` or `fgetpos()` if you push back characters before the start of the file, unless you either read them or discard them using a reposition first.

System action: Function `ftell()` or `fgetpos()` fails.

Symbolic Feedback Code: EDC4SM

EDC5015I The file position value was beyond the limits that ftell() can represent in a long integer value.

Explanation: The current position lies outside the bounds that can be represented by the `ftell()` encoding scheme. For fixed format binary streams, this is a file position beyond relative byte number 2147483647. For other files, the limit is based on the relative block or record number. For CMS files with LRECL between 32KB and 64KB, this limit is 65536 records. For all other files, the maximum depends on the block size. Smaller block sizes allow more records to be encoded. The maximum is at least 131072 records.

Programmer response: The `fgetpos()` function can be used to report file positions that `ftell()` cannot. The `fsetpos()` function must then be used to perform the repositioning at a later time.

System action: The `ftell()` function fails (return -1 (EOF)).

Symbolic Feedback Code: EDC4SN

EDC5016I Byte I/O was attempted on a file that was opened for record I/O.

Explanation: One of the byte I/O functions was invoked with a file that was opened using 'type=record'.

Programmer response: Only the `fread()`, `fwrite()`, and `fupdate()` functions may be used to read, write, or update a file opened for record I/O.

System action: The requested function fails.

Symbolic Feedback Code: EDC4SO

EDC5017I A write operation was attempted on a file that was not opened for writing.

Explanation: When a write operation (for example, `fputc()`, `fwrite()`) is invoked, the file specified must have been opened for writing.

Programmer response: Open the file with a mode that supports writing. The following modes do not support writing: 'r', 'rb'.

System action: The write operation fails.

Symbolic Feedback Code: EDC4SP

EDC5018I An ungetc() function call cannot immediately follow a write operation.

Explanation: A reposition function or flush must occur between a write operation and `ungetc()`.

Programmer response: Invoke either `fflush()`, `rewind()`, `fseek()`, or `fsetpos()` before calling `ungetc()`.

System action: The `ungetc()` function returns EOF.

Symbolic Feedback Code: EDC4SQ

EDC5019I An unrecoverable error has permanently marked the file in error.

Explanation: A previous I/O operation has failed such that the current file pointer is no longer valid. Only the `fclose()` and `freopen()` functions are permitted.

Programmer response: Check the return codes of previous I/O operations, or set up a SIGIOERR handler to determine the source of the error. When you have determined which C function has generated the error, issue `perror()` to get the original error message.

System action: The operation fails.

Symbolic Feedback Code: EDC4SR

EDC5020I An attempt to allocate memory in the Language Environment has failed.

Explanation: The attempt of Language Environment attempt to obtain memory in order to satisfy the current library request has failed.

Programmer response: Run the program in a larger region, or use the HEAP(,,FREE) run-time option instead of the HEAP(,,KEEP) option.

System action: The requested function fails.

Symbolic Feedback Code: EDC4SS

EDC5021I The file attributes for open create an invalid combination.

Explanation: An invalid combination was caused by merging the characteristics specified on the `fopen()/freopen()` call, the ddname declaration, or the existing file attributes.

Programmer response: Adjust the LRECL and BLKSIZE parameters on the `fopen()/freopen()` call, or adjust the attributes specified with the ddname so a valid combination will be created. See *z/OS XL C/C++ Programming Guide* for details on attribute merging and valid combinations.

System action: The `fopen()/freopen()` function returns NULL.

Symbolic Feedback Code: EDC4ST

EDC5022I An error occurred while generating a temporary name.

Explanation: The `tmpnam()` function was called to generate a temporary file name. However, a name, which did not already exist, could not be generated within the maximum number of attempts.

Programmer response: Verify that the `time()` and `clock()` functions are working correctly on your system. If so, try erasing all unused files that were created by the `tmpnam()` function, and retry the function. Contact your IBM Support Center if error still occurs.

System action: The `tmpnam()` function fails and returns NULL.

Symbolic Feedback Code: EDC4SU

EDC5023I An attempt to back up position has failed.

Explanation: One or more `ungetc()` calls are outstanding when an `fgetpos()` or `ftell()` is called such that the file position is really in the previous physical block. An attempt by the Language Environment to back up the file to acquire the position has failed.

Programmer response: Check the __amrc structure for more information. See *z/OS XL C/C++ Programming Guide* for more information on the __amrc structure.

System action: The `ftell()/fgetpos()` function fails.

Symbolic Feedback Code: EDC4SV

EDC5024I An attempt was made to close a file that had been opened on another thread.

Explanation: A file that was opened on one thread was closed on another.

Programmer response: All files must be closed on the same thread on which they were opened.

System action: The close operation fails.

Symbolic Feedback Code: EDC4T0

EDC5025I An I/O function was invoked when a read was pending for a file that had been intercepted.

Explanation: A file that was intercepted under the debugging tool was expecting input when an I/O function for that file was invoked from the debugging session.

Programmer response: Do not recursively invoke library I/O functions when a read is pending. Supply the expected data and then invoke the I/O function.

System action: The I/O function fails.

Symbolic Feedback Code: EDC4T1

EDC5026I An error occurred when expanding hiperspace.

Explanation: An error occurred trying to expand a hiperspace to more than its current size, but less than equal to its maximum allowable size.

Programmer response: Check the __amrc structure for the return code from HSPSERV. See *z/OS XL C/C++ Programming Guide* for more information on the __amrc structure.

System action: The write operation fails.

Symbolic Feedback Code: EDC4T2

EDC5027I The position specified to fseek() was invalid.

Explanation: One of the following occurred: 1) a whence value other than SEEK_SET, SEEK_CUR, or SEEK_END was specified; 2) the specified position was before the start of the stream; or 3) the specified position was beyond the end of the stream and the stream was not a binary file.

Programmer response: Correct the offset/whence parameters on the `fseek()` function to be a valid position, or open the file in binary mode if the user wants file positions beyond EOF to result in null extension to the file.

System action: The `fseek()` function fails.

Symbolic Feedback Code: EDC4T3

EDC5028I A previous I/O error has marked the stream invalid for further I/O processing.

Explanation: A serious error has occurred in a previous I/O operation such that further I/O cannot be continued. The routine that caused the original error had set errno previously, but it will have changed because of this errno value. The `clearerr()` function will not clear this type of error.

Programmer response: Check the return code values of previous I/O operations to detect which operation originated the system I/O failure and get the errno value. Use this list to find a prescribed action, or attempt a `rewind()` or `fsetpos()` to clear the internal error marker and reestablish the file position.

System action: The current I/O operation fails.

Symbolic Feedback Code: EDC4T4

EDC5029I An unrecognized signal value was passed to the signal() or raise() function.

Explanation: The signal value passed into the `signal()` or `raise()` function was not one of the valid signals as defined in signal.h.

Programmer response: Pass either SIGIOERR, SIGFPE, SIGSEGV, SIGILL, SIGABRT, SIGTERM, SIGINT, SIGTERM, SIGUSR1, SIGUSR2, or SIGABND to the `signal()` or `raise()` function.

System action: The `signal()` or `raise()` function returns SIG_ERR.

Symbolic Feedback Code: EDC4T5

EDC5030I An invalid argument was passed.

Explanation: The `setenv()` function has been called with a '=' sign in the environment variable name. This is an invalid argument.

Programmer response: Remove the '=' sign from the environment variable name.

System action: The `setenv()` function fails.

Symbolic Feedback Code: EDC4T6

EDC5031I An attempt was made to close a stream not belonging to the current main program.

Explanation: The user has passed a file pointer across a system call boundary and has attempted to close or reopen the file in the child program.

Programmer response: The program is invalid and must be changed. The suggested change is to close the file in the parent program before the `system()` call. The file can then be reopened in the child program, if required. Upon returning to the parent program, the file can again be reopened.

System action: The `fclose()/freopen()` function fails.

Symbolic Feedback Code: EDC4T7

EDC5032I An error was detected in the input string passed to the system() function.

Explanation: When the `system()` function was invoked and 'PGM=' was specified for an MVS-style parameter list, either the 'PARM=' string was not specified or invalid characters were found on the 'PARM=' string.

Programmer response: Correct the parameter string passed to the `system()` function, or use a VM-style parameter list.

System action: The `system()` function fails.

Symbolic Feedback Code: EDC4T8

EDC5033I An attempt was made to extend a non-extendable file.

Explanation: While updating a partitioned data set member or a concatenated dataset, an attempt was made to extend the file.

Programmer response: If extension is required to a member, open the old member in read mode and copy the contents to a new member that is opened for write. Because the new member is opened in write mode, it can be extended. Close both the old and new members, and then delete the old member with the `remove()` function. Rename the new member using the `rename()` function so that it appears to be the old member, now extended. You cannot extend the concatenation for a concatenated data set.

System action: The I/O write operation fails.

Symbolic Feedback Code: EDC4T9

EDC5034I An unsupported buffering mode was specified for the setvbuf() function.

Explanation: The buffer type specified as a parameter for the `setvbuf()` function was unsupported, or a buffer mode other than line buffered (_IOLBF) was specified for a terminal device type.

Programmer response: Specify one of the following supported buffer types: _IOFBF (full buffering) or _IOLBF (line buffering).

System action: The `setvbuf()` function fails.

Symbolic Feedback Code: EDC4TA

EDC5035I An attempt was made to change the buffering mode after an operation on a file.

Explanation: A call was made to the `setvbuf()` function after the file had been read or written.

Programmer response: Use the `setvbuf()` function to set the buffering mode before any read or write I/O operations are done.

System action: The `setvbuf()` function returns EOF.

Symbolic Feedback Code: EDC4TB

EDC5036I The specification of a member is invalid.

Explanation: On a `remove()` or `rename()` function call, a member has been specified for a file that does not have members, or a member has been specified for one name of a `rename()` function call, but not for the second name.

Programmer response: If the file does not have members, remove the member specification. If this is a `rename()` call and only one name specifies a member, either remove the member specification or add a member specification to the second name.

System action: The `remove()`/`rename()` function fails.

Symbolic Feedback Code: EDC4TC

EDC5037I The specified ddname was not found.

Explanation: A ddname was specified for a file name parameter, but the ddname was not defined. The functions that support ddnames as file names are `fopen()`, `freopen()`, and `remove()`.

Programmer response: See *z/OS XL C/C++ Programming Guide* for details on how to define a ddname in the environments supported. See *z/OS XL C/C++ Run-Time Library Reference* for the math functions.

System action: The function fails.

Symbolic Feedback Code: EDC4TD

EDC5038I An error occurred when the system flushed terminal output before retrieving terminal input.

Explanation: When a terminal read cannot get any data from the buffer and must perform a system terminal read, all terminal output data not yet flushed to the system must be output. While writing the unflushed terminal output data, an error occurred.

Programmer response: Change the code so all terminal output is completed and flushed to the terminal before the terminal input operation. Check all return codes to find out if any output operation gets an error; then check the errno value for further information regarding any errors encountered.

System action: The terminal input operation fails.

Symbolic Feedback Code: EDC4TE

EDC5039I A writable CMS minidisk could not be found to hold the output file specified.

Explanation: An attempt has been made to open a CMS minidisk file for ’write’ or ’append’, but there is no CMS minidisk accessed ’r/w’ to write the output file.

Programmer response: Access a CMS minidisk in ’r/w’ mode.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4TF

EDC5040I An attempt was made to open a flat file as a PDS.

Explanation: When a memory file is created without members, its name cannot be used with a member specified.

Programmer response: Either specify a memory file that already has members, or remove the flat memory file before specifying the open with the member specified and open the file for ’write’.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4TG

EDC5041I An error was detected at the system level when opening a file.

Explanation: A system level error was detected.

Programmer response: Look in the __amrc structure for further details regarding the error. See *z/OS XL C/C++ Programming Guide* for more information on the __amrc structure. Or check the MVS job log for an error message.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4TH

EDC5042I A special internally-generated memory file name was specified for opening, but a memory file with this name does not exist.

Explanation: A name was specified of the form: '((x))', where 'x' is a decimal number, but the name did not match an existing memory file. A name of this format may not be used to create a memory file. The name is generated internally by C/MVS when a user opens a memory file with a name of '*' for output. C/MVS generates the name so that the user can acquire it using the `fldata()` function and then can read, update, append, or remove the memory file.

Programmer response: If using memory files created with a name of '*', issue `fldata()` to acquire the correct name. If the name was properly acquired using `fldata()`, make sure it has not been closed and removed before opening the generated name.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4TI

EDC5043I An attempt was made to open a non-memory file as a memory file.

Explanation: An open for read has specified 'type=memory', but the file is not a memory file.

Programmer response: Remove the 'type=memory' specification on the `fopen()`/`freopen()`.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4TJ

EDC5044I An error occurred when attempting to erase a CMS file.

Explanation: When the `remove()` function was invoked, either the CMS file was not found to be erased, or an error occurred when the system attempted to erase the file.

Programmer response: Ensure that the file name specified to the `remove()` function exists and is on a disk accessed for write.

System action: The `remove()` function fails.

Symbolic Feedback Code: EDC4TK

EDC5045I The operation attempted could not be performed because the file was open.

Explanation: An attempt was made to remove or rename a file that was still open, or an attempt was made to open a file for output or append that was already open.

Programmer response: The `remove()` function can only be invoked with files that have been closed. The `fopen()` function cannot open a memory or disk file for write/update/append if the file is already opened. A memory file opened with a member specified will prevent the name from being used without a member, and vice-versa. For example, it is not possible to have memory files: 'a.b' and 'a.b(c)' opened at the same time. In either case, the original open file must be closed.

System action: The `remove()`, `rename()`, `fopen()`, or `freopen()` function fails.

Symbolic Feedback Code: EDC4TL

EDC5046I The file could not be deleted.

Explanation: The `remove()` function could not remove the file specified on MVS.

Programmer response: Verify that the file name specified to the `remove()` function is erasable.

System action: The `remove()` function fails.

Symbolic Feedback Code: EDC4TM

EDC5047I An invalid file name was specified as a function parameter.

Explanation: The name specified to the `remove()`, `rename()`, `fopen()`, or `freopen()` functions was invalid. The name is either not valid for the system (MVS, CMS), is not a valid memory file name, or is a '*' or GDG data set name specified to the `rename()` function.

Programmer response: Specify a valid file name according to the system, or to the memory file name rules to the `remove()`, `rename()`, `fopen()`, and `freopen()` functions.

System action: The invoked function fails.

Symbolic Feedback Code: EDC4TN

EDC5048I A Language Environment internal routine has failed unexpectedly.

Explanation: An internal call to a Language Environment internal routine has failed, but the failure is not anticipated, and recovery is not possible.

Programmer response: Contact your IBM Support Center.

System action: Current library function using an internal routine fails.

Symbolic Feedback Code: EDC4TO

EDC5049I The specified file name could not be located.

Explanation: When the `rename()` function was invoked, the old file name could not be found or the new file name could not be allocated or, when the `fopen()`/`freopen()` function was invoked, the specified file name opened for read could not be found.

Programmer response: Verify that the specified file exists.

System action: The `fopen()`, `freopen()`, or `rename()` function fails.

Symbolic Feedback Code: EDC4TP

EDC5051I An error occurred when renaming a file.

Explanation: A rename error has occurred.

Programmer response: For disk files, ensure that the old file name exists. For memory files, ensure that different names are specified to the `rename()` function and that PDS-style naming conventions are used consistently for old and new names. For MVS, check the __amrc for further details.

System action: The `rename()` function fails.

Symbolic Feedback Code: EDC4TR

EDC5052S The application is running with AMODE=24 while the run-time library was installed above the line.

Explanation: The application which is accessing the run-time library is running with AMODE=24. But the run-time library was installed above the 16MB line, which the application cannot address.

Programmer response: Ensure the AMODE of the application matches that of the run-time library. Language Environment no longer supports C applications in AMODE=24. Relink the application to have AMODE=31.

System action: Application is terminated with 3000 abend.

Symbolic Feedback Code: EDC4TS

EDC5053S The Language Environment run-time library load module EDCZ24 could not be loaded.

Explanation: An error has occurred when Language Environment tried to load the run-time library load module EDCZ24.

Programmer response: Check the data sets in the go steplib for the job to ensure that EDCZ24 is available. (For example, check SCEERUN.)

System action: The program ends and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000 is returned.

Symbolic Feedback Code: EDC4TT

EDC5054I An attempt to override the disposition was ignored. The file may still be removed.

Explanation: The `remove()` function attempted to delete the data set by using a disposition of DELETE. The data set would not allow an override of the disposition.

Programmer response: Change the disposition on the original allocation, or remove the data set outside of your C program.

System action: The `remove()` function fails, but the data set may have been removed if the original allocation specified DELETE as the normal disposition.

Symbolic Feedback Code: EDC4TU

EDC5055I An error occurred trying to remove the file before its expiration date.

Explanation: When the `remove()` function attempted to erase the file, an error was returned indicating that the expiration date had not yet occurred.

Programmer response: Change the expiration date of the data set.

System action: The `remove()` function fails.

Symbolic Feedback Code: EDC4TV

EDC5057I The open mode string was invalid.

Explanation: The mode string passed to the `fopen()`/`freopen()` function was found to have invalid keywords, combinations, or characters. For example, if you are opening a ddname, be sure the DISP= specified on the DD statement is compatible with the open mode you specified.

Programmer response: Correct the mode string and reissue the `fopen()`/`freopen()`.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4U1

EDC5059I An attempt to reposition a VSAM file failed.

Explanation: When the `flocate()` function was invoked, the reposition was not successful, or `rewind()` could not position to the beginning of the file.

Programmer response: For `flocate()`, verify that the attributes of the VSAM file match the type of repositioning being attempted. For a `rewind()` error, check the __amrc structure. See *z/OS XL C/C++ Programming Guide* for more information on the __amrc structure.

System action: The `flocate()` function fails. The `rewind()` function does not reposition to the start of the data set.

Symbolic Feedback Code: EDC4U3

EDC5060I An invalid file position was passed to the fsetpos() function.

Explanation: When `fsetpos()` was invoked, the fpos_t structure passed did not represent a valid position in the current file.

Programmer response: Verify that the fpos_t structure set by `fgetpos()` is a valid file position before calling `fsetpos()`. Also verify that the file has not changed between the time of the `fgetpos()` and `fsetpos()`.

System action: The `fsetpos()` function fails.

Symbolic Feedback Code: EDC4U4

EDC5061I An error occurred when attempting to define a file to the system.

Explanation: The `fopen()`/`freopen()` function or the `remove()` function could not successfully allocate or FILEDEF the specified file.

Programmer response: Check the __amrc structure for more information. See *z/OS XL C/C++ Programming Guide* for more information on the __amrc structure.

System action: The `fopen()`, `freopen()`, or `remove()` function fails.

Symbolic Feedback Code: EDC4U5

EDC5063I An error was detected in an internal control block.

Explanation: One of the internal I/O control blocks was corrupted and is causing unexpected behavior.

Programmer response: Ensure that the application program is not overwriting storage. If the error cannot be located, contact the IBM Support Center.

System action: The I/O operation fails and the stream is marked as invalid for further I/O.

Symbolic Feedback Code: EDC4U7

EDC5065I A write system error was detected.

Explanation: A system level write error has occurred.

Programmer response: Check the __amrc structure for more information. See *z/OS XL C/C++ Programming Guide* for more information on the __amrc structure.

System action: The write operation fails.

Symbolic Feedback Code: EDC4U9

EDC5066I A read system error was detected.

Explanation: A system level read error has occurred.

Programmer response: Check the __amrc structure for more information. See *z/OS XL C/C++ Programming Guide* for more information on the __amrc structure.

System action: The read operation fails.

Symbolic Feedback Code: EDC4UA

EDC5067I An attempt was made to open a nonexistent file for read.

Explanation: The `fopen()`/`freopen()` function was invoked for read, but the file specified did not exist, or the data set name '*' was attempted to be opened for read in MVS batch.

Programmer response: Ensure that the file to be opened for read exists, or that the interactive terminal is not being opened for read in MVS batch.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4UB

EDC5072I An attempt was made to open a KSDS or Path VSAM data set without specifying record I/O.

Explanation: Key Sequenced VSAM data sets and Path VSAM data sets may not be opened as streams for writing. Only Entry Sequenced VSAM data sets and Relative Record VSAM data sets may be opened this way.

Programmer response: Change the type string parameter on the `fopen()` function to include 'type=record'.

System action: The `fopen()`/`freopen()` function returns NULL.

Symbolic Feedback Code: EDC4UG

EDC5073I The maximum number of attempts to obtain temporary names was exceeded.

Explanation: The `tmpnam()` function was invoked more than the maximum number of times allowed.

Programmer response: The programmer should alter the application to minimize the number of calls to `tmpnam()`. The system can only ensure that TMP_MAX calls will work.

System action: The `tmpnam()` function returns NULL and does not generate any more unique names.

Symbolic Feedback Code: EDC4UH

EDC5074I The open parameters were missing the ’type=record’ specifier.

Explanation: The open type keyword parameter ’acc=’ is not valid unless ’type=record’ is also specified.

Programmer response: Specify ’type=record’ on the `fopen()`/`freopen()` statement.

System action: The `fopen()`/`freopen()` function returns NULL.

Symbolic Feedback Code: EDC4UI

EDC5076I An fread() was not performed before calling the fdelrec() or fupdate() functions.

Explanation: The `fdelrec()` and `fupdate()` functions may not be invoked without first calling the `fread()` function.

Programmer response: Invoke the `fread()` function directly before these functions.

System action: The `fdelrec()` and `fupdate()` functions fail.

Symbolic Feedback Code: EDC4UK

EDC5077I An error occurred trying to erase a VSAM record.

Explanation: The `fdelrec()` function could not successfully erase the last record read from the specified VSAM file.

Programmer response: Examine the values of __amrc__code__feedback__rc and __amrc__code__feedback__fdbk immediately after receiving this errno. Look up the __rc and __fdbk values in a VSAM Macro Reference manual, such as *MVS/ESA VSAM Administration: Macro Instruction Reference*. __rc corresponds to the register 15 value, __fdbk corresponds to the Reason Code. See *z/OS XL C/C++ Programming Guide* for more information on the __amrc structure.

System action: The `fdelrec()` function fails.

Symbolic Feedback Code: EDC4UL

EDC5078I The requested operation is valid only for VSAM data sets.

Explanation: The `fdelrec()`, `flocate()` and `fupdate()` functions may only be invoked with VSAM data sets.

Programmer response: Use the `fseek()`/`ftell()` or `fgetpos()`/`fsetpos()` functions for positioning within a non-VSAM file. To update, use `fread()`/`fwrite()` or the byte I/O functions instead of `fupdate()`.

System action: The `fdelrec()`, `flocate()` and `fupdate()` functions fail.

Symbolic Feedback Code: EDC4UM

EDC5079I The file was not opened with a ’type=record’ specification.

Explanation: The `fdelrec()` and `fupdate()` functions are not valid for VSAM data sets opened as streams.

Programmer response: Change the `fopen()` type parameter string to include ’type=record’.

System action: The `fdelrec()` and `fupdate()` functions failed.

Symbolic Feedback Code: EDC4UN

EDC5080I An invalid option was passed to the flocate() function.

Explanation: One of the supplied parameters, options or key to the `flocate()` function contained an invalid value.

Programmer response: Use one of the following as the options parameter: __KEY_FIRST, __KEY_LAST, __KEY_EQ, __KEY_EQ_BWD, __KEY_GE, __RBA_EQ or __RBA_EQ_BWD, as defined in `stdio.h`.

If using either __RBA_EQ or __RBA_EQ_BWD, ensure that the key parameter is a multiple of the LRECL.

System action: The `flocate()` function fails.

Symbolic Feedback Code: EDC4UO

EDC5083I An error occurred attempting to load a module into storage.

Explanation: The library has attempted to dynamically load a module and a failure resulted. This is usually as a result of a `system()` call.

Programmer response: Verify that the specified program/command has been made accessible for loading. You may also need to adjust your region size. For MVS batch, check the job log for messages which will help to pinpoint the name of the module.

System action: The called library function fails.

Symbolic Feedback Code: EDC4UR

EDC5084I The program was not run because of redirection errors on the command line.

Explanation: An error was detected when the input string to `main()` was being parsed. One of the following may have occurred: 1) the file name specified with the redirection symbols could not be opened (for read, write, or append); 2) the file name specified with the write redirection symbol was already opened; or 3) the same redirection symbol was specified more than once in the command string.

Programmer response: Correct the input string passed to main and if the `system()` call is being used to invoke another C `main()` program, the files that are still open in the first program will be considered open when the redirection statements are being verified.

System action: This errno is used internally to generate the redirection error message. The program is terminated or the `system()` call returns the failure and does not invoke the second program.

Symbolic Feedback Code: EDC4US

EDC5086I An unsupported open mode was specified for a PDS member.

Explanation: The `fopen()`/`freopen()` function was incorrectly invoked specifying write-update, append, or append-update for a PDS member.

Programmer response: Open a PDS member with open modes: read, read-update, or write.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4UU

EDC5087I The specified file characteristics did not match those of the existing file.

Explanation: The `fopen()`/`freopen()` was attempting to perform an open that used an existing data set, but found that the specified attributes did not match the existing file attributes; specifically, LRECL, BLKSIZE, or record format.

Programmer response: Verify that the attributes of the physical file are as expected by the application program.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4UV

EDC5088I An invalid open mode was specified for the current device.

Explanation: The following open modes and device types are invalid combinations: 1) opening the interactive terminal for update; 2) reading a display or printer; 2) writing to a character reader; 3) updating a magnetic tape device; 4) opening SYSIN or SYSOUT for ’append’ or ’update’; 5) opening SYSIN for anything except ’read’; or 6) opening SYSOUT ’read’.

Programmer response: Correct the open mode on the `fopen()`/`freopen()` call and/or verify the that the current device type is what is expected.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4V0

EDC5089I Open mode is invalid for a SYSIN or SYSOUT data set.

Explanation: One of the following was attempted: 1) opening a JCL instream data set for ’update’, ’write’, or ’append’; 2) opening a SYSOUT data set for ’read’ or ’update’.

Programmer response: Correct the open mode on the `fopen()`/`freopen()` call.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4V1

EDC5091I The requested function could not be performed because a system utility failed.

Explanation: A system level utility used by the library unexpectedly returned a failure code.

Programmer response: Check the __amrc structure and *z/OS XL C/C++ Programming Guide* for further details.

System action: The requested function fails.

Symbolic Feedback Code: EDC4V3

EDC5092I An I/O abend was trapped.

Explanation: An I/O abend has occurred during an I/O operation (open, read, write, position, or close) and has been trapped. Recovery was attempted.

Programmer response: Check the __amrc structure defined in *z/OS XL C/C++ Programming Guide* for an explanation of the fields.

System action: The I/O operation fails. The stream is marked in error and all further I/O operations on this stream fail.

Symbolic Feedback Code: EDC4V4

EDC5093I An unsupported I/O operation has been attempted.

Explanation: An unsupported I/O operation has been attempted by the application program. The unsupported operations resulting in this message are the following:

- An attempt was made to open a large format sequential data set for read with repositioning (seek), while the data set was already opened for write without repositioning (noseek). This action is prevented because the writer can extend the data set beyond the point where the reader is allowed to read (65535 tracks). A possible solution is to open for read without repositioning (noseek).

Programmer response: Change the application program to avoid using the unsupported I/O operation.

System action: The function call fails.

Symbolic Feedback Code: EDC4V5

EDC5094I An attempt was made to push back the EOF character using ungetc().

Explanation: The `ungetc()` function may not be invoked with the EOF character.

Programmer response: Do not call `ungetc()` with EOF.

System action: The `ungetc()` function fails.

Symbolic Feedback Code: EDC4V6

EDC5095I The requested CMS minidisk was not accessed.

Explanation: The `fopen()`/`freopen()` function could not open the CMS file specified because the specified minidisk was not accessed.

Programmer response: Access the correct disk when attempting to open a file.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4V7

EDC5098I An invalid RECFM was specified when opening a PDS member.

Explanation: The resultant record format for the open function is invalid. Under CMS, record formats containing ASA characters or machine characters are invalid for PDS members, as well as files with the spanned attribute. Under MVS, spanned record formats are not valid.

Programmer response: Issue the `fopen()`/`freopen()` function with valid attributes, or verify the attributes specified when the ddname is defined.

System action: The `fopen()`/`freopen()` function fails.

Symbolic Feedback Code: EDC4VA

EDC5099I The function specified is not supported under CICS.

Explanation: The function specified is not supported under CICS.

Programmer response: Refer to *z/OS XL C/C++ Programming Guide* for more information on running with C/MVS under CICS.

System action: The specified function fails.

Symbolic Feedback Code: EDC4VB

EDC5100I An attempt was made to perform disk file I/O under CICS.

Explanation: The `fopen()`, `freopen()`, `rename()` and `remove()` functions only support memory files. The standard streams must be memory files or use the specified queues.

Programmer response: Only invoke `fopen()`, `freopen()`, `rename()` or `remove()` with memory files when running under CICS.

System action: The `fopen()`, `freopen()`, `rename()`, and `remove()` functions fail, and all writes to stdout/stderr fail.

Symbolic Feedback Code: EDC4VC

EDC5101I The transient data queue was not enabled for the standard streams.

Explanation: An attempt was made to write to stdout or stderr, when running under CICS, when the requested transient data queue was not enabled.

Programmer response: Ensure that the DFHDCT macro has been assembled and defined correctly in the start-up CICS JCL. The systems programmer will know the name, type of queue, and associated ddnames at your installation.

System action: The write I/O operation fails.

Symbolic Feedback Code: EDC4VD

EDC5102I The transient data queue was not opened for the standard streams.

Explanation: When the first I/O operation was requested for stdout or stderr when running under CICS, the Transient Data Queue inquiry indicated that the requested TD queue was not opened.

Programmer response: Verify that the start-up CICS JCL opened the specified TD queues correctly.

System action: The write I/O operation fails.

Symbolic Feedback Code: EDC4VE

EDC5103I An attempt was made to map remote queues to the standard streams under CICS.

Explanation: When the first I/O operation was requested for stdout or stderr when running under CICS, the Transient Data Queue inquiry indicated that the requested queue was a REMOTE queue.

Programmer response: Correct the start-up JCL to specify (using the DFHDCT macro) the standard stream queues to be EXTRAPARTITION, INTRAPARTITION, or INDIRECT.

System action: The write I/O operation fails.

Symbolic Feedback Code: EDC4VF

EDC5106I An error occurred creating a hiperspace memory file.

Explanation: An error has occurred while trying to create a hiperspace for a 'type=memory(hiperspace)' file. The error may result from a shortage of resources.

Programmer response: Check with your system programmer if hiperspace facilities are available at your installation, and if available, if there is currently a shortage of resources for hiperspaces.

System action: The `fopen()`/`freopen()` or other I/O operation fails.

Symbolic Feedback Code: EDC4VI

EDC5107I An error occurred writing to a hiperspace memory file.

Explanation: An error occurred when writing to a hiperspace memory file. The return code from the HSPSERV macro was greater than 4.

Programmer response: Check the __amrc structure for the return code from HSPSERV. Refer to *z/OS MVS Programming: Assembler Services Reference ABE-HSP* for more information regarding the return code. See *z/OS XL C/C++ Programming Guide* for more information on the __amrc structure.

System action: The write I/O operation fails.

Symbolic Feedback Code: EDC4VJ

EDC5108I An error occurred reading from a hiperspace memory file.

Explanation: An error occurred when reading from a hiperspace memory file. The return code from the HSPSERV macro was greater than 4.

Programmer response: Check the __amrc structure for the return code from HSPSERV. Refer to *z/OS MVS Programming: Assembler Services Reference ABE-HSP* for more information regarding the return code. See *z/OS XL C/C++ Programming Guide* for more information on the __amrc structure.

System action: The read I/O operation fails.

Symbolic Feedback Code: EDC4VK

EDC5111I Permission denied.

Explanation: An attempt was made to access a file in a way that violates its file access permissions. This message is equivalent to the POSIX.1 EACCES errno.

Programmer response: The specific reason for the access denial depends on the function being attempted. Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The access request is denied. The application continues to run.

Symbolic Feedback Code: EDC4VN

EDC5112I Resource temporarily unavailable.

Explanation: A (temporary) condition has occurred which makes the resource unavailable. Later calls may complete normally. This message is equivalent to the POSIX.1 EAGAIN errno.

Programmer response: The reason for the resource being unavailable depends on the function being attempted. Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request has failed. The application continues to run.

Symbolic Feedback Code: EDC4VO

EDC5113I Bad file descriptor.

Explanation: The file descriptor used referred to a file which was not open or was out of range, or a read request was made to a file that was only open for writing, or a write request was made to a file that was open only for reading. This message is equivalent to the POSIX.1 EBADF errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request has failed. The application continues to run.

Symbolic Feedback Code: EDC4VP

EDC5114I Resource busy.

Explanation: An attempt was made to use a system resource that was not available because it was being used by another process or thread in a manner that would have conflicted with the request being made by this process/thread. This message is equivalent to the POSIX.1 EBUSY errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request has failed. The application continues to run.

Symbolic Feedback Code: EDC4VQ

EDC5115I No child processes.

Explanation: A `wait()` or `waitpid()` function was executed by a process that had no existing or unwaited-for child processes. This message is equivalent to the POSIX.1 ECHILD errno.

Programmer response: Ensure that a child process exists.

System action: The request has failed. The application continues to run.

Symbolic Feedback Code: EDC4VR

EDC5116I Resource deadlock avoided.

Explanation: An attempt was made to lock a system resource that would have resulted in a deadlock situation. This message is equivalent to the POSIX.1 EDEADLK errno.

Programmer response: Refer to *z/OS XL C/C++ Programming Guide* for the function being attempted for the specific reason for failure.

System action: The request has failed. The application continues to run.

Symbolic Feedback Code: EDC4VS

EDC5117I File exists.

Explanation: An inappropriate action was requested for an existing file. For instance, a `mkdir()` is attempted for a file that already exists. This message is equivalent to the POSIX.1 EEXIST errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request has failed. The application continues to run.

Symbolic Feedback Code: EDC4VT

EDC5118I Incorrect address.

Explanation: The system detected an invalid address when using an argument of a call. Note that not all functions detect this error. This message is equivalent to the POSIX.1 EFAULT errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure. This failure is usually caused by an invalid argument address.

System action: The request has failed. The application continues to run.

Symbolic Feedback Code: EDC4VU

EDC5119I File too large.

| **Explanation:** The size of a file would exceed the maximum allowed. The maximum file size allowed for z/OS UNIX
| file system files is 2 gigabytes. This message is equivalent to the POSIX.1 EFBIG errno.

Programmer response: Ensure that enough space is available in the file.

System action: The request has failed. The application continues to run.

Symbolic Feedback Code: EDC4VV

EDC5120I Interrupted function call.

Explanation: An asynchronous signal was caught by the (POSIX) process during the execution of an interruptible function, and the signal handler (or default action) resulted in a normal return. This resulted in the interrupted function returning this error condition. This message is equivalent to the POSIX.1 EINTR errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for possible side effects from the function being interrupted.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC500

EDC5121I Invalid argument.

Explanation: An argument supplied was invalid. This message is equivalent to the POSIX.1 EINVAL errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request has failed. The application continues to run.

Symbolic Feedback Code: EDC501

EDC5122I Input/output error.

Explanation: An input or output error occurred. This message is equivalent to the POSIX.1 EIO errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC502

EDC5123I Is a directory.

Explanation: The program attempted to write to a file descriptor that is a directory. This message is equivalent to the POSIX.1 EISDIR errno.

Programmer response: Ensure that the file being written to is not a directory.

System action: The request has failed. The application continues to run.

Symbolic Feedback Code: EDC503

EDC5124I Too many open files.

Explanation: An attempt was made to open more than the maximum number of file descriptors allowed for this (POSIX) process. This message is equivalent to the POSIX.1 EMFILE errno.

Programmer response: The maximum number of files allowed per (POSIX) process is controlled by the OPEN_MAX run-time invariant, which can be determined during program execution using the `sysconf()` function.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC504

EDC5125I Too many links.

Explanation: An attempt was made to have the link count of a file exceed the maximum value allowed. For instance, this can occur while using the `link()` or `rename()` functions. This message is equivalent to the POSIX.1 EMLINK errno.

Programmer response: The maximum number of links for a file is established by the LINK_MAX pathname variable value, and which can be determined at execution time using the `pathconf()` function.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC505

EDC5126I Filename too long.

Explanation: The size of a pathname string exceeded the maximum allowed, or a pathname component was longer than the maximum allowed and no truncation is in effect. This message is equivalent to the POSIX.1 ENAMETOOLONG errno.

Programmer response: The maximum allowable pathname is controlled by the PATH_MAX pathname variable value, which can be determined at execution time using the `pathconf()` function. The maximum allowable file name is controlled by the NAME_MAX pathname variable value, which can be determined at execution time using the `pathconf()` function. Truncation of the pathname component is not allowed if the _POSIX_NO_TRUNC execution time symbolic is set. This can be determined by using the `pathconf()` function.

System action: The request fails. The application continues to run.

| If this is a DLL failure, and the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3597I
| message is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(),
| signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG
| environment variable.

| If this is a DLL failure, a CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC506

EDC5127I Too many open files in system.

Explanation: The system reached its predefined limit for files open at one time. This message is equivalent to the POSIX.1 ENFILE errno.

Programmer response: The request may succeed later if fewer files are in use.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC507

EDC5128I No such device.

Explanation: The function being attempted is not allowed by the specified device. For instance, the program attempted to read from a printer. This message is equivalent to the POSIX.1 ENODEV errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC508

EDC5129I No such file or directory.

Explanation: A name in the pathname does not exist, or the pathname is an empty string. This message is equivalent to the POSIX.1 ENOENT errno.

Programmer response: Ensure the pathname for the object being accessed is correct. Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC509

EDC5130I Exec format error.

Explanation: A request was made to execute a file that was not in a format that may be executed (however, the file does have the appropriate permissions). This message is equivalent to the POSIX.1 ENOEXEC errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50A

EDC5131I No locks available.

Explanation: No file and/or record locks are available. The system limit has been reached. This message is equivalent to the POSIX.1 ENOLCK errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50B

EDC5132I Not enough memory.

Explanation: There is not enough memory space available to create the new object. This message is equivalent to the POSIX.1 ENOMEM errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50C

EDC5133I No space left on device.

| **Explanation:** During a `write()` function to a file, there was no free space left in the z/OS UNIX file system. This
| error may also occur when extending a directory. This message is equivalent to the POSIX.1 ENOSPC errno.

Programmer response: Allocate additional space for the file system. Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50D

EDC5134I Function not implemented.

Explanation: The function to be executed has not been implemented. This message is equivalent to the POSIX.1 ENOSYS errno.

Programmer response: The function may not be used by the application program.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50E

EDC5135I Not a directory.

Explanation: A component in a pathname or directory specified an object that was not a directory. This message is equivalent to the POSIX.1 ENOTDIR errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50F

EDC5136I Directory not empty.

Explanation: A directory that was expected to be empty was not. This message is equivalent to the POSIX.1 ENOTEMPTY errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50G

EDC5137I Inappropriate I/O control operation.

Explanation: A control function was attempted for a file or a special file for which the operation was inappropriate. For instance, the file is not a terminal. This message is equivalent to the POSIX.1 ENOTTY errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50H

EDC5138I No such device or address.

Explanation: Input or output on a special file referred to a device that did not exist, or made a request beyond the limits of the device. For instance, I/O was sent to a tape drive that is not online. This message is equivalent to the POSIX.1 ENXIO errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50I

EDC5139I Operation not permitted.

Explanation: An attempt was made to perform an operation limited to processes with appropriate privileges, or to the owner of a file or other resource. This message is equivalent to the POSIX.1 EPERM errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50J

EDC5140I Broken pipe.

Explanation: A write was attempted on a pipe or FIFO for which there was no process to read the data. This message is equivalent to the POSIX.1 EPIPE errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50K

EDC5141I Read-only file system.

Explanation: An attempt was made to modify a file or directory on a file system that was read-only. This message is equivalent to the POSIX.1 EROFS errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50L

EDC5142I Invalid seek.

Explanation: A seek function was issued on a pipe or FIFO. It is invalid to do a positioning operation on a pipe or FIFO. This message is equivalent to the POSIX.1 ESPIPE errno.

Programmer response: Do not attempt a seek function on a device that cannot seek.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50M

EDC5143I No such process.

Explanation: The process ID does not correspond to an existing process. The UID or userid is not defined, or the OMVS segment is not setup correctly. This error may also apply to the process group ID. This message is equivalent to the POSIX.1 ESRCH errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50N

EDC5144I Improper link.

Explanation: A link to a file on another file system was attempted. This message is equivalent to the POSIX.1 EXDEV errno.

Programmer response: Files may be linked only within the same file system. Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50O

EDC5145I Arg list too long.

Explanation: The sum of the number of bytes for the new process image's argument list and the environment list was greater than the system limit.

Programmer response: The system limit is defined by the ARG_MAX run-time invariant value, and can be determined at execution time using the `sysconf()` function. Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50P

EDC5146I Too many levels of symbolic links.

Explanation: Only POSIX_SYMLOOP symlinks are allowed during pathname resolution. This message is equivalent to the POSIX.1 ELOOP errno. POSIX_SYMLOOP is an invariant variable.

Programmer response: Verify that the specified pathname can be resolved, and that no loop exists (of a symbolic link referring to itself). Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50Q

EDC5147I Illegal byte sequence.

Explanation: The string contains an illegal sequence of bytes. For example, an unmatched shift out / shift in condition exists. This message is equivalent to the z/OS UNIX System Services errno, EILSEQ.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50R

EDC5149I Value Overflow Error.

Explanation: A value is too large to be stored in the data type.

Programmer response: Rework the program to reissue the request using a larger data type. For example, if working with large files, rework the program using the _LARGE_FILES feature. This message is equivalent to the errno, EOVERFLOW.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50T

EDC5150I UNIX System Services is not active.

Explanation: The function being requested cannot be performed because the UNIX System Services kernel is not active. This message is equivalent to the EMVSNOTUP errno.

Programmer response: Have the operator start UNIX System Services (START OMVS).

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50U

EDC5151I Dynamic allocation error.

| **Explanation:** The mount of an z/OS UNIX file system data set or VM/ESA Byte File System (BFS) failed in dynamic allocation. The dynamic allocation reason code returned in errno2 is documented in *z/OS MVS Programming: Authorized Assembler Services Guide* or *VM/ESA Application Development Guide: Authorized Assembler Language Programs.* This message is equivalent to the errno, EMVSDYNALC.

| **Programmer response:** Have the system programmer correct the allocation of the z/OS UNIX file system data set or VM/ESA BFS.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC50V

EDC5152I Catalog Volume Access Facility error.

| **Explanation:** The mount of an z/OS UNIX file system data set failed on a catalog error. The catalog reason code returned in errno2 is documented in *MVS/DFP System Programming Reference*. This message is equivalent to the z/OS UNIX System Services errno, EMVSCVAF.

| **Programmer response:** Have the system programmer correct the allocation of the z/OS UNIX file system data set.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC510

EDC5153I Catalog obtain error.

| **Explanation:** The mount of an z/OS UNIX file system data set failed on a catalog error. The catalog reason code returned in errno2 is documented in *MVS/DFP System Programming Reference*. This message is equivalent to the z/OS UNIX System Services errno, EMVSCATLG.

| **Programmer response:** Have the system programmer correct the allocation of the z/OS UNIX file system data set.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC511

EDC5156I Process initialization error.

Explanation: A process initialization error has occurred. A further explanation can be found in *z/OS UNIX System Services Programming: Assembler Callable Services Reference* or *z/VM: OpenExtensions Callable Services Reference*.

Programmer response: Use the function `errno2()` to retrieve the value of the UNIX System Services kernel reason code to determine further information from *z/OS UNIX System Services Programming: Assembler Callable Services Reference* or *z/VM: OpenExtensions Callable Services Reference*.

System action: Process does not start.

Symbolic Feedback Code: EDC514

EDC5157I An internal error has occurred.

Explanation: This message is equivalent to the z/OS UNIX System Services errno, EMVSERR or ECMSERR.

Programmer response: Report this problem to your system programmer.

System action: A system dump is taken of the error and MVS or VM attempts to continue.

Symbolic Feedback Code: EDC515

EDC5158I Bad parameters were passed to the service.

Explanation: A bad parameter was passed to an z/OS UNIX System Services. This message is equivalent to the z/OS UNIX System Services errno, EMVSPARMERR.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC516

EDC5159I The Physical File System encountered a permanent file error.

Explanation: This message is equivalent to the z/OS UNIX System Services errno, EMVSPFSFILERR or ECMSPFSFILE.

Programmer response: Report this problem to your system programmer.

System action: A system dump is taken of the error and MVS or VM attempts to continue.

Symbolic Feedback Code: EDC517

EDC5160I Bad character in environment variable name.

Explanation: An invalid character was found in the 'name' or 'value' string specified on a `getenv()` or `setenv()` function. This message is equivalent to the z/OS UNIX System Services errno, EMVSBADCHAR.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC518

EDC5162I The Physical File System encountered a system error.

Explanation: This message is equivalent to the z/OS UNIX System Services errno, EMVSPFSPERMERR or ECMSPFSPERM.

Programmer response: Report this problem to your system programmer.

System action: A system dump is taken of the error and MVS or VM attempts to continue.

Symbolic Feedback Code: EDC51A

EDC5163I SAF/RACF extract error.

Explanation: An authorization failure occurred when attempting the service. This message is equivalent to the z/OS UNIX System Services errno, EMVSSAFEXTRERR.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC51B

EDC5164I SAF/RACF error.

Explanation: An internal SAF/RACF error occurred. This message is equivalent to the z/OS UNIX System Services errno, EMVSSAF2ERR.

Programmer response: Report this problem to your system programmer.

System action: A system dump is taken of the error and MVS or VM attempts to continue.

Symbolic Feedback Code: EDC51C

EDC5165I System TOD clock not set.

Explanation: The system time of day (TOD) clock is in error, stopped, or in a non-operational state. This message is equivalent to the z/OS UNIX System Services errno, EMVSTODNOTSET.

Programmer response: Report this problem to your system programmer.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC51D

EDC5166I Access mode argument on function call conflicts with PATHOPTS parameter on JCL DD statement.

Explanation: An open or reopen was issued to a DD which specified PATHOPTS. The PATHOPTS specified on the DD conflict with those specified in the function call. This message is equivalent to the z/OS UNIX System Services errno, EMVSPATHOPTS.

Programmer response: Correct either the open/reopen call or the PATHOPTS specified on the DD being opened.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC51E

EDC5167I Access to the UNIX System Services version of the C RTL is denied.

Explanation: An attempt was made to issue an Open C library function that has a dependency on UNIX System Services and the subsystem was not available or at the incorrect release level. This message is equivalent to the errno, EMVSNORTL.

Programmer response: You need to either install the correct level of the subsystem or modify your program to not issue the function or conditionally issue the function.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC51F

EDC5168I Password has expired.

Explanation: The verification request has failed because the password has expired. This message is equivalent to the z/OS UNIX System Services errno, EMVSEXPIRE.

Programmer response: Change the password.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC51G

EDC5169I Password is invalid.

Explanation: The verification or change password request has failed because the supplied password is invalid. This message is equivalent to the z/OS UNIX System Services errno, EMVSPASSWORD.

Programmer response: Correct the supplied password, and retry the request.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC51H

EDC5170I An error was encountered with WLM.

Explanation: A WLM error was detected.

Programmer response: Report the failure to your local administrator for the WLM function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC51I

EDC5171I An error was encountered with CPL.

Explanation: A CPL error was detected.

Programmer response: Report the failure to your local administrator for the CPL function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC51J

EDC5172I An error was encountered with Application Response Measurement (ARM) component.

Explanation: An Application Response Measurement (ARM) error was detected.

Programmer response: The return value of the failed service contains the detailed error code. Try again, when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC51I

EDC5200I The application contains a Language Environment member language that cannot tolerate a fork().

Explanation: An application that uses the `fork()` or `vfork()` functions cannot use other Language Environment member languages that do not support `fork()` or `vfork()`.

Programmer response: Restructure your application so that it contains only high level languages that support `fork()` or `vfork()`. C/C++, COBOL, and PL/I tolerate `fork()` and `vfork()`, but FORTRAN does not. This message can only be issued from within a single-thread environment. In a multithread environment, message EDC5232I is issued.

System action: The `fork()` function is not performed.

Symbolic Feedback Code: EDC52G

EDC5201I The Language Environment message file was not found in the hierarchical file system.

Explanation: For Language Environment messaging to work correctly in a child process, the Language Environment message file must reside in the hierarchical file system.

Programmer response: Define your message file as a hierarchical file.

System action: The `fork()` function is not performed.

Symbolic Feedback Code: EDC52H

EDC5202E DLL facilities are not supported under SPC environment.

Explanation: An SPC application is attempting to use DLL callable services, or the SPC application is compiled with DLL compiler options.

Programmer response: Make sure that `dllload()`, `dllqueryvar()`, `dllqueryfn()`, and `dllfree()` functions are not invoked from your application, or your application is not compiled with the DLL compile-time option.

System action: The request is not completed.

Symbolic Feedback Code: EDC52I

EDC5203E DLL facilities are not supported under POSIX environment.

Explanation: A POSIX application is attempting to use DLL callable services, or the POSIX application is compiled with the DLL compiler option.

Programmer response: Make sure that `dllload()`, `dllqueryvar()`, `dllqueryfn()`, and `dllfree()` functions are not invoked from your application, or your application is not compiled with the DLL compiler-time option.

System action: The request is not completed.

Symbolic Feedback Code: EDC52J

EDC5204E Not enough storage to load DLL module.

Explanation: Not enough storage was available to load the requested DLL load module into virtual storage. If this was an implicit load request, this message is preceded by message EDC6063I that identifies the DLL load module name.

Programmer response: Ensure that the REGION size is large enough to run the application. If necessary, delete modules not currently needed by the application, or free unused storage, and retry the load request.

System action: The DLL module is not loaded.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3500S or CEE3554S message is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC52K

EDC5205S DLL module not found.

Explanation: The system could not find the DLL load module name specified on the parameter list to Language Environment load service, in either the job library or link library. If this was an implicit load request, this message is preceded by message EDC6063I that identifies the DLL load module name.

Programmer response: Ensure that the requested DLL name was correctly modified. Make sure that the job step indicates the correct library. Correct the error and run the job step again.

System action: DLL module is not loaded.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET (and not set to MSG for CEE3501S), a corresponding CEE3501S or CEE3570S message is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC52L

EDC5206S DLL module name too long.

Explanation: The module name length is greater than the name length supported by the underlying operating system. If this was an implicit load request, this message is preceded by message EDC6063I that identifies the DLL load module name.

Programmer response: Correct the module name length and run the job again.

System action: The name length is truncated to the name length supported by the underlying operating system. The requested module may or may not be loaded.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3502S message is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC52M

EDC5207S Load request for DLL load module unsuccessful.

Explanation: The system cannot load the DLL load module. If this was an implicit load request, this message is preceded by message EDC6063I that identifies the DLL load module name.

Programmer response: Check the original abend from the operating system, and refer to the underlying operating system message manual for explanation and programmer's action.

System action: The DLL module is not loaded. The application may abend.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding message is issued to the
| Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition,
| and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC52N

EDC5208I dllHandle supplied to the dllqueryvar() function is not available for use.

Explanation: The dllHandle supplied to the `dllqueryvar()` call is inactive, because the DLL is logically freed by a successful call to `dllfree()`

Programmer response: Ensure that the proper dllHandle is supplied to the `dllqueryvar()` service, or that the subject DLL is not freed prematurely.

System action: The request is ignored.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3572I message is issued to
| the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition,
| and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

Symbolic Feedback Code: EDC52O

EDC5209I No variables exported from this dllHandle.

Explanation: Attempting to query an external variable, but the DLL does not contain any imported variables.

Programmer response: Ensure that the DLL load module indicated in the job library or link library is the correct version, and that it contains the external variable.

System action: The request is ignored.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, an error corresponding message is also issued to
| the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition,
| and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC52P

EDC5210I Requested variable not found in this DLL.

Explanation: Attempting to query an external variable, but the variable name is not found in the export section of the DLL.

Programmer response: Ensure that the variable name specified on the `dllqueryvar()` function call is correct, or that the DLL load module indicated in the job library or link library is the correct version, and that it contains the external variable.

System action: The request is ignored.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3576I or CEE3578I message
| is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling
| a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment
| variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC52Q

EDC5211I DLL load module does not contain a writeable static area.

Explanation: DLL load module that you loaded does not contain any writeable static.

Programmer response: Ensure that the load module name specified is correct, or that the DLL load module indicated in the job library or link library is the correct version. Also check that the DLL load module was built properly.

1. Specify #pragma export in your source, or compile with EXPORTALL compiler option.

2. Compile with DLL, RENT, and LONGNAME compiler options.
3. Prelink.

System action: The request is ignored. Load module is deleted from storage.

Symbolic Feedback Code: EDC52R

EDC5212I dllHandle supplied to dllqueryfn() function is not available for use.

Explanation: The dllHandle supplied to `dllqueryfn()` call is inactive because the DLL is logically freed as a result of a successful call to the `dllfree()` function.

Programmer response: Ensure that the proper dllHandle is supplied to the `dllqueryfn()` function, or that the subject DLL is not freed prematurely.

System action: The request is ignored.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, an error message is also issued to the Language
| Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and the
| turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

Symbolic Feedback Code: EDC52S

EDC5213I No functions exported from this dllHandle.

Explanation: Attempting to query an external function, but the DLL does not contain any imported functions.

Programmer response: Ensure that the DLL load module indicated in the job library or link library is the correct version, and that it contains the external function.

System action: The request is ignored.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3573I message is issued to
| the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition,
| and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC52T

EDC5214I Requested function not found in this DLL.

Explanation: Attempting to query an external function, but the function name is not found in the export section of the DLL.

Programmer response: Ensure that the function name specified on `dllqueryfn()` is correct, or that the DLL load module indicated in the job library or link library is the correct version, and that it contains the external function.

System action: The request is ignored.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3574I or CEE3577I message
| is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling
| a condition, and the turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG
| environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC52U

EDC5215I Not enough storage available for writeable static.

Explanation: Not enough heap storage was available for allocation of writeable static for the DLL load module.

Programmer response: Increase heap size or free unused heap storage.

System action: The request is ignored. Load module is deleted from storage.

Symbolic Feedback Code: EDC52V

EDC5216I dllHandle supplied is NULL.

Explanation: The dllHandle supplied to the dllfree call is invalid.

Programmer response: You must call the dllload service to initialize a dllHandle properly before attempting to free a DLL.

System action: The request is ignored.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a message is also issued to the Language
| Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning
| off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

Symbolic Feedback Code: EDC530

EDC5217I No DLLs to be freed.

Explanation: Attempting to free a DLL, but all DLLs are freed already, or the dllHandle passed is inactive.

Programmer response: Ensure that `dllfree()` is invoked after the call to `dllload()` is completed successfully, or that you have no extra calls to `dllfree()` with this dllHandle in your application.

System action: The request is ignored.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3566I message is issued to
| the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition,
| and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC531

EDC5218I Logical delete performed, but the DLL is not physically deleted.

Explanation: The `dllfree()` function completed successfully. The DLL is not physically deleted because either there is an implicit dllload performed against this DLL by the application, or multiple calls were made to the `dllload()` service.

Programmer response: If the DLL was loaded implicitly by referring to an external variable or an external function, it will be physically deleted by the run-time library at enclave termination. Otherwise, to free the DLL, invoke `dllfree()` with the proper dllHandle.

System action: Execution continues.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3567I message is issued to
| the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition,
| and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC532

EDC5220I Invalid dllHandle.

Explanation: The dllHandle supplied to the dllfree call could not be matched to a DLL loaded by this application.

Programmer response: Ensure that the dllHandle supplied to `dllfree()` is the same as the one returned from `dllload()` accidentally, and that it has not been overwritten.

System action: The request is ignored.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3568I message is issued to
| the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition,
| and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC534

EDC5221S Load request for DLL not supported while running C++ destructors.

Explanation: The application is attempting to load a DLL explicitly while running C++ destructors.

Programmer response: Make sure that you are not invoking `dllload()` from your C++ destructors.

System action: The request is ignored. Execution continues but results are unpredictable.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3563S message is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC535

EDC5222S IOStreams do not support Record Mode I/O.

Explanation: The application is attempting to initialize an IOStreams object to perform Record Mode I/O. Record Mode I/O is not supported in IOStreams objects.

Programmer response: Remove the ″type=record″ specification from the constructor or `open()` function call.

System action: The attempt to initialize the object fails. Execution continues.

Symbolic Feedback Code: EDC536

EDC5223S Too many characters.

Explanation: The application called the `form()` function with a format specifier string that caused `form()` to write past the end of the format buffer. `form()` is provided in stream.h for compatibility.

Programmer response: Split the call to `form()` into two or more calls.

System action: Execution ends.

Symbolic Feedback Code: EDC537

EDC5224S Singularity: log((0,0))

Explanation: The application is attempting to take the log of (0.0, 0.0).

Programmer response: Correct the value passed to `log()` and resubmit.

System action: Execution ends.

Symbolic Feedback Code: EDC538

EDC5225E DLL function is not allowed because destructors are running for the DLL.

Explanation: A dllfree(), dllqueryvar(), or dllqueryfn() function was invoked for a DLL that is currently running destructors. Since destructors are running the DLL is about to be freed. Further function requests using this DLL are not allowed.

Programmer response: Do not issue DLL function requests from one thread while the DLL is being freed from another thread.

System action: The request failed. Application execution continues. The dllfree() function returns a value of 7. The dllqueryvar() and dllqueryfn() functions return a null pointer.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3569E message is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

Symbolic Feedback Code: EDC539

| **EDC5226S A load of DLL from the z/OS UNIX file system failed.**

| **Explanation:** A load attempt for a DLL in the z/OS UNIX file system failed. If this was an implicit load request, this
| message is preceded by message EDC6063I that identifies the DLL load module name.

| **Programmer response:** Verify that the DLL is available in the z/OS UNIX file system and that the application has
| access to the file.

System action: If the DLL was explicitly loaded using the dllload() function, the request fails and the DLL is not loaded. If the DLL was implicitly loaded by reference to a variable or function contained in it, the application is ends with return code 3000.

Symbolic Feedback Code: EDC53A

EDC5227I Buffer is not long enough to contain a path definition.

Explanation: The request for a path definition for the specified ddname cannot be satisfied because the path name length of the path associated with this ddname is greater than the specified buffer length.

Programmer response: Specify a larger buffer.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC53B

EDC5228I The file referred to is an external link.

Explanation: An I/O operation cannot be satisfied because the file referred to is an external link.

Programmer response: Refer to *z/OS UNIX System Services User's Guide* for information on how to perform I/O operations on external links.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC53C

EDC5229I No path definition for ddname in effect.

Explanation: The request to obtain the path definition for the specified ddname cannot be satisfied because no OPENVM PATHDEF CREATE command was issued to associate a BFS path definition with this ddname.

Programmer response: Issue OPENVM PATHDEF CREATE command to associate this ddname with a path name definition.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC53D

EDC5230I ESM error.

Explanation: An internal External Security Manager (ESM) error occurred. This message is equivalent to the z/OS UNIX System Services errno ECMSESMERR.

Programmer response: Report this problem to your system programmer.

System action: Messages are displayed on the file pool server operator console indicating the error and VM processing continues.

Symbolic Feedback Code: EDC53E

EDC5231I CP or the external security manager had an error.

Explanation: An error occurred in CP or the external security manager. This message is equivalent to the z/OS UNIX System Services errno ECPERR.

Programmer response: Try the command again. If the problem persists, report it to your system programmer.

System action: None.

Symbolic Feedback Code: EDC53F

EDC5232I The function failed because it was invoked from a multithread environment.

Explanation: An application may not use the `exec()` or `vfork()` functions from within a multithread environment. The *fork()* function is not permitted within a multithread environment where the application contains a high level language other than C/C++. COBOL and PL/I tolerate *fork()* in a single-thread environment, but not in a multithread environment. A multithread *fork()* is not allowed in a Language Environment preinitialization (CEEPIPI) environment.

Programmer response: Restructure your application so that it is not multithreaded, remove the call to the function, use supported high level languages, and/or avoid using the function in a multithread CEEPIPI environment.

System action: The function is not performed.

Symbolic Feedback Code: EDC53G

EDC5233S The linkage of the specified locale doesn't match the current run-time environment.

Explanation: The application attempted to change the current locale but the linkage attributes for the requested locale do not match the linkage attributes of the run-time environment.

This can happen when an application compiled with the NOXPLINK option runs with the XPLINK(OFF) run-time option but supplies the fully qualified locale path name for an XPLINK locale. XPLINK locales stored in the z/OS UNIX file system have a ″.xplink″ suffix. They are usable only when the XPLINK(ON) run-time option is in effect. This can also happen when an application compiled with the XPLINK option supplies the fully qualified locale path name for a non-XPLINK locale.

Programmer response: This can happen when an application compiled with the NOXPLINK option runs with the XPLINK(OFF) run-time option but supplies the fully qualified locale path name for an XPLINK locale. XPLINK locales stored in the z/OS UNIX file system have a ″.xplink″ suffix. They are usable only when the XPLINK(ON) run-time option is in effect. This can also happen when an application compiled with the XPLINK option supplies the fully qualified locale path name for a non-XPLINK locale.

System action: This can happen when an application compiled with the NOXPLINK option runs with the XPLINK(OFF) run-time option but supplies the fully qualified locale path name for an XPLINK locale. XPLINK locales stored in the z/OS UNIX file system have a ″.xplink″ suffix. They are usable only when the XPLINK(ON) run-time option is in effect. This can also happen when an application compiled with the XPLINK option supplies the fully qualified locale path name for a non-XPLINK locale.

EDC5234S The DLL cannot be loaded because it does not contain a CEESTART CSECT.

Explanation: The application is attempting an explicit load of a DLL that does not contain a CEESTART CSECT.

Programmer response: Make sure that when you generate the DLL for which the `dllload()` service is failing, it contains a CEESTART CSECT.

System action: The function is not performed.

If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3550S message is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

EDC5235S The fetched executable does not contain a fetchable entry point.

Explanation: Specifically, one or more of the following errors might have occurred:

- The application is attempting to `fetch()` an executable that has been compiled with the XPLINK option but does not have a fetchable entry point.
- The application is attempting to `fetch()` a non-XPLINK executable that has CEESTART as the entry point, but does not contain a fetchable entry point.

Programmer response: Fetched XPLINK executables, as well as non-XPLINK executables that have CEESTART as the entry point, must have a fetchable entry point specified using the #pragma linkage(..., fetchable) directive.

System action: The function is not performed.

EDC5236S The fetched executable was compiled XPLINK but the XPLINK environment is not active.

Explanation: The application is attempting to `fetch()` an executable that has been compiled with the XPLINK option but the current run-time environment does not support XPLINK executables.

Programmer response: XPLINK executables that are to be fetched must have the XPLINK C run-time environment active. This can be achieved by specifying the XPLINK(ON) run-time option. Another alternative is to compile your main program with the XPLINK compiler option. If main() is compiled XPLINK then the XPLINK C run-time environment will be active by default, and you may also have the added benefit of enhanced performance. For more information on XPLINK, refer to *z/OS Language Environment Programming Guide*

System action: The function is not performed.

EDC5237S The DLL was not found in an authorized library.

Explanation: The application is authorized and is attempting to explicitly load a DLL, but that DLL could not be found in an authorized library or concatenation of libraries.

Programmer response: Make sure the module exists in a system or user-defined authorized library. Correct the error, and run the application again.

System action: The function is not performed.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3518S message is issued to
| the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition,
| and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

EDC5238E Not enough storage to fetch module.

Explanation: Not enough storage was available to fetch the requested load module into virtual storage.

Programmer response: Ensure that the REGION size is large enough to run the application. If necessary, delete modules not currently needed by the application, or free unused storage, and retry the fetch function.

System action: The module is not fetched.

EDC5239S Fetched module not found.

Explanation: The system could not find the load module name specified to the fetch function in either the job library or link library.

Programmer response: Ensure that the requested fetch name was correctly specified. Make sure that the job step indicates the correct library. Correct the error and run the job step again.

System action: The module is not fetched.

EDC5240S Fetched module name too long.

Explanation: The length of the name specified on the `fetch()` function is greater than the name length supported by the underlying operating system.

Programmer response: Correct the module name length and run the job again.

System action: The name length is truncated to the name length supported by the underlying operating system. The requested module may or may not be loaded.

EDC5241S Load request for fetch load module unsuccessful.

Explanation: The system cannot load the load module specified on the `fetch()` function.

Programmer response: Check the original abend from the operating system, and refer to the underlying operating system message manual for explanation and programmer's action.

System action: The fetched module is not loaded. The application may abend.

EDC5242S The fetched module was not found in an authorized library.

Explanation: The application is authorized and is attempting to fetch a module, but that module could not be found in an authorized library or concatenation of libraries.

Programmer response: Make sure the module exists in a system or user-defined authorized library. Correct the error, and run the application again.

System action: The function is not performed.

EDC5244I The program, module or DLL is not supported in this environment.

Explanation: The program, module or DLL specified on the function call contains one or more languages that are not supported on this version of the operating system.

Programmer response: Use a supported program, module or DLL for this function when running on this version of the operating system.

System action: The function is not performed.

| If this is a C dllload() request, and the _EDC_DLL_DIAG environment variable is not set to QUIET, an error message is sent to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| If this is a C dllload() request, a CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

EDC5245I Data is not valid.

Explanation: The parameter is not the correct size or contains an incorrect value. For the Enterprise Identity Mapping services, the EIM return code structure does not meet the minimum size requirement. See EIM_RC_MIN_SIZE in <eim.h> for the minimum size.

Programmer response: Correct the parameter and try the service again.

System action: The invoked function fails.

EDC5246I Unknown system state.

Explanation: A program exception occurred or an unexpected error was detected by an underlying service. EIM applications may receive this errno when an LDAP service returns an unexpected error or a problem was detected in the value of an EIM attribute.

Programmer response: Correct the problem and try the service again. Applications using Enterprise Identity Mapping will receive an informative error string to aid with diagnosing the problem.

System action: The invoked function fails.

EDC5247I Operation not supported.

Explanation: The requested function is not supported on z/OS.

Programmer response: Replace the function with support that is available on z/OS.

System action: The invoked function fails.

EDC5248I The object name specified is not correct

Explanation: The name specified is not valid. EIM applications receive this errno when the domain, registry, or identifier name is not valid or the caller does not have enough permissions to perform the requested functions.

Programmer response: Verify the name exists in the EIM domain and that the caller has the necessary permissions to access the specified LDAP data.

System action: The invoked function fails.

EDC5249I The function is not allowed.

Explanation: The requested function could not be performed because the data has not been properly removed. An EIM application receives this errno when it attempts to remove a system registry when it has an application registry or when it attempts to remove a domain when the domain is not empty.

Programmer response: Identify the functions that must be performed before retrying the service again. EIM applications must remove the application registry before attempting to remove a system registry and empty the domain of identifiers and associations before attempting to delete a domain.

System action: The invoked function fails.

EDC5250I The utmpx database format cannot be read.

Explanation: The requested function was not able to determine the record format in use by the UTMPX database. UTMPX interfaces depend on the version indicator in the first record of the UTMPX file.

Programmer response: The version member of the first record in the UTMPX file contains invalid data. Check for other applications that write to the UTMPX database but that do not use `pututxline()` to manage the `write()`.

System action: The invoked function fails.

EDC5251I Input dllHandle not permitted for this DLL function.

Explanation: The dllHandle supplied to this DLL function call could not be matched to a dllHandle returned from dllload().

| **Programmer response:** Ensure that the dllHandle supplied to `dllfree()`, `dllqueryfn()`, or `dllqueryvar()` is the
| same as the one returned from `dllload()`. Make sure that the dllHandle was not returned from from dlopen().

System action: The request is ignored.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, an error is issued to the Language Environment
| message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off
| _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

EDC5252S A call was made from a NOXPLINK-compiled application to an XPLINK-compiled exported function in DLL and the XPLINK(ON) runtime option was not specified.

Explanation: During Language Environment initialization, XPLINK resources need to be allocated if any XPLINK-compiled functions are going to be called during the execution of the application. Language Environment tries to detect this by inspecting the attributes of the initial program. If the initial program consists of NOXPLINK-compiled functions that may at some point call an XPLINK-compiled function, then the XPLINK(ON) runtime option must be used to indicate to Language Environment initialization that XPLINK resources should be allocated.

Programmer response: Specify the XPLINK(ON) runtime option.

System action: If this was an implicit DLL reference, the condition is signaled. If the condition is not handled, the default action is to terminate the enclave. If this was an explicit DLL Load (CEEPLDE) or DLL Open (CEEPOPDL) request, the feedback code is returned to the caller. If this was an explicit C dlopen() request, this error is returned to the caller through a subsequent call to the C dlerror() function.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3555S message is issued to
| the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition,
| and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

EDC5253S An AMODE64 application is attempting to load an AMODE31 DLL load module.

Explanation: An application with AMODE=64 is attempting to load a DLL load module link-edited as an AMODE=31 load module.

Programmer response: Ensure that the AMODE of the application and the AMODE of the DLL load module are the same. Re-link your DLL load module to have AMODE=64.

System action: DLL module is not loaded. The invoked function fails.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3587S message is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

EDC5254S An AMODE31 application is attempting to load an AMODE64 DLL load module.

Explanation: An application with AMODE=31 is attempting to load a DLL load module link-edited as an AMODE=64 load module.

Programmer response: Ensure that the AMODE of the application and the AMODE of the DLL load module are the same. Re-link your DLL load module to have AMODE=31.

System action: DLL module is not loaded. The invoked function fails.

| If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding CEE3588S message is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

| A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

EDC5255S An AMODE31 application is attempting to fetch() an AMODE64 executable.

Explanation: An application with AMODE=31 is attempting to fetch an executable link-edited as an AMODE=64 program object.

Programmer response: Only applications that have been linked AMODE=64 can fetch an executable that is AMODE=64. Re-link either your application or your fetched executable to have the correct matching AMODE.

System action: Fetched executable is not loaded. The fetch() function fails.

EDC5256S An AMODE64 application is attempting to fetch() an AMODE31 executable.

Explanation: An application with AMODE=64 is attempting to fetch an executable link-edited as an AMODE=31 program object.

Programmer response: Only applications that have been linked AMODE=31 can fetch an executable that is AMODE=31. Re-link either your application or your fetched executable to have the correct matching AMODE.

System action: Fetched executable is not loaded. The fetch() function fails.

EDC5257I Function cannot be called in the child process of a *fork()* **from a multithreaded process until** *exec()* **is called.**

Explanation: Only async-safe functions are supported in this environment.

Programmer response: Restructure your application so that it is not multithreaded at the time of the *fork()*, or only use async-safe functions in the child process up until *exec()* is called. Refer to the *z/OS XL C/C++ Run-Time Library Reference* for the list of async-safe functions.

System action: The function is not performed.

Symbolic Feedback Code: EDC549

| **EDC5258I A CUN_RS_NO_UNI_ENV error was issued by Unicode Services.**

Explanation: An iconv() function received a CUN_RS_NO_UNI_ENV error from z/OS Unicode Services.

Programmer response: Refer to Support for z/OS Unicode: z/OS Unicode Services documentation for user action.

| **EDC5259I A CUN_RS_NO_CONVERSION error was issued by Unicode Services.**

Explanation: An iconv() function received a CUN_RS_NO_CONVERSION error from z/OS Unicode Services.

Programmer response: Refer to Support for z/OS Unicode: z/OS Unicode Services documentation for user action.

| **EDC5260I A CUN_RS_TABLE_NOT_ALIGNED error was issued by Unicode Services.**

Explanation: An iconv() function received a CUN_RS_TABLE_NOT_ALIGNED error from z/OS Unicode Services.

Programmer response: Refer to Support for z/OS Unicode: z/OS Unicode Services documentation for user action.

| **EDC5261S There is not enough storage to obtain a DLL function or variable pointer.**

Explanation: There is insufficient heap storage to satisfy a dlsym(), dllqueryfn(), or dllqueryvar() request.

Programmer response: Increase the region size.

System action: The function is not performed.

If the _EDC_DLL_DIAG environment variable is not set to QUIET, a corresponding message is issued to the Language Environment message file. Other DLL diagnostic options, such as issuing ctrace(), signaling a condition, and turning off _EDC_DLL_DIAG diagnostics, are available through the _EDC_DLL_DIAG environment variable.

A CEEDLLF DLL failure control block is populated with further DLL error diagnostics.

| **EDC5262I An iconv() function encountered an unexpected error while using z/OS Unicode Services.**

Explanation: An iconv() function encountered an unexpected error while using z/OS Unicode Services.

Programmer response: Refer to message EDC6258 for additional information.

EDC6000E The raise() function was issued for the signal SIGFPE.

Explanation: The program has invoked the `raise()` function with the SIGFPE signal specified and the default action specified.

Programmer response: None.

System action: The program is terminated and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000(MVS) or 3000000(VM) is returned.

Symbolic Feedback Code: EDC5RG

EDC6001E The raise() function was issued for the signal SIGILL.

Explanation: The program has invoked the `raise()` function with the SIGILL signal specified and the default action specified.

Programmer response: None.

System action: The program is terminated and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000(MVS) or 3000000(VM) is returned.

Symbolic Feedback Code: EDC5RH

EDC6002E The raise() function was issued for the signal SIGSEGV.

Explanation: The program has invoked the `raise()` function with the SIGSEGV signal specified and the default action specified.

Programmer response: None.

System action: The program will be terminated and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000(MVS) or 3000000(VM) is returned.

Symbolic Feedback Code: EDC5RI

EDC6003E The raise() function was issued for the signal SIGABND.

Explanation: The program has invoked the `raise()` function with the SIGABND signal specified and the default action specified.

Programmer response: None.

System action: The program will be terminated and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000(MVS) or 3000000(VM) is returned.

Symbolic Feedback Code: EDC5RJ

EDC6004E The raise() function was issued for the signal SIGTERM.

Explanation: The program has invoked the `raise()` function with the SIGTERM signal specified and the default action specified.

Programmer response: None.

System action: The program will be terminated and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000(MVS) or 3000000(VM) is returned.

Symbolic Feedback Code: EDC5RK

EDC6005E The raise() function was issued for the signal SIGINT.

Explanation: The program has invoked the `raise()` function with the SIGINT signal specified and the default action specified.

Programmer response: None.

System action: The program will be terminated and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000(MVS) or 3000000(VM) is returned.

Symbolic Feedback Code: EDC5RL

EDC6006E The raise() function was issued for the signal SIGABRT.

Explanation: The program has invoked the `raise()` function with the SIGABRT signal specified and the default action specified.

Programmer response: None.

System action: The program will be terminated and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 2000(MVS) or 2000000(VM) is returned.

Symbolic Feedback Code: EDC5RM

EDC6007E The raise() function was issued for the signal SIGUSR1.

Explanation: The program has invoked the `raise()` function with the SIGUSR1 signal specified and the default action specified.

Programmer response: None.

System action: The program will be terminated and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000(MVS) or 3000000(VM) is returned.

Symbolic Feedback Code: EDC5RN

EDC6008E The raise() function was issued for the signal SIGUSR2.

Explanation: The program has invoked the `raise()` function with the SIGUSR2 signal specified and the default action specified.

Programmer response: None.

System action: The program will be terminated and a traceback or dump is issued, depending on the

TERMTHDACT run-time option. A return code of 3000(MVS) or 3000000(VM) is returned.

Symbolic Feedback Code: EDC5RO

EDC6009E The raise() function was issued for the signal SIGIOERR.

Explanation: The program has invoked the `raise()` function with the SIGIOERR signal specified and the default action specified.

Programmer response: None.

System action: The program will be terminated and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000(MVS) or 3000000(VM) is returned.

Symbolic Feedback Code: EDC5RP

EDC6010E An object was thrown which was not caught by any catch clauses.

Explanation: An object was thrown for which no `catch` clauses exist to catch it.

Programmer response: None.

System action: The program ends abnormally.

Symbolic Feedback Code: EDC5RQ

EDC6052S An AMODE 24 application is attempting to load an AMODE 31 DLL load module.

Explanation: An application with AMODE=24 is attempting to load a DLL load module link-edited as an AMODE=31 load module.

Programmer response: Ensure that the AMODE of the application matches that of the run-time library. Language Environment no longer supports C applications in AMODE=24. Relink the application to have AMODE=31.

System action: The application ends with return code 3000.

Symbolic Feedback Code: EDC5T4

EDC6053S An AMODE 31 application is attempting to load an AMODE 24 DLL load module.

Explanation: An application with AMODE=31 is attempting to load a DLL load module link-edited as a load module with AMODE=24.

Programmer response: Ensure that the AMODE of the application and the AMODE of the DLL load module are the same. Re-link your DLL load module to have AMODE=31.

System action: Application is terminated with return code 3000.

Symbolic Feedback Code: EDC5T5

EDC6054S External variable is not found in DLL load module.

Explanation: The application is attempting to refer to an external variable that is not defined in the DLL load module.

Programmer response: Ensure that the DLL load module indicated in the job library or link library is the correct version, and that it contains the external variable.

System action: The application ends with return code 3000.

Symbolic Feedback Code: EDC5T6

EDC6055S External function is not found in DLL load module.

Explanation: The application is attempting to refer to an external function that is not defined in the DLL load module.

Programmer response: Ensure that the DLL load module indicated in the job library or link library is the correct version, and that it contains the external function.

System action: The application ends with return code 3000.

Symbolic Feedback Code: EDC5T7

EDC6056S Attempting to load a DLL while running C++ destructors.

Explanation: The application is attempting to load a DLL implicitly while running C++ destructors.

Programmer response: Make sure that you are not referring to variables or functions implicitly from your C++ destructors.

System action: The application ends with return code 3000.

Symbolic Feedback Code: EDC5T8

EDC6057S A DLL load module that you are attempting to load does not contain a CEESTART csect.

Explanation: The application is attempting to load a DLL load module implicitly or explicitly, but the CEESTART csect cannot be located within it.

Programmer response: Make sure that when you generate the DLL load module, it contains a CEESTART csect.

System action: The application ends with return code 3000.

Symbolic Feedback Code: EDC5T9

EDC6058S There is not enough heap storage to load DLL.

Explanation: There is insufficient heap storage to satisfy DLL load request.

Programmer response: Increase the allocation of your application heap storage by using the heap run-time option.

System action: The application ends with return code 3000.

Symbolic Feedback Code: EDC5TA

EDC6059S The DLL that you have loaded does not export any variables or functions.

Explanation: A DLL was loaded successfully, but the DLL does not export any variables or functions. Either the definition side-deck supplied to your application is incorrect, or the DLL load module is generated incorrectly.

Programmer response: Ensure that the DLL load module was built properly.

1. Specify #pragma export in your source or compile with EXPORTALL compiler option.
2. Compile with DLL, RENT, and LONGNAME compiler options.
3. Prelink.

System action: The application ends with return code 3000.

Symbolic Feedback Code: EDC5TB

EDC6060S The DLL that you are attempting to load does not contain any C functions.

Explanation: A DLL was loaded successfully, but the load module does not contain any C functions.

Programmer response: Make sure that you are loading the correct load module, and that the DLL load module is built correctly.

System action: The application ends with return code 3000.

Symbolic Feedback Code: EDC5TC

EDC6061S You are attempting to load DLLs that are in a circular list.

Explanation: The run-time library discovered a deadlock condition while processing a DLL load request. The deadlock condition exists because the DLLs that are being loaded depend on each other. The following situation illustrates a deadlock condition. DLL A has static constructors that require objects from DLL B. DLL B has static constructors that require objects from DLL A. When DLL A is loaded, its static constructors require objects from DLL B. This forces DLL B to be loaded, requiring objects from DLL A. Since the loading of DLL A has not completed, a deadlock condition exists.

Programmer response: Remove the circular list dependency from the DLLs.

System action: The application ends with return code 3000.

Symbolic Feedback Code: EDC5TD

EDC6062S DLL constructors or destructors did not complete, so DLL cannot be used.

Explanation: A DLL being loaded or freed was in the process of running static constructors or destructors but the process did not complete (probably because the thread was abnormally terminated). The DLL is left in an indeterminate state. Another thread that was waiting for the constructors or destructors to complete while attempting a load or free of the same DLL detected this error.

Programmer response: Determine the cause of the incomplete constructor or destructor process. Ensure that the constructors or destructors are not the cause of the thread termination that lead to this condition.

System action: The application ends with return code 3000.

Symbolic Feedback Code: EDC5TE

EDC6063I DLL name is *dll_name*.

Explanation: This message accompanies other DLL load error messages (for example EDC5205S). It identifies the name of the DLL for which the load failed.

Programmer response: Refer to the accompanying DLL error message.

System action: None.

Symbolic Feedback Code: EDC5TF

EDC6064I Not enough storage was available to use MEMCHECK vendor heap manager.

Explanation: The MEMCHECK vendor heap manager was loaded but there is insufficient storage to manage this trace and diagnostic tool.

Programmer response: If possible, free unneeded heap storage or Increase the size of the storage.

System action: None.

EDC6200E An invalid argument list was specified.

Explanation: The parameter list specified to DLLRNAME is invalid. See *z/OS XL C/C++ User's Guide* for proper syntax.

Programmer response: Ensure that:

1. At least 1 input module or DLL is specified.
2. The syntax of the parameters matches listed syntax for environment.
3. Valid options are specified correctly.

System action: DLLRNAME: fails with return code 8.

Symbolic Feedback Code: EDC61O

EDC6201S A failure occurred accessing @1.

Explanation: An unexpected error occurred when DLLRNAME accessed an input file.

Programmer response: Look up subsequent error message and perform Programmer Response if possible (for example, file not found error might mean to fix input file name). Otherwise, report the problem to your IBM Support Center.

System action: DLLRNAME prints out a perror() message then terminates with rc=16.

Symbolic Feedback Code: EDC61P

EDC6202S A DLL named ″ @1 ″ is already imported.

Explanation: A DLLRNAME operation has found that an old DLL name to be renamed is being renamed to a new name that is already imported in the current module being processed.

Programmer response: User has specified DLL to rename, but the new name chosen matches a DLL in the import list.

System action: DLLRNAME fails with rc=12.

Symbolic Feedback Code: EDC61Q

EDC6203E A DLL name was specified more than once for a rename.

Explanation: On a DLLRNAME, user has specified a DLL name twice in the ″oldname=newname″ list. Examples are: (A=B,A=C or A=C,B=C or A=B,B=C or A=B,C=A or A=A).

Programmer response: Fix the argument list. A DLL cannot appear twice in the argument list.

System action: DLLRNAME fails with rc=8.

Symbolic Feedback Code: EDC61R

EDC6204E No argument list was provided.

Explanation: User did not provide any argument list.

Programmer response: An argument list must be provided (for example, through SYSIN or standard streams redirection).

System action: DLLRNAME fails with rc=8.

Symbolic Feedback Code: EDC61S

EDC6251C The library function setjmp(), _setjmp(), sigsetjmp(), getcontext(), or swapcontext() failed when it tried to use the passed in jmp_buf, sigjmp_buf or ucontext_t area.

Explanation: The library function failed when it tried to initialize the passed-in buffer. A program check occurred, perhaps because the address of the passed-in buffer was not correct.

Programmer response: Make sure that the address of the buffer passed into `setjmp()`, `_setjmp()`, `sigsetjmp()`, `getcontext()`, or `swapcontext()` is correct.

System action: The application ends.

EDC6252C The library function longjmp(), _longjmp(), siglongjmp(), setcontext(), or swapcontext() failed when it tried to use the passed in jmp_buf, sigjmp_buf or ucontext_t area.

Explanation: The library function failed when it tried to use the the passed-in buffer. Either the data in the buffer was incorrect, or a program check occurred because the address of the passed-in buffer was not correct.

Programmer response: Make sure that the address of the buffer passed into `setjmp()`, `_setjmp()`, `sigsetjmp()`, `setcontext()`, or `swapcontext()` is correct. Make sure that the buffer has been initialized by `setjmp()`, `_setjmp()`, `sigsetjmp()`, or `getcontext()` before it is passed to `longjmp()`, `_longjmp()`, `siglongjmp()`, or `setcontext()`, or `swapcontext()`.

EDC6253C An error occurred attempting to retrieve the C++ state variables table from the PPA1.

Explanation: An invalid C++ PPA1 state variables locator was detected.

Programmer response: Contact your IBM Support Center.

EDC6254C ASCII applications require a POSIX(ON) environment. Change environment to POSIX(ON).

Explanation: ASCII C-RTL support does not support a POSIX(OFF) environment.

Programmer response: Change environment to POSIX(ON).

EDC6255C ASCII/EBCDIC mode change failed.

Explanation: ASCII/EBCDIC mode change failed. An internal error occurred.

Programmer response: Contact your IBM Support Center.

EDC6256C ASCII initialization failed.

Explanation: Insufficient memory available.

Programmer response: Increase the amount of storage available to the application.

EDC6257C There is a socket problem, syslogd may need to be restarted. Until the condition is resolved, all messages will be lost.

Explanation: syslog() has not been able to send a message for at least seven minutes. There is a problem with the socket, syslogd() likely needs to be restarted.

Programmer response: Restart syslogd().

| **EDC6258I An iconv() function encountered an unexpected error while using Unicode Conversion Services. A return code of <return code> and reason code of <reason code> were returned from Unicode Services.**

| **Explanation:** An iconv() function encountered an unexpected error while using Unicode Services.

| **Programmer response:** Refer to Support for Unicode: Unicode Services for user action.

| **EDC6259S This function is not supported running on hardware that does not have the Decimal Floating Point Facility installed.**

| **Explanation:** The invoked Decimal Floating Point function cannot be used unless the hardware has the Decimal Floating Point Facility installed.

| **Programmer response:** Before using this function, make sure that the hardware has the Decimal Floating Point Facility installed.

| **System action:** If unhandled, the enclave is terminated.

| **Symbolic Feedback Code:** EDC63J

EDC7000C Signal delivery has failed because the service BPX1SIA failed.

Explanation: The callable service, BPX1SIA (`sigaction()`), unexpectedly returned a failure code. This service was invoked by the library during delivery of a signal to a user catcher function.

Programmer response: Contact your IBM Support Center.

System action: The application ends.

Symbolic Feedback Code: EDC6QO

EDC7001C Signal delivery has failed because the service BPX1SPM failed.

Explanation: The callable service BPX1SPM (`sigprocmask()`) unexpectedly returned a failure code. This service was invoked by the library during delivery of a signal to a user catcher function.

Programmer response: Contact your IBM Support Center.

System action: The application ends.

Symbolic Feedback Code: EDC6QP

EDC7002C Signal delivery has failed because the MVS service CSRL16J failed.

Explanation: The MVS callable service CSRL16J unexpectedly returned a failure code. This service was invoked by the library following the return from a user signal catcher function.

Programmer response: Contact your IBM Support Center.

System action: The application ends.

Symbolic Feedback Code: EDC6QQ

EDC7003C Invalid signal received from the z/OS UNIX System Services kernel.

Explanation: The library has been interrupted by the z/OS UNIX System Services kernel to perform default signal processing. However, the signal was not one of the supported types (SIGHUP, SIGINT, SIGABRT, SIGILL, SIGFPE, SIGSEGV, SIGPIPE, SIGALRM, SIGTERM, SIGUSR1, SIGUSR2, SIGABND, SIGQUIT, or SIGTRAP).

Programmer response: Contact your IBM Support Center.

System action: The application ends.

Symbolic Feedback Code: EDC6QR

EDC7004C The library function sigsetjmp() or siglongjmp() failed because the service BPX1SPM failed.

Explanation: The callable service BPX1SPM (`sigprocmask()`) unexpectedly returned a failure code. The library was attempting to save or restore the signal mask as part of the `sigsetjmp()` or `siglongjmp()` functions.

Programmer response: Contact your IBM Support Center.

System action: The application ends.

Symbolic Feedback Code: EDC6QS

EDC7005E The getopt() function detected an invalid option character *option_char* when it was invoked from program *program_name*.

Explanation: The `getopt()` function detected that an option character that was parsed was not one of the recognized set of specified option characters.

Programmer response: Respecify a recognized option character.

System action: The `getopt()` function returns the character in error. The application continues to run.

Symbolic Feedback Code: EDC6QT

EDC7006E The getopt() function detected an option character *option_char* that is missing an argument when it was invoked from program *program_name*.

Explanation: The `getopt()` function encountered an option character that required an option-argument, but the option-argument was not found.

Programmer response: Respecify the option character with an option-argument.

System action: The `getopt()` function returns the character in error. The application continues to run.

Symbolic Feedback Code: EDC6QU

EDC7007C No memory available for the random() function family internal structure.

Explanation: The initialization routine for the `random()` function family was unable to allocate memory for the internal structure used by the functions.

Programmer response: Reduce memory use and try again.

System action: The application ends.

Symbolic Feedback Code: EDC6QV

EDC7008E No previous regular expression.

Explanation: The `re_comp()` function was invoked with either a null pointer argument or a null regular expression, and a compiled regular expression does not currently exist.

Programmer response: Invoke the `re_comp()` with a valid regular expression.

System action: The `re_comp()` function returns with a pointer to this error message. The application continues to run.

Symbolic Feedback Code: EDC6R0

EDC7009E Regular expression too long.

Explanation: The input regular expression for the `re_comp()` function is too long. The compiled regular expression cannot fit in the internal work buffer, which is of limited size.

Programmer response: Invoke the `re_comp()` with a shorter regular expression.

System action: The `re_comp()` function returns with a pointer to this error message. The application continues to run.

Symbolic Feedback Code: EDC6R1

EDC7010E *paren_pair* **imbalance.**

Explanation: The `re_comp()` function detected an error in the input regular expression. The character sequences **\(** (left parenthesis) were found without a matching **\)** (right parenthesis), or vice versa.

Programmer response: Correct the regular expression pattern and retry the `re_comp()`.

System action: The `re_comp()` function returns with a pointer to this error message. The application continues to run.

Symbolic Feedback Code: EDC6R2

EDC7011E *brace_pair* **imbalance.**

Explanation: The `re_comp()` function detected an error in the input regular expression. The character sequences **\{** (left brace) were found without a matching **\}** (right brace), or vice versa.

Programmer response: Correct the regular expression pattern and retry the `re_comp()`.

System action: The `re_comp()` function returns with a pointer to this error message. The application continues to run.

Symbolic Feedback Code: EDC6R3

EDC7012E *square_bracket* **imbalance.**

Explanation: The `re_comp()` function detected an error in the input regular expression. The left square bracket **[** was found without a matching right square bracket **]**.

Programmer response: Correct the regular expression pattern and retry the `re_comp()`.

System action: The `re_comp()` function returns with a pointer to this error message. The application continues to run.

Symbolic Feedback Code: EDC6R4

EDC7013E Too many *paren_pair* pairs.

Explanation: The `re_comp()` function detected an error in the input regular expression. Too many \(\) sub-expression pairs were specified. Up to nine such \(\) pairs are allowed.

Programmer response: Correct the regular expression pattern and retry the `re_comp()`.

System action: The `re_comp()` function returns with a pointer to this error message. The application continues to run.

Symbolic Feedback Code: EDC6R5

EDC7014E Incorrect range values in *brace_pair.*

Explanation: The `re_comp()` function detected an error in the input regular expression. The repetition interval specified within the \{*m,n*\} is incorrect. Specifically, one or more of the following errors may have occurred:

- One or more numbers within the \{\} are too large. They must be less than 256.
- Bad numbers (for example, non-numeric values) are used as range values.
- More than two numbers are given within the \{\}.
- First number exceeds the second number within the \{\}.

Programmer response: Correct the regular expression pattern and retry the `re_comp()`.

System action: The `re_comp()` function returns with a pointer to this error message. The application continues to run.

Symbolic Feedback Code: EDC6R6

EDC7015E Back-reference number in *backslash* digit incorrect.

Explanation: The `re_comp()` function detected an error in the input regular expression. The back-reference number, *digit*, in *digit* is incorrect. This value must be between 1 and 9 (inclusive), and must correspond to one of the earlier bracketed sub-expressions (that is, sub-expressions enclosed in \(\)). The expression is invalid if less than *digit* sub-expressions precede the *digit*. For example, if five bracketed sub-expressions are defined in the regular expression, then it is valid to refer to them by specifying from \1 to \5. However, it is incorrect to specify \6, \7, \8, or \9.

Programmer response: Correct the regular expression pattern and retry the `re_comp()`.

System action: The `re_comp()` function returns with a pointer to this error message. The application continues to run.

Symbolic Feedback Code: EDC6R7

EDC7016E Incorrect endpoint in range expression.

Explanation: The `re_comp()` function detected an error in the input regular expression. The ending range point in a *range expression* must collate equal to or higher than the starting range point. For example, it is an error to specify [d-a].

Programmer response: Correct the regular expression pattern and retry the `re_comp()`.

System action: The `re_comp()` function returns with a pointer to this error message. The application continues to run.

Symbolic Feedback Code: EDC6R8

EDC7022I USERID:

Explanation: A userid was not specified, while executing the REXEC command. If a userid was specified, it was not found to be valid at the host, when searching the $HOME/.netrc file. A userid must be input at the invocation of this message.

System action: System waits for user input of userid.

Symbolic Feedback Code: EDC6RE

EDC7023I PASSWORD:

Explanation: A password was not specified, while executing the REXEC command. If a password was specified, it was not found to be valid at the host, when searching the $HOME/.netrc file. A password must be input at the invocation of this message.

System action: System waits for user input.

Symbolic Feedback Code: EDC6RF

EDC7024I $HOME/.netrc file cannot be opened.

Explanation: The $HOME/.netrc file cannot be opened for an FOPEN error other than ENONET.

System action: The system continues, asking the user to enter the user ID and password.

Symbolic Feedback Code: EDC6RG

EDC7025I fstat() failed on $HOME/.netrc file.

Explanation: An fstat() was performed on the $HOME/.netrc file, and had an unsuccessful return code.

System action: The system stops trying to find the user ID and password through the $HOME/.netrc file. The user must enter them instead.

Symbolic Feedback Code: EDC6RH

EDC7026I $HOME/.netrc file is not in the correct mode.

Explanation: If the $HOME/.netrc file contains a login password, the file's permissions must be set to 600 (read and write by owner only). The system detected that the $HOME/.netrc file was not set to 600.

System action: The system asks the user to enter the user ID and and password.

Symbolic Feedback Code: EDC6RI

EDC7027I Remove password or correct $HOME/.netrc mode.

Explanation: This message follows EDC7026 and advises the user how to correct the problem with the $HOME/.netrc file.

System action: The system asks the user to enter a user ID and password.

Symbolic Feedback Code: EDC6RJ

EDC7028I Unknown $HOME/.netrc option.

Explanation: The system has successfully examined the $HOME/.netrc file for the user ID. However, the rest of the $HOME/.netrc file is in a syntax that the system cannot understand.

System action: The system stops trying to find the password in the $HOME/.netrc file. The user must then enter it instead.

Symbolic Feedback Code: EDC6RK

EDC7100E *errval* **: error unknown**

Explanation: An unrecognized XTI error value was passed to t_error or t_strerror.

System action: Pass a valid XTI error value to the function.

Symbolic Feedback Code: EDC61O

EDC7101I incorrect addr format

Explanation: A transport address was passed to an XTI function which had an invalid format.

System action: The function fails. A correct address should be passed to the function.

Symbolic Feedback Code: EDC61P

EDC7102I incorrect option format

Explanation: An option buffer was passed to an XTI function which had inconsistent length indication or contained an invalid option value.

System action: The function fails. An option buffer with valid format should be passed to the function.

Symbolic Feedback Code: EDC61Q

EDC7103I incorrect permissions

Explanation: An XTI caller tried to change a transport option for which they lacked privilege.

System action: The function fails. The caller should not attempt to change the option while operating without adequate privilege.

Symbolic Feedback Code: EDC61R

EDC7104I illegal transport fd

Explanation: The descriptor did not refer to a valid XTI transport endpoint.

System action: The function fails. Pass a descriptor referring to a valid transport endpoint.

Symbolic Feedback Code: EDC61S

EDC7105I couldn’t allocate addr

Explanation: The XTI transport provider couldn’t allocate a transport address.

System action: The function fails. Reattempt when addresses are available.

Symbolic Feedback Code: EDC61T

EDC7106I out of state

Explanation: A transport endpoint was not in a valid state for the function to be performed.

System action: The function fails. Manipulate the endpoint to bring it into the correct state before reattempting.

Symbolic Feedback Code: EDC61U

EDC7107I bad call sequence number

Explanation: An invalid sequence number was specified in a t_accept call.

System action: The function fails. Specify a valid sequence number.

Symbolic Feedback Code: EDC61V

EDC7108I system error

Explanation: A system error occurred during the execution of an XTI function.

System action: The function fails. Correct the underlying problem.

Symbolic Feedback Code: EDC620

EDC7109I event requires attention

Explanation: An event on an XTI endpoint requires attention.

System action: The function fails. Call t_look to process the event.

Symbolic Feedback Code: EDC621

EDC7110I illegal amount of data

Explanation: An invalid amount of user data was passed to an XTI function.

System action: The function fails. Correct the amount of data passed.

Symbolic Feedback Code: EDC622

EDC7111I buffer not large enough

Explanation: The buffer provided to return a value from an XTI function was not large enough.

System action: The function fails. Pass a larger return buffer.

Symbolic Feedback Code: EDC623

EDC7112I flow control

Explanation: O_NONBLOCK was set in a call to an XTI function to send data, but the transport flow control mechanism prevented the transport provider from accepting any data at this time.

System action: The function fails. Call the function again when the flow control condition no longer exists.

Symbolic Feedback Code: EDC624

EDC7113I no data

Explanation: An XTI function to accept a connection or receive data was called with O_NONBLOCK set on the endpoint, and no connection/data was pending.

System action: The function fails. Retry.

Symbolic Feedback Code: EDC625

EDC7114I discon_ind not found on queue

Explanation: No disconnect indication was found on the specified XTI endpoint.

System action: The function fails. Retry.

Symbolic Feedback Code: EDC626

EDC7115I unitdata error not found

Explanation: No unitdata error was found on the specified XTI endpoint.

System action: The function fails. Retry.

Symbolic Feedback Code: EDC627

EDC7116I bad flags

Explanation: An invalid flags value was passed to t_optmgmt.

System action: The function fails. Retry with a valid flags value.

Symbolic Feedback Code: EDC628

EDC7117I no ord rel found on queue

Explanation: No orderly release indication was found on the specified XTI endpoint.

System action: The function fails. Retry.

Symbolic Feedback Code: EDC629

EDC7118I primitive/action not supported

Explanation: An operation unsupported by the underlying transport provider was requested.

System action: The function fails.

Symbolic Feedback Code: EDC62A

EDC7119I state is in process of changing

Explanation: An operation was requested on an XTI endpoint whose state was in the process of changing.

System action: The function fails. Retry.

Symbolic Feedback Code: EDC62B

EDC7120I unsupported struct-type requested

Explanation: Allocation of an unsupported XTI structure type was requested from t_alloc.

System action: The function fails. Pass a correct structure type.

Symbolic Feedback Code: EDC62C

EDC7121I invalid transport provider name

Explanation: An invalid transport provider name was specified when attempting to open an XTI endpoint.

System action: The function fails. Specify a valid transport provider.

Symbolic Feedback Code: EDC62D

EDC7122I qlen is zero

Explanation: An attempt was made to listen on an XTI endpoint whose connection queue length is zero.

System action: The function fails. Specify an endpoint with a non-zero queue length.

Symbolic Feedback Code: EDC62E

EDC7123I address in use

Explanation: An attempt was made to bind to a transport address which is already in use.

System action: The function fails. Specify an address which is available.

Symbolic Feedback Code: EDC62F

EDC7124I outstanding connection indications

Explanation: The XTI endpoint specified for both fd and resfd in a call to t_accept has outstanding connect requests.

System action: The function fails. Specify an endpoint with no outstanding connect requests.

Symbolic Feedback Code: EDC62G

EDC7125I transport provider mismatch

Explanation: The listening and responding endpoints in a t_accept call do not refer to the same transport provider.

System action: The function fails. Specify two endpoints which both refer to the same transport provider.

Symbolic Feedback Code: EDC62H

EDC7126I resfd specified to accept w/qlen >0

Explanation: The XTI endpoint specified as resfd to t_accept is a passive endpoint.

System action: The function fails. Specify an endpoint with zero queue length.

Symbolic Feedback Code: EDC62I

EDC7127I resfd not bound to same addr as fd

Explanation: The responding endpoint in a call to t_accept is not bound to the same address as the listening endpoint.

System action: The function fails. Specify two endpoints both bound to the same address.

Symbolic Feedback Code: EDC62J

EDC7128I incoming connection queue full

Explanation: The connection queue of the endpoint specified in a call to t_listen is full.

System action: The function fails. Accept pending connections on the endpoint and retry.

Symbolic Feedback Code: EDC62K

EDC7129I XTI protocol error

Explanation: A communication problem has been detected between XTI and the transport provider to which an endpoint refers.

System action: The function fails. Refer to diagnostic procedures for the transport provider.

Symbolic Feedback Code: EDC62L

EDC8000I A bad socket-call constant was found in the IUCV header.

Explanation: A problem has occurred between MVS or VM and TCP/IP.

Programmer response: Record this error and report the failure using your local procedure to report failures to the IBM Service support contact.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7Q0

EDC8001I An error was found in the IUCV header.

Explanation: An error was found in the IUCV header, such as a bad length.

Programmer response: Record this error and report the failure using your local procedure to report failures to the IBM Service support contact.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7Q1

EDC8002I A socket descriptor is out of range.

Explanation: A socket number assigned by client interface code (for `socket()` and `accept()`) is out of range.

Programmer response: Record this error and report the failure using your local procedure to report failures to the IBM Service support contact.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7Q2

EDC8003I A socket descriptor is in use.

Explanation: A socket number assigned by client interface code is already in use.

Programmer response: Record this error and report the failure using your local procedure to report failures to the IBM Service support contact.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7Q3

EDC8004I Request failed because of an IUCV error.

Explanation: The request failed because of IUCV error. This error is generated by the client stub code.

Programmer response: Record this error and report the failure using your local procedure to report failures to the IBM Service support contact.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7Q4

EDC8005I Offload box error.

Explanation: A problem has occurred between MVS and TCP/IP.

Programmer response: Record this error and report the failure using your local procedure to report failures to the IBM Service support contact.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7Q5

EDC8006I Offload box restarted.

Explanation: A problem has occurred between MVS and TCP/IP.

Programmer response: Record this error and report the failure using your local procedure to report failures to the IBM Service support contact.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7Q6

EDC8007I Offload box down.

Explanation: A problem has occurred between MVS and TCP/IP.

Programmer response: Record this error and report the failure using your local procedure to report failures to the IBM Service support contact.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7Q7

EDC8008I Already a conflicting call outstanding on socket.

Explanation: A problem has occurred between MVS and TCP/IP.

Programmer response: Record this error and report the failure using your local procedure to report failures to the IBM Service support contact.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7Q8

EDC8009I Request cancelled using a SOCKcallCANCEL request.

Explanation: A problem has occurred between MVS and TCP/IP.

Programmer response: Record this error and report the failure using your local procedure to report failures to the IBM Service support contact.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7Q9

EDC8011I A name of a PFS was specified that either is not configured or is not a Sockets PFS.

Explanation: A problem has occurred between MVS and TCP/IP.

Programmer response: Record this error and report the failure using your local procedure to report failures to the IBM Service support contact.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7QB

EDC8100I Block device required.

Explanation: A non-block file was specified when a block device is required.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7T4

EDC8101I Text file busy.

Explanation: An attempt is made to run a pure-procedure program that is currently open for writing or reading. It also occurs when an attempt is made to open for writing, or to remove, a pure-procedure program or shared library while that program or library is being run.

Programmer response: Proceed with cleanup of the application resources; then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7T5

EDC8102I Operation would block.

Explanation: An operation on a socket marked as non-blocking has encountered a situation, such as no data available, that otherwise would have caused the function to suspend execution.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7T6

EDC8103I Operation now in progress.

Explanation: The socket was marked O_NDELAY or O_NONBLOCK using `fcntl()`, and the connection cannot be immediately established.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7T7

EDC8104I Connection already in progress.

Explanation: A connection or disconnection request is already in progress for the specified socket.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7T8

EDC8105I Socket operation on non-socket.

Explanation: The file descriptor does not refer to a socket.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7T9

EDC8106I Destination address required.

Explanation: The socket operation failed because a destination address was not provided. No bind address was established.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TA

EDC8107I Message too long.

Explanation: The socket data transfer failed because the message exceeded the size limits. A message sent on a transport provider was longer than an internal message buffer or some other network limit.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TB

EDC8108I Protocol wrong type for socket.

Explanation: Either the two sockets to be connected are not of the same type, or the protocol used does not support this type of socket.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TC

EDC8109I Protocol not available.

Explanation: The protocol option specified to `setsockopt()` is not supported by this implementation.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TD

EDC8110I Protocol not supported.

Explanation: This protocol is not supported by the address family, or the protocol is not supported by this implementation.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TE

EDC8111I Socket type not supported.

Explanation: The type of socket specified is not supported. Do not use this type of socket in your program.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TF

EDC8112I Operation not supported on socket.

Explanation: This socket, with its particular type, domain, and protocol, does not allow the requested operation.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TG

EDC8113I Protocol family not supported.

Explanation: The socket protocol specified is not supported. Do not use this protocol in your program.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TH

EDC8114I Address family not supported.

Explanation: This implementation does not support the specified address family, or the specified address is not valid for the address family of the specified socket.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TI

EDC8115I Address already in use.

Explanation: A bind or connect operation was attempted using a socket name that is already in use.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TJ

EDC8116I Address not available.

Explanation: The requested socket address is not available to this machine. Either an incorrect socket address was used, or there is a problem at the remote node where the socket address should be.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TK

EDC8117I Network is down.

Explanation: A socket operation failed because the network is not available. The local interface to use or reach the destination is not available.

Programmer response: Proceed with cleanup of the application resources and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TL

EDC8118I Network is unreachable.

Explanation: A socket operation failed because the destination is at a remote node that cannot be reached over the network. No route to the network exists.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TM

EDC8119I Network dropped connection on reset.

Explanation: The host to which the socket was connected went down. The connection can be reestablished after the remote node is restarted.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TN

EDC8120I Connection ended abnormally.

Explanation: The connection between a socket and a remote node was terminated at the local node, the remote node, or the network level.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TO

EDC8121I Connection reset.

Explanation: The connection was forcibly closed by the peer. This errno can be set because of an error, or because of a connection that was closed.

Programmer response: Proceed with cleanup of the application resources and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TP

EDC8122I No buffer space available.

Explanation: Not enough buffer space is available in the system to perform the requested socket operation.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TQ

EDC8123I Socket already connected.

Explanation: A connect operation was attempted on a socket that is already connected.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TR

EDC8124I Socket not connected.

Explanation: A socket operation, other than a connect, was attempted on a socket that is not currently connected, or a send operation that does not require a connection was attempted without a destination address.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TS

EDC8125I Can't send after socket shutdown.

Explanation: An attempt was made to send data after a socket was shut down.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TT

EDC8126I Too many references; can't splice.

Explanation: Too many references have been specified.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TU

EDC8127I Connection timed out.

Explanation: A remote socket did not respond within the timeout period set by the protocol of the socket on this node. If the connection timed out during execution of the function that reported this error (as opposed to timing out before the function being called), results are unpredictable.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7TV

EDC8128I Connection refused.

Explanation: A remote node refused to allow the attempted connect operation. The attempt to connect to a socket was refused because there was no process listening, or because the queue of connection requests was full and the underlying protocol does not support retransmissions.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7U0

EDC8129I Host is not available.

Explanation: A socket operation failed because the remote node specified is not available.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7U1

EDC8130I Host cannot be reached.

Explanation: A socket operation failed because no route to the remote node was available because of an incorrect address, an incorrect routing table, or network hardware problems.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7U2

EDC8131I Too many processes.

Explanation: The system process limit has been exceeded.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7U3

EDC8132I Too many users.

Explanation: The maximum number of users has been reached.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7U4

EDC8133I Disk quota exceeded.

Explanation: A write to an ordinary file, the creation of a directory or symbolic link, or the creation of a directory entry failed because the user's quota of disk blocks is exhausted.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7U5

EDC8134I Stale file handle.

Explanation: The current directory, the root directory, or a file descriptor to a file refers to a file system that is no longer accessible. This error may be caused by the local or remote file system being unmounted, or by a remote file server disabling currently open file handles for implementation-defined reasons.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7U6

EDC8136I File is not a STREAM.

Explanation: A STREAM operation was attempted on a file descriptor which was not associated with a STREAM.

Programmer response: Report the failure to your local administrator.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7U8

EDC8137I STREAMS ioctl() timeout.

Explanation: The timer set for a STREAMS `ioctl()` call has expired. The cause of this error is device-specific and indicates either a hardware or software failure, or a timeout value that is too short for the specific operation. The status of the `ioctl()` operation is unpredictable.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7U9

EDC8138I No STREAMS resources.

Explanation: Insufficient STREAMS memory resources are available to perform a STREAMS-related function. This is a temporary condition; recovery is possible if other processes release resources.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7UA

EDC8139I The message identified by set_id and msg_id is not in the message catalog.

Explanation: This message is equivalent to the ENOMSG errno.

Programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7UB

EDC8140I Bad message.

Explanation: During a `read()`, `getmsg()`, or `ioctl()` I_RECVFD request to a STREAMS device, a message arrived at the head of the STREAMS that is inappropriate for the function receiving the message:

- `read()`—The message waiting to be read on a STREAMS is not a data message.
- `getmsg()`—A file descriptor was received instead of a control message.
- `ioctl()`—Control or data information was received instead of a file descriptor when I_RECVFD was specified.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7UC

EDC8141I Identifier removed.

Explanation: Returned during interprocess communication if an identifier has been removed from the system.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7UD

EDC8144I The link has been severed.

Explanation: This error may be reported by a function that refers to a remote file, when the communications link to the server for that resource has been lost, any file descriptor associated with this remote file should not be used for future I/O.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7UG

EDC8148I Protocol error.

Explanation: A protocol error occurred. This error is device-specific, but is usually not caused by a hardware failure.

Programmer response: Proceed with cleanup of the application resources, and then close the socket. When the socket has been freed, the application may begin the process again.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7UK

EDC8149I Multihop not allowed.

Explanation: For a function that has a pathname as one of its arguments, resolution of that pathname requires multihop access to a remote resource, and multihop access is not supported by the underlying implementation.

Programmer response: Report the failure to your local administrator for the TCP/IP function. Try the application again when the problem has been corrected.

System action: The request fails. The application continues to run.

Symbolic Feedback Code: EDC7UL

| **EDC8152I The asynchronous I/O request has been canceled.**

| **Explanation:** An asynchronous operation was in blocking state and has been canceled with aio_cancel().

| **Programmer response:** Avoid using aio_cancel() to allow operations to wait or complete.

| **System action:** The asynchronous I/O request fails with ECANCELED.

EDC8159I Function call was interrupted before any data was received.

Explanation: An asynchronous signal was caught by the (POSIX) process during the execution of an interruptible function, and the signal handler (or default action) resulted in a normal return. This caused the interrupted function to return this errno. The signal arrived after the socket connection was established but before any data was received over the connection.

Programmer response: See *z/OS XL C/C++ Run-Time Library Reference* for information about possible side effects of interrupting the function.

System action: The request fails and no data is returned. The socket connection is established. The application continues to run.

Symbolic Feedback Code: EDC7UV

EDC8160I Socket reuse is not supported.

Explanation: An attempt was made to reuse the specified socket for this function. Reuse of this socket by this function is not allowed.

Programmer response: See *z/OS XL C/C++ Run-Time Library Reference* for the function being attempted for the specific reason for failure, and for any side effects from the function.

System action: The socket is not reused. Refer to the *z/OS XL C/C++ Run-Time Library Reference* for more information on how each function reacts to this error. The application continues to run.

Symbolic Feedback Code: EDC7V0

EDC8161I The file system cannot currently be moved.

Explanation: The function call is attempting to move a file system which currently cannot be moved. This occurs in a USS sysplex environment. One reason is that an application has issued byte range locks for one or more files in the file system. The move cannot succeed until the files are closed and/or the byte range locks are released.

Programmer response: Ensure the program has no byte range locks outstanding. It may also be necessary to close any open files which had byte range locks. If that does not help, ask the system programmer to use the F BPXOINIT,FILESYS..., console command to determine the state of the file system and to perform file system diagnosis. All applications which hold byte range locks may also need to be determined.

System action: The file system is not moved.

EDC9500I An unknown error occurred.

Explanation: Function `getaddrinfo()` or `getnameinfo()` failed. The reason is unknown.

Programmer response: Report the problem to your system administrator.

System action: The gai_strerror() function returns this message when it is given an error code that is not one of the documented return values from either the getaddrinfo() or getnameinfo() function.

EDC9501I The name does not resolve for the supplied parameters.

Explanation: Function `getaddrinfo()` or `getnameinfo()` failed. For `getnameinfo()`, flag NI_NAMEREQD is set and the host's name cannot be located, or both node and service names were null. For `getaddrinfo()`, both nodename and servname were null.

Programmer response: For getaddrinfo(), correct the function call to supply either or both the nodename and servname parameters. For getnameinfo(), either the node or service name supplied cannot be located and should be verified for a correct name.

System action: The getaddrinfo() or getnameinfo() function fails. The value EAI_NONAME is returned.

EDC9502I The name or address could not be resolved at this time. Future attempts may succeed.

Explanation: Function `getaddrinfo()` or `getnameinfo()` failed. The host address specified for `getnameinfo()` or the host name specified for `getaddrinfo()` could not be resolved in the configured time interval or the system resolver address space has not been started.

Programmer response: Retry your request at a later time.

System action: The getaddrinfo() or getnameinfo() function fails. The value EAI_AGAIN is returned.

EDC9503I A non-recoverable error occurred when attempting to resolve the name or address.

Explanation: Function `getaddrinfo()` or `getnameinfo()` failed. The host address specified for `getnameinfo()` or the host name specified for `getaddrinfo()` could not be resolved.

Programmer response: Report the problem to your system administrator.

System action: The getaddrinfo() or getnameinfo() function fails. The value EAI_FAIL is returned.

| **EDC9504I An argument buffer overflowed.**

| **Explanation:** Function getnameinfo() failed. The buffer pointed to by the node or service argument for getnameinfo()
| was too small.

| **Programmer response:** Correct the size of the argument buffer.

| **System action:** The getnameinfo() function fails. The value EAI_OVERFLOW is returned.

EDC9505I The address family was not recognized or the address length was invalid for the specified family.

Explanation: Function `getaddrinfo()` or `getnameinfo()` failed. The address family information was incorrect.

Programmer response: Consult the *z/OS XL C/C++ Run-Time Library Reference* for the valid address families allowed and correct the application to use a supported address family and its correct length.

System action: The getaddrinfo() or getnameinfo() function fails. The value EAI_FAMILY is returned.

EDC9506I There was a memory allocation failure.

Explanation: Function `getaddrinfo()` or `getnameinfo()` failed. The service was unable to obtain the memory necessary to build the return information.

Programmer response: Report the problem to your system administrator.

System action: The `getaddrinfo()` or `getnameinfo()` function fails. The value EAI_MEMORY is returned.

EDC9507I The flags parameter had an invalid value.

Explanation: Function `getaddrinfo()` or `getnameinfo()` failed. For the `getaddrinfo()` function, the *ai_flags* member of the hints structure specified an invalid combination of flags. For the `getnameinfo()` function, the *flags* parameter specified an invalid combination of flags.

Programmer response: Consult the *z/OS XL C/C++ Run-Time Library Reference* for the valid flags allowed and correct the application to use a proper combination of flags.

System action: The `getaddrinfo()` or `getnameinfo()` function fails. The value EAI_BADFLAGS is returned.

EDC9508I The service passed was not recognized for the specified socket type.

Explanation: Function `getaddrinfo()` failed. The servname parameter specified on `getaddrinfo()` does not support ai_socktype member specified in the hints parameter.

Programmer response: Correct the servname parameter or the ai_socktype value. If these are correct, then contact your system administrator.

System action: The `getaddrinfo()` function fails. The value EAI_SERVICE is returned.

EDC9509I The intended socket type was not recognized.

Explanation: Function `getaddrinfo()` fails. The `getaddrinfo()` function does not support the socket type specified in the ai_socktype member of the hints parameter.

Programmer response: Correct the ai_socktype value. If it is correct, then contact your system administrator.

System action: The `getaddrinfo()` function fails. The value EAI_SOCKTYPE is returned.

EDC9510I A system error occurred. The error code can be found in errno.

Explanation: Function `getaddrinfo()` or `getnameinfo()` failed. The request failed because of a system error.

Programmer response: Retrieve the errno value and use the information it supplies to correct the problem.

System action: The `getaddrinfo()` or `getnameinfo()` function fails. The value EAI_SYSTEM is returned.

Symbolic Feedback Code:

Chapter 5. Fortran Run-Time Messages

This topic shows the ranges of Fortran message numbers by message type, and explains qualifying data, permissible resume actions, and locator-text in the Fortran messages. Finally, the list of the Fortran messages is given.

Fortran Run-Time Message Number Ranges

Component or Language Element	**Range of Message Numbers**
(Reserved)	0000–0099
Service Subroutines	0100–0299
Common Blocks	0300–0339
Operator Messages	0340–0344
(Reserved)	0345–0400
Run-time Environment	0401–0499
Implicit Routines	0500–0599
Intrinsic Functions	0600–0699
(Reserved)	0700–0999
I/O	1000–1999
Input Conversion	1000–1019
Sequential I/O	1020–1069
Direct I/O	1070–1099
Keyed I/O	1100–1179
Formatted I/O	1180–1199
Unformatted I/O	1200–1209
List Directed I/O	1210–1219
Namelist I/O	1220–1249
Striped I/O	1250–1269
Asynchronous I/O	1270–1329
VSAM I/O	1330–1339
INQUIRE	1340–1359
CLOSE semantics	1360–1379
OPEN / DEFINE FILE semantics	1380–1449
(Reserved)	1450–1499
System-detected errors	1500–1549
Command / Macro / Service failure	1550–1599
File Disconnection	1900–1909
End of Data	1910–1914
Invalid unit	1915–1919
Miscellaneous	1920–1999
Multitasking Facility (MTF)	2000–2099
AUTOTASK	2000–2029
AUTOTASK DD Statement	2030–2039
Function invalid	2040–2049
Miscellaneous	2050–2099
Vector	2100–2119
Run-Time Options	2120–2129
Miscellaneous	2130–2199
Separation Tool	2200–2249
Static Debug	2250–2279
Miscellaneous	2280–2999
(Reserved)	3000–9999

Qualifying Data

Many of the messages listed have a section called Qualifying Data describing qualifying data (q_data) associated with the condition. (Qualifying data (or q_data) consists of variables that contain information about the occurrence of a particular condition, such as the input values to the service that detected the condition. This information is useful for a condition handler to determine what corrective actions to take.)

Many of the conditions have q_data descriptors (indicated by data type Q_DATA_DESC) as part of their qualifying data. A q_data descriptor indicates the data type and length of the immediately following element of qualifying data.

The first qualifying data for any condition is *parm-count*, the total number of elements of qualifying data including *parm-count* itself, associated with that condition token.

For I/O errors, the first four qualifying data are defined as shown in Table 1:

Table 1. Basic Set of Qualifying Data for I/O Conditions

No.	Name	Input/Output	Type	Value
1	*parm-count*	Input	INTEGER*4	The total number of elements of qualifying data including this one. If there is no additional qualifying data beyond the basic set shown here, then this value is 4. Otherwise, it includes the first four shown here plus whatever additional qualifying data is applicable to the condition.
2	*statement*	Input	CHARACTER*12	The name of the I/O statement being processed.
3	*unit*	Input	INTEGER*4	−1 if the I/O statement is directed to an internal file; otherwise, the unit number specified on the I/O statement.
4	*file*	Input	CHARACTER*62	Blank if if the I/O statement refers to an internal file or if the name of the file is not provided as part of the message text for this condition. Otherwise, a structure, whose contents are shown below, which gives the name of the external file to which the I/O statement refers.

When the *file* qualifying data is not blank, it gives the file name of the file involved in the I/O statement, and has the following format:

Position	Length	Contents
1	8	The ddname for the file.
9	1	A code that indicates whether the file is identified in the Fortran program either by its ddname or by its data set name. A value of blank indicates that the file is referred to through its ddname. Any non-blank character indicates that it is referred to by its data set name.

For z/OS (when position 9 is other than blank):

9	44	Data set name
53	8	PDS member name (or blank if not a PDS)
61	2	Not used

For more information on qualifying data, see *z/OS Language Environment Programming Guide*.

Permissible Resume Actions

Many of the messages listed have a section called Permissible Resume Actions describing which resume actions a user condition handler can request when the resume cursor has not been moved.

The following table shows the names (that is, the two-character codes) of the resume actions. It also contains a description of the values that the user condition handler must set for the indicated parameters to request that resume action. (*result_code* and *new_condition* are defined in *z/OS Language Environment Programming Guide*.)

Name	**Corresponding Resume Action**	*result_code* **Parameter**	*new_condition* **Parameter**
RN	Resume without moving the resume cursor	10	—
RI	Resume with new input value	60	CEE0CE
RO	Resume with new output value	60	CEE0CF

If a user condition handler requests a resume action that is not listed as one of the permissible resume actions for the condition being processed, either the condition CEE088 (invalid request for the resume action) or the condition CEE087 (invalid request for the fix-up and resume action) is signaled. If a user condition handler attempts to resume without moving the resume cursor for the condition CEE088 or CEE087, the condition is percolated to the next condition handler to avoid a program loop.

Regardless of what is listed for a message under Permissible Resume Actions, you can always move the resume cursor by invoking the callable service CEEMRCE and then requesting the resume action. In this case, the only actions that are taken are those described under System Action; none of those listed under Permissible Resume Actions is taken.

Name	**Resume Action**	*result_code* **Parameter**	*new_condition* **Parameter**
—	Resume after moving the resume cursor	10	—

locator-text in the Run-Time Message Texts

In many message texts for conditions involving I/O statements, *locator-text* is shown as part of the message text. This *locator-text* identifies the Fortran statement for which the error was detected and can be one of the following:

- The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed.
- The *statement* statement for an internal file failed.

- An error occurred during enclave termination.
- The *statement* statement for unit *unit-number* failed.
- The INQUIRE statement failed.

List of Run-Time Messages

The messages listed pertain to Fortran. Messages are followed by an explanation describing the condition that caused the message (except for those messages for which the message text is self-explanatory), a programmer response suggesting how you might prevent the message from occurring again, and a system action indicating how the system responds to the condition that caused the message.

The messages also contain a symbolic feedback code, which represents the first 8 bytes of a 12-byte condition token. You can think of the symbolic feedback code as the nickname for a condition. As such, the symbolic feedback code can be used in user-written condition handlers to screen for a given condition, even if it occurs at different locations in an application.

Some messages also contain qualifying data and permissible resume actions, as discussed in "Qualifying Data" on page 228, and "Permissible Resume Actions" on page 229, respectively. The VS FORTRAN Version 2 error number is shown for those messages that existed in VS FORTRAN Version 2.

The messages in this topic contain alphabetic suffixes that have the following meaning:

I	Informational message
W	Warning message
E	Error message
S	Severe error message
C	Critical error message

FOR0096W The symbol table in storage was corrupted and couldn't be used to produce a dump.

Explanation: During the printing of a dump, a Language Environment routine detected an inconsistency in a symbol table, which contains information about the type and location of the variables in a Fortran program unit. Most likely the symbol table in virtual storage was overlaid by some routine (but not necessarily by the routine with the overlaid symbol table).

Programmer response: Determine and correct the cause of the overlaid symbol table. In Fortran program units, this is often caused by:

- Using subscripts that reference virtual storage outside the declared bounds of an array.
- Referring to variables that are in EQUIVALENCE statements when the variables are declared to overlay too much storage.
- Referring to storage that's addressed through a pointer whose value isn't properly established.
- In a CALL statement or function reference, providing actual arguments that are not consistent with the dummy arguments declared in the subprogram. The actual arguments could be of the wrong type, rank, or have the wrong array bounds. There could be an incorrect number of actual arguments.

System action: The dump or the remainder of the dump for this program unit is not created.

Symbolic Feedback Code: FOR0096

FOR0100S The DIV callable service *service-name* for the dynamic common block *common-name* failed. A *macro-name* macro instruction had a system completion (abend) code of *abend-code*, and a reason code of *reason-code*. Seek assistance from your Language Environment support personnel. VS FORTRAN Version Error 2 Number: AFB143I-1

Explanation: The DIV callable service *service-name* failed because the macro instruction *macro-name*, which was used internally by Language Environment, failed. The system completion (abend) codes and reason codes are described in *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.

Programmer response: Refer to the one of the publications listed under “Explanation” for the cause of the error. You might require the assistance of your Language Environment support personnel to resolve many of these errors.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is not completed, a return code of 128 is set, and execution resumes.

Symbolic Feedback Code: FOR0100

FOR0101S **The DIV callable service** *service-name* **failed for the dynamic common block** *common-name***. A** *macro-name* **macro instruction had a return code of** *return-code***, and a reason code of** *reason-code***. Seek assistance from your Language Environment support personnel. VS FORTRAN Version Error 2 Number: AFB143I-2**

Explanation: The DIV callable service *service-name* failed because the macro instruction *macro-name*, which was used internally by Language Environment, failed. The return codes and reason codes are described in *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.

Programmer response: Refer to the one of the publications listed under “Explanation” for the cause of the error. You might require the assistance of your Language Environment support personnel to resolve many of these errors.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is not completed, a return code of 128 is set, and execution resumes.

Symbolic Feedback Code: FOR0101

FOR0102S **The DIV callable service** *service-name* **failed. The return code was** *return-code***. VS FORTRAN Version Error 2 Number: AFB144I-1, AFB144I-2**

Explanation: The DIV callable service *service-name* failed because of one of the errors in the table, which is identified by the return code *return-code*. The arguments mentioned in the explanations are those described for the DIV callable services in *VS FORTRAN Version 2 Language and Library Reference*.

Return Code	Explanation
8	The value of the *dyncom* argument wasn’t the name of a dynamic common block.
12	The value of the *type* argument was neither DDNAME, DSNAME, nor DSN.
16	If the value of the *type* argument was DSNAME or DSN, the value of the *access* argument was neither READ nor READWRITE. If the value of the *type* argument was DDNAME, the value of the *access* argument was neither READ, READWRITE, nor blank.
20	The value of the *access* argument was READ, but the data object was empty.
24	The object specified by the *divobj* argument was already associated with a dynamic common block or an object ID through a different ddname.
28	The ddname or data set name given as the *divobj* argument did not refer to a VSAM linear data set.
32	The value supplied for *divobj* argument was not a valid ddname or data set name (as determined by the value of the *type* argument).

Return Code	Explanation
36	The value of the *divobj* argument conflicted with the value of the *type* argument. For example, this return code could indicate that the *type* argument had a value of DDNAME, and the *divobj* had a value that could only be a data set name rather than a ddname.
40	The data set that had the name given as the value of the *divobj* argument and that should have been a VSAM linear data set could not be dynamically allocated, possibly because it didn't exist.
44	The dynamic common block whose name was given as the value of the *dyncom* argument was already associated with another data object through a the use of the DIVINF or DIVVWV callable service.
48	The argument list passed to the DIV callable service was invalid for one or more of these reasons: • The call was made from a program compiled by the VS FORTRAN Version 1 or the VS FORTRAN Version 2 compiler with the the LANGLVL(66) compiler option. • The call was made from a program compiled by the VS FORTRAN Version 1 compiler at a level before Release 3. • The call was made from a program compiled by the FORTRAN IV H Extended or the FORTRAN IV G1 compiler. • An incorrect number of arguments was provided. • One or more of the arguments wasn't of the type required by the callable service. • The call was made from an assembler language program, and the arguments were not provided in the form required when there are character arguments.
52	The value of the *mapnum* argument implied a range in the data object that overlaps a range that was already mapped.
56	The value of the *obj-id* argument did not have an association with any data object.
60	The value of the *offset* argument was negative.
64	The dynamic common block whose name was given as the value of the *dyncom* argument was not associated with any data object.
68	The DIVSAV callable service was invoked, but the data object associated with the dynamic common whose name was given as the value of the *dyncom* argument was not accessed using a value of READWRITE for the *access* argument.
72	The value of the *mapnum* argument was zero or negative.
76	The DIV callable service was called from within an MTF parallel subroutine.

Programmer response: Based on the return code identified by *return-code*, take the action indicated. The arguments mentioned are those described for the DIV callable services in *VS FORTRAN Version 2 Language and Library Reference*.

Return Code	Explanation
8	Specify the name of the common block as one of the suboptions of the DC compiler option.
12	Change the value of the *type* argument to DDNAME, DSNAME, or DSN depending on whether a ddname or a data set name is given as the value of the *divobj* argument. (Lowercase characters are allowed.)
16	Change the value of the *access* argument to READ or READWRITE. (Lowercase characters are allowed.)
20	Ensure that name given as the *divobj* argument refers to the data object (VSAM linear data set) that was intended, or change the value of the *access* argument to READWRITE.
24	Remove this call to the DIVINF or DIVINV callable service if an existing association can be used. Alternatively, terminate the existing association using the DIVTRF or DIVTRV callable service before calling DIVINF or DIVINV.
28	Ensure that the ddname or the data set name given as the *divobj* argument refers to a VSAM linear data set.

Return Code	Explanation
32	Ensure that the value of the *divobj* argument is a valid ddname or a data set name that refers to a VSAM linear data set. Also ensure that it is correctly specified as either a ddname or a data set name in the *type* argument.
36	If the *type* argument has a value of DDNAME, then ensure that a ddname referring to a VSAM linear data set is given as the value of the *divobj* argument. If the *type* has a value of DSNAME or DSN, then ensure that a data set name of a VSAM linear data set is given as the value of the *divobj* argument. Change either or both of these arguments to make them consistent.
40	Ensure that the data set name refers to a VSAM linear data set, which can be created using Access Method Services.
44	Make one or more of these changes: • Remove the call to the DIVINF or DIVVWV if an existing association can be used. • Use a different dynamic common block name. • First terminate the existing association using the DIVTRF or DIVTRV callable service.
48	If the calling program is written in Fortran, compile it with the VS FORTRAN Version 2 compiler, and do not specify the LANGLVL(66) compiler option. If it is written in assembler language, use the Fortran conventions for argument lists with character arguments. These conventions are described in the topic “Passing Character Arguments Using the Standard Linkage Convention” in Appendix B of *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.
52	Use a different value for the *mapnum* argument to avoid overlapping an existing mapping of the data object. Use the DIVCML callable service if necessary to determine the length of the dynamic common blocks so that overlapping mappings can be avoided.
56	Ensure that the value of the *obj-id* argument is the same as what was returned by a previous call to the DIVINV callable service. Also ensure that the previous call to the DIVINV callable service completely successfully. If it’s possible that a user-written condition handler requested that execution resume in the event of an error, then provide logic to handle the nonzero return code.
60	Provide a value for the *offset* argument that is not less than 0.
64	Ensure that the name given as the value of the *dyncom* argument has been associated with a data object using the DIVINF callable service. Also ensure that the previous call to the DIVINF callable service completely successfully by checking the return code if it’s possible that a user-written condition handler requested that resumption of execution occur.
68	If the changes made in the dynamic common block are to be saved in the data object, then ensure that the value of the *access* argument in the call to the DIVINF or DIVINV callable service is READWRITE. If the changes are not to be saved, then remove the call to the DIVSAV callable service.
72	Provide a positive value for the *mapnum* argument. Also see the actions for return code 52.
76	Restructure the application so that there are no calls to the data-in-virtual callable service in MTF parallel subroutines. However, these services can be used in the main task program, and the SHRCOM callable service can be used to allow sharing of the dynamic common blocks among the main task program and the parallel subroutines.

System action: The service is not completed, and the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	3
2	*subroutine-name*	Input	CHARACTER*8	The name of the DIV subroutine
3	*return-code*	Input	INTEGER*4	The return code from the Fortran DIV subroutine.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is not completed, a return code of *return code* is set, and execution resumes.

Symbolic Feedback Code: FOR0102

FOR0103W The DIV callable service *service-name* completed successfully, but the dynamic common block *common-name* had a length of *length*, which was not a multiple of 4096.

Programmer response: If dynamic common block *common-name* will be modified and the changes saved in the data object or if you want to avoid the signaling of this condition, change the declarations of the variables in *common-name* such that the length of this common block becomes an exact multiple of 4096 (4096, 8192, 12288, and so on). Otherwise, you can ignore this condition.

System action: The service is completed and execution resumes.

Qualifying Data: None

Permissable Resume Actions:

Name	Action Taken after Resumption
RN	The service is not completed, a return code of 4 is set, and execution resumes.

Symbolic Feedback Code: FOR0103

FOR0104S The DIV callable service *service-name* failed. It was called with no argument list. VS FORTRAN Version Error 2 Number: AFB154I

Programmer response: Provide the arguments that are required for the *service-name* callable service. The data-in-virtual callable services are described in detail in *VS FORTRAN Version 2 Language and Library Reference*.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissable Resume Actions:

Name	Action Taken after Resumption
RN	The service is not completed, a return code of 48 is set, and execution resumes.

Symbolic Feedback Code: FOR0104

FOR0105S The DIV callable service *service-name* failed. It was called with an incorrect number of arguments. VS FORTRAN Version Error 2 Number: AFB154I

Programmer response: Provide the arguments that are required for the *service-name* callable service. The data-in-virtual callable services are described in detail in *VS FORTRAN Version 2 Language and Library Reference*.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, a return code of 48 is set, and execution resumes.

Symbolic Feedback Code: FOR0105

FOR0106S The DIV callable service *service-name* failed. It was called with an argument list in an incorrect format. This probably occurred because a required character argument was not provided. VS FORTRAN Version Error 2 Number: AFB154I

Explanation: The argument list provided to the *service-name* callable service wasn't in the internally-generated form produced by the Fortran compiler when there are character arguments. This could have occurred for one or more of these reasons:

- The call was made from a program compiled by the VS FORTRAN Version 1 or the VS FORTRAN Version 2 compiler with the the LANGLVL(66) compiler option.
- The call was made from a program compiled by the VS FORTRAN Version 1 compiler at a level before Release 3.
- The call was made from a program compiled by the FORTRAN IV H Extended or the FORTRAN IV G1 compiler.
- An incorrect number of arguments was provided.
- One or more of the arguments wasn't of the type required by the callable service.
- The call was made from an assembler language program, and the arguments were not provided in the form required when there are character arguments.

Programmer response: Provide the arguments that are required for the *service-name* callable service. The data-in-virtual callable services are described in detail in *VS FORTRAN Version 2 Language and Library Reference*.

If the program is written in Fortran, compile it with the VS FORTRAN Version 2 compiler, and do not specify the LANGLVL(66) compiler option. If it is written in assembler language, use the Fortran conventions for argument lists with character arguments. These conventions are described in the topic "Passing Character Arguments Using the Standard Linkage Convention" in Appendix B of *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, a return code of 48 is set, and execution resumes.

Symbolic Feedback Code: FOR0106

FOR0120S The FILEINF callable service failed. It was called with an argument list in an incorrect format. VS FORTRAN Version Error 2 Number: AFB096I-1

Explanation: The argument list provided in the call to the FILEINF callable service was incorrect in one of these ways:

- There was no argument list.
- The argument list had an even number of arguments.
- The argument list wasn't in the internally-generated form produced by the Fortran compiler when there are character arguments. This could have occurred for one or more of these reasons:
  - One or more of the keyword arguments (CYL, RECFM, and so on) weren't provided as character expressions.
  - The call was made from a program compiled by the VS FORTRAN Version 1 or the VS FORTRAN Version 2 compiler with the the LANGLVL(66) compiler option.
  - The call was made from a program compiled by the VS FORTRAN Version 1 compiler at a level before Release 3.
  - The call was made from a program compiled by the FORTRAN IV H Extended or the FORTRAN IV G1 compiler.
  - The call was made from an assembler language program, and the arguments were not provided in the form required when there are character arguments.

Programmer response: Be sure that the argument list contains an odd number of arguments and that the even-numbered arguments are character expressions whose values are the permissible keyword arguments.

If the program is written in Fortran, compile it with the VS FORTRAN Version 2 compiler, and do not specify the LANGLVL(66) compiler option. If it is written in assembler language, use the Fortran conventions for argument lists with character arguments. These conventions are described in the topic "Passing Character Arguments Using the Standard Linkage Convention" in Appendix B of *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

Refer to “System Action” regarding the detection of error FOR1926 on a subsequent OPEN or CLOSE statement.

System action: The condition is signaled. If the condition is unhandled, the application is terminated. However, if the **RN** action listed under “Permissible Resume Actions” is taken to resume execution following the call, then the file information provided is ignored, and error FOR1926 is detected during execution of a subsequent OPEN or INQUIRE statement. Detection of error FOR1926 can be suppressed if, following the failing call to the FILEINF callable service, another call is made either with no arguments or with arguments that don’t cause another error to be detected.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution resumes. Refer to “System Action” regarding the detection of error FOR1926 following this resumption.

Symbolic Feedback Code: FOR0120

FOR0121S The FILEINF callable service failed. The argument in position *position* of the argument list was not one of the character values that the FILEINF callable service understands as an argument. VS FORTRAN Version Error 2 Number: AFB096I-2

Explanation: Position *position* of the argument list for the FILEINF callable service was not a character expression whose value was one of the permissible keyword arguments. These permissible keyword arguments are values such as RECFM, CYL, and so on.

Programmer response: Correct the argument list by coding the first argument as an integer variable and the remaining pairs of arguments as one of the permissible keyword arguments followed by its value. The keyword arguments are listed in the description of the FILEINF callable service in *VS FORTRAN Version 2 Language and Library Reference*.

Be sure that each keyword argument is coded as a character expression. Remember that if a character constant is used, the keyword argument, such as RECFM, must be enclosed in quotes or apostrophes.

Refer to “System Action” regarding the detection of error FOR1926 on a subsequent OPEN or CLOSE statement.

System action: The condition is signaled. If the condition is unhandled, the application is terminated. However, if the **RN** action listed under “Permissible Resume Actions” is taken to resume execution following the call, then the file information provided is ignored, and error FOR1926 is detected during execution of a subsequent OPEN or INQUIRE statement. Detection of error FOR1926 can be suppressed if, following the failing call to the FILEINF callable service, another call is made either with no arguments or with arguments that don’t cause another error to be detected.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution resumes. Refer to “System Action” regarding the detection of error FOR1926 following this resumption.

Symbolic Feedback Code: FOR0121

FOR0122S The FILEINF callable service failed. An incorrect value was provided for the actual argument immediately following the actual argument with the value of *keyword*. VS FORTRAN Version Error 2 Number: AFB096I-3

Programmer response: Change the argument list by providing a value that’s allowed to follow and correspond to the keyword argument *keyword*. The permissible values are shown in the description of the FILEINF callable service in *VS FORTRAN Version 2 Language and Library Reference*.

If the value is of character type, such as FB, and it is coded as a character constant, be sure to enclose the value in quotes or apostrophes.

Refer to “System Action” regarding the detection of error FOR1926 on a subsequent OPEN or CLOSE statement.

System action: The condition is signaled. If the condition is unhandled, the application is terminated. However, if the **RN** action listed under "Permissible Resume Actions" is taken to resume execution following the call, then the file information provided is ignored, and error FOR1926 is detected during execution of a subsequent OPEN or INQUIRE statement. Detection of error FOR1926 can be suppressed if, following the failing call to the FILEINF callable service, another call is made either with no arguments or with arguments that don't cause another error to be detected.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution resumes. Refer to "System Action" regarding the detection of error FOR1926 following this resumption.

Symbolic Feedback Code: FOR0122

FOR0123S The FILEINF callable service failed. VSAM record level sharing (RLS) was specified, but execution was on a system without both MVS/ESA SP Version 5 Release 2 or later and DFSMS/MVS Version 1 Release 3 or later.

Programmer response: Ensure that the Fortran application that connects a VSAM file using RLS mode is run on MVS/SP Version 5 Release 2 or later and DFSMS/MVS Version 1 Release 3 or later. If these levels aren't available, then you can't use RLS mode. In this case, remove the RLS keyword argument and its corresponding value from the argument list for the FILEINF callable service.

Refer to "System Action" regarding the detection of error FOR1926 on a subsequent OPEN or CLOSE statement.

System action: The condition is signaled. If the condition is unhandled, the application is terminated. However, if the **RN** action listed under "Permissible Resume Actions" is taken to resume execution following the call, then the file information provided is ignored, and error FOR1926 is detected during execution of a subsequent OPEN or INQUIRE statement. Detection of error FOR1926 can be suppressed if, following the failing call to the FILEINF callable service, another call is made either with no arguments or with arguments that don't cause another error to be detected.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution resumes. Refer to "System Action" regarding the detection of error FOR1926 following this resumption.

Symbolic Feedback Code: FOR0123

FOR0130S The ARGSTR callable service failed. It was called with an argument list in an incorrect format.

Explanation: The argument list provided in the call to the ARGSTR callable service was incorrect in one of these ways:

- There was no argument list.
- The argument list had other than two arguments.
- The argument list wasn't in the internally-generated form produced by the Fortran compiler when there are character arguments. This could have occurred because:
  - The first argument wasn't of character type.
  - The call was made from a program compiled by the VS FORTRAN Version 1 or the VS FORTRAN Version 2 compiler with the the LANGLVL(66) compiler option.
  - The call was made from a program compiled by the VS FORTRAN Version 1 compiler at a level before Release 3.
  - The call was made from a program compiled by the FORTRAN IV H Extended or the FORTRAN IV G1 compiler.
  - The call was made from an assembler language program, and the arguments were not provided in the form required when there are character arguments.

Programmer response: Be sure that the argument list contains two arguments, the first of which is a character variable and the second of which is an integer variable of length 4.

If the program is written in Fortran, compile it with the VS FORTRAN Version 2 compiler, and do not specify the LANGLVL(66) compiler option. If it is written in assembler language, use the Fortran conventions for argument lists with character arguments. These conventions are described in the topic “Passing Character Arguments Using the Standard Linkage Convention” in Appendix B of *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

System action: The service is ignored, and the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution resumes.

Symbolic Feedback Code: FOR0130

FOR0300S One program unit specified common block *common-name* in a DC compiler option, but another program unit did not specify it in a DC compiler option. VS FORTRAN Version Error 2 Number: AFB158I-2

Programmer response: If you want *common-name* to be a dynamic common block, compile all program units that refer to it using a DC compiler option that has as a suboption either *common-name* or an asterisk. If you want *common-name* to be a static common block, do not compile any program units that refer to it using a DC compiler option that has as a suboption either *common-name* or an asterisk unless *common-name* is used as a suboption of the SC compiler option.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*common name*	Input	CHARACTER*31	The name of the common block

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	*common-name* is not made available to one or more program units and execution continues. The results of execution are unpredictable if data in the dynamic common block is subsequently referenced.

Symbolic Feedback Code: FOR0300

FOR0301S The common block *common-name* of length *length* could not be created because there was insufficient virtual storage. VS FORTRAN Version Error 2 Number: AFB156I

Programmer response: Run your application in a larger region. You can change the region size with the REGION parameter on the EXEC statement in your JCL.

If the application allows it, you could also recompile all program units that refer to common block *common-name* with declarations that result in a smaller length for this or for other common blocks.

If there are allocatable arrays that are allocated but not currently in use, then deallocate them to make more storage available.

If one of your routines is running in 24-bit addressing mode, remember that dynamic common blocks acquired for it are created in virtual storage below 16 Mb, where storage is limited. However, when the routine is running in 31-bit addressing mode, dynamic common blocks acquired for it are created in virtual storage above 16 Mb, where there is normally much more storage available. Therefore, if your application is running in 24-bit addressing mode and if it could run in 31-bit addressing mode instead, then making this change could alleviate this storage constraint. But before link editing the application with the AMODE=31 option, you should be sure that there aren't any program units, such as those compiled with the FORTRAN IV H Extended compiler, that aren't capable of running in 31-bit addressing mode.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	3
2	*common-name*	Input	CHARACTER*31	The name of the common block.
3	*length*	Input/Output	INTEGER*4	The length of the common block.

Permissable Resume Actions:

Name	Action Taken after Resumption
RN	The service is not completed, a return code of *return code* is set, and execution resumes.
RI	An attempt is made to acquire virtual storage for the common block. *common-name* using the length provided in *length*. If this is successful, execution continues but the results of execution are unpredictable if data in the dynamic common block beyond the length provided as *length* is referenced.

Symbolic Feedback Code: FOR0301

FOR0302S Common block *common-name* was defined with a length of *length1*, but it was defined with a length of *length2* in a program unit that was invoked earlier. VS FORTRAN Version Error 2 Number: AFB158I-1

Programmer response: In all program units that refer to the common block *common-name* ensure that the declarations of the common block are such that the length of the common block is the same.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	4
2	*common-name*	Input	CHARACTER*31	The name of the common block
3	*length-1*	Input	INTEGER*4	The length of the common block as defined in program unit 1.
4	*length-2*	Input	INTEGER*4	The length of the common block as defined in program unit 2.

Permissable Resume Actions:

Name	Action Taken after Resumption
RN	*common-name* is not made available to one or more program units and execution continues. The results of execution are unpredictable if data in the dynamic common block is subsequently referenced.

Symbolic Feedback Code: FOR0302

FOR0303S **The common block callable service** ***service-name*** **failed. It was called with an argument list in an incorrect format. VS FORTRAN Version Error 2 Number: AFB920I-2, AFB157I-3**

Explanation: The argument list provided in the call to the *service-name* callable service was incorrect in one of these ways:

- There was no argument list.
- The argument list had the wrong number of arguments.
- The argument list wasn't in the internally-generated form produced by the Fortran compiler when there are character arguments. This could have occurred for one or more of these reasons:
  - The common block name wasn't provided as a character expression.
  - The call was made from a program compiled by the VS FORTRAN Version 1 or the VS FORTRAN Version 2 compiler with the the LANGLVL(66) compiler option.
  - The call was made from a program compiled by the VS FORTRAN Version 1 compiler at a level before Release 3.
  - The call was made from a program compiled by the FORTRAN IV H Extended or the FORTRAN IV G1 compiler.
  - The call was made from an assembler language program, and the arguments were not provided in the form required when there are character arguments.

Programmer response: Be sure that the argument list contains the number of arguments required by *service-name* and that they are of the correct type. In particular, if the common block name is coded as a character constant, be sure to enclose the value in quotes or apostrophes.

If the program is written in Fortran, compile it with the VS FORTRAN Version 2 compiler, and do not specify the LANGLVL(66) compiler option. If it is written in assembler language, use the Fortran conventions for argument lists with character arguments. These conventions are described in the topic "Passing Character Arguments Using the Standard Linkage Convention" in Appendix B of *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

System action: The service is ignored, and the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*service-name*	Input	CHARACTER*8	The name of the common block callable service that was called.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution resumes.

Symbolic Feedback Code: FOR0303

FOR0304S **The common block callable service** ***service-name*** **failed. The common block name had an incorrect format. The invalid name was '*common-name*'. VS FORTRAN Version Error 2 Number: AFB157I-2**

Explanation: The name of the dynamic common block provided to the *service-name* was not a valid name because it either:
- Began with a blank or was all blank,
- Was longer than 31 characters, or
- Contained an imbedded blank

Programmer response: Be sure the character expression for the dynamic common block name passed to *service-name* is a valid Fortran name. In particular, it must:
- Be left-adjusted with trailing blanks,
- Begin with a letter, underscore (_), or dollar sign ($),

- Contain only alphameric characters, that is, letters, digits, underscores (_), or dollar signs ($),
- Contain at least 1 but no more than 31 nonblank characters, and
- Have no imbedded blanks.

If the common block name is coded as a character constant, be sure to enclose the value in quotes or apostrophes.

If the program is written in Fortran, compile it with the VS FORTRAN Version 2 compiler, and do not specify the LANGLVL(66) compiler option. If it is written in assembler language, use the Fortran conventions for argument lists with character arguments. These conventions are described in the section "Passing Character Arguments Using the Standard Linkage Convention" in Appendix B of *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	3
2	*service-name*	Input	CHARACTER*8	The name of the common block callable service that was called.
3	*common-name*	Input	CHARACTER*31	The common block name (or the first 31 characters of the name) that was provided for *service-name*.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution resumes.

Symbolic Feedback Code: FOR0304

FOR0310S **The ALLOCATE statement could not be completed. The object** *object_name* **of length** *object_length* **could not be created because there was insufficient virtual storage.**

Programmer response: Run your application in a larger region. You can change the region size with the REGION parameter on the EXEC statement in your JCL.

If the application allows it, you could also reduce the size of the allocatable array so that it doesn't require as much storage.

If there are other allocatable arrays that are allocated but not currently in use, then deallocate them to make more storage available.

If there are common blocks or other large storage areas that could be reduced in size, then doing so could make more storage available.

If a routine is running in 24-bit addressing mode, remember that allocatable arrays acquired for it are created in virtual storage below 16 Mb, where storage is limited. However, when the routine is running in 31-bit addressing mode, allocatable arrays acquired for it are created in virtual storage above 16 Mb, where there is normally much more storage available. Therefore, if your application is running in 24-bit addressing mode and if it could run in 31-bit addressing mode instead, then making this change could alleviate this storage constraint. But before link editing the application with the AMODE=31 option, you should be sure that there aren't any program units, such as those compiled with the FORTRAN IV H Extended compiler, that aren't capable of running in 31-bit addressing mode.

System action: If the STAT specifier is not present on the ALLOCATE statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	3
2	*object-name*	Input	CHARACTER*250	The name of the object specified in the ALLOCATE statement.
3	*object-length*	Input	INTEGER*4	The length of the object specified in the ALLOCATE statement.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The current operation is ignored. The remainder of the allocation is processed and execution continues.

Symbolic Feedback Code: FOR0310

FOR0311S The ALLOCATE statement could not be completed. The object *object_name* was already allocated.

Programmer response: Correct the logic of your program so that the same allocatable array isn't allocated again until its first occurrence is deallocated.

System action: If the STAT specifier is not present on the ALLOCATE statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*object-name*	Input	CHARACTER*250	The name of the object specified in the ALLOCATE subroutine.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The current operation is ignored. The remainder of the allocation list is processed and execution continues.

Symbolic Feedback Code: FOR0311

FOR0312S The DEALLOCATE statement could not be completed. The object *object_name* was not allocated.

Programmer response: Correct the logic of your program so that you don't deallocate an array isn't allocated.

System action: If the STAT specifier is not present on the DEALLOCATE statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*object-name*	Input	CHARACTER*250	The name of the object specified in the DEALLOCATE statement.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The current operation is ignored. The remainder of the deallocation list is processed and execution continues.

Symbolic Feedback Code: FOR0312

FOR0340A PAUSE *message* VS FORTRAN Version Error 2 Number: AFB001I

Explanation: A PAUSE statement has been executed from a Fortran routine. The message text *message* is whatever information was provided by the programmer with the PAUSE statement.

Programmer response: Follow the instructions given by *message* or by the person who submitted the job for execution. These instructions should indicate the action to be taken.

To resume execution, provide any single character as a response to the outstanding console message after taking the actions requested.

System action: Execution of the program waits for a response, which can be any character. After the response is entered, execution of the program continues with the statement following the PAUSE statement.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0340

FOR0341I STOP *message* VS FORTRAN Version Error 2 Number: AFB002I

Explanation: A STOP statement has been executed from a Fortran routine. The message text *message* is whatever information was provided by the programmer with the STOP statement.

System action: The termination imminent condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0341

FOR0400S A Fortran main program was executed from within an enclave that had already started executing. VS FORTRAN Version Error 2 Number: AFB905I

Programmer response: If you intended to call a Fortran subroutine rather than a main program, then code a SUBROUTINE statement as the first statement of that called routine.

If you want to call a main program, which will be in a new enclave, do this in one of these two ways:

- Invoke an assembler language program that uses a LINK macro instruction to pass control to the main program and implicitly create a new enclave.
- Invoke the callable service CEE3CRE, which creates a new enclave and passes control to the main program.

In both cases, the main program that is specified must be in a separate load module.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0400

FOR0401S **The execution of program unit** *program-unit* **failed at ISN** *statement-number* **because an error was detected by the compiler at that statement. VS FORTRAN Version Error 2 Number: AFB230I**

Programmer response: Refer to the printed output of the compilation of program unit *program-unit* to determine the error that occurred at ISN *statement-number*. Correct the error, then compile, link edit, and execute the job again.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0401

FOR0402S *program-name2***, which has one or more dummy arguments of character type with an assumed length, was called by** *program-name1* **with an argument list that didn't provide the lengths of the character arguments. VS FORTRAN Version Error 2 Number: AFB153I**

Explanation: The subprogram *program-name2* had a dummy argument of character type with an assumed length, that is, for which the current length needs to be provided by the calling routine. However, the argument list provided by program unit *program-name1* wasn't in the internally-generated form produced by the Fortran compiler when there are character arguments. This could have occurred because:

- A dummy argument in *program-name2* was inadvertently coded as a character dummy argument with an assumed length as in this example:

  ```
  CHARACTER*(*)  INPUT_ARG
  ```
- *program-name1* didn't provide one or more of the character arguments that were required by *program-name2*.
- *program-name1* was compiled by the VS FORTRAN Version 1 or the VS FORTRAN Version 2 compiler with the the LANGLVL(66) compiler option.
- *program-name1* was compiled by the VS FORTRAN Version 1 compiler at a level before Release 3.
- *program-name1* was compiled by the FORTRAN IV H Extended or the FORTRAN IV G1 compiler.
- *program-name1* was an assembler language program, and the arguments were not provided in the form required when there are character arguments.

Programmer response: Be sure that the argument list provided by *program-name1* contains the number of arguments required by *program-name2* and that they are of the correct type.

If a character argument is coded as a character constant, be sure to enclose the value in quotes or apostrophes.

If *program-name1* is written in Fortran, compile it with the VS FORTRAN Version 2 compiler, and do not specify the LANGLVL(66) compiler option.

If *program-name1* is written in assembler language, use the Fortran conventions for argument lists with character arguments. These conventions are described in the topic "Passing Character Arguments Using the Standard Linkage Convention" in Appendix B of *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

If *program-name1* is neither a Fortran nor an assembler language program, the required argument list cannot be generated. In this case, change *program-name2* so the character data in the dummy argument list is of fixed, rather than of assumed, length.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0402

FOR0404C **The LIBPACK (composite module)** *module-name1* **was at release level** *module-level1***, but the LIBPACK** *module-name2* **was at release level** *module-level2***. VS FORTRAN Version Error 2 Number: AFB142I**

Programmer response: If your JCL specifies the correct Language Environment library for execution, then this is likely to be a problem either with the installation of Language Environment or with the availability of the library. Refer the problem to your Language Environment support personnel.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0404

FOR0405C *module-name* **was not a valid LIBPACK (composite module). VS FORTRAN Version Error 2 Number: AFB145I**

Programmer response: If your JCL specifies the correct Language Environment library for execution, then this is likely to be a problem either with the installation of Language Environment or with the availability of the library. Refer the problem to your Language Environment support personnel.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0405

FOR0406C **The shareable load module** *module-name* **was loaded at an address above 16 Mb by the nonshareable part of program unit** *program-unit***, which was running in 24-bit addressing mode. VS FORTRAN Version Error 2 Number: AFB146I**

Explanation: Program unit *program-unit* was compiled with the RENT compiler option and was separated into its nonshareable and shareable parts. The nonshareable part was entered in 24-bit addressing mode but was unable to pass control to its shareable part because the shareable part was loaded above 16 Mb.

Programmer response: Either:

- Run the program in 31-bit addressing mode by link editing the nonshareable parts with AMODE=31 as a linkage editor parameter. This can be done only if the load module has no routines, such as those compiled with the FORTRAN IV H Extended compiler, that are not capable of executing in 31-bit addressing mode.
- Link edit the shareable load module using AMODE=24 as a linkage editor parameter.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0406

FOR0407C **The shareable load module** *module-name* **that was loaded by the nonshareable part of program unit** *program-unit* **had an incorrect format. VS FORTRAN Version Error 2 Number: AFB147I**

Explanation: Program unit *program-unit* was compiled with the RENT compiler option and was separated into its nonshareable and shareable parts. During execution, the program's nonshareable part loaded a load module that was supposed to contain the program's shareable part. However, the load module was not in the expected format.

Programmer response: Use the Fortran reentrant program separation tool to separate the shareable and nonshareable parts of the program that was compiled with the RENT compiler option. (This tool is invoked by the use of the cataloged procedures AFHWRL and AFHWRLG.) Then ensure that during the execution of the program, the load module containing the shareable part is available either in a library referenced by a STEPLIB DD statement or in a link pack area.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0407

FOR0409C The shareable load module *module-name* that was loaded by the nonshareable part of program unit *nonshareable-part-name* had a timestamp of *timestamp1* in the shareable part of program unit *shareable-part-name*. This timestamp differed from the timestamp of *timestamp2* in the nonshareable part of program unit *nonshareable-part-name*. VS FORTRAN Version Error 2 Number: AFB149I

Explanation: Program unit *program-unit* was compiled with the RENT compiler option and was separated into its nonshareable and shareable parts. During execution, the program's nonshareable part loaded a load module that was supposed to contain the program's shareable part. However, the load module contained a copy of the code that was compiled at a different time than the nonshareable part. The parts are assumed to be incompatible.

Programmer response: Use the Fortran reentrant program separation tool to separate the shareable and nonshareable parts of the program that was compiled with the RENT compiler option. (This tool is invoked by the use of the cataloged procedures AFHWRL and AFHWRLG.) Then ensure that during the execution of the program, the load module containing the shareable part is available either in a library referenced by a STEPLIB DD statement or in a link pack area. Also ensure that some previous copy isn't accessible so that only the corresponding copy of the shareable part load module is available to the executing program.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0409

FOR0410C A Fortran subprogram was called before the Fortran run-time environment was initialized. VS FORTRAN Version Error 2 Number: AFB932I

Explanation: Language Environment did not become aware of the existence of a Fortran subprogram in the application before the invocation of that subprogram. Usually the presence of a Fortran routine in the application is detected either at the time a main program is started or at the time a subsequent load module is dynamically loaded using various languages' dynamic call facilities. However, because Fortran compiled code doesn't conform to the current Language Environment linkage conventions, sometimes a Fortran subprogram isn't detected, especially if it doesn't require the use of an run-time library services such as input/output.

Programmer response: Ensure that there is a main program (not necessarily written in Fortran) in the application and that it is executed before any Fortran subprograms. If this was already the case, then provide the following linkage editor control statement in the input that link edits the main program:

```
INCLUDE SYSLIB(CEESG007)
```

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0410

FOR0411C VS FORTRAN Version 2 error *error-number* was detected by Language Environment.

Explanation: The obsolete VS FORTRAN Version 2 error condition with error number *error-number* was detected. This in an internal error in the Fortran portion of Language Environment.

Programmer response: Contact the people who provide system support at your installation for Language Environment.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0411

FOR0414C The shareable load module *module-name***, which was loaded by the nonshareable part of program unit** *program-unit***, did not contain the shareable part** *shareable-part-name***. VS FORTRAN Version Error 2 Number: AFB148I**

Explanation: Program unit *program-unit* was compiled with the RENT compiler option and was separated into its nonshareable and shareable parts. During execution, the program's nonshareable part loaded a load module that was supposed to contain the program's shareable part. However, the load module did not contain the expected shareable part.

Programmer response: Use the Fortran reentrant program separation tool to separate the shareable and nonshareable parts of the program that was compiled with the RENT compiler option. (This tool is invoked by the use of the cataloged procedures AFHWRL and AFHWRLG.) Then ensure that during the execution of the program, the load module containing the shareable part is available either in a library referenced by a STEPLIB DD statement or in a link pack area. Also ensure that some previous copy isn't accessible.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0414

FOR0415C The shareable load module *module-name* **that was loaded by the nonshareable part of program unit** *program-unit* **did not contain the shareable part** *shareable-part-name* **at a storage location accessible to the program. VS FORTRAN Version Error 2 Number: AFB148I**

Explanation: Program unit *program-unit* was compiled with the RENT compiler option and was separated into its nonshareable and shareable parts. During execution, the program's nonshareable part loaded a load module that was supposed to contain the program's shareable part. However, not all of that load module was loaded so that it could be used.

Programmer response: Use the Fortran reentrant program separation tool to separate the shareable and nonshareable parts of the program that was compiled with the RENT compiler option. (This tool is invoked by the use of the cataloged procedures AFHWRL and AFHWRLG.) Then ensure that during the execution of the program, the load module containing the shareable part is available either in a library referenced by a STEPLIB DD statement or in a link pack area.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR0415

FOR0416S The program unit *program-unit* **called the subprogram** *routine-name* **with the array** *array-name* **(***array-bounds***) having a dimension with the lower bound greater than the upper bound. VS FORTRAN Version Error 2 Number: AFB257I**

Programmer response: Ensure that the declarations of the array and of the dimension arguments are consistent in *program-unit* and in *routine-name*. Also ensure that the values of the bounds that are provided as actual arguments for the call do not make the lower bound greater than the upper bound for any dimension of the array.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	Execution continues, but invalid results are probable if a reference is made to the array whose dimensions are wrong.

Symbolic Feedback Code: FOR0416

FOR0417S **The program unit** *program-unit* **called the subprogram** *routine-name* **with the array located at address** *array-address*, **offset** *array-offset*, **and array bounds** *array-bounds*. **The array bounds had a dimension with the lower bound greater than the upper bound. VS FORTRAN Version Error 2 Number: AFB257I**

Programmer response: Ensure that the declarations of the array and of the dimension arguments are consistent in *program-unit* and in *routine-name*. Also ensure that the values of the bounds that are provided as actual arguments for the call do not make the lower bound greater than the upper bound for any dimension of the array.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	Execution continues, but invalid results are probable if a reference is made to the array whose dimensions are wrong.

Symbolic Feedback Code: FOR0417

FOR0500S **A relational expression using character values with the relational operator** *relational-operator* **could not be evaluated. Character value** *operand-number* **had a length of** *operand-length*, **which was not between 1 and 32767, inclusive. VS FORTRAN Version Error 2 Number: AFB193I**

Programmer response: Ensure that the length of the character value is neither less than 1 nor greater than 32767. Examine any variables that define a character substring to be sure that they don't have values that result in an invalid length.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*operator-name*	Input	CHARACTER*2	The name of the relational operator

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The comparison is not performed, and execution continues.

Symbolic Feedback Code: FOR0500

FOR0501S **The assignment of the character value could not be performed. The storage area that was being copied overlapped with the storage area to which that data was to be copied. VS FORTRAN Version Error 2 Number: AFB195I**

Programmer response: Examine any variables that define a character substring to be sure that they don't have values that result in an excessive character length or overlapping substrings. Also look at EQUIVALENCE statements to ensure that the character variables in question don't overlap.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The character assignment is not performed, and execution continues.

Symbolic Feedback Code: FOR0501

FOR0502S **The assignment of the character value could not be performed. The length of the storage area to which the data was to be copied was not between 1 and 32767, inclusive. VS FORTRAN Version Error 2 Number: AFB196I**

Programmer response: Ensure that the length of the character value is neither less than 1 nor greater than 32767. Examine any variables that define a character substring to be sure that they don't have values that result in an invalid length.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The character assignment is not performed, and execution continues.

Symbolic Feedback Code: FOR0502

FOR0503S **The assignment of the character value could not be performed. The length of the storage area from which the data was to be copied was not between 1 and 32767, inclusive. VS FORTRAN Version Error 2 Number: AFB197I**

Programmer response: Ensure that the length of the character value is neither less than 1 nor greater than 32767. Examine any variables that define a character substring to be sure that they don't have values that result in an invalid length.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The character assignment is not performed, and execution continues.

Symbolic Feedback Code: FOR0503

FOR0504S **The concatenation of character values could not be performed. The length of one of the values was not between 1 and 32767, inclusive. VS FORTRAN Version Error 2 Number: AFB199I**

Programmer response: Ensure that the lengths of the character values are not less than 1 and not greater than 32767. Examine any variables that define a character substring to be sure that they don't have values that result in an invalid length.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The character concatenation is not performed, and execution continues.

Symbolic Feedback Code: FOR0504

FOR0601S The *funcname* function could not be evaluated. The value of the argument was not between 0 and *limit*, inclusive. VS FORTRAN Version Error 2 Number: AFB258I

Programmer response: Ensure that the argument to the ACHAR function in not less than 0 nor greater than 127 or that the argument to the CHAR function in not less than 0 nor greater than 255. The values of 127 and 255 are the greatest values in the ASCII and EBCDIC collating sequences, respectively.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*function*	Input	CHARACTER*8	funcname

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The CHAR or ACHAR function is ignored, and execution continues.

Symbolic Feedback Code: FOR0601

FOR0602S The INDEX function could not be evaluated. Argument number *argument-number* had a length of *argument-length*, which was not between 1 and 32767, inclusive. VS FORTRAN Version Error 2 Number: AFB259I

Programmer response: Ensure that the length of the character value in argument *argument-number* is neither less than 1 nor greater than 32767. Examine any variables that define a character substring to be sure that they don't have values that result in an invalid length.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*function*	Input	CHARACTER*8	INDEX

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The INDEX function is ignored, and execution continues.

Symbolic Feedback Code: FOR0602

FOR0603S The lexical relational function could not be evaluated. Argument number *argument-number* had a length of *argument-length*, which was not between 1 and 32767, inclusive. VS FORTRAN Version Error 2 Number: AFB191I

Programmer response: Ensure that the length of the character value in argument *argument-number* for the the LGE, LGT, LLE, or LLT function is neither less than 1 nor greater than 32767. Examine any variables that define a character substring to be sure that they don't have values that result in an invalid length.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The lexical compare is ignored, and execution continues.

Symbolic Feedback Code: FOR0603

FOR0610S The value of argument number *argno* for the the MVBITS subroutine was not between 0 and *limit*, inclusive. VS FORTRAN Version Error 2 Number: AFB159I

Programmer response: For argument *argno* provide a value that is between 0 and *limit*.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The lexical compare is ignored, and execution continues.

Symbolic Feedback Code: FOR0610

FOR0611S The sum of the values of argument numbers *argno1* and *argno2* for the MVBITS subroutine was greater than the number of bits in the first argument. VS FORTRAN Version Error 2 Number: AFB176I

Programmer response: Adjust the values of arguments *argno1* and *argno2* so that the specified string of bits doesn’t extend beyond the end of the integer.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*function*	Input	CHARACTER*8	MVBITS

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The MVBITS subroutine is ignored, and execution continues.

Symbolic Feedback Code: FOR0611

FOR0612S The *funcname* function could not be evaluated. The value of argument number *argno* was not between *lowlimit* and *hilimit*, inclusive. VS FORTRAN Version Error 2 Number: AFB159I

Programmer response: For the function *funcname*, ensure that the value provided for argument *argno* is in the range of *lowlimit* through *hilimit*.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*function*	Input	CHARACTER*8	funcname

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The bit manipulation function is ignored, and execution continues.

Symbolic Feedback Code: FOR0612

FOR0613S The IBITS function could not be evaluated. The sum of the second and the third arguments was greater than the number of bits in the first argument. VS FORTRAN Version Error 2 Number: AFB159I

Programmer response: For the IBITS function, adjust the values of arguments 2 and 3 so that the specified string of bits doesn't extend beyond the end of the integer.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*function*	Input	CHARACTER*8	IBITS

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The IBITS function is ignored, and execution continues.

Symbolic Feedback Code: FOR0613

FOR0614S The IBITS function could not be evaluated. The value of the second or third argument was less than 0. VS FORTRAN Version Error 2 Number: AFB159I

Programmer response: For the IBITS function, adjust the values of arguments 2 and 3 so that they are nonnegative integer values that indicate the starting bit number (relative to 0) and the number of bits to be extracted.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*function*	Input	CHARACTER*8	IBITS

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The IBITS function is ignored, and execution resumes.

Symbolic Feedback Code: FOR0614

FOR0650S The callable service *service-name* failed. Qualifying datum number *index* was specified as the second argument, but that qualifying datum was not available for the condition.

Explanation: Either QDFETCH or QDSTORE was called to obtain or update an element of qualifying data, but *index*, which specifies the ordinal number of the qualifying datum to be referenced, was either less than 1 or greater than the total number of elements of qualifying data associated with this instance of the condition.

Programmer response:

Ensure that the second argument is a positive integer whose value is the ordinal number of an element of qualifying data.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution resumes.

Symbolic Feedback Code: FOR0650

FOR0651S **The callable service** *service-name* **failed. The first argument, the condition token, was not a character variable or array element of length 12.**

Explanation: Either the first argument was not a character variable or character array element of length 12 or the argument list wasn't in the internally-generated form produced by the Fortran compiler when there are character arguments. The latter could have occurred because:

- The call was made from a program compiled by the VS FORTRAN Version 1 or the VS FORTRAN Version 2 compiler with the the LANGLVL(66) compiler option.
- The call was made from a program compiled by the VS FORTRAN Version 1 compiler at a level before Release 3.
- The call was made from a program compiled by the FORTRAN IV H Extended or the FORTRAN IV G1 compiler.
- The call was made from an assembler language program, and the arguments were not provided in the form required when there are character arguments.

Programmer response: Be sure that the argument list contains the number of arguments required by *service-name* and that they are of the correct type. In particular, ensure that the first argument is a condition token whose declaration is a character variable or character array element of length 12.

If the program is written in Fortran, compile it with the VS FORTRAN Version 2 compiler, and do not specify the LANGLVL(66) compiler option. If it is written in assembler language, use the Fortran conventions for argument lists with character arguments. These conventions are described in the topic “Passing Character Arguments Using the Standard Linkage Convention” in Appendix B of *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution continues.

Symbolic Feedback Code: FOR0651

FOR1000S *locator-text* **The formatted input data** *input-field* **had a value that was outside the range of values that could be contained within the integer item in the input item list. VS FORTRAN Version 2 Error Number: AFB206I**

Explanation: *input-field* is a character string that was interpreted as an integer value; an integer variable or array element that was given in the input item list of a READ statement was supposed to become defined with this value. However, the value of *input-field* was outside the acceptable range. These are the ranges of integer values corresponding to integer data items of different lengths:

Data Type	Maximum Positive Value	Maximum Negative Value
INTEGER*1	127 (2^7 –1)	–128 (–2^7)
INTEGER*2	32767 (2^{15}–1)	–32768 (–2^{15})

Data Type	Maximum Positive Value	Maximum Negative Value
INTEGER*4	2147483647 (2^{31}–1)	–2147483648 (–2^{31})
INTEGER*8	9223372036854775807 (2^{63}–1)	–9223372036854775808 (–2^{63})

locator-text gives more information about the location of the error, and can be one of the following:
The READ statement for an internal file failed.
The READ statement for unit *unit-number* which was connected to *file-name* failed.

Programmer response: Ensure that the following are true:
- The edit descriptor specifies the correct field width.
- The value specified by *input-field* is within the required range for the integer variable size.
- *input-field* doesn't have any imbedded or trailing blanks that are incorrectly treated as zeros, thus changing the intended magnitude of the number. Blanks are treated as zeros in these cases:
  - The BLANK specifier is given on the OPEN statement with value of ZERO.
  - The BZ edit descriptor is included in the format specification.
  - There is no OPEN statement, and there is no BN edit descriptor in the format specification.

System action: The input item being processed and the remainder of the items in the input item list become undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

No.	Name	Input/Output	Data Type and Length	Value
5	*input-field-desc*	Input	Q_DATA_DESC	The q_data descriptor for *input-field*. It contains the data type and the length of *input-field*.
6	*subroutine-name*	Input	CHARACTER*n	The formatted input data; that is, the character string that is being interpreted as an integer value. The length *n* is part of *input_field-desc* and has a maximum possible value of 255.
7	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
7	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
8	*record*	Input	CHARACTER*n	The formatted input record that contained the character string being interpreted as an integer value. The length *n* which includes only the data portion of the record, is part of *record-desc*.
9	*result-desc*	Input	Q_DATA_DESC	The q_data descriptor for *result*. It contains the data type and the length of *result*.
10	*result*	output	INTEGER*n	The new output value, where *n*, the length, is part of *result-desc* and could be 1, 2, 4, or 8.

Name	Action Taken after Resumption
R0	The input item receives the value placed in *result*, execution continues, and the remainder of the input item list is processed.

Symbolic Feedback Code: FOR100

FOR1001E *locator-text* **The length of the record to be written exceeded the maximum data length,** *data-length*, **allowed for records in the file. VS FORTRAN Version 2 Error Number: AFB164I, AFB201I (format 1), AFB201I (format 2), AFB204I, AFB212I, AFB213I**

Explanation: Based on the type of formatting described by the FMT specifier (or by its absence) on the WRITE or REWRITE statement, one of the following exceeded *data-length*, which is the smaller of either the maximum length of the data that can fit in a record in the file or the value given in the RECL specifier, if any, on the OPEN statement:

For output using a format specification:

– The length of the record described by the output item list and the format specification.

For list-directed output or namelist output:

– For an output item of neither character type nor complex type, the formatted length of the item.

– For an output item of complex type, the formatted length of the real or the imaginary part.

The formatted lengths of the output from list-directed formatting for the various data types is listed in "WRITE Statement — List-Directed I/O to External Devices" in *VS FORTRAN Version 2 Language and Library Reference*. For namelist formatting, the lengths of the data are the same.

For unformatted output:

– The total length of all the items in the output item list.

The maximum length of a record that can be written on a particular file depends on the values of certain specifiers given on the OPEN statement and on various file characteristics managed by the underlying operating system's access methods. Length is taken from one or more of the following:

- **Files connected for sequential access:**
  - **Non-VSAM files**
    - From the LRECL value given in the DD statement or ALLOCATE command (when dynamic file allocation is not involved).
    - From the LRECL value given in the invocation of the FILEINF callable service (when dynamic file allocation is involved).
    - For a DASD or labeled tape file for which the data set existed previously, from the LRECL value given when the file was either allocated or created.
    - From the default LRECL value specified for the unit in the Unit Attribute Table either as shipped by IBM or as customized for the installation.
    - From the LRECL value derived from the RECFM and BLKSIZE values when there is a conflict among these three parameters. Refer to "Considerations for Specifying RECFM, LRECL, and BLKSIZE" in Chapter 12, or in *VS Fortran Version 2 Programming Guide for CMS and MVS*.

    Note: When the record format is one of the variable-length formats, that is, variable (V), variable blocked (VB), variable spanned (VS), or variable blocked spanned (VBS), the maximum length of the data that can be written is four bytes less than the LRECL value.
  - **VSAM files**
    - From the maximum record length given as the second sub-parameter of the RECORDSIZE parameter of the Access Method Services DEFINE command that was used to define the cluster.
- **Files connected for direct access:**
  - From the value given by the RECL specifier on the OPEN statement.

    In certain cases, this value must be consistent with the record length specified previously:
    - **VSAM files (RRDSs):**

      With the maximum record length given as the second sub-parameter of the RECORDSIZE parameter of the Access Method Services DEFINE command that was used to define the cluster.
    - **Non-VSAM files that existed previously and are not being reformatted:**

      With the record length given in the RECL specifier on the OPEN statement that was used to create the file.

    Note: An existing file connected for direct access is not reformatted unless one or more of the following is true:
    - WRITE is given as the value of the ACTION specifier on the OPEN statement.
    - NEW is given as the value of the STATUS specifier on the OPEN statement.

  - No records have been written into the file previously.
- **Files connected for keyed access:**

  From the maximum record length given as the second sub-parameter of the RECORDSIZE parameter of the Access Method Services DEFINE command that was used to define the cluster.
- **Internal files:**

  From the length of the record or records that comprise the internal file. This is the length of the character variable, of the character substring, or of the character array element that comprises the internal file. For an internal file that is a character array, this is the length of the corresponding character array element.

locator-text gives more information about the location of the error, and can be one of the following:
  The WRITE statement for an internal file failed.
  The *statement* statement for unit *unit-number* which was connected to *file-name*, failed.

Programmer response:

Ensure that the length of the record described or implied by the output item list and the format identifier, if any, is no longer than the maximum length record that can be written to the file. Either the length of the record to be written or the maximum record length allowed for the file must be changed to correct the condition or conditions described in "Explanation." If the length of the record being written isn't what you intended, then you might have to change:

- The type of formatting indicated by FMT specifier (or its absence) in the I/O statement. For example, perhaps you intended to write the file using a format specification rather than using list-directed formatting.
- The format specification. This can include errors in the edit descriptors, field widths, repetition factors, nesting levels, Hollerith constants (H edit descriptor), and character constants.
- The output item list. There could be errors in the number of items in the list, in the data types and lengths of individual items, and in the specification of implied DOs in the output item list.

If the length of the record is what you intended, then the maximum record length allowed for the file must be increased. If you are creating a new file, you might have to change:

- The record length given in the RECL specifier of the OPEN statement
- The LRECL parameter on a DD statement or a TSO ALLOCATE command (when dynamic file allocation is not involved) or the LRECL parameter for the FILEINF callable service (when dynamic allocation is involved)

  For record formats of variable spanned or variable blocked spanned, you can make the record length larger than the block size. You can also define the record to be of unlimited length in one of these ways:
  - When dynamic file allocation is not involved, provide a value of X for the LRECL parameter of a DD statement or a TSO ALLOCATE command.
  - When dynamic file allocation is involved, provide a value of –1 for LRECL in the arguments for the FILEINF callable service.

  For any of the variable-length record formats, that is, for variable (V), variable blocked (VB), variable spanned (VS), or variable blocked spanned (VBS), the length given as LRECL includes a 4-byte record descriptor word. Therefore, the LRECL value must be four bytes larger than the largest amount of data that you want to write in a single record.
- The RECFM and BLKSIZE parameters on a DD statement or a TSO ALLOCATE command when dynamic file allocation is not involved or as parameters for the FILEINF callable service when dynamic file allocation is involved. Note that when not all of the RECFM, LRECL, and BLKSIZE parameters have been specified for a particular program, the defaults in the Unit Attribute Table are applied for the omitted ones. Sometimes this causes inconsistencies among the parameters; this is resolved as described in "Considerations for Specifying RECFM, LRECL, and BLKSIZE" in Chapter 12, or in *VS Fortran Version 2 Programming Guide for CMS and MVS*.
- The use of a particular device or file itself if the device or file isn't capable of accepting a large enough record
- The maximum record length given as the second sub-parameter of the RECORDSIZE parameter on the DEFINE command that defined a VSAM cluster

If you are writing on an existing file whose previous contents you want to retain, then you generally cannot increase the record length without recreating the file.

The person at your installation who gives system support for Language Environment can change your installation's default values for record format, record length, and block size for various units. This is done by customizing the the

Unit Attribute Table. As part of this process, the SFLRECL or SULRECL parameters on the AFHODCBM macro instructions can specify larger default values for formatted or unformatted I/O. Each unit (other than the error message unit) can be given different default values.

System action: If either the ERR or the IOSTAT specifier is present on the I/O statement, then before control returns to the program, the record described for the action **RN** is written.

If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition either is unhandled or is handled by moving the resume cursor and resuming, then the record described for the action **RN** is written. If the condition is unhandled, the enclave stops executing after the record has been written.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 10. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*access*	Input	CHARACTER*10	For an external file, the value SEQUENTIAL, DIRECT, or KEYED, depending on whether the file is connected for sequential, direct, or keyed access, respectively. For an internal file, this qualifying datum contains the value SEQUENTIAL.
6	*fmt-type*	Input	CHARACTER*8	One of the following values to indicate the type of formatting indicated by the FMT specifier (or its absence) on the WRITE or REWRITE statement: **Value** / **Type of Formatting**: blanks — Unformatted; FORMAT — Format specification; * — List-directed formatting; NAMELIST — Namelist formatting
7	*data_len*	Input	INTEGER*4	The maximum length of the data that can be written into the records in the file during this connection. If this length is controlled by a RECL specifier on the OPEN statement, then this qualifying datum has that value given by that RECL specifier; otherwise, this is the maximum amount of data that can be written in the records in the file. When the record format is one of the variable-length formats, that is, variable (V), variable blocked (VB), variable spanned (VS), or variable blocked spanned (VBS), the length given here is the LRECL value less 4 (unless the RECL specifier had a smaller value).

Name	Action Taken after Resumption
RN	A record is written, but its length does not exceed the maximum length allowed for the file, and no additional record is written. Formatted and unformatted output are handled slightly differently: **Formatted output:** For a data item of other than character type that doesn't fit in the record, that data item is ignored, that is, none of it is placed in the record. (If the records are of fixed-length format, blanks fill the rest of the record. Otherwise, no additional characters are added to the record.) The rest of the output item list and the rest of the format specification, if any, are ignored. For a data item of character type, including a character constant in the format specification, that doesn't fit in the record, as much of the item as can fit is placed in the record. The rest of this item, the rest of the output item list, and the rest of the format specification, if any, are ignored. **Unformatted output:** As much of the data from the output item list as can fit is placed in the record. This includes the data item that would overflow the record; as much of it as can fit is placed in the record. The rest of this item and the rest of the output item list are ignored. In either case, execution then continues.

Name	Action Taken after Resumption
RF	This action depends on several factors: • The type of formatting, if any, specified in the WRITE or REWRITE statement, • Whether the file is connected for sequential, direct, or keyed access, and • The record format. **For output using a format specification**: • **Files connected for sequential access**: For a formatted data item of other than character type that doesn't fit in the record, none of it is placed in the record from which it would overflow. (If the records are of fixed-length format, blanks fill the rest of the record. Otherwise, no additional characters are added to the record.) The entire formatted data item is placed into the next record beginning at character position 1. Should the maximum length record be too short to hold this data item, as much of the data as can fit is placed in this next record, and the rest of the data is lost. No error is detected for this loss of data. For a data item of character type (including a character constant in the format specification) that doesn't fit in the record, as much of the item as can fit is placed in this record, and the remainder continues into the next record or records for as many records as it takes to hold the entire item. After the formatted data item is placed in the next record, normal processing of the rest of the format specification and the output item list continues. This same condition could be detected again for the same WRITE or REWRITE statement. • **Files connected for direct access**: The action is the same as for sequential access. When applied to direct access, the term *next record* refers to the record with the next higher record number. • **Files connected for keyed access**: This action is the same as **RN** on page 258. **For list-directed and namelist output**: For the formatted data item (or the real or imaginary part of an item of complex type) that's longer than the record, none of the data is placed in the record. (If the records are of fixed-length format, blanks fill the rest of the record. Otherwise, no additional characters are added to the record.) The rest of the output item list is ignored. For data of character type, this error is not detected. Such data is automatically spanned across records without this being considered an error. **For unformatted output**: • **Files connected for sequential access**: For non-VSAM files that have a record format of variable spanned (VS) or variable blocked spanned (VBS), a record of the size required by the output item list is written. This is just as though the LRECL value specified an unlimited record length. Such records generally cannot be read using languages other than Fortran because the lengths of records in the file exceed the LRECL value associated with the file. For other files connected for sequential access, this action is the same as **RN** on page 258. • **Files connected for direct access**: As much of the data from the output item list as can fit is placed in the record. This includes the data item that would overflow the record; as much of it as can fit is placed in the record. The rest of this item plus the remaining data items in the output item list are written into as many records as necessary to hold the data. Each successive record is written as the record with the next higher record number. • **Files connected for keyed access**: This action is the same as **RN** on page 258.

Symbolic Feedback Code: FOR1001

FOR1002E *locator-text* **An input item required data from beyond the end of the data that was available in a record. The length of the available data was** *data-length*. **VS FORTRAN Version 2 Error Number: AFB164I, AFB201I (format 1), AFB201I (format 2), AFB204I, AFB212I, AFB213I**

Explanation: In a READ statement, one or more of the input items in the input item list required that data be transferred from beyond position *data-length* of the record. *data-length* is the smaller of either the value given in the RECL specifier, if any, on the OPEN statement **or** the length of the record available from the underlying system access methods. This latter length is limited by the value of the LRECL parameter, if any, in the file definition if this LRECL value is less than the actual length of the previously written record. The specific error that is detected differs based on these factors:

- The type of formatting described by the FMT specifier (or by its absence) on the READ statement
- The record format (the RECFM value) for the file
- The value given in the PAD specifier, if any, on the OPEN statement
- The value of the RECPAD run-time option

These are the specific errors that this condition represents:

- **For input using a format specification**:
  Both of the items (1 and 2) were true:
  1. Either of these two conditions occurred:
     - An input item referred to a field that started beyond position *data-length* in the record.
     - An input item with a corresponding edit descriptor of A referred to a field that extended beyond position *data-length* in the record.

     (Note that because the T edit descriptor could have been used to specify the position at which data transfer was to begin, this condition doesn't necessarily imply that just the input items along with their corresponding repeatable edit descriptors represented more data than the record contained.)
  2. Either of these two conditions applied:
     - The OPEN statement had a PAD specifier whose value was NO.
     - There was no PAD specifier on the OPEN statement (or there was no OPEN statement because the file is a preconnected file or an internal file). In addition, one of these cases applied:
       - The RECPAD(NONE) run-time option was in effect.
       - The RECPAD(VAR) run-time option was in effect and the file was an external file that was a non-VSAM file with a record format (that is, the RECFM value that was applied to the file) of either fixed (F) or fixed blocked (FB).
       - The RECPAD(VAR) run-time option was in effect and the file was an external file that was a VSAM relative record data set (RRDS).
- **For unformatted input**:
  All three of the items were true:
  1. The total length of the items in the input item list exceeded the number of bytes of data (*data-length*) available from the record.
  2. The NUM specifier was not given in the READ statement.
  3. If the unit was connected for direct access, the OPEN statement rather than the DEFINE FILE statement from the FORTRAN 66 language standard was used to connect the file.
- **For list-directed input**:
  – Both of the items were true:
  1. There were no characters other than blanks in the record that corresponded to one or more input items.
  2. The record format (that is, the RECFM value that was applied to the file) was either variable spanned (VS) or variable blocked spanned (VBS).

locator-text gives more information about the location of the error, and can be one of the following:
The READ statement for an internal file failed.
The READ statement for unit *unit-number* which was connected to *file-name* failed.

Programmer response:

If you want to read an existing record, change one or more of the following:

- The type of formatting indicated by FMT specifier (or its absence) in the I/O statement. For example, you might have to read the record with list-directed formatting rather than with a format specification in order to be consistent with the structure of the existing record.
- The format specification. This could include errors in the edit descriptors field widths, repetition factors, and nesting levels.
- The input item list. There could be errors in the number of items in the list, in the data types and lengths of individual items, and in the specification of implied DOs in the input item list.

- The value given for the RECL specifier on the the OPEN statement. If this value is limiting the available amount of data to less than what the record actually contains, you should increase it to allow additional data to be read.
- The NUM specifier for an unformatted READ statement. When there is a NUM specifier on an unformatted READ statement, encountering a record that is shorter than the total length of all of the input items does not cause this error to be detected. In this case:
  1. All of the data from the record is transferred to the input items. This includes the data item for which there was not enough data in the record; as much of it as is available is transferred to the item.
  2. The rest of this input item and the remaining input items, if any, are not modified.
  3. The variable or array element *num* given in the NUM=*num* specifier becomes defined with the number of bytes of data that were transferred to the input items.
- The position of the file. Perhaps previous I/O statements (READ, WRITE, BACKSPACE, REWIND, and so on) caused the file to be positioned to a record other than the one you intended to read.
- Blank padding, which can be used when your READ statement has a format specification. When it is used, records are treated as though they were extended with a sufficient number of blanks to supply data for all of the input items. Blank padding is in effect either when the OPEN statement has a PAD specifier that has a value of YES or when there is no PAD specifier when certain sub-options of the RECPAD run-time option are in effect.

  Here are the RECPAD run-time option values that cause blank padding to be in effect:

This run-time option	Applies blank padding to these types of files
RECPAD(VAR)	Any of the following: – An external file that is a non-VSAM file with a record format (the RECFM value) of **other than** fixed (F) or fixed blocked (FB) – An external file that is a VSAM entry-sequenced data set (ESDS) or key-sequenced data set (KSDS) – An internal file (Note that for an external file connected for direct access, a non-VSAM file must have a record format of fixed, and a VSAM file must be an RRDS; therefore, the run-time option RECPAD(VAR) won't alleviate the problem for a file connected for direct access.)
RECPAD(ALL)	Any file (internal or external; VSAM of any type; non-VSAM with any record format)

  When blank padding is used to treat a record as though it were extended with blanks, the values that would be provided for the input items that would otherwise have caused this error to be detected are as follows:
  – For a field that corresponded to an edit descriptor of A and that would have extended beyond the number of characters available from the record, the portion of the field in the record would be extended with blanks and transferred to the input item.
  – For a field that corresponded to an edit descriptor of A and that would have started beyond the number of characters available from the record, the input item becomes defined with a value of blanks.
  – For a field that corresponded to an edit descriptor of L and that would have started beyond the number of characters available from the record, the input item becomes defined with a value of false.
  – For a field that corresponded to one of the numeric edit descriptors and that would have started beyond the number of characters available from the record, the input item becomes defined with a value of zero.

  Note that because a run-time option controls this extension of records with blanks, the appropriate action is taken for all records and files to which it applies.

If the existing record of the file doesn't have the length or structure that you intended, you might have to recreate the file after correcting either the program that originally wrote the file or the file definitions that were in effect when the file was created. There is more detailed information on this subject in the "Programmer Response" topic for condition FOR1001 on page 256.

System action: If either the ERR or the IOSTAT specifier is present on the READ statement, then before control returns to the program, the **RN** action described on page 262 is taken.

If neither the ERR nor the IOSTAT specifier is present on the READ statement, the condition is signaled. If the condition either is unhandled or is handled by moving the resume cursor and resuming, then the **RN** action described on page 262 is taken. If the condition is unhandled, execution of the enclave terminates.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within the basic set, *statement* has a value of READ, and *parm_count* has a value of 7. In addition, there are these qualifying sets:

No.	Name	Input/Output	Data Type and Length	Value
5	*access*	Input	CHARACTER*10	For an external file, the value SEQUENTIAL, DIRECT, or KEYED depending on whether the file is connected for sequential, direct, or keyed access, respectively. For an internal file, this qualifying datum contains the value SEQUENTIAL.
6	*fmt-type*	Input	CHARACTER*8	One of the following values to indicate the type of formatting indicated by the FMT specifier (or its absence) on the READ statement: **Value** / **Type of Formatting**; blanks: Unformatted; FORMAT: Format specification; *: List-directed formatting
7	*data-length*	Input	INTEGER*4	The maximum length of data available to be read. If the OPEN statement had a RECL specifier, then *data-length* could be less than what is reflected in *record-desc*.
8	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
9	*record*	Input	CHARACTER**n*	For a formatted READ statement or for an unformatted READ statement directed to a file with other than spanned records (VS or VBS), the whole input record. For an unformatted READ statement that read from a record that spans more than one block, only the single record segment that includes position *data-length* of the record. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	Formatted and unformatted input are handled slightly differently:

Formatted input:
The input item that required data from beyond the end of the record as well as any remaining input items, if any, are not modified. The remainder of the format specification, if any, is ignored.

Unformatted input:
All of the data from the record is transferred to the items in the input item list. This includes the item for which there was not enough data in the record; as much of it as is available is transferred to the item. The rest of this item and the remaining input items, if any, are not modified.

In both cases, the file is positioned to the end of the record that was read (that is, to the end of the record that was too short).

RF	If the READ statement is for unformatted I/O and it refers to a unit that is connected for direct access, then data from as many of the succeeding records as necessary is transferred to the input items, and the file is positioned to the end of the last record from which data was transferred. (This action is similar to the semantics of the FORTRAN 66 standard except that there is no associated variable.) For other READ statements, this action is the same as **RN**.

Symbolic Feedback Code: FOR1002

FOR1003S *locator-text* **A character that wasn't numeric was found in the formatted input data where a numeric character was expected. The input field was '***input-field***'. VS FORTRAN Version 2 Error Number: AFB215I**

Explanation: For a READ statement, *input-field* is a character string that was interpreted as an integer, real or complex value either because the input item in the input item list was an integer, real or complex data type, or because the format specification had an I, E, F, Q or D edit descriptor. *input-field* contained characters other than 0 through 9 where only these characters were allowed.

When the input is interpreted as a complex value, the expected format of *input-field* depends on whether a format specification is used. If a format specification is used, two Fortran real numbers are used to describe a complex number: one describing the real part of the complex number, the other describing the imaginary part of the complex number. For list-directed or namelist, *input-field* should have the form of a complex constant. Note, however, that for list-directed or namelist input, condition FOR1006 could be detected in some cases for complex input.

locator-text gives more information about the location of the error, and can be one of the following:
  The READ statement for an internal file failed.
  The READ statement for unit *unit-number* which was connected to *file-name* failed.

Programmer response: Ensure that:

- *input-field* contains formatted data that can be converted to the expected data type (refer to *VS FORTRAN Version 2 Language and Library Reference* for the form that integer, real and complex constants can have),
- The edit descriptor does not indicate numeric input where other input should be indicated,
- The edit descriptor specifies the correct field width.

System action: The input item being processed, and the remainder of the items in the input item list are undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm-count* has a value of 8. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*input-field-desc*	Input	Q_DATA_DESC	The q_data descriptor for *input-field*. It contains the data type and the length of *input-field*.
6	*input-field*	input/output	CHARACTER**n*	The formatted input data; that is, the character string that is being interpreted as either an integer, real, or complex value. The length *n* is part of *input-field-desc* and has a maximum possible value of 255.
7	*index*	Input	INTEGER*4	The index in *input_* field of the character in error.
8	*record_desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.

No.	Name	Input/Output	Data Type and Length	Value
9	*record*	Input	CHARACTER**n*	The formatted input record that contained the character string being interpreted as a numeric value. The length *n*, which includes only the data portion of the record, is part of *record-desc*.
10	*result-desc*	Input	Q_DATA_DESC	The q_data descriptor for *result*. It contains the data type and the length of *result*.
11	*result*	output	See *result_desc*.	The result value that is used when the release_option action is requested by the user condition handler. The data type can be an integer with a length of 1, 2, 4, or 8, or it can be real with a length of 4, 8, or 16. When a format specification is used, th real or imaginary part of a number is supplied as qualifying data as a real number of half the length.

Name	Action Taken after Resumption
RI	The character string that is in *input-field* (whose length is part of *input-field-desc*) is converted to a value that has the data type specified by *result-desc* and the data type specified by *result-desc*, and the input item becomes defined with this value. Execution continues, and the remainder of the input item list is processed.
release-option	The input item, or the real or imaginary part of a complex input item, becomes defined with the value *result*.

Symbolic Feedback Code: FOR1003

FOR1004S *locator-text* **A character that wasn't a hexadecimal character was found in the formatted input data where a hexadecimal character was expected. The input field was '***input-field***'. VS FORTRAN Version 2 Error Number: AFB225I**

Explanation: For a READ statement, *input-field* is a character string that was interpreted as hexadecimal data because there was a Z edit descriptor in the format specification. Hexadecimal data consists of the characters 0 through 9 and A through F, but *input-field* contained other characters.

locator-text gives more information about the location of the error, and can be one of the following:
The READ statement for an internal file failed.
The READ statement for unit *unit-number* which was connected to *file-name* failed.

Programmer response: Ensure that:

- *input-field* contains formatted data that can be converted to hexadecimal data,
- The edit descriptor does not indicate hexadecimal input where other input should be indicated.
- The edit descriptor specifies the correct field width.

System action: The input item being processed, and the remainder of the items in the input item list are undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 11. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*input-field-desc*	Input	Q_DATA_DESC	The q_data descriptor for *input-field*. It contains the data type and the length of *input-field*.
6	*input-field*	input/output	CHARACTER**n*	The formatted input data, that is, the character string that is being interpreted as a hexadecimal value. The length *n* is part of *input-field-desc* and has a maximum possible value of 255.
7	*index*	Input	INTEGER*4	The index in *input_* field of the character in error.
8	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
9	*record*	Input	CHARACTER**n*	The formatted input record that contained the character string being interpreted as a numeric value. The length *n*, which includes only the data portion of the record, is part of *record-desc*.
10	*result-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
11	*result*	output	Q_DATA_DESC	The result value that is used when the release_option action is requested by the user condition handler.

Name	Action Taken after Resumption
RI	The character string that is in *input-field* whose length is part of *input-field-desc*) is converted to a hexadecimal value, and the input item becomes defined with this value. Execution continues, and th remainder of the input list is processed.
release_option	The input item becomes defined with the value *result*.

Symbolic Feedback Code: FOR1004

FOR1005S *locator-text* **The formatted input data** *input-field* **was outside the range of values that could be contained within the real or complex input item. VS FORTRAN Version 2 Error Number: AFB226I**

Explanation: For a READ statement, *input-field* is a character string that was interpreted either as the value of a real number or as the value of the real or the imaginary part of a complex number. This value exceeded the permissible range for a floating-point number. The largest magnitude is approximately 7.2E+75, and the smallest magnitude is approximately 5.4E−79.

locator-text gives more information about the location of the error, and can be one of the following:
The READ statement for an internal file failed.
The READ statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Ensure that:

- The formatted data in *input-field* is within the required range for a real number,
- *input-field* does not contain trailing blanks if either the BLANK specifier is given on the OPEN statement with value of ZERO or the BZ edit descriptor is included on the format specification.

System action: The input item being processed, and the remainder of the items in the input item list are undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 10. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*input-field-desc*	Input	Q_DATA_DESC	The q_data descriptor for *input-field*. It contains the data type and the length of *input-field*.
6	*input-field*	Input	CHARACTER*n	The formatted input data, that is, the character string that is being interpreted as a a real value or as the real or imaginary part of a complex value. The length *n* is part of *input-field-desc* and has a maximum possible value of 255.
7	*return-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
9	*record_desc*	Input	Q_DATA_DESC	The q_data descriptor for *result*. It contains the data type and the length of *result*.
9	*result*	output	REAL**n*	The new output value, where *n*, the length is part of *result-desc* an dcould be 4, 8, or 16.

Name	Action Taken after Resumption
release_option	The input item becomes defined with the value in *result*. Execution continues, and the remainder of the input item list is processed.

Symbolic Feedback Code: FOR1005

FOR1006S *locator-text* **For list-directed input, the variable was of complex type, but the formatted input data was not in the correct format for a complex constant. The formatted input data was** *input-field***. VS FORTRAN Version 2 Error Number: AFB238I**

Explanation: For a READ statement, *input-field* is a portion of the character string that was interpreted as a complex constant for list-directed input, and can be either the real part, the imaginary part, or both. *input-field* either contained embedded blanks in the real part or the imaginary part of the complex number, did not contain a comma as a separator between the real part and the imaginary part, or was not enclosed in parentheses. Or, the end of the record occurred other than between the real part and the comma or between the comma and the imaginary part.

locator-text gives more information about the location of the error, and can be one of the following:
The READ statement for an internal file failed.
The READ statement for unit *unit-number* which was connected to *file-name*, failed.

Programmer response: Ensure that *input-field* contains no embedded blanks in the real part or the imaginary part of the complex number, contains a comma as a separator, is enclosed by a left and a right parenthesis, and, if the the complex number does not fit into one record, that the end of the record occurs between the real part and the comma or between the comma and the imaginary part.

System action: The input item being processed, and the remainder of the items in the input item list are undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 8. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*input-field-desc*	Input	Q_DATA_DESC	The q_data descriptor for *input-field*. It contains the data type and the length of *input-field*.
5	*input-field*	Input	CHARACTER*n	The formatted input data, that is, a part of the character string that is being interpreted as a complex number. It can be either the real part, the imaginary part, or both. The length *n* is part of *input-field-desc* and has a maximum possible value of 255.
7	*return-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
8	*record*	Input	CHARACTER**n*	The formatted input record that contained the character string being interpreted as a complex number. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	The current operation is ignored. The remainder of the deallocation list is processed and execution continues.

Permissable Resume Actions:

Name	Action Taken after Resumption
RN	Execution continues, and the remainder of the input item list is ignored.

Symbolic Feedback Code: FOR1006

FOR1007S *locator-text* **The input item was of type character, but the formatted input data did not begin with an apostrophe or with a quote. The input field was** *input-field***. VS FORTRAN Version 2 Error Number: AFB238I**

Explanation: For a READ statement, *input-field* is a character string that was interpreted as a character constant for list-directed input. *input-field* did not begin with an apostrophe or with a quote.

locator-text gives more information about the location of the error, and can be one of the following:
The READ statement for an internal file failed.
The READ statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Ensure that *input-field* is delimited by apostrophes or quotes, and that the beginning and ending delimiters are the same.

System action: The input item being processed and the remainder of the input items in the input item list are undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 8. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*input-field-desc*	Input	Q_DATA_DESC	The q_data descriptor for *input-field*. It contains the data type and the length of *input-field*.
6	*input-field*	Input	CHARACTER**n*	The formatted input data, that is, the character string that is being interpreted as a character value. The length *n* is part of *input-field-desc* and has a maximum possible value of 255.
7	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
8	*record*	Input	CHARACTER**n*	The formatted input record that contained the character string being interpreted as a character value. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	The remainder of the input items in the input item list are ignored, and execution continues.

Symbolic Feedback Code: FOR1008

FOR1009S *locator-text* **The list-directed input data did not have a value separator following the ending delimiter. The input data ended with** *input-field*. **VS FORTRAN Version 2 Error Number: AFB238I**

Explanation: For a READ statement, *input-field* is a character string that was interpreted as a character constant (delimited by apostrophes or quotes) or a complex constant (delimited by parentheses) for list-directed input. *input-field* was not followed by a value separator, where a value separator can be either a comma (**,**), a blank, or a slash (**/**).

locator-text gives more information about the location of the error, and can be one of the following:
The READ statement for an internal file failed.
The READ statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Ensure that *input-field* is followed by a value separator.

System action: The input item being processed and the remainder of the items in the input item list are undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 8. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*input-field-desc*	Input	Q_DATA_DESC	The q_data descriptor for *input-field*. It contains the data type and the length of *input-field*.
6	*input-field*	Input	CHARACTER**n*	The formatted input data; that is, the character string that is being interpreted either as a character value or as a complex number. The length *n* is part of *input-field-desc* and has a maximum possible value of 255.

No.	Name	Input/Output	Data Type and Length	Value
7	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
8	*record*	Input	CHARACTER**n*	The formatted input record that contained the character constant with no delimiter. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1009

FOR1010S *locator-text* **A character that wasn't a binary character was found in the formatted input data where a binary character was expected. The input field was '*input-field*'.**

Explanation: For a READ statement, *input-field* is a character string that was interpreted as binary data because there was a B edit descriptor in the format specification. Binary data consists of the characters 0 and 1, but *input-field* contained other characters.

locator-text gives more information about the location of the error, and can be one of the following:
The READ statement for an internal file failed.
The READ statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Ensure that:

- *input-field* contains formatted data that can be converted to binary data,
- The edit descriptor does not indicate binary input where other input should be indicated,
- The edit descriptor specifies the correct field width.

System action: The input item being processed and the remainder of the items in the input item list are undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 11. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*input-field-desc*	Input	Q_DATA_DESC	The q_data descriptor for *input-field*. It contains the data type and the length of *input-field*.
6	*input-field*	Input / output	CHARACTER**n*	The formatted input data; that is, the character string that is being interpreted as a binary value. The length *n* is part of *input-field-desc* and has a maximum possible value of 255.
7	*index*	Input	INTEGER*4	The index in *input_* field of the character in error.
8	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.

No.	Name	Input/Output	Data Type and Length	Value
9	*record*	Input	CHARACTER**n*	The formatted input record that contained the character string being interpreted as a binary value. The length *n*, which includes only the data portion of the record, is part of *record-desc*.
10	*result-desc*	Input	Q_DATA_DESC	The q_data descriptor for *result*. It contains the data type and the length of *result*.
11	*result*	output	See *result-desc*	The result value that is used when the release_option action is requested by the user condition handler.

Permissable Resume Actions:

Name	Action Taken after Resumption
RI	The character string that is in *input-field* (whose length is part of *input-field-desc*) is converted to a binary value, and the input item becomes defined with this value. Execution continues, and the remainder of the input item list is processed.
release_option	The input item becomes defined with the value *result*.

Symbolic Feedback Code: FOR1010

FOR1011S *locator-text* **A character that wasn't an octal character was found in the formatted input data where an octal character was expected. The input field was '***input-field***'.**

Explanation: For a READ statement, *input-field* is a character string that was interpreted as octal data because there was an O edit descriptor in the format specification. Octal data consists of the characters 0 through 7, but *input-field* contained other characters.

locator-text gives more information about the location of the error, and can be one of the following:
The READ statement for an internal file failed.
The READ statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Ensure that:

- *input-field* contains formatted data that can be converted to octal data,
- The edit descriptor does not indicate octal input where other input should be indicated.
- The edit descriptor specifies the correct field width.

System action: The input item being processed and the remainder of the items in the input item list are undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 11. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*input-field-desc*	Input	Q_DATA_DESC	The q_data descriptor for *input-field*. It contains the data type and the length of *input-field*.
6	*input-field*	Input/Output	CHARACTER**n*	The formatted input data, that is, the character string that is being interpreted as an octal value. The length *n* is part of *input-field-desc* and has a maximum possible value of 255.

No.	Name	Input/Output	Data Type and Length	Value
7	*index*	Input	INTEGER*4	The index in *input_* field of the character in error.
8	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
9	*record*	Input	CHARACTER**n*	The formatted input record that contained the character string being interpreted as an octal value. The length *n*, which includes only the data portion of the record, is part of *record-desc*.
10	*result-desc*	Input	Q_DATA_DESC	The q_data descriptor for *result*. It contains the data type and the length of *result*.
11	*result*	output	See *result-desc*	The result value that is used when the release_option action is requested by the user condition handler.

Name	Action Taken after Resumption
RI	The character string that is in *input-field* (whose length is part of *input-field-desc*) is converted to a binary value, and the input item becomes defined with this value. Execution continues, and the remainder of the input item list is processed.
release_option	The input item becomes defined with the value *result*.

Symbolic Feedback Code: FOR1011

FOR1020S **The** *statement* **statement for sequential access for unit** *unit-number***, which was connected to** *file-name***, failed. The file had been connected for** *access* **access. VS FORTRAN Version 2 Error Number: AFB163I, AFB231I (format 1)**

Programmer response: Ensure that you use only the I/O statements that are consistent with the access mode in use for the file connection. For example, if you intend to use direct access, then don't use the file positioning statements (BACKSPACE, ENDFILE, REWIND). Instead, use only READ or WRITE statements with a REC specifier; the value given for the REC specifier is the number of the record that you want to read or write.

If you want to read the file sequentially, then you must connect the file for sequential access by executing an OPEN statement in which the access specifier has a value of SEQUENTIAL (or in which the access specifier is omitted). If the file was already connected for direct access and you want to process it with sequential access, then you must execute a CLOSE statement before the OPEN statement.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1020

FOR1022S **The** *statement* **statement for unit** *unit-number***, which was connected to** *file-name***, failed. The file was already positioned after the endfile record. VS FORTRAN Version 2 Error Number: AFB101I, AFB218I (format 8)**

Explanation: The file was positioned after the endfile record either because the end-of-file condition was just detected for a READ statement or because you just executed an ENDFILE statement. Your *statement* statement implied the use of a subsequent subfile, but your file did not have multiple subfiles because it was a dynamically allocated scratch file, a named file, or a striped file.

Programmer response: Do not try to read or write records beyond the endfile record for the files that don't support multiple subfiles. The only files that can have multiple subfiles are unnamed files that are neither striped files nor dynamically allocated.

Check the logic of your program to ensure that you haven't inadvertently executed a READ or WRITE statement either after reaching the end of the file or after executing an ENDFILE statement. Use the END specifier, if necessary, to detect the end of the file.

If you want to extend an existing file after reading through the data records and detecting the end of the file, you can do so by executing a BACKSPACE statement (which positions the file just beyond the last data record and just before the endfile record) followed by WRITE statements.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1022

FOR1023S **The** *statement* **statement for unit** *unit***, which was connected to** *file-name***, failed. The file definition statement referred to a file on a device or with a file organization for which the** *statement* **statement is not supported. VS FORTRAN Version 2 Error Number: AFB095I (format 1), AFB166**

Explanation: Your program executed a *statement* statement, but the file to which the unit is connected doesn't support this statement. For example, the following statements are inconsistent with the types of files indicated:

Statement	Inconsistent file type
BACKSPACE	Printer
BACKSPACE	Striped file
ENDFILE	PDS member
REWIND	Language Environment message file

Programmer response: Either

- Change the logic of your program to avoid the use of the prohibited I/O statements for the file that you're using, or
- Change the file definition (DD statement, ALLOCATE command) to refer to a different type of file, and maintain the data in that other type of file.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1023

FOR1024S **The BACKSPACE statement for unit** *unit-number***, which was connected to** *file-name***, failed. The file definition statement refers to a PDS member and the previous statement was a WRITE statement. VS FORTRAN Version 2 Error Number: AFB095I (format 2)**

Programmer response: Because the sequence of a WRITE statement followed by a BACKSPACE statement isn't allowed for a file that is a PDS member, make or both of these changes:

- Modify the program so it doesn't use this sequence of statements.

- Change the file definition (DD statement, ALLOCATE command) to refer to a different type of file, and maintain the data in that other type of file.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of BACKSPACE, and *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1024

FOR1025S **The BACKSPACE statement for unit *unit-number*, which was connected to *file-name*, failed. The file definition statement referred to a PDS member, and BUFNO had a value greater than 1. VS FORTRAN Version 2 Error Number: AFB095I (format 2)**

Programmer response: Because the BACKSPACE statement isn't allowed with a file that is a PDS member and that is processed with more than one buffer, make one or more of these changes:

- Modify the program so it does not use a BACKSPACE statement.
- Change the file definition (DD statement, ALLOCATE command) to refer to a different type of file, and maintain the data in that other type of file.
- Change the file definition statement either by removing the BUFNO parameter that has a value greater than 1 or by providing a BUFNO=1 parameter.
- For a dynamically allocated file that has an associated call to the FILEINF callable service, either remove the BUFNO parameter that has a value greater than 1 or provide a a BUFNO parameter with an associated value of 1.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of BACKSPACE, and *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1025

FOR1026S **The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The file definition statement referred to a PDS member, and *statement* statement nor a REWIND statement. VS FORTRAN Version 2 Error Number: AFB123I (format 6), AFB123I (format 7)**

Explanation: The sequence of I/O statements that was executed for a file that was a PDS member caused a change either from input to output processing or from output to input processing without an intervening REWIND statement.

Programmer response: Change the logic of the program to avoid switching between input and output processing.

If you must do both input and output processing on the file, either

- Insert an intervening REWIND statement (or a CLOSE followed by an OPEN statement) between the two types of processing if the logic of the program and the desired file positioning allows it, or
- Change the file definition (DD statement, ALLOCATE command) to refer to a different type of file, and maintain the data in that other type of file.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1026

FOR1027S The READ statement for unit *unit-number*, which was connected to *file-name*, failed. The file definition statement referred to a system output (sysout) data set. VS FORTRAN Version 2 Error Number: AFB218I (format 3)

Programmer response: If the file is supposed to be a system output data set (SYSOUT parameter on the DD statement), then change the logic of your program so that you don't execute a READ statement for a unit that's connected to this file.

If the file should be another type, then change the file definition (DD statement, ALLOCATE command) to refer to the file that you intend to use.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1027

FOR1028S The *statement* statement for unit *unit-number*, failed. The file was already positioned after the endfile record of the 999th subfile, which is the last subfile allowed.

Explanation: Your program has already processed 999 subfiles and is trying to position itself into the next one; however, 999 is the maximum number of subfiles that can be handled. Note that each subsequent subfile is referenced when you execute a READ, WRITE, or ENDFILE statement either after the end-of-file condition was just detected for a READ statement or after you just executed an ENDFILE statement.

Programmer response: Ensure that your program doesn't inadvertently execute a READ, WRITE, or ENDFILE statement for the same unit either after reaching the end of the file or after executing an ENDFILE statement. Use the END specifier, if necessary, on the READ statement to detect the end of the file.

If you want to extend an existing file after reading through the data records and detecting the end of the file, you can do so by executing a BACKSPACE statement (which positions the file just beyond the last data record and just before the endfile record) followed by WRITE statements

System action: The file is closed. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, and *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1028

FOR1070S The direct access *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The REC specifier had a value of *record-number*, which either was not positive or exceeded *num-records*, the number of records in the file. VS FORTRAN Version 2 Error Number: AFB232I

Programmer response: Ensure that the REC specifier on the direct access READ or WRITE statement has a value that is neither less than 1 or greater than the number of records in the file.

If you're sure that the file contains the correct number of records, then just correct the logic of your program to provide the correct value for the REC specifier.

If you expect the file to contain more records than it does, then you'll have to delete the existing file and create it again to give it additional space. In this case, the term *delete* means that the data set's disk space must be released and the data set allocated again. Note that unless the file is dynamically allocated, neither of the following release the data set's existing space on the disk volume:

- A CLOSE statement with a STATUS specifier value of DELETE or
- An OPEN statement with a STATUS specifier value of REPLACE.

However, if the file is dynamically allocated, you can use these forms of the OPEN and CLOSE statements to release the space from within your Fortran program.

For a non-VSAM file, you should indicate the amount of space in the file by one of the following:

- In MVS JCL: the SPACE parameter on the DD statement
- In the TSO ALLOCATE command: the SPACE operand along with the BLOCK, TRACKS, or CYLINDERS operand
- In the FILEINF argument list for a dynamically allocated file: the CYL, TRK, or MAXREC keyword argument

For a VSAM relative record data set (RRDS), you should indicate the amount of space on the Access Method Services DEFINE CLUSTER command with the CYLINDERS, RECORDS, or TRACKS parameter.

When the new file is first connected with a Fortran OPEN statement whose ACCESS specifier has a value of DIRECT, it is automatically formatted with as many records as will fit in the space allocated to the data set. (However, when the DEFINE FILE statement from the FORTRAN 66 language standard is used, only the number of records specified in that statement is formatted.)

Alternatively, you can format a newly created file by using sequential access rather than direct access. In this case, simply write as many records as the file is supposed to to hold, then close the file, and reconnect it for direct access. Remember that a file connected for direct access must have a record format that indicates fixed-length unblocked records and a record length that is the same as the value of the RECL specifier on the OPEN statement that's used to connect the file for direct access. Therefore, if you're using sequential access to format the file, ensure that the record format and record length are correctly specified in the DD statement, in the TSO ALLOCATE command, or in the the arguments for the FILEINF callable service.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 6. In addition, there are these qualifying data:

No.	**Name**	**Input/Output**	**Data Type and Length**	**Value**
5	*record-number*	Input	INTEGER*4	The invalid record number that was given in the REC specifier.
6	*num-records*	Input	INTEGER*4	The number of records for which there is space within the file.

Permissable Resume Actions:

Name	**Action Taken after Resumption**
RN	The remainder of the input item is ignored, and execution continues.

Symbolic Feedback Code: FOR1070

FOR1071S The direct access *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The file had been connected for *access* access. VS FORTRAN Version 2 Error Number: AFB235I

Explanation: Your program executed a *statement* statement with a REC specifier, which indicates direct access, but the file was connected for *access* rather than direct access.

Programmer response: Ensure that you use only the I/O statements that are consistent with the access mode in use for the file connection. For example, if you intend to use sequential access, then don't use READ or WRITE statements with a REC specifier because these apply only to direct access.

If you want to read the file using direct access, then connect the file by executing an OPEN statement in which the access specifier has a value of DIRECT. (In a program that uses the FORTRAN 66 language standard, the DEFINE FILE statement has a function similar to the OPEN statement.)

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1071

FOR1072S The direct access READ statement for unit *unit-number*, which was connected to *file-name*, failed. The file was empty. VS FORTRAN Version 2 Error Number: AFB236I (format 1)

Programmer response: Ensure that whatever program created the file actually wrote records into the file. The three most common ways of writing these records are:

- Using the OPEN statement with a value of DIRECT for the ACCESS specifier when the file doesn't yet exist. If the OPEN statement does **not** have either:
  - A STATUS specifier with a value of OLD, or
  - An ACTION specifier with a value of READ.

  then during execution of the OPEN statement the file should be formatted automatically with as many records as it can hold.
- Using sequential access to write the records. In this case, simply write as many records as the file is supposed to to hold, then close the file, and reconnect it for direct access. Remember that a file connected for direct access must have a record format that indicates fixed-length unblocked records and a record length that is the same as the value of the RECL specifier on the OPEN statement that's used to connect the file for direct access. Therefore, if you're using sequential access to format the file, ensure that the record format and record length are correctly specified in the DD statement, in the TSO ALLOCATE command, or in the the arguments for the FILEINF callable service.
- Using a utility program or a program written in some other language. If the file is to be used with direct access by a Fortran, it must have a record format that indicates fixed-length unblocked records and a record length that is the same as the value of the RECL specifier on the Fortran OPEN statement that will be used to connect the file for direct access.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1072

FOR1100S The REWRITE statement for unit *unit-number*, which was connected to *file-name*, failed. The key of reference had a value of *new-rec-key* (X'*new-rec-key-hex*'), which was different from *orig-rec-key* (X'*orig-rec-key-hex*'), the value of that key in the record that was read. The key of reference had a KEYID value of *keyid*. VS FORTRAN Version 2 Error Number: AFB139I

Explanation: You read a record and, in trying to rewrite it, provided a record in your output item with a key of reference whose value differed from that in the original record. You cannot use the REWRITE statement to replace a record if you change the value of the key of reference.

The term *key of reference* means the key (that is, certain positions within the record) that was used for the sequential or direct retrieval of the record that was read. The term *KEYID value* means the relative position of the start-end pair for this key within the start-end pairs listed in the KEYS specifier on the OPEN statement.

Programmer response: If you intended to replace an existing record rather than to write one with a modified key value, ensure that:

- The value of the key in the output item list is the same as what was just read and that it is in the correct position in the record,
- The output item list contains all the fields of the record to be rewritten, and
- Any changes in the order or length of various fields in the record have not caused the value of the key of reference within the record to be shifted from its original position.

If you want to replace the record with one having a different key value, use the DELETE statement to delete the record that was read, and then

- If there isn't already a record in the file with the different key value, then use the WRITE statement, rather than the REWRITE statement, to add the record with a new key value.
- If there is already a record in the file with the different key value, then use the READ statement to read that record, and then use the REWRITE statement to replace that record.

If you want to add a new record with a key value that doesn't already exist in the file, then use the WRITE statement to add it.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of REWRITE, and *parm_count* has a value of 6. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
6	*record*	Input	CHARACTER**n*	The record. The length *n* is part of *record-desc*.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1100

FOR1102S The WRITE statement for unit *unit-number*, which was connected to *file-name*, failed. Records were being loaded into the file, and the primary key had a value of *new-key* (X'*new-key-hex*'), which was not greater than *prev-key* (X'*prev-key-hex*'), the value of the primary key in the previous record. VS FORTRAN Version 2 Error Number: AFB140I

Explanation: Because the OPEN statement for this file connection had a value of WRITE for the ACTION specifier, the file was connected for use in loading records in order of ascending primary key value. You attempted to load a record in which the value of the primary key was not greater than the value of the primary key in the previous record.

The term *primary key* refers to the main key for the VSAM key-sequenced data set as it was declared in the Access Method Services DEFINE CLUSTER command.

Programmer response: Ensure that your output item list has the primary key at its proper position within the record being written. If you're not sure where this key is located in the record, use the Access Method Services LISTCAT command to find out. Then change your output item list (and the KEYS specifier, if present, on the OPEN statement) so that it is consistent with the primary key position known to VSAM. On the other hand, if the output of the LISTCAT command shows the key in a different position than you intended, then delete and redefine the file with the Access Method Services DELETE and DEFINE CLUSTER commands.

Change the logic of your program or the order of the records being loaded so that the records are presented in increasing sequence of their primary key values.

If you cannot present the records in increasing key sequence, then on the OPEN statement change the value of the ACTION specifier to READWRITE. This allows records to be added to the file without regard for the order of their keys.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of WRITE, and *parm_count* has a value of 6. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
6	*record*	Input	CHARACTER**n*	The record. The length *n* is part of *record-desc*.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1102

FOR1103S **The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The previous I/O statement for this unit resulted in a condition that caused the loss of position in the file. VS FORTRAN Version 2 Error Number: AFB123I (format 1), AFB123I (format 2), AFB123I (format 3), AFB123I (format 4), AFB123I (format 5)**

Explanation: The *statement* statement was not allowed for either or both of these reasons:

- It depended on a previous statement to establish or retain a position (or record pointer) within the file.
- The execution of a previous statement caused the loss of position in the file. This file position could have been lost because of any of a number of conditions:
  - Record-not-found condition
  - Duplicate-key condition
  - End-of-file condition
  - Any error condition

You can neither read records sequentially nor use a BACKSPACE statement until you have reestablished file position. In addition, you cannot use a DELETE, or REWRITE statement except immediately after successfully reading a record.

Programmer response: Ensure that a BACKSPACE statement or a sequential retrieval READ statement isn't executed until a position has been established within the file. This position can be established by a successfully executed OPEN, REWIND, or direct retrieval READ statement.

Ensure that a DELETE or REWRITE statement isn't executed unless a record has just been read.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1103

FOR1104S **The READ statement for unit** *unit-number***, which was connected to** *file-name***, failed. The KEYID specifier had a value of** *keyid***, which conflicted with** *num-keys***, the number of keys specified in the KEYS specifier on the OPEN statement. VS FORTRAN Version 2 Error Number: AFB124I**

Explanation: The value of the KEYID specifier on the OPEN statement is either less than 1 or greater than the number of start-end pairs in the KEYS specifier. Therefore, no pair (and hence no key) can be associated with the value of the KEYID specifier.

This conflict can arise even if no KEYS specifier is coded: a default of one key is assumed, so if KEYID has a value greater than 1, an error is detected.

Programmer response: If the KEYS specifier on the OPEN statement accurately indicates the keys that you want to use, then change the value of the KEYID specifier so that it is a positive integer that is no larger than the number of start-end pairs in the KEYS specifier. Its value should be the ordinal number of the start-end pair for the key that you want to use.

If you intended to have additional keys available for use, then specify their starting and ending record positions as start-end pairs in the KEYS specifier on the OPEN statement. In addition, provide the file definitions (DD statements or TSO ALLOCATE commands) to refer VSAM paths for the additional keys.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1104

FOR1106S **The READ statement for unit** *unit-number***, which was connected to** *file-name***, failed. The argument to be used in searching for a key had a length of** *arg-length***, which was greater than** *key-length***, the length of the key on which the search is being made. The KEYID value was** *keyid***. VS FORTRAN Version 2 Error Number: AFB125I**

Explanation: The argument to be used in searching for a key was given in the KEY, KEYGE, or KEYGT specifier of a READ statement. However, the length of this argument is greater than the length of the key of reference.

The term *key of reference* means the key (that is, certain positions within the record) that was used for the direct retrieval of the record that was to be read. The key of reference is indicated by the *KEYID value*, which is the relative position of the start-end pair for this key within the start-end pairs listed in the KEYS specifier on the OPEN statement. The key of reference can be established through the KEYID specifier on a direct retrieval READ statement and remains in effect until it is changed in another direct retrieval READ statement. The KEYID value is 1 if it has not been specified since the file was connected.

Programmer response: Provide a search argument in the KEY, KEYGE, or KEYGT specifier whose length does not exceed that of the key of reference. If you want to search with a different key of reference, then for the KEYID specifier specify the ordinal number of the desired key's start-end pair in the KEYS specifier on the OPEN statement.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 7. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*key-desc*	Input	Q_DATA_DESC	The q_data descriptor for *key*. It contains the data type and the length of *key*.
6	*key*	Input	See *key-desc*	The value of the key argument. The data type can be integer of length 1, 2, 4, or 8, or it can be character with any positive length not exceeding 255.
7	*keyid*	Input	INTEGER*4	The value of the KEYID specifier.

Name	Action Taken after Resumption
RN	The current operation is ignored. The remainder of the deallocation list is processed and execution continues.

Symbolic Feedback Code: FOR1106

FOR1107S **No record with the specified key could be found for the keyed access READ statement for unit** *unit-number***, which was connected to** *file-name***. The** *key* **specifier had a value of** *value* **(X'***hex-value***'). The KEYID value was** *keyid* **(***start***:***end***). VS FORTRAN Version 2 Error Number: AFB126I**

Explanation: For the key of reference there was no record in the file meeting the search criterion indicated by the search argument in the KEY, KEYGE, or KEYGT specifier on the READ statement. This is the *record not found condition* and might not be an error.

The term *key of reference* means the key (that is, certain positions within the record) that was used for the direct retrieval of the record that was to be read. The key of reference is indicated by the *KEYID value*, which is the relative position of the start-end pair for this key within the start-end pairs listed in the KEYS specifier on the OPEN statement. The key of reference can be established through the KEYID specifier on a direct retrieval READ statement and remains in effect until it is changed in another direct retrieval READ statement. The KEYID value is 1 if it has not been specified since the file was connected.

Programmer response: Ensure that the KEYID value on this or an a previously executed READ statement represents the desired key of reference.

Check the value of the KEY, KEYGE, or KEYGT specifier to ensure that it provides the desired search argument.

Check the program and the data that was used to create the file to ensure that the expected records are actually in the file.

If you want your program to get control in the event of a record not found condition, then on the READ statement add a NOTFOUND specifier with the label of the statement to be given control should the record not found condition occur.

System action: If the IOSTAT=*ios* specifier is present on the READ statement, *ios* becomes defined either with the value 1107 if *ios* is an integer variable or with the condition token for FOR1107 if *ios* is a character variable of length 12.

If the NOTFOUND=*nfd* specifier is present on the READ statement, control passes to the label *nfd*. If the NOTFOUND specifier is not present on the READ statement but the ERR=*err* specifier is present, control passes to the label *err*.

If neither the ERR, the NOTFOUND, nor the IOSTAT specifier is present on the READ statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 7. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*key-desc*	Input	Q_DATA_DESC	The q_data descriptor for *key*. It contains the data type and the length of *key*.
6	*key*	Input	See *key-desc*	The value of the key argument. The data type can be integer of length 1, 2, 4, or 8, or it can be character with any positive length not exceeding 255.
7	*keyid*	Input	INTEGER*4	The value of the KEYID specifier.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1107

FOR1112S **The** *statement* **statement for unit** *unit-number***, which was connected to** *file-name***, failed. The record being written had a length of** *length***, which was too short to contain all the keys. VS FORTRAN Version 2 Error Number: AFB129I (format 1), AFB129I (format 2)**

Explanation: The record described by the output item list on the *statement* statement wasn't long enough to contain all of the keys that were defined for the file through Access Method Services. The record being written must be long enough to contain all of these keys even if one or more of the keys wasn't listed as a start-end pair in the KEYS specifier of the OPEN statement.

Programmer response: Ensure that the output item list defines a record long enough to contain all of the keys made known to VSAM through the DEFINE CLUSTER, DEFINE ALTERNATE INDEX, and DEFINE PATH commands for Access Method Services. Use the LISTCAT command if necessary to determine the position of these keys.

Remember that with a REWRITE statement the output item list must provide output items that describe the whole record rather than just the portions that are to be rewritten.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 6. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
6	*record*	Input	CHARACTER*n	The record. The length *n* is part of *record_desc*.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1112

FOR1113S **The** *statement* **statement for unit** *unit-number***, which was connected to** *file-name***, failed. The file had been connected for** *access* **access, and the** *statement* **statement was of a form that applies to keyed access. VS FORTRAN Version 2 Error Number: AFB127I, AFB128I**

Explanation: One of the following specifiers was given on the *statement* statement:
KEY
KEYGE

KEYGT
NOTFOUND
DUPKEY
KEYID

These specifiers are applicable only to files connected for keyed access, but the file connection was for *access* access.

Programmer response: If your file is a VSAM key-sequenced data set (KSDS), then provide a value of KEYED for the ACCESS specifier on the OPEN statement. This will allow you to use I/O statements that apply to keyed access.

If your file is not a VSAM KSDS, then you cannot use keyed access nor can you use I/O statements that apply only to keyed access. In this case, you have two choices:

- Change the logic of your program to use only the statements that apply to sequential or direct access.
- Define or redefine the file using Access Method Services to make it a VSAM KSDS. Then you can connect the file for keyed access and use any of the I/O statements that apply to keyed access.

Do not confuse sequential or direct access with sequential or direct retrieval statements. Sequential and direct access are indicated by the ACCESS specifier on the OPEN statement. Sequential and direct retrieval statements are forms of the READ statement that can be used when the file is connected for keyed access, that is, when the ACCESS specifier on the OPEN statement has a value of KEYED.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1113

FOR1114S **The WRITE statement for unit *unit-number*, which was connected to *file-name*, failed. A key value within the record to be written was a duplicate of a key in a record already in the file. The key of reference had a position of *start* : *end* in the record, a hexadecimal value of *value*, and a KEYID of *keyid*. VS FORTRAN Version 2 Error Number: AFB135I**

Explanation: A WRITE statement that was used with a file that was connected for keyed access provided in its output item list a record with a key whose value was the same as one that was already in the file. The duplicate key value was either for the primary key, which never allows duplicate key values, or for an alternate-index key that does not allow duplicate values because it was defined to be unique through the Access Method Services DEFINE ALTERNATEINDEX command. The key whose value is duplicated is not necessarily the key of reference, which is indicated in the message text, and it isn't even necessarily among the keys listed in the KEYS specifier of the OPEN statement for the file. However, the key is one that was defined using the Access Method Services DEFINE CLUSTER or DEFINE ALTERNATE INDEX command.

This is the *duplicate key condition* and might not be an error.

(The term *key of reference* means the key (that is, certain positions within the record) that is currently in use for reading and writing records. The key of reference is indicated by the *KEYID value*, which is the relative position of the start-end pair for this key within the start-end pairs listed in the KEYS specifier on the OPEN statement. The key of reference can be established through the KEYID specifier on a direct retrieval READ statement and remains in effect until it is changed in another direct retrieval READ statement. The KEYID value is 1 if it has not been specified since the file was connected.)

Programmer response: Ensure that the output item list defines a record such that no key positions have values that are the same as values that are already in the file. This applies both to the primary key and to all alternate index keys whose values are supposed to be unique. Carefully check the record to be sure that it is in the intended format.

Remember that the key whose value has been duplicated in your record might not be one that your program uses. You'll have to check all of the keys made known to VSAM through the DEFINE CLUSTER, DEFINE ALTERNATE

INDEX, and DEFINE PATH commands for Access Method Services. Use the LISTCAT command if necessary to determine the position of these keys.

If you want your program to get control in the event of a duplicate key condition, then on the WRITE or REWRITE statement add a DUPKEY specifier with the label of the statement to be given control should the duplicate key condition occur.

System action: If the IOSTAT=*ios* specifier is present on the I/O statement, *ios* becomes defined either with the value 1114 if *ios* is an integer variable or with the condition token for FOR1114 if *ios* is a character variable of length 12.

If the DUPKEY=*dky* specifier is present on the I/O statement, control passes to the label *dky*. If the DUPKEY specifier is not present on the I/O statement but the ERR=*err* specifier is present, control passes to the label *err*.

If neither ERR, the DUPKEY, nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of WRITE, and *parm_count* has a value of 6. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
6	*record*	CHARACTER**n*	The record. The length *n* is part of *record-desc*.	

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1114

FOR1180S **The formatted** *statement* **statement for unit** *unit-number***, which was connected to** *file-name***, failed. The file was connected for unformatted I/O. VS FORTRAN Version 2 Error Number: AFB174I**

Explanation: A formatted *statement* statement was executed for a unit that was connected for unformatted input/output operations.

A formatted I/O statement is a PRINT statement, or it is identified by the presence of an FMT specifier in the I/O statement's control list as follows:

- The presence of a FMT specifier with the FMT keyword. For example:

  ```
  READ (FMT=*, UNIT=8) A

  READ (8, FMT=10) A
  ```
- The absence of the UNIT keyword in the unit specifier and the presence of an immediately following specifier that has no keyword. For example:

  ```
  READ (8, NL1) A

  READ (8, *) A
  ```

The file was connected for unformatted input/output operations because either:

- A value of UNFORMATTED was given for the FORM specifier on the OPEN statement,
- The FORM specifier was omitted from the OPEN statement for a file that was connected for direct or keyed access, or
- The first I/O statement that was executed for a preconnected file was an unformatted I/O statement, that is, an I/O statement with no FMT specifier.

Programmer response: Determine whether you want to perform formatted or unformatted input/output operations on the file. If you want to use unformatted I/O statements, remove or change the formatted I/O statements.

If you want to use formatted I/O statements, then:

- For an OPEN statement that connects a file for sequential access, either omit the FORM specifier or provide one with a value of FORMATTED.
- For an OPEN statement that connects a file for direct or keyed access, provide a FORM specifier with a value of FORMATTED.
- Do not execute an unformatted I/O statement for the unit.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 7.

Permissable Resume Actions:

Name	Action Taken after Resumption
RN	The remainder of the input item list is ignored, and execution continues.

Symbolic Feedback Code: FOR1180

FOR1181S *locator-text* **A format specification contained an invalid edit descriptor of** *code***. VS FORTRAN Version 2 Error Number: AFB211I**

Explanation: A format specification referenced by the I/O statement contained an edit descriptor (sometimes called a *format code*) that was

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for an internal file failed.
The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: If the FMT specifier on the I/O statement referred to the format specification in a variable (rather than as a constant or as a label of a FORMAT statement), ensure that the format specification was constructed according to the same rules that apply for a FORMAT statement. In particular, it must start with a left parenthesis, end with a right parenthesis, have no imbedded blanks except within a pair of quotes or apostrophes, and use only the edit descriptors that are allowed by the Fortran language.

Ensure that the logic of your program didn't inadvertently overlay storage such as by referring to array elements outside the declared bounds of the array.

System action: The input or output item being processed and the remainder of the items in the input/output item list are undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The invalid edit descriptor is treated as the end of the format specification, and execution continues.

Symbolic Feedback Code: FOR1181

FOR1182S *locator-text* **A format specification contained more than 51 nested parenthesis groups. VS FORTRAN Version 2 Error Number: AFB160I**

Programmer response: In your format specification, do not use more than 51 levels of nesting for parenthesis groups.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The parenthesis group is ignored, and execution continues. The results are unpredictable.

Symbolic Feedback Code: FOR1182

FOR1183S *locator-text* **The format specifier on the I/O statement did not refer to a FORMAT statement. VS FORTRAN Version 2 Error Number: AFB211I**

Programmer response: Ensure that the format specifier refers to a correctly structured format specification. Either provide the label of a FORMAT statement or character expression whose value is a format specification.

Ensure that the logic of your program didn't inadvertently overlay storage such as by referring to array elements outside the declared bounds of the array.

System action: The input or output item being processed and the remainder of the items in the input/output item list are undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The format field is treated as an end of format, and execution continues.

Symbolic Feedback Code: FOR1184

FOR1200S **The unformatted** *statement* **statement for unit** *unit-number***, which was connected to** *file-name***, failed. The file was connected for formatted I/O. VS FORTRAN Version 2 Error Number: AFB173I**

Explanation: An unformatted *statement* statement was executed for a unit that was connected for formatted input/output operations.

An unformatted I/O statement is recognized by the absence of an FMT specifier. For example, these are unformatted I/O statements:

```
WRITE (10) A, B

READ (7, END=88) A, B
```

and these are formatted I/O statements:

```
WRITE (FMT=*, UNIT=8) A

WRITE (8, FMT=*) A

READ (8, NL1) A

READ (8, *, END=77) A

PRINT 10, A, B, C
```

The file was connected for formatted input/output operations because either:

- A value of FORMATTED was given for the FORM specifier on the OPEN statement,
- The FORM specifier was omitted from the OPEN statement for a file that was connected for sequential access, or
- The first I/O statement that was executed for a preconnected file was a formatted I/O statement, that is, an I/O statement with an FMT specifier.

Programmer response: Determine whether you want to perform formatted or unformatted input/output operations on the file. If you want to use formatted I/O statements, remove or change the unformatted I/O statements.

If you want to use unformatted I/O statements, then:

- For an OPEN statement that connects a file for sequential access, provide a FORM specifier with a value of UNFORMATTED.
- For an OPEN statement that connects a file for direct or keyed access, either omit the FORM specifier or provide one with a value of UNFORMATTED.
- Do not execute a formatted I/O statement for the unit.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken sfter Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1200

FOR1201S **The unformatted READ statement for unit** *unit-number*, **which was connected to** *file-name*, **failed. The file definition statement referred to an ASCII tape. VS FORTRAN Version 2 Error Number: AFB214I**

Explanation: An unformatted READ statement referred to a unit that was connected to a file whose DCB information indicated variable-length ASCII tape records. These ASCII tape records are indicated by a RECFM value of D.

Programmer response: If you have an ASCII tape to be read, then change your program to use formatted rather than unformatted READ statements.

If your input file is not an ASCII tape, then ensure that you provide DCB information that does not include a value of D for the RECFM parameter.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1201

FOR1210S *locator-text* **There was an invalid repeat specification in a record being read with list-directed formatting. VS FORTRAN Version 2 Error Number: AFB227I**

Explanation: An input field in a record that was read with a list-directed READ statement had the form *n***value*, where *n*, which is the repeat specification, should be an integer constant and *value* should be the value to be assigned to *n* successive variables or array elements in the input item list. However, there was an error in the format of *n***value*. Possible errors might include an incorrect integer value for the repeat specification, a second asterisk, or an invalid value following the asterisk.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for an internal file failed.
The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: In the record to be read, ensure that the repeat specification and the constant following the asterisk are coded in the correct format.

System action: The input item being processed and the remainder of the items in the input item list are undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 6. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
6	*record*	Input	CHARACTER**n*	The formatted input record that contained the invalid repeat specification. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	3
2	*subroutine-name*	Input	CHARACTER*8	The name of the DIV subroutine
3	*return-code*	Input	INTEGER*4	The return code from the Fortran DIV subroutine

Name	Action Taken after Resumption
RN	The current operation is ignored. The remainder of the deallocation list is processed and execution continues.

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1210

FOR1220S *locator-text* **In the namelist group** *group-name* **in a namelist input file, a variable name or array name exceeded 31 characters in length. The first 31 characters were** *object-name*. **VS FORTRAN Version 2 Error Number: AFB221I**

Programmer response: Ensure that the variable names in the namelist group in the namelist input file are all listed in the corresponding NAMELIST statement in the program and that the delimiters, such commas, equal signs, and quotes, are used as required.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 9. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name*	Input	CHARACTER*31	The namelist group name in the namelist input file.
6	*object-name-desc*	Input	Q_DATA_DESC	The q_data_descriptor for *object-name*. It contains the data type and the length of *object-name*.

No.	Name	Input/Output	Data Type and Length	Value
7	*object-name*	Input	CHARACTER**n*	The variable name or array name that was too long in *group-name* in the namelist input file. The length *n* is part of *object-name-desc* and has a maximum possible value of 255.
8	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
9	*record*	Input	CHARACTER**n*	The formatted input record that contained the name that was too long. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	The current operation is ignored. The remainder of the deallocation list is processed and execution continues.

Symbolic Feedback Code: FOR1220

FOR1221S *locator-text* **In the namelist group** *group-name* **in a namelist input file, the variable name or array name** *object-name* **was not in the namelist group in the NAMELIST statement. VS FORTRAN Version 2 Error Number: AFB222I**

Programmer response: Ensure that the variable names in the namelist group in the namelist input file are all listed in the corresponding NAMELIST statement in the program and that the delimiters, such commas, equal signs, and quotes, are used as required.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 8. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name*	Input	CHARACTER*31	The namelist group name in the namelist input file.
6	*object-name*	Input	CHARACTER*31	The variable name or array name that was not in *group-name* in the NAMELIST statement.
7	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
8	*record*	Input	CHARACTER**n*	The formatted input record that contained the name that was not in the namelist group. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken After Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1221

FOR1222S *locator-text* **In the namelist group** *group-name* **in a namelist input file, there was a syntax error involving the name** *object-name* **or its value. VS FORTRAN Version 2 Error Number: AFB223I**

Programmer response: Ensure that the variable names in the namelist group in the namelist input file are all listed in the corresponding NAMELIST statement in the program and that the delimiters, such commas, equal signs, and quotes, are used as required.

System action: The variable being processed and the remainder of the variables given in the namelist input file become undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 10. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *group-name*. It contains the data type and the length of *group-name*.
6	*group-name*	Input	CHARACTER**n*	The namelist group name in the namelist input file. The length *n* is part of *group-name-desc* and has a maximum possible value of 250.
7	*object-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *object-name*. It contains the data type and the length of *object-name*.
8	*object-name*	Input	CHARACTER**n*	The variable name or array name that had an incorrect value or syntax in *group-name* in the namelist input file. The length *n* is part of *object-name-desc* and has a maximum possible value of 250.
9	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
10	*record*	Input	CHARACTER**n*	The formatted input record that contained the name that had an incorrect value or syntax. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1222

FOR1223S *locator-text* **In the namelist group** *group-name* **in a namelist input file, a subscript for array** *object-name* **had the value** *subsc-val***, which was not within the bounds for that array. VS FORTRAN Version 2 Error Number: AFB224I**

Programmer response: Ensure that the subscript has a value that lies within the bounds of the array *object-name*. Either correct the subscript or change the declaration of the array in the program.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 8. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name*	Input	CHARACTER*31	The namelist group name in the namelist input file.
6	*object-name*	Input	CHARACTER*31	The name of the array that had an incorrect subscript value in *group-name* in the namelist input file.
7	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
8	*record*	Input	CHARACTER**n*	The formatted input record that contained the name of the array that had an incorrect subscript value. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1223

FOR1224S *locator-text* **In the namelist group** *group-name* **in a namelist input file,** *object-name* **had a subscript but was not an array. VS FORTRAN Version 2 Error Number: AFB224I**

Programmer response: Correct the inconsistency between the use of *object-name* with a subscript in the namelist input file and the declaration of *object-name* in the Fortran program as a scalar variable. Either remove the subscript in the namelist input file or correct the declaration in the program.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 8. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name*	Input	CHARACTER*31	The namelist group name in the namelist input file.
6	*object-name*	Input	CHARACTER*31	The name of the variable that had a subscript in *group-name* in the namelist input file.
7	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
8	*record*	Input	CHARACTER**n*	The formatted input record that contained the name of the variable that had a subscript. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1224

FOR1225W *locator-text* **In a namelist input file, a namelist group name or variable name exceeded** *max-length* **characters in length. The first** *max-length* **characters were** *name***. VS FORTRAN Version 2 Error Number: AFB221I**

Programmer response: Ensure that the namelist input file is coded in the correct format with the information beginning in column 2 or later in the records. Check for an ampersand preceding the namelist group name with no intervening spaces, and check for missing delimiters, such as commas, quotes, or apostrophes.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, execution continues, and the name *name* is ignored.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 8. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *group_name*. It contains the data type and the length of *group-name*.
6	*group-name*	Input	CHARACTER**n*	The namelist group name that was too long in *group-name* in the namelist input file. The length *n* is part of *group-name-desc* and has a maximum possible value of 255.
7	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
8	*record*	Input	CHARACTER**n*	The formatted input record that contained the name that was too long. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1225

FOR1226S *locator-text* **There was an error in the specification of a name in a name-value pair within the namelist group** *group-name* **in the namelist input file. (***Description of the error that was detected.***) VS FORTRAN Version 2 Error Number: AFB222I**

Explanation: The namelist group *group-name* in the namelist input file had an error involving one of the variable names given in what should be a name-value pair. The detailed description of the error is in the message text.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for an internal file failed.
The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Correct the incorrectly coded variable name in the namelist input file or provide any missing delimiters.

System action: The variable being processed and the remainder of the variables given in the namelist input file become undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 8. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name-desc*	Input	Q_DATA_DESC	The q_data_descriptor for *group-name*. It contains the data type and the length of *group-name*.
6	*group-name*	Input	CHARACTER**n*	The namelist group name that contained the variable that was coded in error. The length *n* is part of *group-name-desc* and has a maximum possible value of 255.
7	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
8	*record*	Input	CHARACTER**n*	The formatted input record that contained the name that was too long. The length *n* which includes only the data portion of the record, is part of *record_desc*.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1226

FOR1227S *locator-text* **Within the namelist group** *group-name* **in the namelist input file the variable** *var-name* **was not followed by an equal sign. VS FORTRAN Version 2 Error Number: AFB222I**

Explanation: The namelist group *group-name* in the namelist input file contained the variable name *var-name*. However, this name was not immediately followed by the equal sign (=), which should separate the name and a value.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for an internal file failed.
The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Code the name-value pair in the form of the variable name followed by an equal sign followed by a value or values.

System action: The variable being processed and the remainder of the variables given in the namelist input file become undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 10. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *group-name*. It contains the data type and the length of *group-name*.

No.	Name	Input/Output	Data Type and Length	Value
6	*group-name*	Input	CHARACTER**n*	The namelist group name that contained the variable that wasn't followed by an equal sign. The length *n* is part of *group-name-desc* and has a maximum possible value of 255.
7	*var-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *var-name*. It contains the data type and the length of *var-name*.
8	*var-name*	Input	CHARACTER**n*	The name of the variable that wasn't followed by an equal sign. The length *n*, which is part of *var-name-desc*, has a maximum possible value of 255 even if the variable name is actually longer.
9	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
10	*record*	Input	CHARACTER**n*	The formatted input record that contained the name that wasn't followed by an equal sign. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1227

FOR1228S *locator-text* **Within the namelist group** *group-name* **in the namelist input file there were too many values given for the variable** *var-name***. VS FORTRAN Version 2 Error Number: AFB222I**

Explanation: The namelist group *group-name* in the namelist input file contained the variable name *var-name* followed by an the equal sign (=), which separates the name and the values. However, following that equal sign there were more values than there were intrinsic data items comprising the variable. There were either:
- more values than there were elements in an array variable,
- more values than there were intrinsic data items within a variable of derived type, or
- more than one value for a scalar variable of an intrinsic data type.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for an internal file failed.
The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Code the name-value pair in the form of the variable name followed by an equal sign followed by no more values than comprise the variable. For example, a scalar variable of an intrinsic data type cannot have more than one value. If an assumed-shape or deferred-shape array is involved, ensure that the number of values doesn't exceed the number of elements represented by the current shape of the array. If a variable of derived type is involved, ensure that there aren't more values than there are intrinsic data items within the derived type, and ensure that the values are of the proper type to correspond with the intrinsic data items.

If you intended for the extraneous value to be interpreted as the next variable name rather than as a value, then code this variable name followed by an equal sign followed by this variable's value or values.

System action: The variable being processed and the remainder of the variables given in the namelist input file become undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 10. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *group-name*. It contains the data type and the length of *group-name*.
6	*group-name*	Input	CHARACTER**n*	The namelist group name that contained the variable that had too many values. The length *n* is part of *group-name-desc* and has a maximum possible value of 255.
7	*var-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *var-name*. It contains the data type and the length of *var-name*.
8	*var-name*	Input	CHARACTER**n*	The name of the variable that had too many values. The length *n*, which is part of *var-name-desc*, has a maximum possible value of 255 even if the variable name is actually longer.
9	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
10	*record*	Input	CHARACTER**n*	The formatted input record that contained the value that exceeded the number of allowable ones. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1228

FOR1229S *locator-text* **Within the namelist group** *group-name* **in the namelist input file the character** *not-name-char* **was found instead of the beginning of a variable name.**

Explanation: In the namelist group *group-name* in the namelist input file, there wasn't a variable name at a place where a variable name should have been. Instead, there was the character *not-name-char*, which cannot begin a variable name. This could have occurred at the beginning of the namelist group. Alternatively, following some other variable and its values there could have been some string of characters that wasn't recognized either as a value or as another variable name.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for an internal file failed.
The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Code the name-value pairs in the form of the variable name followed by an equal sign followed by no more values than comprise the variable. Be sure that each variable name is the name of a variable given for the namelist group in the NAMELIST statement in the Fortran program. Also be sure that the value or values are coded as literal constants rather than as named constants.

System action: The variable being processed and the remainder of the variables given in the namelist input file become undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 8. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *group-name*. It contains the data type and the length of *group-name*.
6	*group-name*	Input	CHARACTER**n*	The namelist group name that was too long in *group-name* in the namelist input file. The length *n* is part of *group-name-desc* and has a maximum possible value of 255.
7	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
8	*record*	Input	CHARACTER**n*	The formatted input record that contained the string of characters that was expected to be a variable name but wasn't in the correct format for a name. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1229

FOR1230S *locator-text* **In the namelist group** *group-name* **in a namelist input file, the value for the variable** *object-name* **was not in the form of a complex constant even though the variable was of complex type. The formatted input data was** *input-field***.**

Explanation: For a READ statement, *input-field* is a portion of the character string that is being interpreted as a complex constant for namelist input, and can be either the real part, the imaginary part, or both. *input-field* either contained embedded blanks in the real part or the imaginary part of the complex number, did not contain a comma as a separator between the real part and the imaginary part, or was not enclosed in parentheses.

locator-text gives more information about the location of the error, and can be one of the following:
  The READ statement for an internal file failed.
  The READ statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Ensure that *input-field* contains no embedded blanks in the real part or the imaginary part of the complex number, contains a comma as a separator, is enclosed by a left and a right parenthesis, and, if the the complex number does not fit into one record, that the end of the record occurs between the real part and the comma or between the comma and the imaginary part.

System action: The variable being processed and the remainder of the variables given in the namelist input file become undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 10. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *group_name*. It contains the data type and the length of *group-name*.
6	*group-name*	Input	CHARACTER**n*	The namelist group name in the namelist input file. The length *n* is part of *group-name-desc*.

No.	Name	Input/Output	Data Type and Length	Value
7	*object-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *object_name*. It contains the data type and the length of *group-name*.
8	*object-name*	Input	CHARACTER**n*	The name of the complex variable that has an incorrect value in *group-name* in the namelist input file. The length *n* is part of *object-name-desc*.
9	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
10	*record*	Input	CHARACTER**n*	The formatted input record that contained the name of the array that had an incorrect subscript value. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	Execution continues, and the remainder of the input item list is ignored.

Symbolic Feedback Code: FOR1230

FOR1231S *locator-text* **In the namelist group** *group-name* **in a namelist input file, the value for the variable** *object-name* **was not a delimited character constant even though the variable was of character type. The formatted input data was** *input-field***.**

Explanation: For a READ statement, *input-field* is a portion of the character string that is being interpreted as a character constant for namelist input. It was not delimited by apostrophes (’) or by quotes (″) as required for namelist input.

locator-text gives more information about the location of the error, and can be one of the following:
The READ statement for an internal file failed.
The READ statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Ensure that the character constant used as the value for the variable is delimited either by apostrophes (’) or by quotes (″).

System action: The variable being processed and the remainder of the variables given in the namelist input file become undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 10. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *group-name*. It contains the data type and the length of *group-name*.
6	*group-name*	Input	CHARACTER**n*	The namelist group name in the namelist input file. The length *n* is part of *group-name-desc*.
7	*object-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *object-name*. It contains the data type and the length of *object-name*.

No.	Name	Input/Output	Data Type and Length	Value
8	*object-name*	Input	CHARACTER**n*	The name of the character variable that has a value that was not properly delimited in *group-name* in the namelist input file. The length *n* is part of *object-name-desc*.
9	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
10	*record*	Input	CHARACTER**n*	The formatted input record that contained the value that was not properly delimited. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	Execution continues, and the remainder of the input item list is ignored.

Symbolic Feedback Code: FOR1231

FOR1232S *locator-text* **In the namelist group** *group-name* **in a namelist input file, the variable** *object-name* **was a delimited character constant for which there was no ending delimiter. The character constant contained or began with the characters** *input-field***.**

Explanation: For a READ statement, *input-field* is a portion of the character string that is being interpreted as a character constant for namelist input. It had a starting delimiter of either an apostrophe or a quote, but there was corresponding ending delimiter before the end of the file.

locator-text gives more information about the location of the error, and can be one of the following:
 The READ statement for an internal file failed.
 The READ statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Ensure that the character constant used as the value for the variable is delimited either by apostrophes (') or by quotes ("). Both the starting and ending delimiter must both be apostrophes or both be quotes.

Check for a doubled occurrence of the delimiter that started the character constant. Such a doubled delimiter is not interpreted as the ending delimiter but rather as one occurrence of that delimiter as a character within the character constant. In this case, a single occurrence of the delimiter must follow to indicate the end of the character constant.

System action: The variable being processed becomes undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 10. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *group-name*. It contains the data type and the length of *group-name*.
6	*group-name*	Input	CHARACTER**n*	The namelist group name in the namelist input file. The length *n* is part of *group-name-desc*.
7	*object-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *object-name*. It contains the data type and the length of *object-name*.

No.	Name	Input/Output	Data Type and Length	Value
8	*object-name*	Input	CHARACTER**n*	The name of the character variable that has a value that did not have an ending delimiter in *group-name* in the namelist input file. The length *n* is part of *object-name-desc*.
9	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
10	*record*	Input	CHARACTER**n*	The formatted input record that contained the value that did not have an ending delimiter. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	Execution continues, and the remainder of the input item list is ignored.

Symbolic Feedback Code: FOR1232

FOR1233S *locator-text* **In the namelist group** *group-name* **in a namelist input file, the variable** *object-name* **was a delimited character constant whose length,** *length***, exceeded** *max-char-length***, the maximum length allowed for a character constant. The character constant began with the characters** *input-field***.**

Explanation: For a READ statement, *input-field* is a portion of the character string that is being interpreted as a character constant for namelist input. This length of this constant, not counting the starting and ending delimiters, was *length*, but this was longer than *max-char-length*, the product-imposed maximum length of a character constant that can be interpreted as namelist input.

locator-text gives more information about the location of the error, and can be one of the following:
The READ statement for an internal file failed.
The READ statement for unit *unit-number*, which was connected to *file-name*, failed.

Programmer response: Ensure that the character constant used as the value for the variable is delimited either by apostrophes (') or by quotes ("). Both the starting and ending delimiter must both be apostrophes or both be quotes.

Check for a doubled occurrence of the delimiter that started the character constant. Such a doubled delimiter is not interpreted as the ending delimiter but rather as one occurrence of that delimiter as a character within the character constant. In this case, a single occurrence of the delimiter must follow to indicate the end of the character constant.

System action: The variable being processed and the remainder of the variables given in the namelist input file become undefined. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 10. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*group-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *group-name*. It contains the data type and the length of *group-name*.
6	*group-name*	Input	CHARACTER**n*	The namelist group name in the namelist input file. The length *n* is part of *group-name-desc*.

No.	Name	Input/Output	Data Type and Length	Value
7	*object-name-desc*	Input	Q_DATA_DESC	The q_data descriptor for *object-name*. It contains the data type and the length of *object-name*.
8	*object-name*	Input	CHARACTER**n*	The name of the character variable that has a value that was too long in *group-name* in the namelist input file. The length *n* is part of *object-name-desc*.
9	*record-desc*	Input	Q_DATA_DESC	The q_data descriptor for *record*. It contains the data type and the length of *record*.
10	*record*	Input	CHARACTER**n*	The formatted input record that contained the value that was too long. The length *n*, which includes only the data portion of the record, is part of *record-desc*.

Name	Action Taken after Resumption
RN	Execution continues, and the remainder of the input item list is ignored.

Symbolic Feedback Code: FOR1233

FOR1250S *locator-text* **The file definitions for the stripes of** *file-name* **did not define consistent characteristics for all the stripes. VS FORTRAN Version 2 Error Number: AFB092I**

Explanation: A striped file, that is, one with ddnames of the form FT*nn*P001, FT*nn*P002, and so on, had different characteristics given in its file definitions (DD statements or ALLOCATE commands) or data set labels for the different stripes. One or more of the following parameters had different values among the stripes:

- RECFM
- LRECL
- BLKSIZE
- BUFOFF
- IN or OUT (fourth subparameter of the LABEL parameter)
- DISP

locator-text gives more information about the location of the error, and can be one of the following:

The READ statement for an internal file failed.
The READ statement for unit *unit-number*, which was connected to file-name, failed.

Programmer response: Ensure that the file definitions for all of the stripes have identical values for the parameters listed in “Explanation.” If an existing data set was used, be sure that the whatever was in the existing data set label doesn’t cause this conflict; override such a conflicting value if necessary.

System action: If the error occurred during the execution of an OPEN statement, the unit is not connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1250

FOR1251S *locator-text* **The file definition statements for the stripes of** *file-name* **had inconsistent ddnames or data set names. VS FORTRAN Version 2 Error Number: AFB093I**

Explanation: A striped file, that is, one with ddnames of the form FT*nn*P001, FT*nn*P002, and so on, had inconsistent ddnames and data set names in one or more of its file definitions (DD statements or ALLOCATE commands). The inconsistency could be one of the following:

- A file definition for ddname FT*nn*P*mmm* did not refer to a data set name that ends in the form *xxx*P*yyy*, where *xxx* is the number of stripes and *yyy* is a particular stripe number.
- A file definition for ddname FT*nn*P*mmm* did refer to a data set name that ends in the form *xxx*P*yyy*, and either:
  - The stripe numbers *mmm* and *yyy* did not match,
  - The portion of the data set name other than the stripe number, that is, other than the *yyy*, differed from that for other stripes, or
  - The stripe number *yyy* and the maximum stripe number *xxx* did not have the same number of digits.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for *unit-number* failed.
The INQUIRE statement failed.

Programmer response: Assuming that there are *xxx* stripes, ensure that:

- There are *xxx* file definitions and that the file definitions have the ddnames FT*nn*P001, FT*nn*P002, ... FT*nn*P*sss*, where *nn* is the two-digit unit number and *sss* is the three-digit representation of the number of stripes (*xxx*),
- In the file definition with the ddname FT*nn*P*mmm*, the final characters of the end of the data set name are *xxx*P*yyy*, where *yyy* is the stripe number, that is, *yyy* has the same numeric value as *mmm*,
- The number of digits in *xxx* is the same as the number of digits in each *yyy*, and
- All the data set names are identical except for the trailing *mmm*.

Here is an example of DD statements for a striped file with six stripes:

```
//FT10P001 DD  DSN=MYNAME.MYFILE.ABC6P1,DISP=OLD
//FT10P002 DD  DSN=MYNAME.MYFILE.ABC6P2,DISP=OLD
//FT10P003 DD  DSN=MYNAME.MYFILE.ABC6P3,DISP=OLD
//FT10P004 DD  DSN=MYNAME.MYFILE.ABC6P4,DISP=OLD
//FT10P005 DD  DSN=MYNAME.MYFILE.ABC6P5,DISP=OLD
//FT10P006 DD  DSN=MYNAME.MYFILE.ABC6P6,DISP=OLD
```

System action: If the error occurred during the execution of an OPEN statement, the unit is not connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4. In addition, there are these qualifying data:

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1251

FOR1252S *locator-text* **A file definition statement for one of the stripes of** *file-name* **referred to a file type or device type that cannot be used for a striped file. VS FORTRAN Version 2 Error Number: AFB094I**

Explanation: A file definition (DD statement or ALLOCATE command) for a striped file referred to a file on a device that was neither a tape nor a non-VSAM disk file other than a PDS member.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for *unit-number* failed.
The INQUIRE statement failed.

Programmer response: Ensure that for each of the stripes of a striped file the file definition refers either to a tape or to a non-VSAM disk file other than a PDS member.

System action: If the error occurred during the execution of an OPEN statement, the unit is not connected to a file.

If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1252

FOR1270S The WAIT statement for unit *unit-number*, which was connected to *file-name*, failed. There was no corresponding READ or WRITE statement. VS FORTRAN Version 2 Error Number: AFB287I

Programmer response: Ensure that the program executes a WAIT statement only after a corresponding asynchronous READ or WRITE statement.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of WAIT, and *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The WAIT statement is ignored, and execution continues.

Symbolic Feedback Code: FOR1270

FOR1271S The asynchronous *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. In the file definition statement, the BLKSIZE parameter either was omitted or had a value of 0. VS FORTRAN Version 2 Error Number: AFB239I

Programmer response: Ensure that for a new file the block size (BLKSIZE parameter in the DD statement or ALLOCATE command) has a nonzero value.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1271

FOR1272S The asynchronous *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The last array element in the input item list had a lower subscript value than the first. VS FORTRAN Version 2 Error Number: AFB228I

Programmer response: Ensure that the starting and ending elements in the input/output item list are specified with the lower-valued subscript first. If the subscripts involve an variable, ensure that the variables are set to their intended values.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

Symbolic Feedback Code: FOR1272

FOR1273S The asynchronous *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The previous I/O statement for this unit was neither a REWIND nor another asynchronous I/O statement. VS FORTRAN Version 2 Error Number: AFB286I

Programmer response: If you want to switch back and forth between asynchronous I/O statements and the sequential I/O statement defined by the Fortran language standard, then you must execute a REWIND statement each time you switch between the two. (Of course you can also execute a CLOSE statement followed by an OPEN statement.) While positioned within a file, you cannot switch between the two forms of I/O statements. If executing the REWIND statement doesn't provide the file positioning that your program requires, then change the program so that either asynchronous I/O statements or standard sequential I/O statements are used exclusively.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1273

FOR1274S The *statement* statement, which was not of the asynchronous form, for unit *unit-number*, which was connected to *file-name*, failed. The previous asynchronous I/O statement was not followed by a REWIND statement. VS FORTRAN Version 2 Error Number: AFB286I

Programmer response: If you want to switch back and forth between asynchronous I/O statements and the sequential I/O statements defined by the Fortran language standard, then execute a REWIND statement each time you switch between the two. (Alternatively, you could execute a CLOSE statement followed by an OPEN statement.) While positioned within a file, you cannot switch between the two forms of I/O statements. If executing the REWIND statement doesn't provide the file positioning that your program requires, then change the program so that either asynchronous I/O statements or standard sequential I/O statements are used exclusively.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1274

FOR1275S There was no corresponding WAIT statement for an asynchronous *statement* statement that was executed for unit *unit-number*, which was connected to *file-name*. VS FORTRAN Version 2 Error Number: AFB288I

Programmer response: Ensure that the Fortran program executes a WAIT statement after each asynchronous READ or WRITE statement.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	Execution continues with an implied WAIT.

Symbolic Feedback Code: FOR1275

FOR1276S **The asynchronous *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The file definition statement referred to a file type or device type that cannot be used for asynchronous I/O. VS FORTRAN Version 2 Error Number: AFB090I (format 2), AFB194I (format 1)**

Explanation: An asynchronous *statement* statement was executed for unit *unit*, and one of the following was true:

- The file definition (DD statement or ALLOCATE statement) for the ddname FT*nn*F001 referred to a file that was neither a tape nor an non-VSAM disk file other than a PDS member.
- There was a file definition with the ddname FT*nn***P**001.

Programmer response: If you want to use asynchronous I/O, then provide a file definition that refers to either a tape nor an non-VSAM disk file other than a PDS member. Also, do not provide a file definition with the ddname FT*nn*P001 because asynchronous I/O cannot be performed on striped files.

If you didn't intend to use an asynchronous I/O statement, which is identified by the ID specifier, then correct your program so that you don't use one for a unit that's connected to one of the prohibited file or device types.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1276

FOR1277S **The asynchronous *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The unit was being used as one of the Fortran standard I/O units. VS FORTRAN Version 2 Error Number: AFB192I**

Explanation: The asynchronous *statement* referred to a unit that was either the standard input unit, the error message unit, the print unit (which could be the same as the error message unit), or the punch unit. These unit numbers are specified by the RDRUNIT, ERRUNIT, PRTUNIT, and PUNUNIT run-time options, respectively.

Programmer response: If you want to use asynchronous I/O, do one of the following:

- Change the unit number in your asynchronous I/O statements to refer to some unit other than one of the prohibited units listed under "Explanation."
- Change the value of one or more of the RDRUNIT, ERRUNIT, PRTUNIT, or PUNUNIT run-time options so that one of them refers to the unit that you want to use for asynchronous I/O, ensuring, of course, that you don't create a conflict with some other unit that's used by your program.

If you didn't intend to use asynchronous I/O, then change the form of the I/O statement by removing the ID specifier and making whatever other changes are needed. Note that unformatted I/O statements are similar in function to asynchronous I/O statements.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1277

FOR1278S **The asynchronous *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The program was executed on other than an MVS system. VS FORTRAN Version 2 Error Number: AFB161I**

Programmer response: If you didn't intend to use asynchronous I/O, then change the form of the I/O statement by removing the ID specifier and making whatever other changes are needed. Note that unformatted I/O statements are similar in function to asynchronous I/O statements.

If the performance requirements of your program are such that asynchronous I/O is needed, then you'll have to run the program on MVS.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1278

FOR1279S **The asynchronous** *statement* **statement for unit** *unit-number***, which was connected to** *file-name***, failed. The I/O subtask terminated abnormally with the system completion code** *completion-code***. VS FORTRAN Version 2 Error Number: AFB205I**

Explanation: Much of the processing for an asynchronous I/O statement is done in an MVS subtask so that its processing can overlap that of your program. The asynchronous I/O subtask terminated abnormally with a system completion (abend) code of *completion-code*.

Programmer response: Ensure that the file definition (DD statement or ALLOCATE command) for the file you are using is coded correctly.

Ensure that your program uses the correct sequence of asynchronous READ, WRITE, and WAIT statements with no intervening I/O statements of the standard sequential form unless a REWIND statement is used to separate the uses of the two forms.

For the meaning of *completion-code*, and for possible corrective actions, refer to *z/OS DFSMS Macro Instructions for Data Sets*.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is not complete, and execution continues.

FOR1280S **The asynchronous** *statement* **statement for unit** *unit-number***, which was connected to** *file-name***, failed. The record format was other than variable spanned. VS FORTRAN Version 2 Error Number: AFB214I (format 2)**

Programmer response: Ensure that the file definition (DD statement or ALLOCATE command) for the file indicates a RECFM value of VS.

If you're reading an existing file that has some a record format of other than variable spanned, then you can't use asynchronous I/O statements to read it. In this case, either change your program to use the standard sequential I/O statements, or recreate the file so that it has variable spanned records. If you choose to recreate the file using a Fortran program, you can produce variable spanned records with either asynchronous I/O statements or with standard unformatted I/O statements. Ensure that when you recreate the file, the file definition has a RECFM value of VS.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The READ statement is ignored, and execution continues.
RF	For an input operation, the READ statement is not completed and execution continues. For an output operation, VS is assumed, and execution continues.

Symbolic Feedback Code: FOR1280

FOR1281S **The asynchronous *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The program resides in authorized library. VS FORTRAN Version 2 Error Number: AFB952I (format 8)**

Explanation: Your program was link edited with an AC option that provided an authorization code of other than 0, and your program was in an authorized library. In addition, your program used an asynchronous I/O statement, but this is inconsistent with executing in an authorized state because of the internal implementation of asynchronous I/O.

Language Environment does not support execution of Fortran programs running in an authorized state. Running such programs, while not diagnosed in all cases, causes a system integrity exposure.

Programmer response: Link edit your program without the AC option so that it does not run in an authorized state.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The READ statement is ignored, and execution continues.

Symbolic Feedback Code: FOR1281

FOR1330S **The *statement* statement for unit *unit-number* failed. The file definition statement for *file-name* referred to a VSAM file, but the file had not been connected to the unit with an OPEN statement. VS FORTRAN Version 2 Error Number: AFB168I**

Programmer response: If you intend to process a VSAM file, ensure that your program executes an OPEN statement before any other I/O statements.

If you don't intend to process a VSAM file, change the file definition (DD statement or ALLOCATE command) to refer to some other file.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The READ statement is ignored, and execution continues.

Symbolic Feedback Code: FOR1330

FOR1331S **The WRITE statement for sequential access for unit *unit-number*, which was connected to *file-name*, failed. The file was a VSAM RRDS that already contained records. VS FORTRAN Version 2 Error Number: AFB162I**

Programmer response: Ensure that you don't use the sequential access form of the WRITE statement for a VSAM relative record data set (RRDS) if the data set already contains records.

If you want to replace individual records, use a value of DIRECT for the ACCESS specifier on the OPEN statement, and use the direct access form of the WRITE statement, that is, with a REC specifier whose value indicates the specific record to be written.

If you want to extend the file at the end or rewrite the file sequentially from somewhere other than the end, the file can't be a VSAM RRDS, so change the file definition (DD statement or ALLOCATE command) to refer to a non-VSAM file.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1331

FOR1340S **The INQUIRE statement failed. The FILE specifier had a value of blanks, but the UNIT specifier was not given. VS FORTRAN Version 2 Error Number: AFB106I**

Programmer response: Correct your program to use of the four acceptable forms of the INQUIRE statement:

INQUIRE by file
No UNIT specifier; FILE specifier with a value that is either a data set name preceded by a slash (**/**) or a ddname other than one of the default ddnames such as FT*nn*F001

INQUIRE by unit
UNIT specifier; no FILE specifier

INQUIRE by unnamed file, format 1
UNIT specifier; FILE specifier with a value of blanks

INQUIRE by unnamed file, format 2
No UNIT specifier; FILE specifier with a value that is one of the defaults ddnames such as FT*nn*F001

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	4
2	*statement*	Input	CHARACTER*12	INQUIRE
3	*unit*	Input	INTEGER*4	Undefined
4	*file*	Input	CHARACTER*62	Undefined

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1340

FOR1341S **The INQUIRE statement failed. The FILE specifier had a value of *file-name*, which, because it was other than all blanks, conflicted with the presence of the UNIT specifier. VS FORTRAN Version 2 Error Number: AFB109I**

Programmer response: Correct your program to use of the four acceptable forms of the INQUIRE statement:

INQUIRE by file
No UNIT specifier; FILE specifier with a value that is either a data set name preceded by a slash (**/**) or a ddname other than one of the default ddnames such as FT*nn*F001

INQUIRE by unit
UNIT specifier; no FILE specifier

INQUIRE by unnamed file, format 1
UNIT specifier; FILE specifier with a value of blanks

INQUIRE by unnamed file, format 2
No UNIT specifier; FILE specifier with a value that is one of the defaults ddnames such as FT*nn*F001

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of INQUIRE, and *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1341

FOR1342S The INQUIRE statement failed. The FILE specifier had a value of *file-name*, which was not in the correct format for a file name. VS FORTRAN Version 2 Error Number: AFB180I (format 1)

Programmer response: Correct your program to use of the four acceptable forms of the INQUIRE statement:

INQUIRE by file
No UNIT specifier; FILE specifier with a value that is either a data set name preceded by a slash (**/**) or a ddname other than one of the default ddnames such as FT*nn*F001

INQUIRE by unit
UNIT specifier; no FILE specifier

INQUIRE by unnamed file, format 1
UNIT specifier; FILE specifier with a value of blanks

INQUIRE by unnamed file, format 2
No UNIT specifier; FILE specifier with a value that is one of the defaults ddnames such as FT*nn*F001

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	4
2	*statement*	Input	CHARACTER*12	INQUIRE
3	*unit*	Input	INTEGER*4	Undefined
4	*file*	Input	CHARACTER*62	The name of the file to which the INQUIRE statement was directed.

Name	Action Taken after Resumption
RN	The I/O operation is ignored, and execution continues.

Symbolic Feedback Code: FOR1342

FOR1360E On the CLOSE statement for unit *unit-number*, which was connected to *file-name*, the STATUS specifier had a value of KEEP, but the STATUS specifier on the OPEN statement had a value of SCRATCH. VS FORTRAN Version 2 Error Number: AFB171I

Programmer response: Either change the OPEN statement so that its STATUS specifier has a value of other than SCRATCH, or change the CLOSE statement so that its STATUS specifier is either omitted or has a value of DELETE. In the latter case, the scratch file will be deleted.

System action: The file is closed as though STATUS='DELETE' had been specified. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of CLOSE, and *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	Execution continues.

Symbolic Feedback Code: FOR1360

FOR1361E On the CLOSE statement for unit *unit-number***, the STATUS specifier had a value of** *status***, which was other than KEEP or DELETE. VS FORTRAN Version 2 Error Number: AFB186I**

Programmer response: Correct the value of the STATUS specifier on the CLOSE statement so that it has a value of either KEEP or DELETE. You can also omit the STATUS specifier in which case the default value is:

- DELETE if the OPEN statement had a STATUS specifier with a value of SCRATCH, or
- KEEP if the OPEN statement either had no STATUS specifier or a STATUS specifier with a value of other than SCRATCH.

System action: The file is closed as though STATUS='KEEP' had been specified. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: The basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of CLOSE, and *parm_count* has a value of 4.

Name	Action Taken after Resumption
RN	Execution continues.

Symbolic Feedback Code: FOR1361

FOR1380S The OPEN statement could not connect unit *unit-number* **to** *file-name***. The STATUS specifier had a value of** *status***, which was other than NEW, OLD, REPLACE, SCRATCH, or UNKNOWN. VS Fortran Version 2 Error Number: AFB251I**

Programmer response: Based on whether the file you're connecting exists or not, change the value of the STATUS specifier on the OPEN statement to NEW, OLD, REPLACE, SCRATCH, or UNKNOWN. If you code the value as a character constant, enclose the value in quotes or apostrophes.

For the error message unit, either omit the STATUS specifier, or provide a value of UNKNOWN.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1380

FOR1381S The OPEN statement for unit *unit-number* **failed. The FILE specifier had a value of** *ddname***, which is a ddname reserved for unnamed files. VS Fortran Version 2 Error Number: AFB107I**

Programmer response: If you want to connect the unit to an unnamed file, that is, to a file with a ddname of FT*nn*F*mmm*, FT*nn*K*mm*, FTERR*sss*, or FTPRT*sss*, then omit the FILE specifier from the OPEN statement. A file definition (DD statement or ALLOCATE command) for that ddname would still be required, however.

If you want to refer to a named file, then for the FILE specifier provide a value than isn't one of the ddnames that are used for unnamed files.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the

ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1381

FOR1382S The OPEN statement could not connect unit *unit-number* **to** *file-name***. The ACTION specifier had a value of** *action***, which was other than READ, WRITE, or READWRITE. VS Fortran Version 2 Error Number: AFB136I**

Programmer response: Depending on whether you intend to use only input statements, only output statements, or both, change the value of the ACTION specifier on the OPEN statement to READ, WRITE, or READWRITE. If you code the value as a character constant, enclose the value in quotes or apostrophes.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1382

FOR1383S The OPEN statement for unit *unit-number* **failed. The FILE specifier had a value of** *file-name***, which was not in the correct format for a file name.**

Explanation: VS Fortran Version 2 Error Number: AFB180I (format 2)

Programmer response: Correct the value of the FILE specifier. If you're providing a ddname, ensure that it consists of no more than eight characters, all of which must be alphanumeric, and that its first character is alphabetic.

If you're providing a data set name, code the value of the FILE specifier as a slash (**/**) followed by the data set name. The data set name, which can be followed by a member name surrounded by parentheses, must be in the format required by the DSNAME parameter of a DD statement as described in *z/OS DFSMS Macro Instructions for Data Sets*.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1383

FOR1384S The OPEN statement could not connect unit *unit-number* to *file-name*. The ACCESS specifier had a value of *access*, which was other than SEQUENTIAL, DIRECT, or KEYED.

Explanation: VS Fortran Version 2 Error Number: AFB182I

Programmer response: Depending on the type of I/O statements that you want to use for the file that you're connecting, change the value of the ACCESS specifier on the OPEN statement to SEQUENTIAL, DIRECT, or KEYED. If you code the value as a character constant, enclose the value in quotes or apostrophes.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1384

FOR1385S The OPEN statement could not connect unit *unit-number* to *file-name*. The BLANK specifier had a value of *blank*, which was other than ZERO or NULL.

Explanation: VS Fortran Version 2 Error Number: AFB183I

Programmer response: Depending on whether you want blanks in input fields that are read with formatted READ statements to be interpreted as zeros or nulls, change the value of the BLANK specifier on the OPEN statement to ZERO or NULL, respectively. If you code the value as a character constant, enclose the value in quotes or apostrophes.

If you omit the FILE specifier from the OPEN statement, a value of NULL is assumed.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1385

FOR1386S The OPEN statement could not connect unit *unit-number* to *file-name*. The FORM specifier had a value of *format*, which was other than FORMATTED or UNFORMATTED.

Explanation: VS Fortran Version 2 Error Number: AFB184I

Programmer response: Depending on whether you intend to use formatted or unformatted input/output statements for the file, change the value of the FORM specifier on the OPEN statement to FORMATTED or UNFORMATTED, respectively. If you code the value as a character constant, enclose the value in quotes or apostrophes.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1386

FOR1387S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The CHAR specifier had a value of** *char***, which was other than DBCS or NODBCS. Fortran Version 2 Error Number: AFB104I**

Programmer response: Depending on whether your input file contains double-byte characters that are to be read with a formatted READ statement, change the value of the CHAR specifier on the OPEN statement to DBCS or NODBCS. If you code the value as a character constant, enclose the value in quotes or apostrophes.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1387

FOR1389S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The RECL specifier had a value of** *recl***, which was not within the range 1 to 32760, inclusive. Fortran Version 2 Error Number: AFB233I**

Programmer response: Ensure that the value of the RECL specifier on the OPEN statement is an integer that is neither less than 1 nor greater than 32760. In addition, ensure that this value is the same as the value that's associated with the file through one or more of the following, as applicable:

- The label of an existing data set
- The LRECL parameter of the DD statement or ALLOCATE command
- The LRECL value given in an invocation of the FILEINF callable service
- The record length given in the RECORDSIZE parameter of the Access Method Services DEFINE command that was used to define the VSAM cluster.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 5.

No.	Name	Input/Output	Data Type and Length	Value
5	*record-length*	Input/Output	INTEGER*4	The value of the RECL specifier on the OPEN statement.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues. The remainder of the deallocation list is processed and execution continues.

Name	Action Taken after Resumption
RI	The value placed in *record_length* is used as the new value for the RECL specifier, and execution continues.

Symbolic Feedback Code: FOR1389

FOR1390S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The SMSVSAM server was not available.**

Explanation: VSAM record level sharing was requested for the file either because of the RLS parameter on the DD statement or ALLOCATE command or because of the RLS argument in the call to the FILEINF callable service. Record level sharing couldn't be provided because the SMSVSAM server was not available.

Programmer response: If VSAM record level sharing isn't required, then don't request it in the DD statement, in the ALLOCATE command, or in the call to the FILEINF callable service. Otherwise, correct the situation that caused the SMSVSAM server not to be available.

If you cannot resolve the problem, seek assistance from your Language Environment support personnel.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1390

FOR1393S *locator-text* **The VSAM** *macro-name* **macro instruction executed for** *file-name* **had a return code of** *return-code* **and an error code of X'** *hex-code* **' (***decimal-code***). The SMSVSAM server was not available.**

Explanation: In support of the Fortran I/O statement indicated by the message text, Language Environment executed a VSAM *macro-name* macro instruction. DFSMS/MVS detected the error indicated by the return code *return-code* and the error code with a hexadecimal value of *hex-code* (decimal value of *decimal-code*). This error code indicates that VSAM record level sharing couldn't be provided because the SMSVSAM server was not available. (VSAM record level sharing was requested for the file either because of the RLS parameter on the DD statement or ALLOCATE command or because of the RLS argument in the call to the FILEINF callable service.)

Programmer response: For the meaning of return code *return-code* and error code *hex-code* (or *decimal-code*), refer to *z/OS DFSMS Macro Instructions for Data Sets*.

If VSAM record level sharing isn't required, then don't request it in the DD statement, in the ALLOCATE command, or in the call to the FILEINF callable service. Otherwise, correct the situation that caused the SMSVSAM server not to be available.

If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1393

FOR1394S **The OPEN statement could not connect unit** *unit-number***, the error message unit, to** *file-name***. The FILE specifier had a value that started with a slash (/).**

Explanation: On the OPEN statement the UNIT specifier had a value that was the error message unit number, and the FILE specifier had a value with a leading slash (**/**). The leading slash indicated dynamic file allocation, that is, that a data set name rather than a ddname was given.

Programmer response: Dynamic allocation cannot be done from a Fortran program for a file that's being connected to the error message unit. Therefore, take one of these actions:

- Provide a file definition (DD statement or ALLOCATE command) for the file, and use its ddname without a slash as the value of the FILE specifier on the OPEN statement.
- Connect that file to a unit other than the error message unit by changing the unit identifier.
- Remove the slash from the value of the FILE specifier if an intended ddname followed the slash.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1394

FOR1395S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The ACCESS specifier had a value of KEYED, and the FILE specifier had a value that started with a slash (/). Fortran Version 2 Error Number: AFB102I**

Explanation: The OPEN statement had a value for the ACCESS specifier that indicated keyed access and a value for the FILE specifier with a leading slash (**/**). The leading slash indicated dynamic file allocation, that is, that a data set name rather than a ddname was given.

Programmer response: Dynamic allocation cannot be done from a Fortran program for a file that's being connected for keyed access. If you intend to use keyed access because the file is a VSAM key-sequenced data set, take on of these actions:

- Provide a file definition (DD statement or ALLOCATE command) for the file, and use its ddname without a slash as the value of the FILE specifier on the OPEN statement.
- Remove the slash from the value of the FILE specifier if an intended ddname followed the slash.

If the file isn't a VSAM key-sequenced data set, then change the value of the ACCESS specifier to either SEQUENTIAL or DIRECT.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1395

FOR1396S The OPEN statement could not connect unit *unit-number* to *file-name*. The STATUS specifier had a value of NEW, but the file already existed. Fortran Version 2 Error Number: AFB108I (format 1)

Explanation: The OCSTATUS run-time option was in effect, and the STATUS specifier on the OPEN statement had a value of NEW, but the file already existed according to the Fortran definitions of file existence. These definitions are explained in *VS FORTRAN Version 2 Programming Guide for CMS and MVS* in the chapter "What Determines File Existence" and generally reflect the operating system view of file existence; however, there are a few differences. For example, using a DD statement or ALLOCATE command to allocate space for a file on a disk volume does not mean that the file exists from the Fortran point of view. Such a file doesn't exist until an OPEN or WRITE statement for it has been used in a Fortran program or until it has had records written into it by some non-Fortran program or utility.

Programmer response: The changes that you must make depend on how you intend to processed the file in your program.

If you want to read an existing file, then change the value of the STATUS specifier to OLD.

If you want to read or update an existing file being connected for direct or keyed access, then change the value of the STATUS specifier to OLD.

If you want to extend an existing file being connected for sequential access, then change the value of the STATUS specifier to OLD, and take one of these actions:

- Provide the DISP=MOD parameter on the DD statement, the MOD parameter in the ALLOCATE command, or a value of MOD for the DISP argument in the invocation of the FILEINF callable service.
- Provide a value of APPEND for the POSITION specifier.

If you want to replace all of the records in an existing file being connected for sequential access, then make one of these changes:

- Provide a value of OLD for the STATUS specifier and do not provide either the DISP=MOD parameter on the DD statement, the MOD parameter on the ALLOCATE command, nor a value of MOD for the DISP argument in the invocation of the FILEINF callable service, or
- Provide a value of REPLACE for the STATUS specifier. In this case, if the disk file is dynamically allocated, the original space, if any, is released, and new space is allocated.

If you want to replace all of the records in an existing file, being connected for direct or keyed access, then provide a value of REPLACE for the STATUS specifier. In this case, if the disk file is dynamically allocated, the original space, if any, is released, and new space is allocated.

If you want to create and write to a new file in your program, then the STATUS specifier value of NEW is correct. Ensure that all of the following are true:

- The file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement refers to the file that you intended.
- No other unit is connected to the same file and has updated it before this OPEN statement was executed.
- The file does not exist according to the Fortran rules of file existence. (For example, under certain circumstances, a file that is present on a disk volume but contains no data can still exist according to this definition.)

If the file does exist according to the Fortran definition of file existence, but you still want the STATUS specifier value to be NEW, then use the NOOCSTATUS run-time option. However, consider the following:

- Because file existence is not checked, the STATUS specifier value of NEW takes precedence, meaning that any records already in the file could be overwritten.
- This run-time option causes file existence checking to be bypassed for all OPEN statements.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1396

FOR1397S The OPEN statement could not connect unit *unit-number* to *file-name*. The STATUS specifier had a value of NEW, but the file definition referred to a file or device that was restricted to input only. Fortran Version 2 Error Number: AFB108I (format 2)

Explanation: The STATUS specifier had a value of NEW, which implied that a file was to be created. However, the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement referred to a file or device that doesn't allow output operations. Examples of such files include:

- An in-stream data set (DD *)
- A data set whose DD statement specifies LABEL=(,,,IN)
- A file for which the system's access control facility (such as RACF) prevents you from updating the data set.

Programmer response: If you want to read from a file or a device that doesn't permit output, either omit the STATUS specifier or provide a value of OLD or UNKNOWN for the STATUS specifier. Ensure that the file really exists and that you can read from it.

If you want to create a new file and write records on it, then the STATUS specifier value of NEW is correct. In this case, change either of the following, as applicable, to refer to a file or device on which you can write records:

- The file definition (DD statement or ALLOCATE command)
- For a dynamically allocated data, the data set name that follows the slash (**/**) in the FILE specifier.

Don't refer to a data set such as an in-stream data set (DD *), a data set for which you don't have RACF authority to update, or a data set whose DD statement has a LABEL=(,,,IN) parameter.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1397

FOR1398S The OPEN statement could not connect unit *unit-number* to *file-name*. The STATUS specifier had a value of OLD, but the file did not exist. Fortran Version 2 Error Number: AFB108I (format 3)

Explanation: The OCSTATUS run-time option was in effect, and the STATUS specifier on the OPEN statement had a value of OLD, but the file didn't exist according to the Fortran definitions of file existence. These definitions are explained in *VS FORTRAN Version 2 Programming Guide for CMS and MVS* in the chapter "What Determines File Existence" and generally reflect the operating system view of file existence; however, there are a few differences. For example, using a DD statement or ALLOCATE command to allocate space for a file on a disk volume does not mean that the file exists from the Fortran point of view. Such a file doesn't exist until an OPEN or WRITE statement for it has been used in a Fortran program or until it has had records written into it by a non-Fortran program or utility.

Programmer response: If you want to create a new file and write records on it, then change the value of the STATUS specifier to NEW, REPLACE, or UNKNOWN.

If you want to read from or write to an existing file, then the STATUS specifier value of OLD is correct. Ensure that all of the following are true:

- The file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement refers to the file that you intended.

- No other unit has been connected to the same file and has caused the file to be deleted before this OPEN statement was executed.
- The file exists according to the Fortran rules of file existence. (For example, under certain circumstances, a file that is present on a disk volume but contains no data doesn't exist according to this definition.)

If the file doesn't exist according to the Fortran definition of file existence, but you still want the STATUS specifier value to be OLD, then use the NOOCSTATUS run-time option. However, consider the following:

- Because file existence is not checked, the STATUS specifier value of OLD will take precedence, meaning that there must be records in the file if you want to read them.
- This run-time option causes file existence checking to be bypassed for all OPEN statements.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1398

FOR1399S The OPEN statement could not connect unit *unit-number* to *file-name*. The ACTION specifier had a value of READ, but the file did not exist. Fortran Version 2 Error Number: AFB108I (format 4)

Explanation: The ACTION specifier on the OPEN statement had a value of READ, but the file didn't exist according to the Fortran definitions of file existence. These definitions are explained in *VS FORTRAN Version 2 Programming Guide for CMS and MVS* in the chapter "What Determines File Existence" and generally reflect the operating system view of file existence; however, there are a few differences. For example, using a DD statement or ALLOCATE command to allocate space for a file on a disk volume does not mean that the file exists from the Fortran point of view. Such a file doesn't exist until an OPEN or WRITE statement for it has been used in a Fortran program or until it has had records written into it by some non-Fortran program or utility.

Programmer response: If you want to write records on the file, then either omit the ACTION specifier, or change the value of the ACTION specifier to WRITE or READWRITE.

If you want to just read from an existing file, then the ACTION specifier value of READ is correct. Ensure that all of the following are true:

- The file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement refers to the file that you intended.
- No other unit has been connected to the same file and has caused the file to be deleted before this OPEN statement was executed.
- The file exists according to the Fortran rules of file existence. (For example, under certain circumstances, a file that is present on a disk volume but contains no data doesn't exist according to this definition.)

If the file doesn't exist according to the Fortran definition of file existence, but you still want to read from the file, then omit the ACTION specifier to avoid detecting this error. In this case, ensure that the file has records that can be read.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1399

FOR1400S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The ACTION specifier had a value of READ, but the file definition referred to a file or device that was restricted to output only. Fortran Version 2 Error Number: AFB108I (format 5)**

Explanation: The ACTION specifier had a value of READ, which implied that only input processing would be performed on the file. However, the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement referred to a file or device that allows only output operations. Examples of such files include:

- A system output data set (SYSOUT parameter on the DD statement)
- A data set whose DD statement specifies LABEL=(,,,OUT)

Programmer response: Ensure that the value of the ACTION specifier is consistent with the capabilities of the file or device referenced by the file definition (DD statement or ALLOCATE command) or by the data set name given in the FILE specifier on the OPEN statement. You might have to change either the ACTION specifier, the file definition, or the data set name.

If you want to write on a file or device that allows only output, either omit the ACTION specifier or provide a value of WRITE for the action specifier.

If you want to read from a file, then change the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement to refer to a file or device from which you can read records. Ensure that the file exists or that are records to be read.

If you want to read to and write from the file, then ensure that the file or device allows you to perform both input and output. In this case, either omit the ACTION specifier, or provide a value of READWRITE for the ACTION specifier.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1400

FOR1401S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The ACTION specifier had a value of** *action***, but the file definition referred to a file or device that was restricted to input only. Fortran Version 2 Error Number: AFB108I (format 6)**

Explanation: The ACTION specifier had a value of *action*, which implied that output statements, such WRITE or ENDFILE, would to be executed. However, the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement referred to a file or device that doesn't allow output operations. Examples of such files include:

- An in-stream data set (DD *)
- A data set whose DD statement specifies LABEL=(,,,IN)
- A file for which the system's access control facility (such as RACF) prevents you from updating the data set.

Programmer response: Ensure that the value of the ACTION specifier is consistent with the capabilities of the file or device referenced by the file definition (DD statement or ALLOCATE command) or by the data set name given in the FILE specifier on the OPEN statement. You might have to change either the ACTION specifier, the file definition, or the data set name.

If you want to read from a file or a device that doesn't permit output, either omit the ACTION specifier or provide a value of READ for the STATUS specifier. Ensure that the file really exists and that you can read from it.

If you want to create a new file and write records on it, then provide a value of WRITE for the ACTION specifier. In this case, ensure that either of the following, as applicable, refers to a file or device on which you can write records:

- The file definition (DD statement or ALLOCATE command)
- For a dynamically allocated data, the data set name that follows the slash (/) in the FILE specifier

If you want to read to and write from the file, then ensure that the file or device allows you to perform both input and output. In this case, either omit the ACTION specifier, or provide a value of READWRITE for the ACTION specifier.

If you provide a value of WRITE or READWRITE for the ACTION specifier, don't refer to a data set such as an in-stream data set (DD *), a data set for which you don't have RACF authority to update, or a data set whose DD statement has a LABEL=(,,,IN) parameter.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1401

FOR1402S **The OPEN statement could not connect unit** *unit-number* **to** *file-name*. **The ACTION specifier, which had a value of READ, conflicted with the STATUS specifier, which had a value of** *status*. **Fortran Version 2 Error Number: AFB108I (format 8)**

Explanation: The ACTION specifier on the OPEN statement had a value of READ, which implied that you would read from but not write to a file which should already exist. However, the value of *status* for the STATUS implied that a new file was to be created. This was inconsistent because you can't read from a newly created file until records have been written on it.

Programmer response: If you want to read from an existing file without writing on it, then the value of READ for the ACTION specifier is correct. Either omit the STATUS specifier, or provide a value of OLD or UNKNOWN for the STATUS specifier. Ensure that the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement refers to a file that exists so that you can read from it.

If you want to write records on the file, change the value of the ACTION specifier to WRITE, or, if you want to both write to and read from the file, either change the value of the ACTION specifier to READWRITE or omit the ACTION specifier. For the STATUS specifier, use a value of NEW if you want to create a file that doesn't already exist, a value of REPLACE if you want to create a new file replacing the existing one if it already exists, or a value of OLD if you want to use an existing file even if you want to rewrite it. Omitting the STATUS specifier or providing value of UNKNOWN is the equivalent of NEW if the file doesn't exist or of OLD if it does exist.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1402

FOR1403S The OPEN statement could not connect unit *unit-number* **to** *file-name***. The file did not exist, and the file definition referred to a file or device that was restricted to input only. Fortran Version 2 Error Number: AFB108I (format 9)**

Explanation: The file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement referred to a file or device that doesn't allow output operations. Examples of such files include:

- A data set whose DD statement specifies LABEL=(,,,IN)
- A file for which the system's access control facility (such as RACF) prevents you from updating the data set.

However, the file didn't exist according to the Fortran definitions of file existence; this precludes any meaningful access to the file.

The Fortran definitions of file existence are explained in *VS FORTRAN Version 2 Programming Guide for CMS and MVS* in the chapter "What Determines File Existence" and generally reflect the operating system view of file existence; however, there are a few differences. For example, using a DD statement or ALLOCATE command to allocate space for a file on a disk volume does not mean that the file exists from the Fortran point of view. Such a file doesn't exist until an OPEN or WRITE statement for it has been used in a Fortran program or until it has had records written into it by some non-Fortran program or utility.

Programmer response: If you want to create a new file, ensure that the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement refers to a file or device that allows you to perform output operations.

If you want to read from an existing file, ensure that all of the following are true:

- The file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement refers to the file that you intended.
- No other unit has been connected to the same file and has caused the file to be deleted before this OPEN statement was executed.
- The file exists according to the Fortran rules of file existence. (For example, under certain circumstances, a file that is present on a disk volume but contains no data doesn't exist according to this definition.)

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1403

FOR1404S The OPEN statement could not connect unit *unit-number* **to** *file-name***. The ACCESS specifier had a value of DIRECT, and the file already existed, but the file was empty. Fortran Version 2 Error Number: AFB108I (format 10)**

Explanation: The file that was to be connected for direct access existed before the execution of the OPEN statement. However, there were no records in the file, and such a file can't be used for direct access. The fact that the file seemed to exist is based on one of the following:

- The file was previously connected for sequential access within the same executable program and was closed without any records having been written. Unless the STATUS specifier on that CLOSE statement had a value of DELETE, such a file is seen as an existing file according to the Fortran definitions of file existence, which are explained in *VS FORTRAN Version 2 Programming Guide for CMS and MVS* in the chapter "What Determines File Existence."
- The file was dynamically allocated, that is, the FILE specifier had a data set name preceded by a slash (**/**), and before execution of the OPEN statement the data set existed on the disk volume but contained no records.

- The NOOCSTATUS run-time option was in effect, the file didn't contain any records, and the file had not been used previously within the same executable program.

Programmer response: If you want to connect an existing file for direct access, ensure that all of the following are true:

- The file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement refers to the file that you intended.
- The file contains at least one record.
- The program that created the file successfully closed it.
- The file contains unblocked fixed-length records, that is, the RECFM value is F.

If you want to create a new file, then change the value of the STATUS specifier to NEW or SCRATCH.

If the file might have existed before the execution of the OPEN statement and if you want to replace whatever might have been in that file, then take one of these actions, as applicable:

- If the file is a named file, provide a value of REPLACE for the STATUS specifier.
- If the file was used earlier in a Fortran program, provide a value of DELETE for the STATUS specifier on the CLOSE statement for that previous use of the file.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

FOR1405S The OPEN statement could not connect unit *unit-number* to *file-name*. The ACTION specifier, which had a value of WRITE, conflicted with the KEYS specifier, which listed more than one key. Fortran Version 2 Error Number: AFB121I

Explanation: A value of WRITE was provided for the ACTION specifier on an an OPEN statement that specified keyed access; this implied that records were to be loaded into the file using the file's primary key. However, the KEYS specifier listed more than one start-end pair, and this is not permitted when loading records into the file.

Programmer response: If you want to load records into the file with records that are presented in increasing sequence of the primary key, then either remove the KEYS specifier or specify only the start-end pair that represents the primary key for the file. Ensure that the file definition (DD statement or ALLOCATE command) refers to the base cluster of the VSAM key-sequenced data set rather than to a path that corresponds to one of the alternate index keys.

If you want to process a file that is not empty, change the value of the ACTION specifier to READ or READWRITE.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1405

FOR1406S The OPEN statement could not connect unit *unit-number* to *file-name*. The KEYS specifier was given, but the ACCESS specifier did not have a value of KEYED. Fortran Version 2 Error Number: AFB137I

Programmer response: If you were connecting a VSAM key-sequenced data set (KSDS), then change the value of the ACCESS specifier to KEYED. Otherwise, remove the KEYS specifier from the OPEN statement, and ensure that the ACCESS specifier has a value of either SEQUENTIAL or DIRECT, as appropriate to your use of the file. If you code the value of the ACCESS specifier as a character constant, enclose the value in quotes or apostrophes.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1406

FOR1407S The OPEN statement could not connect unit *unit-number* to *file-name*. The ACCESS specifier had a value of KEYED, and the ACTION specifier had a value of READ, but the file was empty. Fortran Version 2 Error Number: AFB138I

Programmer response: Ensure that the VSAM key-sequenced data set (KSDS) that you want to process is referenced by the file definition (DD statement or ALLOCATE command).

If you want to read from a file that has records, then:

- If the file referenced by the file definition is a base cluster, that is, the file with the primary key, then ensure that is was successfully loaded with records, either in a Fortran program or by a program written in some other language.
- If the file referenced by the file definition is a path for an alternate index, then ensure that the Access Method Services BLDINDEX command successfully built the alternate index after the base cluster was loaded.

If you want to start with an empty file and add records to it in other than ascending sequence of the primary key, then change the value of the ACCESS specifier to READWRITE. In this case the OPEN statement processing simulates the loading of the file and deletes all loaded records; then VSAM no longer considers the file to be empty.

If you want to load records into an empty file in the fastest way, follow these steps:

1. On the file definition (DD statement or ALLOCATE command), provide the data set name of the file's base cluster, that is, of the file with the primary key.
2. Execute an OPEN statement with a value of WRITE for the ACTION specifier and a value of KEYED for the ACCESS specifier; either omit the KEYS specifier or provide a KEYS specifier with a single start-end pair that represents the position in the record of the file's primary key.
3. Execute at least one WRITE statement to write the records; ensure that you write the records in ascending sequence of the primary key.
4. Execute a CLOSE statement.

After these steps, the file is available to be connected using an ACTION specifier of either READ or READWRITE.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1407

FOR1408S The OPEN statement could not connect unit *unit-number* to *file-name*. The RECL specifier was given, but the ACCESS specifier did not have a value of DIRECT. Fortran Version 2 Error Number: AFB155I (format 1)

Programmer response: If you intend to execute direct access READ or WRITE statements, which have a REC specifier, then change the value of the ACCESS specifier to DIRECT. If you code the value as a character constant, enclose the value in quotes or apostrophes.

If you want to use sequential or keyed access, then remove the RECL specifier from the OPEN statement.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1408

FOR1409S The OPEN statement could not connect unit *unit-number* to *file-name*. The ACCESS specifier had a value of DIRECT, but the RECL specifier was not given. Fortran Version 2 Error Number: AFB155I (format 2)

Programmer response: If you intend to execute direct access READ or WRITE statements, which have a REC specifier, then add the RECL specifier to the OPEN statement. Ensure that this value given for the RECL specifier is the same as the value that's associated with the file through one or more of the following, as applicable:

- The label of an existing data set
- The LRECL parameter of the DD statement or ALLOCATE command
- The LRECL value given in an invocation of the FILEINF callable service
- The record length given in the RECORDSIZE parameter of the Access Method Services DEFINE command that was used to define the VSAM cluster

If you want to use sequential or keyed access, then change the value of the ACCESS specifier to SEQUENTIAL or KEYED. If you code the value as a character constant, enclose the value in quotes or apostrophes.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1409

FOR1410S The OPEN statement could not connect unit *unit-number* to *file-name*. The FILE specifier was given, but the STATUS specifier had a value of SCRATCH. Fortran Version 2 Error Number: AFB255I

Programmer response: If you want to connect a named file, then on the OPEN statement provide the FILE specifier and either omit the STATUS specifier or change the value of the STATUS specifier to NEW, OLD, REPLACE, or UNKNOWN.

If you want to connect a file as a scratch (or temporary) file, then omit the FILE specifier and provide a value of SCRATCH for the STATUS specifier.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1410

FOR1411S The OPEN statement could not connect unit *unit-number* to *file-name*. The ACCESS specifier had a value of DIRECT, but the file definition referred to a file or device that was not supported for direct access. Fortran Version 2 Error Number: AFB090I, AFB114I (format 1), AFB165I (format 1)

Programmer response: If you intend to use direct access, ensure that the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement refers to one of the following:
- For a non-VSAM file, a disk file that is not a PDS member.
- A VSAM relative record data set (RRDS).

If you intend to use direct access and you want to use an unnamed file (that is, you've omitted the FILE specifier on the OPEN statement), then ensure that there is no corresponding file definition with a ddname of the form that is used for a striped file. (This prohibited form is FT*nn*P*mmm*, where *nn* is the unit number given on the OPEN statement.) Instead, use a ddname of the form FT*nn*F001. If you need the file definition with a ddname of the form FT*nn*P*mmm* because you're also processing a striped file, then change the unit number either for the direct access I/O or for the striped file I/O.

If you didn't intend to use direct access I/O, change the ACCESS specifier on the OPEN statement to either SEQUENTIAL or KEYED.

If you want to process a VSAM key-sequenced data set (KSDS), then change the ACCESS specifier on the OPEN statement to KEYED. Do not confuse sequential or direct access with the sequential or direct retrieval statements that are used with keyed access. Sequential and direct access are indicated by the ACCESS specifier on the OPEN statement. Sequential and direct retrieval statements are forms of the READ statement that can be used when the file is connected for keyed access, that is, when the ACCESS specifier on the OPEN statement has a value of KEYED.

If you want to process a VSAM linear data set, don't use Fortran I/O statements; instead, use the data-in-virtual callable services described in *VS FORTRAN Version 2 Language and Library Reference*.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1411

FOR1412S The OPEN statement could not connect unit *unit-number* to *file-name*. The ACCESS specifier had a value of SEQUENTIAL, but the file definition referred to a file or device that was not supported for sequential access. Fortran Version 2 Error Number: AFB165I (format 1)

Programmer response: If you want to use sequential access, ensure that the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement does not refer to a VSAM key-sequenced data set (KSDS) nor to a VSAM linear data set.

If you want to process a VSAM key-sequenced data set (KSDS), then change the ACCESS specifier on the OPEN statement to KEYED. Do not confuse sequential or direct access with the sequential or direct retrieval statements that are used with keyed access. Sequential and direct access are indicated by the ACCESS specifier on the OPEN statement. Sequential and direct retrieval statements are forms of the READ statement that can be used when the file is connected for keyed access, that is, when the ACCESS specifier on the OPEN statement has a value of KEYED.

If you want to process a VSAM linear data set, don't use Fortran I/O statements; instead, use the data-in-virtual callable services described in *VS FORTRAN Version 2 Language and Library Reference*.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1412

FOR1413S The OPEN statement for unit *unit-number* failed. The unit was already connected to *file-name*, and the *specifier* specifier had a value that differed from the value that was already established. Fortran Version 2 Error Number: AFB120I, AFB100I

Explanation: The OPEN statement referred to a unit that was already connected to a file. Because the FILE specifier was either omitted or had a value that was same as the name of the file to which the unit was already connected, the OPEN statement did not cause the file to be disconnected and opened again; instead, the OPEN statement applied to the already existing connection. In addition, the *specifier* specifier on the OPEN statement had a value that was different from the value established earlier. This is not allowed because only the BLANK, CHAR, DELIM, and PAD specifiers can change the properties of the connection when the OPEN statement refers to an existing connection between a unit and a file.

Programmer response: Ensure that both the OPEN statement and any previously executed I/O statements refer to the unit number that you intend.

If you want to retain the existing connection between the unit and the file, then take one of these actions:

- Remove the *specifier* specifier from the OPEN statement.
- If you do use the *specifier* specifier, then ensure that its value is the same as what is already in effect.

If you want to use the OPEN statement to establish a new connection between the unit and the file, then take one of the following actions to disconnect the unit from the file to which it's already connected and to connect the unit to a different file:

1. Execute a CLOSE statement followed by another OPEN statement.

2. For a named file only, execute an OPEN statement with a FILE specifier whose value is different from the name of the file to which the unit is already connected. This has the effect of executing a CLOSE statement with no STATUS specifier followed by the OPEN statement.

For either of these two cases, all specifiers coded on the OPEN statement and the default values for all omitted specifiers provide the properties of the new connection between the unit and the file.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1413

FOR1414S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The KEYS specifier referred to a key with a record position of** *start* **:** *end***, but none of the file definitions for this file referred to a file with this key. Fortran Version 2 Error Number: AFB134I**

Programmer response: Ensure that the start-end pairs given in the KEYS specifier on the OPEN statement refer to files that have keys in the indicated positions in the record.

Provide the same number of file definitions (DD statements or ALLOCATE commands) as there are start-end pairs in the KEYS specifier. Use file definitions with these ddnames:

- For an unnamed file: FT*nn*K01, FT*nn*K02, ... FT*nn*K*kk*, where *nn* is the unit number and *kk* is the number of start-end pairs in the KEYS specifier. Both *nn* and *kk* consist of exactly two digits.
- For a named file: *file-name*, *file-name* 1, *file-name* 2, ... *file-name k*, where *file-name* is the file name given in the FILE specifier and *k* is an integer whose value is one less than the number of start-end pairs in the KEYS specifier. If *file-name* is eight characters long, the numeric suffixes overlay the last character of *file-name*.

Ensure that each of the file definitions with the ddnames used for keyed access files refers to a file with one of the keys listed as a start-end pair in the KEYS specifier. Check the VSAM Access Method Services DEFINE CLUSTER, DEFINE ALTERNATE INDEX, and DEFINE PATH commands that were used for the file to ensure that the intended keys are referenced. In referring to the positions with a record, the first position in a record is position 1 from the point of view of the KEYS specifier on the OPEN statement, but the first position in a record is position 0 from the point of view of KEYS parameter on the DEFINE CLUSTER or DEFINE ALTERNATEINDEX command. Therefore, the following KEYS specifier on an OPEN statement:

```
KEYS(9:11)
```

is equivalent to the following KEYS parameter on the DEFINE CLUSTER or DEFINE ALTERNATEINDEX command:

```
KEYS(3 8)
```

Each indicates a three-character key starting in the ninth position in the record.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1414

FOR1415S The OPEN statement could not connect unit *unit-number* **to** *file-name***. The file was already connected to another unit. Fortran Version 2 Error Number: AFB172I**

Programmer response: Ensure that the file name *file-name* isn't used in the FILE specifier on an OPEN statement while this same file is still connected to another unit. Correct the logic of the program to take one of these actions:

- Execute a CLOSE statement if necessary to disconnect the file from the first unit before you execute the second OPEN statement.
- Change the name given in the FILE specifier on one of the applicable OPEN statements to refer to a different file.
- If you intended to change only the values of the BLANK, CHAR, DELIM, or PAD specifiers for an existing connection of a unit and file (rather than establishing a connection), then ensure that the OPEN statement refers to the same unit as the one to which the file is already connected.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1415

FOR1416S The OPEN statement could not connect *unit-number* **to** *ddname***, the error message unit. The** *specifier* **specifier had a value that is not allowed for the error message unit.**

Programmer response: If you want to execute an OPEN statement that refers to the error message unit and therefore to the Language Environment message file, then ensure that the following specifiers are either omitted or given the values indicated:

Specifier	Acceptable value
STATUS	UNKNOWN
ACCESS	SEQUENTIAL
FORM	FORMATTED
ACTION	WRITE
POSITION	ASIS

Facilities other than what are implied by the acceptable values in the preceding list are not available with the error message unit. If you require the one of these unavailable facilities, change the OPEN statement and the other I/O statements to refer to a unit other than the error message unit.

Designating the error message unit and the print unit as different units lets you use the print unit in a manner similar to what was available with the error message unit in VS FORTRAN Version 2. This is because the print unit doesn't have the usage restrictions of the error message unit unless it's the same unit as the error message unit. Use the ERRUNIT and PRTUNIT run-time options to provide the unit numbers for the error message unit and the print unit.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1416

FOR1417S **The OPEN statement could not connect unit** *unit-number* **to** *file-name*. **Because there were** *key-count* **keys listed in the KEYS specifier, conflicting ddnames were generated to refer to different parts of the file. Fortran Version 2 Error Number: AFB131I**

Explanation: The OPEN statement for keyed access referred to the named file *file-name* and specified more that one start-end pair in the keys specifier. The required file definitions have the following ddnames: *file-name*, *file-name* 1, *file-name* 2, ... *file-name k*, where *file-name* is the file name given in the FILE specifier and *k* is an integer whose value is one less than the number of start-end pairs in the KEYS specifier. However, in this case *file-name* was eight characters long, and the numeric suffixes had to overlay the last character of *file-name*. But *file-name* itself ended in one of the required suffix characters, thus causing a conflict in the ddnames. For example, suppose that the OPEN statement contained the following specifiers:

```
FILE='CONFILE2'
KEYS=(2:4, 6:10, 15:20)
```

In this example, because there are three keys and because the ddname is eight characters long, the ddnames CONFILE2, CONFILE1, and CONFILE2 would be needed. The conflict is that there are only two rather than three unique ddnames.

Programmer response: If you intend to use *key-count* keys, then change the the file name in the FILE specifier on OPEN statement to one that has at least one of these characteristics:
- Fewer than eight characters
- An alphabetic character in the last position
- A digit in the last position, where the numeric value of that digit is not less than *key-count*

Also provide file definitions (DD statements or ALLOCATE commands) for the ddnames listed under “Explanation.”

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1417

FOR1425S **The OPEN statement for unit** *unit-number* **failed. For a keyed file, the file definition statement for** *ddname1* **referred to a file that had a record length of** *length1*, **but** *ddname2* **referred to a file that had a record length of** *length2*. **Fortran Version 2 Error Number: AFB132I**

Programmer response: Change the file definitions (DD statements or ALLOCATE commands) for this file so that they all refer to files that represent the same base cluster. For each of the alternate index keys that you want to use, do not refer to the data set name of the alternate index itself. Instead, use the data set name of the path that refers to the alternate index. Such a path is defined with the Access Method Services DEFINE PATH command. This path gives you access to the records in the base cluster through the alternate index.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1425

FOR1419S **The OPEN statement for unit *unit-number* failed. More than one key was specified for a keyed file, but the file definition statement for *ddname* referred to a file that was empty. Fortran Version 2 Error Number: AFB133I**

Programmer response: You cannot load records into an empty file and refer to more than one key using a single OPEN statement.

If you have already loaded the file with records and have established the alternate indexes, then ensure that the file definitions (DD statements or ALLOCATE commands) refer to the appropriate base cluster and to the various paths that are associated with the alternate indexes. Follow steps 8 through 9 in the list that follows.

If you just want to load records into an empty file, then don't refer to more than one key in the KEYS specifier. To load records into the file, follow steps 1 through 4 in the list that follows.

To establish the file so that you can use more that one key, as indicated by the KEYS specifier, then follow these steps to load records into the base cluster and to make alternate indexes available:

1. On the file definition (DD statement or ALLOCATE command), provide the data set name of the file's base cluster, that is, of the file with the primary key.
2. Execute an OPEN statement with a value of WRITE for the ACTION specifier and a value of KEYED for the ACCESS specifier; either omit the KEYS specifier or provide a KEYS specifier with a single start-end pair that represents the position in the record of the file's primary key.
3. Execute at least one WRITE statement to write the records; ensure that you write the records in ascending sequence of the primary key.
4. Execute a CLOSE statement.

Perform the following with Access Method Service commands rather than with Fortran I/O statements:

5. Define each alternate index with the DEFINE ALTERNATEINDEX command.
6. Build each alternate index with the BLDINDEX command.
7. Create a path for each alternate index with the DEFINE PATH command.

In a Fortran program you can then use keyed access to refer to more than one key as follows:

8. In the KEYS specifier on the OPEN statement, provide a start-end pair for each of the keys, either the primary key or one or more alternate index keys, that you want to use in your program.
9. Provide file definitions (DD statements or ALLOCATE commands) that refer to files representing each of the keys indicated by the KEYS specifier. For an alternate index keys, ensure that the file definition refers to the data set name of the path rather than of the alternate index itself.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1419

FOR1420S The OPEN statement for unit *unit-number* failed. The FILE specifier had a value of CEEDUMP, which is the ddname of the Language Environment dump file.

Programmer response: Either use a ddname of other than CEEDUMP in the FILE specifier on the OPEN statement or omit the FILE specifier to refer to an unnamed file.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1420

FOR1421S The OPEN statement for unit *unit-number* failed. The FILE specifier had a value of *ddname*, which was the ddname of the Language Environment message file.

Explanation: The ddname of *ddname*, which was given as the value of the FILE specifier on the OPEN statement, is already a Language Environment message file. This message file could be the current one, or it could be some previously used message file that hasn't yet been closed.

Programmer response: Take one or more of these actions:

- Use a ddname other than *ddname* as the value of the FILE specifier on the OPEN statement. Ensure that the ddname that you select isn't also a message file.
- Change the ddname of the message file to a value other than *ddname*. Do this by changing the ddname given in the MSGFILE run-time option or by changing the value of the FILE specifier in an OPEN statement that refers to the error message unit.
- Close the current message file with a CLOSE statement that refers to the error message unit, and connect the error message unit to a different message file with an OPEN statement that refers to the error message unit. (Closing the current message file causes the current message file to revert back to the ddname given in the MSGFILE run-time option.)

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1421

FOR1423S The OPEN statement could not connect unit *unit-number*. The STATUS specifier had a value of REPLACE, but there was no FILE specifier.

Programmer response: Ensure that if the OPEN statement has a value of REPLACE for the STATUS specifier then it also has a FILE specifier.

If you don't want to replace a possibly existing file, then don't use a value of REPLACE for the STATUS specifier.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1423

FOR1424S The OPEN statement could not connect unit *unit-number* **to** *file-name***. The PAD specifier was given, but the file was to be connected for unformatted input/output.**

Programmer response: Make one of these changes in the OPEN statement:

- If you want to use formatted input/output statements, then:
  - For sequential access either omit the FORM specifier, or change the value of the FORM specifier to FORMATTED.
  - For direct or keyed access, provide a value of FORMATTED of the FORM specifier.
- If you want to use unformatted input/output statements, then remove the PAD specifier.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1424

FOR1425S The OPEN statement could not connect unit *unit-number* **to** *file-name***. The PAD specifier had a value of** *pad***, which was other than YES or NO.**

Programmer response: Based on whether you want formatted records to be treated as though they were padded with blanks when they are read, change the value of the PAD specifier on the OPEN statement to YES or NO. If you code the value as a character constant, enclose the value in quotes or apostrophes.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1425

FOR1426S The OPEN statement could not connect unit *unit-number* **to** *file-name***. The RECL specifier had a value of** *recl-val***, which was incompatible with the maximum record length,** *record-size***, which was given in the DEFINE CLUSTER command for the VSAM relative record data set (RRDS).**

Programmer response: Ensure that the file definition (DD statement or ALLOCATE command) refers to the data set that you intend.

Ensure that the value given in the RECL specifier on the OPEN statement that was used for direct access is the same as the value given in RECORDSIZE parameter of the Access Method Services DEFINE CLUSTER command that defined the VSAM relative record data set (RRDS). Use the LISTCAT command, if necessary, to determine the value that was given when the DEFINE CLUSTER command defined the RRDS.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of OPEN, and *parm_count* has a value of 6.

No.	Name	Input/Output	Data Type and Length	Value
5	*record-size*	Input	INTEGER*4	The length of the data (or the maximum length of the data) established for the file through the DEFINE CLUSTER command for the VSAM data set
6	*recl-val*	Input	INTEGER*4	The length of the data indicated by the RECL specifier.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1426

FOR1427S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The RECL specifier had a value of** *recl-val***, which was incompatible with the length of the data,** *data-length***, which the records in the file can hold.**

Explanation: The value of the RECL specifier on the OPEN statement was incorrect in one or more of these ways:

- It was not a positive value.
- It exceeded the maximum data length that was established for the records in a file with variable-length records.
- It was not identical to the record length established for a file with fixed-length records.

The data length for the file could have been established on the file definition (DD statement or ALLOCATE command), in a call to the FILEINF callable service that applied to a file that was dynamically allocated, or in the Access Method services DEFINE CLUSTER command. For a file that already existed before execution of the OPEN statement, this length might have been established when the file was first created.

Programmer response: Ensure that the file definition or FILE specifier on the OPEN statement refers to the file that you intended.

Change the value of the RECL specifier on the OPEN statement, or change the record length wherever it was established. For a file with fixed-length records, the value of the RECL specifier must be identical to the value established elsewhere for the file.

For a file with variable-length records, the value of the RECL specifier must not exceed the length of the data that the records can hold. In this case, remember that the value of the LRECL parameter (for example, as a DCB subparameter on a DD statement) includes the four-byte record descriptor word, but the value of RECL specifier on the OPEN statement does not. Therefore, when you use variable-length records, the value of the RECL specifier cannot exceed the LRECL value less 4.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of OPEN, and *parm_count* has a value of 6.

No.	Name	Input/Output	Data Type and Length	Value
5	*data-length*	Input	INTEGER*4	The length of the data (or the maximum length of the data) established for the file.
6	*recl-val*	Input	INTEGER*4	The length of the data indicated by the RECL specifier.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1427

FOR1428S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The STATUS specifier had a value of REPLACE, but the file definition referred to a file or device that was restricted to input only.**

Explanation: The STATUS specifier had a value of REPLACE, which implied that a file should be created after deleting the existing one, if any. However, the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement referred to a file or device that doesn't allow output operations. Examples of such files include:

- An in-stream data set (DD *)
- A data set whose DD statement specifies LABEL=(,,,IN)
- A file for which the system's access control facility (such as RACF) prevents you from updating the data set

Programmer response: If you want to read from a file or a device that doesn't permit output, either omit the STATUS specifier or provide a value of OLD or UNKNOWN for the STATUS specifier. Ensure that the file really exists and that you can read from it.

If you want to delete a file that might exist and to create a new file and write records on it, then the STATUS specifier value of REPLACE is correct. As an alternative, use a value of NEW for the STATUS specifier if the file isn't supposed to exist before the execution of the OPEN statement. In either case, change either of the following, as applicable, to refer to a file or device on which you can write records:

- The file definition (DD statement or ALLOCATE command)
- For a dynamically allocated data, the data set name that follows the slash (**/**) in the FILE specifier

Don't refer to a data set such as an in-stream data set (DD *), a data set for which you don't have RACF authority to update, or a data set whose DD statement has a LABEL=(,,,IN) parameter.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1428

FOR1429S The OPEN statement could not connect unit *unit-number* **to** *file-name***. The POSITION specifier had a value of** *pos***, which was other than ASIS, REWIND, or APPEND.**

Programmer response: Based on where you want the file to be positioned after execution of the OPEN statement, change the value of the POSITION specifier on the OPEN statement to ASIS, REWIND, or APPEND. If you code the value as a character constant, enclose the value in quotes or apostrophes.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1429

FOR1430S The OPEN statement could not connect unit *unit-number* **to** *file-name***. The POSITION specifier was given, and the ACCESS specifier had a value of** *access***.**

Programmer response: If you want to connect the file for sequential access, then change the value of the ACCESS specifier on the OPEN statement to SEQUENTIAL.

If you want to connect a file for direct or keyed access, then remove the POSITION specifier from the OPEN statement because the POSITION specifier applies only to sequential access.

Do not confuse sequential or direct access with the sequential or direct retrieval statements that are used with keyed access. Sequential and direct access are indicated by the ACCESS specifier on the OPEN statement. Sequential and direct retrieval statements are forms of the READ statement that can be used when the file is connected for keyed access, that is, when the ACCESS specifier on the OPEN statement has a value of KEYED.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1430

FOR1431S The OPEN statement could not connect unit *unit-number* **to** *file-name***. The DELIM specifier had a value of** *delim***, which was other than APOSTROPHE, QUOTE, or NONE.**

Programmer response: Based on what delimiters you want to surround character values in output written with list-directed and namelist formatting, change the value of the DELIM specifier on the OPEN statement to APOSTROPHE, QUOTE, or NONE. If you code the value as a character constant, enclose the value in quotes or apostrophes.

A value of APOSTROPHE causes delimiters of ' to be used in the output, and a value of QUOTE causes delimiters of ″ to be used.

If you omit the DELIM specifier, then for list-directed formatting the character values are written without delimiters, and for namelist formatting the character values are surrounded by apostrophes.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1431

FOR1432S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The DELIM specifier was given, but the file was to be connected for unformatted input/output.**

Programmer response: Make one of these changes in the OPEN statement:

- If you want to use formatted input/output statements, then:
  - For sequential access either omit the FORM specifier, or change the value of the FORM specifier to FORMATTED.
  - For direct or keyed access, provide a value FORMATTED of the FORM specifier.
- If you want to use unformatted input/output statements, then remove the DELIM specifier.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1432

FOR1433S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The POSITION specifier had a value of APPEND, but the file definition referred to a file or a device for which positioning to the end is not allowed.**

Explanation: The POSITION specifier had a value of APPEND on an OPEN statement for a file that resides on a device that does not honor positioning commands in a way that would permit positioning the file to its terminal point. Most tape and disk files support positioning commands, whereas the following do not:
- In-stream data sets (DD *)
- System output data sets (SYSOUT parameter on the DD statement)
- Terminals
- Card readers
- Printers
- Card punches

Programmer response: Ensure that the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement refers to the file that you intended.

If the logic of your program requires that the file be positioned at its terminal point when it is connected, change the file definition or the data set name given in the FILE specifier to refer to a file or a device that supports this type of positioning. If the positioning to the terminal point is not required, either remove the POSITION specifier on the OPEN statement or change its value to ASIS or REWIND.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1433

FOR1434S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The file definition referred to a member of a partitioned data set (PDS), the POSITION specifier had a value of APPEND, and the ACTION specifier didn't have a value of READ.**

Explanation: The file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement referred to a member of a partitioned data set (PDS). The OPEN statement had a POSITION specifier with a value of APPEND, which indicates that the file should be positioned to its endfile record. However, because the ACTION specifier was either omitted or was coded with a value of other than READ, writing of records onto the file is implied. It is not possible to write records onto the end of an existing member of a PDS.

Programmer response: Ensure that the file definition or FILE specifier on the OPEN statement refers to the file that you intended.

If you want to write a new or replacement member of a PDS, then remove the POSITION specifier and provide a value of WRITE for the ACTION specifier.

If you want to read an existing member of a PDS from the beginning, provide a value of READ for the ACTION specifier and either remove the POSITION specifier or provide a value of REWIND for it.

If you want to read an existing member of a PDS after first positioning the file to the end (so that you can execute BACKSPACE statements, for example), then the value of APPEND for the POSITION specifier is correct. In this case, provide a value of READ for the ACTION specifier.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1434

FOR1435S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The file definition referred to a concatenation of data sets, and the POSITION specifier had a value of APPEND.**

Explanation: The file definition (DD statement or ALLOCATE command) referred to a concatenation of data sets, that is, to a sequence data sets that was to be processed as though it consisted of a continuous sequence of records in a single file. The POSITION specifier on the OPEN statement had a value of APPEND, which implied that this file was to be positioned at its terminal point. Such positioning cannot be done for a file that is a concatenation of data sets.

Programmer response: Ensure that the file definition refers to the file that you intended.

If the file that you want to process is a concatenation of data sets, either remove the POSITION specifier on the OPEN statement or change its value to REWIND or ASIS. You won't be able to position this file to its terminal point.

If the logic of your program requires that the file be positioned at its terminal point when it is connected, change the file definition to refer to a file or a device that supports this type of positioning.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	**Action Taken after Resumption**
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1435

FOR1436S **The OPEN statement could not connect unit** ***unit-number*** **to** ***file-name*****. The file definition referred to a concatenation of data sets, and the STATUS specifier had a value of REPLACE.**

Explanation: The file definition (DD statement or ALLOCATE command) referred to a concatenation of data sets, that is, to a sequence data sets that was to be processed as though it consisted of a continuous sequence of records in a single file. The STATUS specifier on the OPEN statement had a value of REPLACE, which implied that this file was to be deleted and recreated. This deletion and recreation cannot be done for a file that is a concatenation of data sets.

Programmer response: Ensure that the file definition refers to the file that you intended.

If the file that you want to process is a concatenation of data sets, either remove the STATUS specifier on the OPEN statement or change its value to OLD or UNKNOWN. You won't be able to delete the existing file.

If the logic of your program requires that the file be deleted and recreated, change the file definition to refer to a file or a device that supports file deletion.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	**Action Taken after Resumption**
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1436

FOR1437S **The OPEN statement could not connect unit** ***unit-number*** **to** ***file-name*****. The file definitions referred to a striped file, and the POSITION specifier had a value of APPEND.**

Explanation: The file definitions (DD statements or ALLOCATE commands) referred to a striped file, that is, to a file with ddnames of the form FT*nn*P*mmm*, where *nn* is the unit number and *mmm* is the stripe number. The POSITION specifier on the OPEN statement had a value of APPEND, which implied that this file was to be positioned at its terminal point. Such positioning cannot be done for a striped file.

Programmer response: Ensure that the file definition refers to the file that you intended.

If the file that you want to process is a striped file, either remove the POSITION specifier on the OPEN statement or change its value to REWIND or ASIS. You won't be able to position this file to its terminal point.

If the logic of your program requires that the file be positioned at its terminal point when it is connected, change the ddname on the file definition to the form FT*nn*F001. Ensure that the file or a device referenced by the file definition is one that supports this type of positioning.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	**Action Taken after Resumption**
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1437

FOR1438S **The OPEN statement could not connect unit** *unit-number* **to** *file-name***. The RECL specifier had a value of** *recl-val***, which is smaller than the minimum length,** *high-key-pos***, needed to contain all of the file's keys.**

Programmer response: Either remove the RECL specifier from the OPEN statement or increase its value so that it is large enough to contain all record positions of each of the keys listed as start-end pairs in the KEYS specifier on the OPEN statement.

If you don't need to refer to certain of the keys, then don't list the unneeded ones as start-end pairs in the KEYS specifier.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of OPEN, and *parm_count* has a value of 6.

No.	Name	Input/Output	Data Type and Length	Value
5	*recl-val*	Input	INTEGER*4	The value from the RECL specifier in the OPEN statement.
6	*high-key-pos*	Input	INTEGER*4	The minimum record size needed to include all the keys specified in the KEYS specifier.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1438

FOR1439S **The OPEN statement for unit** *unit-number***, which was already connected to file** *file-name***, failed. The POSITION specifier had a value of** *position***, which was other than ASIS.**

Explanation: The OPEN statement referred to a unit that was already connected to a file. Because the FILE specifier was either omitted or had a value that was same as the name of the file to which the unit was already connected, the OPEN statement did not cause the file to be disconnected and opened again; instead, the OPEN statement applied to the already existing connection. In addition, the POSITION specifier on the OPEN statement had a value of either REWIND or APPEND. This is not allowed because only the BLANK, CHAR, DELIM, and PAD specifiers can change the properties of the connection when the OPEN statement refers to an existing connection between a unit and a file.

Programmer response: Ensure that both the OPEN statement and any previously executed I/O statements refer to the unit number that you intend.

If you want to retain the existing connection between the unit and the file, then either remove the POSITION specifier from the OPEN statement or change the value of the POSITION specifier to ASIS.

If you want to use the OPEN statement to establish a new connection between the unit and the file, then take one of the following actions to disconnect the unit from the file to which it's already connected and to connect the unit to a different file:

- Execute a CLOSE statement followed by another OPEN statement.
- For a named file only, execute an OPEN statement with a FILE specifier whose value is different from the name of the file to which the unit is already connected. This has the effect of executing a CLOSE statement with no STATUS specifier followed by the OPEN statement.

For either of these two cases, all specifiers coded on the OPEN statement and the default values for all omitted specifiers provide the properties of the new connection between the unit and the file.

System action: If the unit is other than the error message unit, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1439

FOR1440S **The OPEN statement could not connect unit *unit-number* to *file-name*. The STATUS specifier had a value of SCRATCH, but the file definition referred to a file or device that is restricted to input only.**

Explanation: The STATUS specifier had a value of SCRATCH, which implied that a temporary file was to be created and that output statements, such WRITE or ENDFILE, would be executed. However, the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement referred to a file or device that doesn't allow output operations. Examples of such files include:

- An in-stream data set (DD *)
- A data set whose DD statement specifies LABEL=(,,,IN)
- A file for which the system's access control facility (such as RACF) prevents you from updating the data set

Programmer response: Ensure that the value of the STATUS specifier is consistent with the capabilities of the file or device referenced by the file definition (DD statement or ALLOCATE command) or by the data set name given in the FILE specifier on the OPEN statement. You might have to change either the STATUS specifier, the file definition, or the data set name.

If you want to read from a file or a device that doesn't permit output, either remove the STATUS specifier or change its value to OLD. Ensure that the file really exists and that you can read from it.

If you want to create a temporary file and write records on it, then the value of SCRATCH for the STATUS specifier is correct. In this case, ensure that the file definition refers to a file or device on which you can write records. Note that when the STATUS specifier has a value of SCRATCH, you must omit the FILE specifier.

If you want to create a new file that is other than a temporary file, either remove the STATUS specifier or change its value to NEW or REPLACE. The value of REPLACE allows you to create a new file if it doesn't already exist or to delete an existing one and create a new one.

If you want to overwrite the records in an existing file, either remove the STATUS specifier or change its value to OLD. Unless you specify either a value of APPEND for the POSITION specifier, the DISP=MOD parameter in the DD statement, or the MOD specifier in the ALLOCATE command, the file will be positioned to the beginning and the first WRITE statement will overwrite any existing records.

In any of the cases for which a file is to be created or records are to be written, ensure that the file definition refers to a file or device on which you can write records. For example, don't refer to a data set such as an in-stream data set (DD *), a data set for which you don't have RACF authority to update, or a data set whose DD statement has a LABEL=(,,,IN) parameter.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1440

FOR1441S The OPEN statement could not connect unit *unit-number* to *file-name*. The STATUS specifier had a value of SCRATCH, but the ACTION specifier had a value of READ

Explanation: The STATUS specifier had a value of SCRATCH, which implied that a temporary file was to be created and that output statements, such WRITE or ENDFILE, would be executed. However, the ACTION specifier had a value of READ, which implied that that no output statements would be executed.

Programmer response: Ensure that the value of the STATUS specifier is consistent with the value of the ACTION specifier and with the capabilities of the file or device referenced by the file definition (DD statement or ALLOCATE command) or by the data set name given in the FILE specifier on the OPEN statement. You might have to change either the STATUS specifier, the ACTION specifier, the file definition, or the data set name.

If you just want to read from a file, then the value of READ for the ACTION specifier is correct. In this case, either remove the STATUS specifier or change its value to OLD. Ensure that the file really exists and that you can read from it.

If you want to create a temporary file and write records on it, then the value of SCRATCH for the STATUS specifier is correct. In this case, either remove the ACTION specifier or change its value to READWRITE. Also ensure that the file definition refers to a file or device on which you can write records. Note that when the STATUS specifier has a value of SCRATCH, you must omit the FILE specifier.

System action: The unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The OPEN statement is not completed, and execution continues.

Symbolic Feedback Code: FOR1441

FOR1500S *locator-text System-message* Fortran Version 2 Error Number: AFB225I (format 1), AFB225I (format 2)

Explanation: An I/O error was detected by one of the underlying operating system's access methods; the error is described by *System-message*. Examples of causes include these situations:

- A permanent I/O error was encountered.
- The length of the data to be read or written was inconsistent with the block size specified in the file definition (DD statement or ALLOCATE command) or in the call to the FILEINF callable service for a dynamically allocated file.
- The length of the data to be read or written was inconsistent with the capabilities of the I/O device.
- The physical end of a tape was encountered while reading or writing a record.
- A storage medium error occurred on either tape or disk.

locator-text gives more information about the location of the error, and can be one of the following:
  The *statement* statement for unit *unit-number* failed.
  The INQUIRE statement failed.
  An error occurred during enclave termination.

Programmer response: Examine the description of the error described by *System-message*, and try to determine and fix the cause of the error. Check the possibilities listed under "Explanation." If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1501

FOR1502S *locator-text* **The volume** *volser* **did not have enough space available to create the new data set** *data-set-name*. **Fortran Version 2 Error Number: AFB103I (format 1)**

Explanation: The data set *data-set-name* was to be created using dynamic file allocation. (Dynamic allocation occurred either because *data-set-name* was given in the FILE specifier on the OPEN statement or because a value of SCRATCH was given in the STATUS specifier and there was no corresponding file definition, that is, no DD statement or ALLOCATE command.) The volume serial number *volser* was given as a value for the VOLSER or VOLSERS argument on the immediately preceding call to the FILEINF callable service, and this disk volume didn't have the amount of space either indicated by the CYL, TRK, MAXBLK, or MAXREC argument on the FILEINF call.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.

where *statement* is the OPEN statement.

Programmer response: Take one or more of these actions:

- If amount of space indicated by the CYL, TRK, MAXBLK, or MAXREC argument on the call to the FILEINF callable service is larger than you need, reduce the value.
- Use a volume other than *volser* if you know of one that might have more space.
- Remove the VOLSER or VOLSERS argument on the call to the FILEINF callable service so that space can be found on any available disk volume.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1502

FOR1503S *locator-text* **The data set** *data-set-name* **had been allocated to another job and was not available. Fortran Version 2 Error Number: AFB103I (format 2), AFB103I (format 3)**

Explanation: The data set *data-set-name* was to be connected using dynamic file allocation. (Dynamic allocation occurred either because *data-set-name* was given in the FILE specifier on the OPEN statement or because a value of SCRATCH was given in the STATUS specifier and there was no corresponding file definition, that is, no DD statement or ALLOCATE command.) However, your use of the data set was denied because one or both of these conditions existed:

- Another job had exclusive use of *data-set-name*.
- Another job was using *data-set-name* but your job wanted exclusive use of it.

The job, either your job or the other job, that requested exclusive use of the data set did so in one or more of these ways:

- In the immediately preceding call to the FILEINF callable service, the DISP argument had a value of NEW, OLD, or MOD.
- On the DD statement the DISP parameter had a value of NEW, OLD, or MOD.
- The ALLOCATE command had a NEW, OLD, or MOD parameter. For a non-VSAM data set or for a VSAM data set when VSAM record level sharing was not used (that is, there was no RLS parameter on the DD statement, on the ALLOCATE command, or on the call to FILEINF), either of the following implied exclusive use of the data set:
- On the OPEN statement the STATUS specifier had a value of NEW.
- On the OPEN statement the ACTION specifier was either omitted or had a value of READWRITE or WRITE.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.

Programmer response: If you can run the conflicting jobs, or at least the portions of them that use the data set *data-set-name*, at different times, then schedule the jobs accordingly.

If you must run the conflicting jobs at the same time and if both need the data set at the same time, then change either or both jobs so that they request shared use of the data set. To do this, take one or more of the following actions, as applicable, in both jobs:

- Change the value of the DISP argument for the FILEINF callable service to SHR.
- Change the value of the DISP parameter on the DD statement to SHR.
- Remove the NEW, OLD, or MOD parameter on the ALLOCATE command and replace it with SHR. For a non-VSAM data set or for a VSAM data set when VSAM record level sharing is not used:
- Change the value of the STATUS specifier on the OPEN statement to OLD.
- Change the value of the ACTION specifier on the OPEN statement to READ.

Note that except when VSAM record level sharing is used, the changes listed don't allow a file to be created nor do they allow either job to update the file.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	**Action Taken after Resumption**
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1503

FOR1504S *locator-text* **The volume** *volser***, which should have contained the data set** *data-set-name***, could not be found. Fortran Version 2 Error Number: AFB103I (format 4)**

Explanation: The data set *data-set-name* was to be connected using dynamic file allocation. (Dynamic allocation occurred either because *data-set-name* was given in the FILE specifier on the OPEN statement.) The volume serial number *volser* was given as a value for the VOLSER or VOLSERS argument on the immediately preceding call to the FILEINF callable service, but this disk volume wasn't available.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.

Programmer response: Ensure that volume serial number given as the value of the VOLSER or VOLSERS argument on the call to the FILEINF callable service is one that resides on the device given by the DEVICE argument.

If the data set is cataloged, you don't need to specify the volume serial number, so remove the VOLSER or VOLSERS argument.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1504

FOR1505S *locator-text* **An incorrect device name,** *dev-name***, was specified for the data set** *data-set-name***. Fortran Version 2 Error Number: AFB103I (format 5)**

Explanation: The data set *data-set-name* was to be connected using dynamic file allocation. (Dynamic allocation occurred either because *data-set-name* was given in the FILE specifier on the OPEN statement or because a value of SCRATCH was given in the STATUS specifier and there was no corresponding file definition, that is, no DD statement or ALLOCATE command.) The device name *dev-name* was given either as the value for the DEVICE argument on the immediately preceding call to the FILEINF callable service or as the default value that was established during the installation of Language Environment. However, this device name wasn't known on your system.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.

Programmer response: Ensure that the value given for the DEVICE argument in the call to the FILEINF callable service is a valid device on your system. If you didn't provide the DEVICE argument, then *dev-name* was the default value assigned during the installation of Language Environment. In this latter case or if you are unable to resolve the problem, seek assistance from your Language Environment support personnel to determine the device names that can be used at your site.

You can code the device name in the same three ways that you can code this same information in the UNIT parameter on the DD statement:

- A three-character hexadecimal number of the device. For example, 130.
- The generic name of the device that identifies a device by machine type and model. For example, 3380.
- A group name, which identifies a group of devices by a symbolic name. For example, SYSDA.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1505

FOR1506S *locator-text* **The volume did not have enough space for the directory for the data set** *data-set-name***. Fortran Version 2 Error Number: AFB103I (format 6)**

Explanation: The data set *data-set-name* was to be created using dynamic file allocation. (Dynamic allocation occurred because *data-set-name* was given in the FILE specifier on the OPEN statement.) The data set was to be a partitioned data set (PDS) and the number of directory blocks was given as the value of the DIR argument on the immediately preceding call to the FILEINF callable service. However, the number of directory blocks was so large that there wasn't enough space for the whole directory on the volume given by the VOLSER or VOLSERS argument, if any, on the FILEINF call.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.

Programmer response: Take one or more of these actions:

- If the number of directory blocks indicated by the DIR argument on the call to the FILEINF callable service is larger than you need, reduce the value.
- On the VOLSER or VOLSERS argument on the call to FILEINF, specify a volume that might have more space.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1506

FOR1507S *locator-text* **The space requested for the directory was greater than the amount of primary space requested for the data set** *data-set-name*. **Fortran Version 2 Error Number: AFB103I (format 7)**

Explanation: The data set *data-set-name* was to be created using dynamic file allocation. (Dynamic allocation occurred because *data-set-name* was given in the FILE specifier on the OPEN statement.) The data set was to be a partitioned data set (PDS) and the number of 256-byte directory blocks was given as the value of the DIR argument on the immediately preceding call to the FILEINF callable service. However, the number of directory blocks was so large that it wouldn't fit into the amount of disk space indicated by the CYL, TRK, MAXBLK, or MAXREC argument on the call to FILEINF.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.

where *statement* is the OPEN statement.

Programmer response: Take one or more of these actions:

- If the number of directory blocks indicated by the DIR argument on the call to the FILEINF callable service is larger than you need, reduce the value.
- Increase the amount of disk space indicated by the CYL, TRK, MAXBLK, or MAXREC argument on the call to FILEINF. Ensure that you have available to you a disk volume with enough space to allocate a data set of this size.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1507

FOR1508S *locator-text* **A required catalog was not available for the data set** *data-set-name***. Fortran Version 2 Error Number: AFB103I (format 8)**

Explanation: The data set *data-set-name* was to be referenced using dynamic file allocation. (Dynamic allocation occurred because *data-set-name* was given in the FILE specifier on the OPEN statement.) However, a catalog in which the data set was supposed to be cataloged wasn't available.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.

Programmer response: If the data set was created using Access Method Services and if the CATALOG parameter was used on the DEFINE command. then supply a JOBCAT or STEPCAT DD statement that refers to that same catalog.

In this doesn't resolve the problem, seek assistance from your Language Environment support personnel.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1508

FOR1509S *locator-text* **The catalog did not have enough space to add an entry for the data set** *data-set-name***. Fortran Version 2 Error Number: AFB103I (format 9)**

Explanation: The data set *data-set-name* was to be created using dynamic file allocation. (Dynamic allocation occurred because *data-set-name* was given in the FILE specifier on the OPEN statement.) However, there wasn't enough space available in a catalog for the new data set to be cataloged.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.

where *statement* is the OPEN statement.

Programmer response: Seek assistance from your Language Environment support personnel.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1509

FOR1510S *locator-text* **The data set** *data-set-name* **could not be dynamically allocated because the limit had been reached for the number of dynamically allocated files that could be in use at the same time. Fortran Version 2 Error Number: AFB169I (format 2)**

Explanation: The data set *data-set-name* was to be referenced using dynamic file allocation. (Dynamic allocation occurred either because *data-set-name* was given in the FILE specifier on the OPEN statement or because a value of SCRATCH was given in the STATUS specifier and there was no corresponding file definition, that is, no DD statement or ALLOCATE command.) All possible ddnames that can be used for dynamic allocated files were in use, so no more files could be dynamically allocated.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.

Programmer response: Reduce the number of dynamically allocated files.

If you've happened to use ddnames of either of these forms:
*s*DF*nnnnn*
DF*snnnnn*

where *s* is @, #, or $, and *nnnnn* is in the range from 00000 to 99999, then don't use these as ddnames of your own because they are what Language Environment uses internally for dynamically allocated files.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1510

FOR1511S *locator-text* **A permanent I/O error was detected while searching the partitioned data set (PDS) directory for** *file-name***. Fortran Version 2 Error Number: AFB219I (format 11)**

Programmer response: Ensure that the following are true:
- The data set was a PDS.
- The data set name wasn't used without a member name for input/output operations. Violating this restriction might have caused the directory to be overwritten.
- There weren't two different members in the data set being used for output operations at the same time.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1512

FOR1550S *locator-text* **The SVC 99 function executed for the data set** *data-set-name* **had a return code of** *return-code* **and an error reason code of** *reason-code***.** *System-message* **Seek assistance from your Language Environment support personnel. Fortran Version 2 Error Number: AFB103I (format 10)**

Explanation: The data set *data-set-name* was to be referenced using dynamic file allocation. (Dynamic allocation occurred either because *data-set-name* was given in the FILE specifier on the OPEN statement or because a value of SCRATCH was given in the STATUS specifier and there was no corresponding file definition, that is, no DD statement or ALLOCATE command.) The internally executed SVC 99 service, which performs the dynamic allocation, detected the error reflected by return code *return-code* and error reason code *reason-code*. It also provided *System-message* to explain the error.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.

Programmer response: For the meaning of return code *return-code* and error reason code *reason-code*, refer to *z/OS DFSMS Macro Instructions for Data Sets*.

If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1550

FOR1551W *locator-text* **The SVC 99 function executed for the data set** *data-set-name* **was successful, but an unusual condition occurred. The information reason code was** *information-code***.** *System-message* **Fortran Version 2 Error Number: AFB103I (format 10)**

Explanation: The data set *data-set-name* was to be referenced using dynamic file allocation. (Dynamic allocation occurred either because *data-set-name* was given in the FILE specifier on the OPEN statement or because a value of SCRATCH was given in the STATUS specifier and there was no corresponding file definition, that is, no DD statement or ALLOCATE command.) The internally executed SVC 99 service, which performs the dynamic allocation, detected an exceptional situation (not necessarily an error) reflected by information code *information-code*. It also provided *System-message* to explain the situation.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.

Programmer response: For the meaning of information code *information-code*, refer to *z/OS DFSMS Macro Instructions for Data Sets*.

If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: The file is allocated or deallocated, and execution continues.

Qualifying Data: None

Name	Action Taken after Resumption
RN	Execution resumes.

Symbolic Feedback Code: FOR1551

FOR1552C The SVC 99 function executed for the file *file-name* **during enclave initialization had a return code of** *return-code* **and an error reason code of** *reason-code*. *System-message* **Seek assistance from your Language Environment support personnel. Fortran Version 2 Error Number: AFB936I (format 2)**

Explanation: Dynamic allocation occurred for the print unit, file name *file-name*, because there was no corresponding file definition, that is, no DD statement or ALLOCATE command. The internally executed SVC 99 service, which performs the dynamic allocation, detected the error reflected by return code *return-code* and error reason code *reason-code*. It also provided *System-message* to explain the error.

Programmer response: For the meaning of return code *return-code* and error reason code *reason-code*, refer to *z/OS DFSMS Macro Instructions for Data Sets*.

If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: The condition is signaled, and execution of the enclave terminates.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR1552

FOR1553C The SVC 99 function executed for the file *file-name* **during enclave termination had a return code of** *return-code* **and an error reason code of** *reason-code*. *System-message* **Seek assistance from your Language Environment support personnel. Fortran Version 2 Error Number: AFB936I (format 3)**

Explanation: Dynamic allocation had occurred earlier for the print unit, file name *file-name*, because there was no corresponding file definition, that is, no DD statement or ALLOCATE command. Then during termination, the deallocation of this file occurred. The internally executed SVC 99 service, which performs the dynamic allocation, detected the error reflected by return code *return-code* and error reason code *reason-code*. It also provided *System-message* to explain the error.

Programmer response: For the meaning of return code *return-code* and error reason code *reason-code*, refer to *z/OS DFSMS Macro Instructions for Data Sets*.

If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: The condition is signaled, and execution of the enclave terminates.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR1553

FOR1554S *locator-text* **The** *macro-name* **macro instruction executed for** *file-name* **had a return code of** *return-code* **and a reason code of** *reason-code*. **Seek assistance from your Language Environment support personnel. Fortran Version 2 Error Number: AFB091I, AFB225I (format 5), AFB219I (format 9)**

Explanation: In support of the Fortran I/O statement indicated by the message text, Language Environment executed a non-VSAM *macro-name* macro instruction. DFSMS/MVS detected the error indicated by return code *return-code* and reason code *reason-code*.

locator-text gives more information about the location of the error, and can be one of the following:

The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.
An error occurred during enclave termination.

Programmer response: For the meaning of return code *return-code* and reason code *reason-code*, refer to *z/OS DFSMS Macro Instructions for Data Sets*. If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1556

FOR1557S *locator-text* **The VSAM** *macro-name* **macro instruction executed for** *file-name* **had a return code of** *return-code***. Seek assistance from your Language Environment support personnel. Fortran Version 2 Error Number: AFB111I (format 4), AFB130I, AFB167I**

Explanation: In support of the Fortran I/O statement indicated by the message text, Language Environment executed a VSAM *macro-name* macro instruction. DFSMS/MVS detected the error indicated by return code *return-code*.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.
An error occurred during enclave termination.

Programmer response: For the meaning of return code *return-code* refer to *z/OS DFSMS Macro Instructions for Data Sets*. If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: If the error occurred during enclave termination or from a CLOSE statement, the file is disconnected, but not deleted (as though the STATUS specifier had been coded with a value of KEEP). If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file.

If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1557

FOR1558S *locator-text* **The VSAM** *macro-name* **macro instruction executed for** *file-name* **had a return code of** *return-code* **and an error code of X'***hex-code***' (***decimal-code***). Seek assistance from your Language Environment support personnel. Fortran Version 2 Error Number: AFB111I (format 4), AFB130I, AFB167I**

Explanation: In support of the Fortran I/O statement indicated by the message text, Language Environment executed a VSAM *macro-name* macro instruction. DFSMS/MVS detected the error indicated by the return code *return-code* and the error code with a hexadecimal value of *hex-code* (decimal value of *decimal-code*).

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.
An error occurred during enclave termination.

Programmer response: For the meaning of return code *return-code* and error code *hex-code* (or *decimal-code*), refer to *z/OS DFSMS Macro Instructions for Data Sets*. If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: If the error occurred during enclave termination or from a CLOSE statement, the file is disconnected, but not deleted (as though the STATUS specifier had been coded with a value of KEEP). If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file.

If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1558

FOR1559S *locator-text* **There was a system completion code of** *completion-code* **involving file** *file-name***. Seek assistance from your Language Environment support personnel. Fortran Version 2 Error Number: AFB111I (format 1), AFB225I (format 4)**

Explanation: In support of the Fortran I/O statement indicated by the message text, Language Environment executed a data management macro instruction such as OPEN, READ, WRITE, or CHECK. Either MVS or DFSMS/MVS detected the error indicated by the system completion (abend) code *completion-code*.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.
An error occurred during enclave termination.

Programmer response: Ensure that the file definition (DD statement or ALLOCATE command) for the file you are using is coded correctly. If the the program uses dynamic allocation for the file, ensure that both the data set name given in the FILE specifier on the OPEN statement and the arguments on call to the FILEINF callable service are coded correctly.

For the meaning of *completion-code* and for possible corrective actions, refer to *z/OS DFSMS Macro Instructions for Data Sets*.

System action: If the error occurred during enclave termination or from a CLOSE statement, the file is disconnected, but not deleted (as though the STATUS specifier had been coded with a value of KEEP). If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file.

If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1559

FOR1560S *locator-text* **There was a system completion code of** *completion-code* **and a reason code of** *reason-code* **involving file** *file-name***. Seek assistance from your Language Environment support personnel. Fortran Version 2 Error Number: AFB111I (format 1), AFB225I (format 4), AFB219I (format 10)**

Explanation: In support of the Fortran I/O statement indicated by the message text, Language Environment executed a data management macro instruction such as OPEN, READ, WRITE, or CHECK. Either MVS or DFSMS/MVS detected the error indicated by system completion (abend) code *completion-code* and reason code *reason-code*.

locator-text gives more information about the location of the error, and can be one of the following:

The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.
An error occurred during enclave termination.

Programmer response: Ensure that the file definition (DD statement or ALLOCATE command) for the file you are using is coded correctly. If the the program uses dynamic allocation for the file, ensure that both the data set name given in the FILE specifier on the OPEN statement and the arguments on call to the FILEINF callable service are coded correctly.

For the meaning of system completion code *completion-code* and reason code *reason-code*, and for possible corrective actions, refer to *z/OS DFSMS Macro Instructions for Data Sets*.

System action: If the error occurred during enclave termination or from a CLOSE statement, the file is disconnected, but not deleted (as though the STATUS specifier had been coded with a value of KEEP). If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file.

If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1561

FOR1563S The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. A block that was read had an unexpected length. Fortran Version 2 Error Number: AFB091I

Explanation: In a record of the file being read, there were one mor more of the following inconsistencies:

- The record format was either V, VS, D, VB, VBS, or DB, and the block descriptor did not match the actual block size.
- The record format was either V, VS, D, VB, VBS, or DB, and the record descriptor implied that the record extended past the end of the block.
- The record format was VBS, and a record descriptor value was too small; that is, it implied no data but it was not a valid VBS null segment.
- The record format was D, and the actual block size did not exceed the specified block size.
- The record format was FB with a block preface, and the actual block size did not exceed the specified block preface size.
- The record format was FB, and the last record extended beyond the end of the block.

(The *record format* refers to the RECFM value that is provided either in the file definition (DD statement or ALLOCATE command) or in the label of an existing file.)

Programmer response: Ensure that the file definition (DD statement or ALLOCATE command) for the file you are using is coded correctly in the following ways:

- The file definition refers to the data set that you want to process.
- The RECFM value, if any, indicates the same record format that was specified when the file was created.
- The LRECL value, if any, indicates the same record length that was specified when the file was created.
- The BLKSIZE value, if any, indicates the same block length that was specified when the file was created.

Ensure that the file was closed successfully when it was created. If it wasn't, then the file must be created again because the last block might not have been written correctly.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1565

FOR1570S *locator-text* **The file definition with the ddname** *ddname* **was missing. Fortran Version 2 Error Number: AFB219I (format 1), AFB219I (format 7)**

Explanation: There was no file definition (DD statement or ALLOCATE command) with the ddname implied by the I/O statement.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.
An error occurred during enclave termination.

Programmer response: Determine the ddname implied by the I/O statement, and provide a DD statement or ALLOCATE command with this ddname.

If the file is an unnamed file, then the ddname takes the form FT*nn*F001 for sequential and direct access files and the form FT*nn*K*mm* for keyed access files, where *nn* is the unit number. A file is an *unnamed file* in either of these cases:

- The OPEN statement that connects the unit and the file has no FILE specifier.
- The unit is seen as preconnected to a file because a sequential I/O statement is executed for the unit before an OPEN statement.

If the file is a named file, then the ddname has the value given for the FILE specifier on the OPEN statement.

If you want to refer to the file through dynamic allocation, then ensure that there is an OPEN statement with a FILE specifier whose value consists of a slash (**/**) followed by the data set name. In this case, no file definition is needed.

If the OPEN statement has no FILE specifier and you want that OPEN statement to refer to an existing connection between the unit and a file, then ensure that the unit is already connected to a file before the OPEN statement is executed. (Failure to observe this restriction could cause the OPEN statement to be interpreted as referring to an unnamed file.)

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1570

FOR1571S *locator-text* **The OPEN macro instruction executed for** *file-name* **had a system completion code of** *completion-code* **and a reason code of** *reason-code*. **Seek assistance from your Language Environment support personnel. Fortran Version 2 Error Number: AFB219I (format 1)**

Explanation: In support of the Fortran I/O statement indicated by the message text, Language Environment executed an OPEN macro instruction. Either MVS or DFSMS/MVS detected the error indicated by system completion (abend) code *completion-code* and reason code *reason-code*.

locator-text gives more information about the location of the error, and can be one of the following:

The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.
An error occurred during enclave termination.

Programmer response: Ensure that the file definition (DD statement or ALLOCATE command) for the file you are using is coded correctly. If the the program uses dynamic allocation for the file, ensure that both the data set name given in the FILE specifier on the OPEN statement and the arguments on call to the FILEINF callable service are coded correctly.

For the meaning of system completion code *completion-code* and reason code *reason-code*, and for possible corrective actions, refer to *z/OS DFSMS Macro Instructions for Data Sets*.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1571

FOR1900E *locator-text* **The file** *file-name* **was to be deleted, but the file definition referred to a file or device for which file deletion was not possible. Fortran Version 2 Error Number: AFB105I, AFB111I (format 2)**

Explanation: One of the following implied that the file deletion was to occur:

- The CLOSE statement had a STATUS specifier with a value of DELETE.
- The CLOSE statement referred to a unit for which the corresponding OPEN statement had a STATUS specifier with a value of SCRATCH.
- During enclave termination, a unit was still connected to a file for which the corresponding OPEN statement had a STATUS specifier with a value of SCRATCH.

In addition, the OCSTATUS run-time option was in effect, and the file had a characteristic, such as one of the following, that precluded file deletion:

- VSAM data set that is not empty and that is not reusable
- Unlabeled tape data set
- In-stream (DD *) data set
- Sysout data set
- Terminal
- Unit record input data set
- Unit record output data set
- Subsystem file
- Concatenation of multiple data sets
- Keyed file with an alternate index
- LABEL=(,,,IN) parameter on the DD statement
- IN parameter on the ALLOCATE command
- Multiple sub-files
- Connected with an OPEN statement that had an ACTION specifier with a value of READ

locator-text gives more information about the location of the error, and can be one of the following:

The *statement* statement for unit *unit-number* failed.
The INQUIRE statement failed.
An error occurred during enclave termination.

Programmer response: If file deletion is required so that further use of the file isn't possible, then change one or more of the following to avoid referring to a file with any of the characteristics listed under "Explanation":

- The value of the ACTION specifier on the OPEN statement to either WRITE or READWRITE
- The value of the FILE specifier on the OPEN statement to refer to some other ddname
- The value of the FILE specifier on the OPEN statement to refer to some other data set name
- The value of the UNIT specifier on the OPEN statement to refer to some other Fortran unit number
- The DD statement parameters such as the reference to a particular device or data set
- The ALLOCATE command parameters such as the reference to a particular device or data set
- The reusability attribute of a VSAM data set in the Access Method Services DEFINE CLUSTER command
- The combination of DD statements or ALLOCATE command parameters that concatenate multiple data sets under a single ddname

If file deletion isn't required, then either use the NOOCSTATUS run-time option, or modify the program in one or more of the following ways so that file deletion won't occur:

- On the OPEN statement, either omit the STATUS specifier or provide a value other than SCRATCH.
- On the CLOSE statement, either omit the STATUS specifier or provide a value other than DELETE.

System action: The file is disconnected, but not deleted (as though the STATUS specifier on the CLOSE statement had been coded with a value of KEEP). If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The file is closed but not deleted, and execution continues.

Symbolic Feedback Code: FOR1900

FOR1910S *locator-text* **The end of the file was reached. Fortran Version 2 Error Number: AFB200I, AFB217I**

Explanation: The execution of the READ statement requested that a record be read even though the file was already positioned beyond the last data record in the file.

This is the *end-of-file condition* and might not be an error.

For a READ statement that specified namelist formatting, this also could have occurred for the following reason: For the namelist group name given in the FMT specifier on the READ statement, there was no corresponding namelist group in the input file at a point beyond where the file was already positioned. Some syntax error within the input file, such as a missing &END delimiter or a missing quote or apostrophe delimiter, might have caused the namelist group to treated as part of some other construct and thus appear as though it wasn't in the input file.

locator-text gives more information about the location of the error, and can be one of the following:
The *statement* statement for an internal file failed.
The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed.

where *statement* is READ.

Programmer response: Ensure that the READ statement refers to the unit that you intended, that the unit is connected to the file that you intended, and that the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement refers to the file that you intended. Also ensure that the file was created successfully, either in a program or by some manual process such as an editor.

If the file is the one you want to read, modify the program to detect the end of the file in one of these ways:

- Use the END specifier on the READ statement so that your program can gain control at some specific label when the end-of-file condition occurs.
- Use the IOSTAT specifier on the READ statement so that upon completion of the READ statement your program can determine whether the end-of-file condition occurred. A value of −1 in the variable given in the IOSTAT specifier indicates that the end-of-file condition occurred.
- If you know the number of records in the file before reading them, maintain a count of the number of records read so that your program doesn't attempt to read beyond the last data record.

If your program uses statements that position the file, ensure that the logic of the program and the contents of the file don't cause the file to be inadvertently positioned to the wrong place. The following statements affect the position within the file:
- ENDFILE
- BACKSPACE
- REWIND
- OPEN with a POSITION specifier that has a value of APPEND
- READ statement, even with no input item list
- READ statement with a format specification that has a slash (**/**) edit descriptor
- WRITE, which for sequential access causes the record that's written to become the last record in the file

In addition, if the DD statement has the DISP=MOD parameter or the ALLOCATE command has the MOD parameter, the file is positioned beyond the last data record when it's first connected.

If namelist formatting is requested on the READ statement because the FMT specifier refers to the a namelist group name declared in a NAMELIST statement, ensure that the namelist group indicated by the FMT specifier is actually in the file at some point beyond the current position. (There's no attempt to read through the entire file to find the namelist group; the search is only from the current file position.) In the file, identify the namelist group with an ampersand (&) followed by the namelist group name with no intervening blanks. Ensure that the ampersand begins in position 2 or later, and that all positions preceding the ampersand in the record are blank.

In a namelist input file, ensure that all namelist groups, especially any that precede the one referenced by the failing READ statement, are coded in the correct format. Remember that all information must start no earlier than position 2 of the records. Pay attention to all delimiters, such as commas, equal signs, quotes, and apostrophes, to ensure that they are used as required. Also ensure that each namelist group is ended by the characters &END.

System action: If the IOSTAT=*ios* specifier is present on the READ statement, *ios* becomes defined either with the value −1 if *ios* is an integer variable or with the condition token for FOR1910 if *ios* is a character variable of length 12. If the END=*stl* specifier is present on the READ statement, control passes to the label *stl*. If neither the END nor the IOSTAT specifier is present on the READ statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *statement* has a value of READ, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.
RF	If the error occurred on a READ statement for sequential access to an unnamed file that is neither VSAM nor striped, the data set sequence number is increased by 1 and the next subfile is read; otherwise, execution continues.

Symbolic Feedback Code: FOR1910

FOR1915S **The OPEN statement for unit *unit-number* failed. The FILE specifier was not given, and the unit number was greater than 99, the maximum unit number allowed for unnamed files. Fortran Version 2 Error Number: AFB175I**

Programmer response: If you want to connect an unnamed file, ensure that the value given for the the unit number does not exceed the smaller of either 99 or the highest unit number allowed at your site.

If you want to connect a named file, add a FILE specifier to the OPEN statement and provide either a ddname or a data set name, the latter preceded by a slash (**/**).

If you want the OPEN statement to refer to an existing connection between a unit and a file (so that you can use one or more of the BLANK, CHAR, DELIM, and PAD specifiers to change the properties of the connection), then ensure that the unit is connected to a file before executing the OPEN statement.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated. The statement is ignored, and processing continues.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *file* has a value of blanks, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1915

FOR1916S **The** *statement* **statement for unit** *unit-number* **failed. The unit number was either less than 0 or greater than** *max-unit-num***, the highest unit number allowed at your installation. Fortran Version 2 Error Number: AFB220I**

Programmer response: Ensure that the unit number given on the *statement* statement is a positive number that doesn't exceed *max-unit-num*, which was established during the installation or customization of Language Environment as the highest unit number available at your site.

The defaults provided by IBM have 99 as the highest unit number. However, this can be changed by updating the UNTABLE parameter of the AFHOUTCM macro instruction in the module AFHOUTAG. For more information, refer to *z/OS Language Environment Customization*.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated. The statement is ignored, and execution continues.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *file* has a set of blanks, and *parm_count* has a value of 5. In addition, there are these qualifying data:

No.	Name	Input/Output	Data Type and Length	Value
5	*max-unit-name*	Input	INTEGER*4	The highest unit number allowed at your installation.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1916

FOR1917S **The** *statement* **statement could not be executed for unit** *unit-number***. The unit was not connected to a file. Fortran Version 2 Error Number: AFB110I**

Programmer response: Ensure that the *statement* statement refers to the unit that you intended.

Check the logic of your program to ensure that the unit *unit-number* is connected to a file before executing the *statement* statement. Here are some items to examine or correct:

- Determine whether some error occurred during the An error detected for an OPEN statement, for example, usually causes the unit to be disconnected from the file. If such an error occurred, correct the problem.
- Ensure that a CLOSE statement didn't disconnect the unit from the file. If it did, then either execute an OPEN statement to reconnect the unit to a file or don't execute the CLOSE statement.
- If you want to use a preconnected file, that is, one that can be read or written without first executing an OPEN statement, then ensure that there's a file definition (DD statement or ALLOCATE command) with a ddname FT*nn*F001, where *nn* is the two-digit unit number.
- If you want to use a unit number greater than 99, then use an OPEN statement to connect a named file to the unit. To connect a named file, provide the FILE specifier on the OPEN statement.

It's possible for the *statement* statement used for sequential access to automatically reconnect a unit to an unnamed file. To do this, ensure that these conditions are met:

- The unit number doesn't exceed 99.
- There's a file definition with the ddname FT*nn*F001, where *nn* is the two-digit unit number. This is the file to which the unit will be reconnected.
- The NOOCSTATUS run-time option is in effect.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *file* has a value of blanks, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1917

FOR1920S **The** *statement* **statement for unit** *unit-number***, which was connected to** *file-name***, failed. The file definition referred to a file or device that was restricted to input only. Fortran Version 2 Error Number: AFB108I (format 7)**

Explanation: The *statement* is an output statement, but the file definition (DD statement or ALLOCATE command) or the data set name given in the FILE specifier on the OPEN statement referred to a file or device that doesn't allow output operations. Examples of such files include:

- An in-stream data set (DD *)
- A data set whose DD statement specifies LABEL=(,,,IN)
- A file for which the system's access control facility (such as RACF) prevents you from updating the data set

Programmer response: Ensure that I/O statement to be executed is consistent with the capabilities of the file or device referenced by the file definition (DD statement or ALLOCATE command) or by the data set name given in the FILE specifier on the OPEN statement. You might have to change either the file definition, the data set name, or the I/O statements used to process the file.

If you want to perform output operations on a file, don't refer to an input-only data set such as an in-stream data set (DD *), a data set for which you don't have RACF authority to update, or a data set whose DD statement has a LABEL=(,,,IN) parameter.

System action: If the error occurred during the execution of an OPEN statement, the unit is no longer connected to a file. If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1920

FOR1921S **The** *statement* **statement for unit** *unit-number***, which was connected to** *file-name***, failed. Execution of this statement was inconsistent with the ACTION specifier on the OPEN statement, which had a value of** *action***. Fortran Version 2 Error Number: AFB122I**

Programmer response: Either change the value of the ACTION specifier on the OPEN statement that connected the unit to the file, or change the logic of your program so that you don't execute a statement that isn't allowed by the value that you provide for the ACTION specifier. Also ensure that the ACCESS specifier has a value that indicates the type of file access that you want to use. Here are the permissible statements based on the values of the ACTION specifier and of the ACCESS specifier:

Value of ACTION Specifier	Value of ACCESS Specifier	Permissible Input/Output Statements
READ	SEQUENTIAL KEYED	READ, BACKSPACE, REWIND, CLOSE with STATUS='KEEP'
READ	DIRECT	READ, CLOSE with STATUS='KEEP'
READWRITE	SEQUENTIAL	READ, BACKSPACE, REWIND, WRITE, ENDFILE, CLOSE with STATUS='KEEP'
READWRITE	DIRECT	READ, WRITE, CLOSE with any STATUS value
READWRITE	KEYED	READ, BACKSPACE, REWIND, WRITE, REWRITE, DELETE, CLOSE with any STATUS value
WRITE	SEQUENTIAL	WRITE, ENDFILE, CLOSE with any STATUS value
WRITE	DIRECT	WRITE, CLOSE with any STATUS value
WRITE	KEYED	WRITE, CLOSE with STATUS='KEEP'

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1921

FOR1922S The *statement* **statement for unit** *unit-number***, which was connected to** *file-name***, failed. The file was not usable because a permanent I/O error was detected Fortran Version 2 Error Number: AFB152I**

Programmer response: Determine the cause of the error on some previous I/O statement that was executed for this unit, and correct the problem.

To continue using a file connected for sequential access after an error has occurred, execute a REWIND statement.

To use the same unit or file through a newly established file connection, execute a CLOSE statement followed by an OPEN statement.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1922

FOR1923S The *statement* **statement for unit** *unit-number***, which was connected to** *file-name***, failed. An implied DO in the input or output item list inconsistently specified the initial, terminal, and increment values. The initial value was** *initial-value***; the terminal value was** *terminal-value***, and the increment value was** *increment-value***. Fortran Version 2 Error Number: AFB203I**

Explanation: In the input or output item list of a READ or WRITE statement there was an implied DO such as the following:

(A(I), I = *initial-value*, *terminal-value*, *increment-value*)

For one of the levels of nesting in the implied DO, the combination of *initial-value*, *terminal-value*, and *increment-value* was incorrect in one of these ways:

- *increment-value* = 0
- *terminal-value* < *initial-value* and *increment-value* > 0
- *terminal-value* > *initial-value* and *increment-value* < 0

Programmer response: Ensure that the combination of the initial value, terminal value, and increment value in the implied DO doesn't have any of the inconsistencies listed under "Explanation" for any level of nesting.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1923

FOR1924S **The** *statement* **statement for unit** *unit-number* **failed. The statement was executed from within an MTF parallel subroutine and referred to an unnamed file. Fortran Version 2 Error Number: AFB923I**

Programmer response: Change the MTF parallel subroutine so that the I/O statements refer only to named rather than to unnamed files. A file is an *unnamed file* in either of these cases:

- The OPEN statement that connects the unit and the file has no FILE specifier.
- The unit is seen as preconnected to a file because a sequential I/O statement is executed for the unit before an OPEN statement. (Except for the standard input unit, the error message unit, and the print unit, preconnected files don't exist in an MTF parallel subroutine.)

A file is a *named file* if the OPEN statement that connects the unit and the file has a FILE specifier. Therefore, in the parallel subroutine include the FILE specifier on each OPEN statement that's used to connect a unit to a file.

If the OPEN statement has no FILE specifier and you want that OPEN statement to refer to an existing connection between the unit and a file, then ensure that the unit is already connected to a file before the OPEN statement is executed. (Failure to observe this restriction could cause the OPEN statement to be interpreted as referring to an unnamed file.)

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled either in the parallel subroutine or in the main task program, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *file* has a value of blanks, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1924

FOR1925S The *statement* statement for unit *unit-number*, which was the error message unit, failed. The statement or the form of the statement that was used is not permitted for the error message unit. Fortran Version 2 Error Number: AFB234I

Explanation: The *statement* statement referred to the error message unit and therefore to the Language Environment message file, but was other than one of the following statements that is allowed to refer explicitly to the error message unit:
- OPEN statement
- INQUIRE statement
- CLOSE statement
- WRITE statement for sequential access and formatted I/O

The following statements refer to the print unit. They are allowed to refer implicitly to the error message unit when the error message unit and the print unit are the same unit:
- PRINT statement
- WRITE statement that has * as the unit identifier

Programmer response: Change the program so that if you refer to the error message unit you use only the statements or forms of statements listed under “Explanation.” For example, do not use these statements:
- WRITE statement for direct access (REC specifier)
- WRITE statement for keyed access (KEY specifier)
- WRITE statement for asynchronous I/O (ID specifier)
- WRITE statement for unformatted I/O (no format specifier)
- ENDFILE statement
- REWIND statement
- BACKSPACE statement
- DELETE statement
- REWRITE statement

If you need to use any of the prohibited statements or forms of statements, change the unit specifier to refer to some unit other than the error message unit. If you need only the BACKSPACE, REWIND, or ENDFILE statements in conjunction with output that’s directed to the print unit (rather than explicitly to the error message unit), then take both of the following actions:
- Use the ERRUNIT and PRTUNIT run-time options to define the error message unit and the print unit as different units.
- On the BACKSPACE, REWIND, or ENDFILE statement, provide a unit number that is the same as what you’ve used as the value of the PRTUNIT run-time option.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *file* has a value of blanks, and *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1925

FOR1926S *locator-text* The file name was *file-name*. The immediately previous invocation of the FILEINF callable service failed. Fortran Version 2 Error Number: AFB219I (format 9)

Programmer response: Determine the cause of the error that caused the previous call to the FILEINF callable service to fail; then correct the problem.

If the call to the FILEINF callable service doesn’t need to be used for the OPEN or INQUIRE statement, then remove the call.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1926

FOR1927S An I/O statement was executed, but some other I/O statement had not yet completed. Fortran Version 2 Error Number: AFB904I

Explanation: There was a recursive call to the Fortran input/output library routines that are part of Language Environment. Here are some situations that could have caused the recursive call:

- Within the input item list on a READ statement there was a reference to a user-written function that performs some other I/O operation. To illustrate this situation, here's an example of a READ statement with a function reference:

  ```
  READ (8) I, A(INXFUNC(I))
  ```

  and here's an example of the referenced function subprogram with a PRINT statement whose execution causes error FOR1927 to be detected:

  ```
        FUNCTION INXFUNC ( X )
        INTEGER*4  INXFUNC
        INTEGER*4  X
        IF ( X .GT. 4 ) THEN
           PRINT *, 'INCORRECT SUBSCRIPT VALUE.  ASSUMING 1.'
           INXFUNC = 1
        ELSE
           INXFUNC = X
        ENDIF
        END
  ```

- As a result of some error that was detected during the execution of a Fortran I/O statement, the following occurred:
  1. The condition representing the error was signaled.
  2. Assuming that a user-written condition handler had been registered before the I/O statement was executed, the condition handler was entered. This condition handler included a Fortran subprogram.
  3. The Fortran subprogram executed a PRINT statement to report the error.

  The condition handling was considered to be a subordinate part of the original failing I/O statement. Therefore, execution of the PRINT statement in the condition handler caused the prohibited recursive entry into the input/output library and thus caused error FOR1927 to be detected.

Programmer response: Restructure the program to avoid the recursive entry into the Fortran input/output library.

System action: The ERR and IOSTAT specifiers are not honored. The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	4
2	*statement*	Input	CHARACTER*12	The name of the I/O statement being processed
3	*unit*	Input	INTEGER*4	Undefined
4	*file*	Input	CHARACTER*62	Undefined

Permissible Resume Actions: None

Symbolic Feedback Code: FOR1927

FOR1928S During execution of an I/O statement that had an IOSTAT specifier, a condition occurred from the internal use of some Language Environment callable service.

Explanation: Some unusual condition occurred during the execution of an I/O statement that had an IOSTAT=*ios* specifier, where *ios* was an INTEGER*4 variable. This condition wasn't one that was detected by the Fortran library portion of Language Environment but rather by one of the internally used routines that are part of Language Environment. Because the IOSTAT specifier was present, this condition was not signaled. Instead, a value of 1928 was returned in *ios*.

(If *ios* had been a character variable of length 12, the condition token reflecting the condition that was detected would have been returned. If the IOSTAT specifier had not been present, that condition would have been signaled.)

Programmer response: If there isn't some obvious problem involving either the I/O statement, the file definition (DD statement or ALLOCATE command), or the virtual storage available to the application, remove (at least temporarily) the IOSTAT specifier. Then the condition reflecting the problem that was detected will be signaled.

System action: If the IOSTAT=*ios* specifier is present on the I/O statement, and if *ios* is an integer variable, the value 1928 is returned in *ios*. If *ios* is a character variable of length 12, the condition token that reflects the error that was detected by the internally executed callable service is returned in *ios*.

If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition detected by the internally executed callable service is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR1928

FOR1929S The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The ADVANCE specifier had a value of *advance*, which was other than YES or NO.

Programmer response: Based on whether you want to use advancing or nonadvancing input/output, change the value of the ADVANCE specifier on the *statement* statement to YES or NO. If you code the value as a character constant, enclose the value in quotes or apostrophes.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1929

FOR1930S The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The EOR specifier was provided on a formatted READ statement that did not have an ADVANCE specifier with the value NO.

Programmer response: If you want to use the advancing READ statement, that is, the conventional form of the READ statement, remove the EOR specifier from the READ statement. Also remove the SIZE specifier if it is present.

If you want to use nonadvancing input/output, then on the READ statement provide an ADVANCE specifier with a value of NO. If you code the value as a character constant, enclose the value in quotes or apostrophes.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1930

FOR1931S The *statement* statement for unit *unit-number*, which was connected to *file-name*, failed. The SIZE specifier was provided on a formatted READ statement that did not have an ADVANCE specifier with the value NO.

Programmer response: If you want to use the advancing READ statement, that is, the conventional form of the READ statement, remove the SIZE specifier from the READ statement. Also remove the EOR specifier if it is present.

If you want to use nonadvancing input/output, then on the READ statement provide an ADVANCE specifier with a value of NO. If you code the value as a character constant, enclose the value in quotes or apostrophes.

System action: If neither the ERR nor the IOSTAT specifier is present on the I/O statement, the condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: Only the basic set of four qualifying data for I/O conditions as shown in Table 1 on page 228. Within this basic set, *parm_count* has a value of 4.

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The I/O operation is not completed, and execution continues.

Symbolic Feedback Code: FOR1931

FOR2000C The MTF parallel subroutine load module *module-name*, the name of which was specified in the AUTOTASK run-time option, did not exist in the load library specified by the AUTOTASK file definition statement. VS FORTRAN Version 2 Error Number: AFB919I-1

Programmer response: Ensure that your multitasking facility (MTF) parallel subroutine load module was link-edited as a member of the partitioned data set referenced by the file definition (DD statement or ALLOCATE command) with the ddname of AUTOTASK. Use the name of this member in the AUTOTASK run-time option, which must be in the following format:

AUTOTASK(*loadmod*,*numtasks*)

where:

loadmod	Is the name of the parallel subroutine load module; that is, the name of the member in the data set referenced by the file definition with the ddname AUTOTASK.
numtasks	Is the number of tasks to be created by MTF.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2000

FOR2001C The PDS member *member-name*, the name of which was specified in the AUTOTASK run-time option as the name of the MTF parallel subroutine load module, was not a valid load module. VS FORTRAN Version 2 Error Number: AFB919I-2

Programmer response: Ensure that your multitasking facility (MTF) parallel subroutine load module was link-edited as a member of the partitioned data set referenced by the file definition (DD statement or ALLOCATE command) with the ddname of AUTOTASK. Use the name of this member in the AUTOTASK run-time option, which must be in the following format:

AUTOTASK(*loadmod*,*numtasks*)

where:

loadmod	Is the name of the parallel subroutine load module; that is, the name of the member in the data set referenced by the file definition with the ddname AUTOTASK.
numtasks	Is the number of tasks to be created by MTF.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2001

FOR2003C The MTF parallel subroutine load module *module-name***, the name of which was specified in the AUTOTASK run-time option, had the not-editable linkage editor attribute. VS FORTRAN Version 2 Error Number: AFB919I-3**

Programmer response: Ensure that your multitasking facility (MTF) parallel subroutine load module was link-edited as a member of the partitioned data set referenced by the file definition (DD statement or ALLOCATE command) with the ddname of AUTOTASK. Also ensure that there were no failures during the link-editing process.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2003

FOR2004C The MTF parallel subroutine load module *module-name***, the name of which was specified in the AUTOTASK run-time option, did not contain the entry point VFEIS#. VS FORTRAN Version 2 Error Number: AFB919I-4**

Programmer response: Ensure that your multitasking facility (MTF) parallel subroutine load module was link-edited as a member of the partitioned data set referenced by the file definition (DD statement or ALLOCATE command) with the ddname of AUTOTASK. Also ensure that the module VFEIS# was included during the link-edit process. As an example of how to link-edit your parallel subroutine load module, assume that:

- You've compiled your parallel subroutine, and it's in the temporary data set &&LOADSET.
- You want the load module to be given the member name SUB001, and you want it placed in the load load with the data set name MY.SUB.LOAD.

Then you would code the following job step to link-edit your parallel subroutine load module:

```
//LINKPS EXEC  CEEWCL,PGMLIB='MY.SUB.LOAD',GOPGM=SUB001
//SYSLIN   DD  DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD  *
  INCLUDE SYSLIB(VFEIS#)
  ENTRY VFEIS#
/*
```

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2004

FOR2005C The MTF parallel subroutine load module *module-name***, the name of which was specified in the AUTOTASK run-time option, contained VFEIS#, but VFEIS# was not the entry point of the load module. VS FORTRAN Version 2 Error Number: AFB919I-5**

Programmer response: Ensure that your multitasking facility (MTF) parallel subroutine load module was link-edited as a member of the partitioned data set referenced by the file definition (DD statement or ALLOCATE command) with the ddname of AUTOTASK. Also ensure that the your parallel subroutine is a subroutine subprogram, or possibly more than one subroutine subprogram, and that there is no main program.

As an example of how to link-edit your parallel subroutine load module, assume that:

- You've compiled your parallel subroutine, and it's in the temporary data set &&LOADSET.
- You want the load module to be given the member name SUB001, and you want it placed in the load load with the data set name MY.SUB.LOAD.

Then you would code the following job step to link-edit your parallel subroutine load module:

```
//LINKPS EXEC  CEEWCL,PGMLIB='MY.SUB.LOAD',GOPGM=SUB001
//SYSLIN   DD  DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD  *
  INCLUDE SYSLIB(VFEIS#)
  ENTRY VFEIS#
/*
```

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2005

FOR2030C The AUTOTASK file definition statement was missing. VS FORTRAN Version 2 Error Number: AFB925I-1

Programmer response: Ensure that your multitasking facility (MTF) parallel subroutine load module was link-edited as a member of a partitioned data set. Do not use a partitioned data set extended (PDSE). Provide a file definition (DD statement or ALLOCATE command) with the ddname of AUTOTASK; in this file definition, refer to the data set into which you link-edited your parallel subroutine load module.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2030

FOR2031C The AUTOTASK file definition statement did not specify a load library. VS FORTRAN Version 2 Error Number: AFB925I-2

Programmer response: Ensure that your multitasking facility (MTF) parallel subroutine load module was link-edited as a member of a partitioned data set. Do not use a partitioned data set extended (PDSE). In the file definition (DD statement or ALLOCATE command) with the ddname of AUTOTASK, refer to the data set into which you link-edited your parallel subroutine load module.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2031

FOR2032C The OPEN macro instruction executed for the load library that was specified by the AUTOTASK file definition statement and that should contain the MTF parallel subroutine load module had a system completion code of *completion-code* and a reason code of *reason-code*. Seek assistance from your Language Environment support personnel. VS FORTRAN Version 2 Error Number: AFB925I-1

Explanation: During the initialization of the Fortran multitasking facility (MTF), Language Environment executed an OPEN macro instruction that referred to the file definition (DD statement or ALLOCATE command) with a ddname of AUTOTASK. (This is the file definition that is supposed to refer to the partitioned data set (PDS) into which was link-edited the parallel subroutine load module with the name given in the AUTOTASK run-time option.) Either MVS or

DFSMS/MVS detected the error indicated by system completion (abend) code *completion-code* and reason code *reason-code*.

Programmer response: Ensure that the file definition with the ddname AUTOTASK is coded correctly and that it refers to the PDS (not PDSE) into which was link-edited the parallel subroutine load module with the name given in the AUTOTASK run-time option.

For the meaning of system completion code *completion-code* and reason code *reason-code*, and for possible corrective actions, refer to *z/OS MVS System Codes*.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2032

FOR2040S The MTF callable service *service-name* failed. The service was called from an MTF parallel subroutine. VS FORTRAN Version 2 Error Number: AFB920I-1, AFB157I-1

Explanation: The multitasking facility (MTF) callable service *service-name* was called from a parallel subroutine. However, this service is restricted to use in the main task program.

Programmer response: Remove the call to *service-name* from the parallel subroutine, and, if necessary, place the call in the main task program instead.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*service-name*	Input	CHARACTER*6	The name of the MTF callable service

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution resumes.

Symbolic Feedback Code: FOR2040

FOR2041S The MTF callable service *service-name* failed. The service was called with an argument list in an incorrect format. VS FORTRAN Version 2 Error Number: AFB920I-2, AFB157I-3

Explanation: The argument list provided in the call to the *service-name* callable service was incorrect in one of these ways:

- There was no argument list when one was required.
- The argument list had an incorrect number of arguments.
- For either the DSPTCH or SHRCOM callable service, the argument list wasn't in the internally-generated form produced by the Fortran compiler when there are character arguments. This could have occurred because:
  - The first argument was not of character type.
  - The call was made from a program compiled by the VS FORTRAN Version 1 or the VS FORTRAN Version 2 compiler with the the LANGLVL(66) compiler option.
  - The call was made from a program compiled by the VS FORTRAN Version 1 compiler at a level before Release 3.
  - The call was made from a program compiled by the FORTRAN IV H Extended or the FORTRAN IV G1 compiler.
  - The call was made from an assembler language program, and the arguments were not provided in the form required when there are character arguments.

Programmer response: Ensure that the argument list contains the required number of arguments and that the arguments are of the required type. For further information, refer to the chapter “Multitasking Facility (MTF) Subroutines” in *VS FORTRAN Version 2 Language and Library Reference*.

For either the DSPTCH or SHRCOM callable service, follow these rules: If the program is written in Fortran, compile it with the VS FORTRAN Version 2 compiler, and do not specify the LANGLVL(66) compiler option. If it is written in assembler language, use the Fortran conventions for argument lists with character arguments. These conventions are described in the section “Passing Character Arguments Using the Standard Linkage Convention” in Appendix B of *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*service-name*	Input	CHARACTER*6 The name of the MTF callable service	

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution resumes.

Symbolic Feedback Code: FOR2041

FOR2042S **The MTF callable service** ***service-name*** **failed. The multitasking facility was not active. VS FORTRAN Version 2 Error Number: AFB920I-3**

Programmer response: If you intend to use the Fortran multitasking facility (MTF), then do the following to make MTF active for the application:

1. Link-edit your parallel subroutine load module as a member of a partitioned data set.
2. Provide a file definition (DD statement or ALLOCATE command) with the ddname of AUTOTASK. Refer this file definition to the data set into which you link-edited your parallel subroutine load module.
3. Provide the AUTOTASK run-time option in the following format:

   AUTOTASK(*loadmod*,*numtasks*)

   where:

loadmod	Is the name of the parallel subroutine load module; that is, the name of the member in the data set referenced by the file definition with the ddname AUTOTASK.
numtasks	Is the number of tasks to be created by MTF.

If you didn’t intend to use MTF, then make one of these changes:

- If the call to *service-name* was meant to refer to one of your own routines rather than to the MTF callable service, then during the link-editing of your application, ensure that your own routine is included in the load module. Do this either by using the linkage editor INCLUDE statement or by concatenating the library containing your routine ahead of the Language Environment product library in the SYSLIB DD statement.
- If the call to *service-name* was coded to use the MTF callable service, then remove the call because you can’t use this callable service unless MTF is active for the application (as discussed earlier).

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	2
2	*service-name*	Input	CHARACTER*6	The name of the MTF callable service

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2043

FOR2044S The MTF callable service SHRCOM failed. The dynamic common block *common-name* **was not declared in any program unit that was invoked in the main task program. VS FORTRAN Version 2 Error Number: AFB099I**

Programmer response: Ensure that common block name, *common-name*, given as the argument for the SHRCOM callable service is the name that you intended. If you code the name as a character constant, enclose the name in quotes or apostrophes.

Also ensure that at least one program unit that was entered before the SHRCOM callable service was invoked has a declaration of the common block as a dynamic common block. (Declaring it in the program unit that invokes the SHRCOM callable service meets this requirement.) Specify the common block as a dynamic common block by supplying its name as a suboption of the DC compile-time option.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	3
2	*service-name*	Input	CHARACTER*6	SHRCOM
3	*common-name*	Input	CHARACTER*31	The name of the dynamic common block

Permissible Resume Actions:

Permissible Resume Actions:

Name	Action Taken after Resumption
RN	The service is ignored, and execution resumes.

Symbolic Feedback Code: FOR2044

FOR2056S MTF subtask *subtask-number* **abnormally terminated during execution of parallel subroutine** *subroutine-name***. The system completion code was** *system-completion-code***, and the reason code was** *reason-code***. VS FORTRAN Version 2 Error Number: AFB922I-2**

Explanation: At the time that a call was made to one of multitasking facility (MTF) callable services SYNCRO, DSPTCH, or SHRCOM, Language Environment detected that the parallel subroutine *subroutine-name* in MTF subtask *subtask-number* had ended unexpectedly because of the abnormal termination indicated by the system completion code *completion-code* and reason code *reason-code*. The message describing this abnormal termination is in the message file for the subtask. This message file is the one referenced by the file definition (DD statement or ALLOCATE command) with the ddname FTERR*sss*, where *sss* is the three-digit representation of *subtask-number*.

Programmer response: For the meaning of system completion code *completion-code* and reason code *reason-code*, and for possible corrective actions, refer to *z/OS MVS System Codes*.

System action: The condition FOR2056 is signaled in the main task program. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2056

FOR2057S **MTF subtask** *subtask-number* **abnormally terminated during execution of parallel subroutine** *subroutine-name***. The user completion code was** *user-completion-code***, and the reason code was** *reason-code***. VS FORTRAN Version 2 Error Number: AFB922I-2**

Explanation: At the time that a call was made to one of multitasking facility (MTF) callable services SYNCRO, DSPTCH, or SHRCOM, Language Environment detected that the parallel subroutine *subroutine-name* in MTF subtask *subtask-number* had ended unexpectedly because of the abnormal termination indicated by the user completion code *completion-code* and reason code *reason-code*. If the parallel subroutine wrote any output on the message file before terminating, that output is in the message file for the subtask. This message file is the one referenced by the file definition (DD statement or ALLOCATE command) with the ddname FTERR*sss*, where *sss* is the three-digit representation of *subtask-number*.

The abnormal termination was requested by executing an assembler language ABEND macro instruction or by calling one of the CEE3ABD, SYSABN, or SYSABD callable services, among others. The exact meaning of the user completion code *completion-code* and reason code *reason-code* depends on the application that requested the abnormal termination. Some form of information about the meaning of these codes should be available to users of that application.

Programmer response: Correct the problem indicated by the user completion code *completion-code* and reason code *reason-code*.

System action: The condition FOR2057 is signaled in the main task program. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2057

FOR2058S **MTF subtask** *subtask-number* **terminated during execution of MTF parallel subroutine** *subroutine-name* **because of an unhandled condition of severity** *severity***. VS FORTRAN Version 2 Error Number: AFB922I-2**

Explanation: At the time that a call was made to one of multitasking facility (MTF) callable services SYNCRO, DSPTCH, or SHRCOM, Language Environment detected that the parallel subroutine *subroutine-name* in MTF subtask *subtask-number* had ended unexpectedly because of an unhandled condition of severity *severity*. The message describing this condition is in the message file for the subtask. This message file is the one referenced by the file definition (DD statement or ALLOCATE command) with the ddname FTERR*sss*, where *sss* is the three-digit representation of *subtask-number*.

Programmer response: Correct the problem indicated by the unhandled condition.

System action: The condition FOR2058 is signaled in the main task program. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2058

FOR2059S **A CSECT with the name CEEUOPT was present in the MTF parallel subroutine load module.**

Programmer response: Link-edit the parallel subroutine load module without the CSECT with the name CEEUOPT.

If you want to provide run-time options that are link-edited with the application, then link-edit the CSECT with the name CEEUOPT into the main task load module. The run-time options that you specify in this manner or in any other manner will apply in the main task program and in all parallel subroutines.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2059

FOR2060S The MTF callable service DSPTCH could not be completed. The first argument to the DSPTCH callable service specified that *subroutine-name* **was to be invoked as a parallel subroutine, but the parallel subroutine load module did not contain a subroutine with the specified name. VS FORTRAN Version 2 Error Number: AFB921I**

Programmer response: Ensure that the parallel subroutine name, *subroutine-name*, given as the first argument for the DSPTCH callable service is the name that you intended. If you code the name as a character constant, enclose the name in quotes or apostrophes. Do not provide a name that exceeds eight characters in length.

Ensure that the parallel subroutine *subroutine-name* is link-edited into your multitasking facility (MTF) parallel subroutine load module. Also ensure that this load module is link-edited as a member of the partitioned data set referenced by the file definition (DD statement or ALLOCATE command) with the ddname of AUTOTASK. Use the name of this member in the AUTOTASK run-time option, which must be in the following format:

AUTOTASK(*loadmod*,*numtasks*)

where:

loadmod	Is the name of the parallel subroutine load module; that is, the name of the member in the data set referenced by the file definition with the ddname AUTOTASK.
numtasks	Is the number of tasks to be created by MTF.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	Name	Input/Output	Data Type and Length	Value
1	*parm-count*	Input	INTEGER*4	3
2	*service-name*	Input	CHARACTER*6	DSPTCH
3	*subroutine-name*	Input	CHARACTER*8	The name of the MTF parallel subroutine

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2060

FOR2062C During MTF initialization, the *macro-name* **macro instruction had a return code of** *return-code***. Seek assistance from your Language Environment support personnel. VS FORTRAN Version 2 Error Number: AFB924I**

Explanation: During the initialization of the Fortran multitasking facility (MTF), Language Environment executed a *macro-name* macro instruction. Either MVS or DFSMS/MVS detected the error indicated by return code *return-code*.

Programmer response:

For the meaning of return code *return-code*, and for possible corrective actions, refer to one of the following:
z/OS MVS System Codes
z/OS DFSMS Macro Instructions for Data Sets

If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2062

FOR2063C During MTF initialization, the *macro-name* macro instruction had a return code of *return-code* and a reason code of *reason-code*. Seek assistance from your Language Environment support personnel. VS FORTRAN Version 2 Error Number: AFB925I

Explanation: During the initialization of the Fortran multitasking facility (MTF), Language Environment executed a *macro-name* macro instruction. Either MVS or DFSMS/MVS detected the error indicated by return code *return-code* and reason code *reason-code*.

Programmer response:

For the meaning of return code *return-code* and reason code *reason-code*, and for possible corrective actions, refer to one of the following:

z/OS MVS System Codes
z/OS DFSMS Macro Instructions for Data Sets

If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2063

FOR2064S The MTF callable service DSPTCH could not be completed. The MTF main task program was operating in 31-bit addressing mode, but the parallel subroutine load module had the linkage editor attribute that indicated 24-bit addressing mode. VS FORTRAN Version 2 Error Number: AFB927I

Programmer response: Determine whether any routine in the parallel subroutine load module must execute in 24-bit addressing mode or why the linkage editor gave it the attribute indicating that it must. For example, code compiled by the FORTRAN IV H Extended compiler must run in 24-bit addressing mode; therefore, if the parallel subroutine load module contains such code, then this attribute is correct.

If a routine in the parallel subroutine load module must execute in 24-bit addressing mode, then ensure that the main task program is running in 24-bit addressing mode at the time that it calls the DSPTCH callable service. Do this in one of these ways:

- When you link-edit the main task program, provide these linkage editor options:

  AMODE=24
  RMODE=24

  This causes the main task program to be invoked in 24-bit addressing mode and to reside below 16 Mb.
- If a portion of the main task program must run in 31-bit addressing mode, then use an assembler language routine to switch into 24-bit addressing mode at some point before calling the DSPTCH callable service. (Also switch back into 31-bit addressing mode as necessary.) Remember that in order to switch successfully into 24-bit addressing mode, the load module must reside below 16 Mb. To ensure that the load module is loaded below 16 Mb, provide this linkage editor option:

  RMODE=24

If you're sure that no routine in the parallel subroutine load module must be invoked in 24-bit addressing mode, then when you link-edit this load module, provide this linkage editor option:

AMODE=31

In addition, if the parallel subroutine load module can reside above 16 Mb, then provide this linkage editor option as well:

RMODE=ANY

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data:

No.	**Name**	**Input/Output**	**Data Type and Length**	**Value**
1	*parm-count*	Input	INTEGER*4	3
2	*service-name*	Input	CHARACTER*6	DSPTCH
3	*subroutine-name*	Input	CHARACTER*8	The name of the MTF parallel subroutine

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2064

FOR2065C The MTF main task program load module was created with Language Environment, but the MTF parallel subroutine load module was created with VS FORTRAN. VS FORTRAN Version 2 Error Number: AFB928I

Programmer response: Link-edit parallel subroutine load module using Language Environment. This is required because your main task program was link-edited using Language Environment.

If, when you link-edit your parallel subroutine load module, you want to use your executable load module (rather than the original object modules) as input to the linkage editor, then remember to replace the VS FORTRAN library modules that are in that load module. For information on how to do this, refer to *z/OS Language Environment Programming Reference*.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2065

FOR2067S The MTF main task program terminated while subtask *subtask-number* was still executing parallel subroutine *subroutine-name*. VS FORTRAN Version 2 Error Number: AFB930I

Programmer response: Ensure that before the main task program ends it invokes the SYNCRO callable service to wait for the completion of the parallel subroutines.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2067

FOR2068C MTF internal error *error-number* was detected. Seek assistance from your Language Environment support personnel. VS FORTRAN Version 2 Error Number: AFB931I

Programmer response: Because this error is not likely to be caused by your application, seek assistance from your Language Environment support personnel.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2068

FOR2069C MTF subtask *subtask-number* **failed during initialization. VS FORTRAN Version 2 Error Number: AFB922I-1**

Explanation: During the initialization of the Fortran multitasking facility (MTF) by Language Environment, subtask *subtask-number* couldn't be started successfully. The message describing this situation is in the message file for the subtask. This message file is the one referenced by the file definition (DD statement or ALLOCATE command) with the ddname FTERR*sss*, where *sss* is the three-digit representation of *subtask-number*.

Programmer response: Correct the problem indicated in the message file for subtask *subtask-number*.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2069

FOR2070I MTF subtask *subtask-number* **abnormally terminated during execution of parallel subroutine** *subroutine-name***. The system completion code was** *system-completion-code***, and the reason code was** *reason-code***. VS FORTRAN Version 2 Error Number: AFB922I-2**

Explanation: The parallel subroutine *subroutine-name* in MTF subtask *subtask-number* ended unexpectedly because of the abnormal termination indicated by the system completion code *completion-code* and reason code *reason-code*. The message describing this abnormal termination is in the message file for the subtask. This message file is the one referenced by the file definition (DD statement or ALLOCATE command) with the ddname FTERR*sss*, where *sss* is the three-digit representation of *subtask-number*.

Programmer response: For the meaning of system completion code *completion-code* and reason code *reason-code*, and for possible corrective actions, refer to *z/OS MVS System Codes*.

System action: The condition FOR2070 is signaled in the main task program during termination of the application.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2070

FOR2071I MTF subtask *subtask-number* **ended during execution of MTF parallel subroutine** *subroutine-name* **because a statement that requested an immediate termination was executed. The subtask return code was** *return-code***. VS FORTRAN Version 2 Error Number: AFB922I-3**

Explanation: The parallel subroutine *subroutine-name* in MTF subtask *subtask-number* ended because of a request to explicitly terminate the application with return code *return-code*. Examples of such requests include the execution either of a STOP statement or of a call to the EXIT or SYSRCX callable service.

Programmer response: Correct the problem indicated either by return code *return-code* or by any other messages the parallel subroutine wrote.

System action: The condition FOR2071 is signaled in the main task program during termination of the application.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2071

FOR2072I MTF subtask *subtask-number* **terminated during execution of MTF parallel subroutine** *subroutine-name* **because of an unhandled condition of severity** *severity***. VS FORTRAN Version 2 Error Number: AFB922I-2**

Explanation: The parallel subroutine *subroutine-name* in MTF subtask *subtask-number* ended unexpectedly because of an unhandled condition of severity *severity*. The message describing this condition is in the message file for the subtask. This message file is the one referenced by the file definition (DD statement or ALLOCATE command) with the ddname FTERR*sss*, where *sss* is the three-digit representation of *subtask-number*.

Programmer response: Correct the problem indicated by the unhandled condition.

System action: The condition FOR2072 is signaled in the main task program during termination of the application.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2072

FOR2073I **MTF subtask** *subtask-number* **abnormally terminated during execution of parallel subroutine** *subroutine-name*. **The user completion code was** *user-completion-code*, **and the reason code was** *reason-code*. **VS FORTRAN Version 2 Error Number: AFB922I-2**

Explanation: The parallel subroutine *subroutine-name* in MTF subtask *subtask-number* ended unexpectedly because of the abnormal termination indicated by the user completion code *completion-code* and reason code *reason-code*. If the parallel subroutine wrote any output on the message file before terminating, that output is in the message file for the subtask. This message file is the one referenced by the file definition (DD statement or ALLOCATE command) with the ddname FTERR*sss*, where *sss* is the three-digit representation of *subtask-number*.

The abnormal termination was requested by executing an assembler language ABEND macro instruction or by calling one of the CEE3ABD, SYSABN, or SYSABD callable services, among others. The exact meaning of the user completion code *completion-code* and reason code *reason-code* depends on the application that requested the abnormal termination. Some form of information about the meaning of these codes should be available to users of that application.

Programmer response: Correct the problem indicated by the user completion code *completion-code* and reason code *reason-code*.

System action: The condition FOR2073 is signaled in the main task program during termination of the application.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2073

FOR2101S **Based on the value given for the SIZE suboption of the VECTOR compile-time option, the program unit was compiled so that it could run only on a machine with a section size of** *comp-section-size*. **However, execution was on a machine with a section size of** *mach-section-size*. **VS FORTRAN Version 2 Error Number: AFB934I**

Explanation: When the program unit was compiled, the VECTOR compile-time option had one of these forms:

VECTOR(... SIZE(LOCAL) ...)
: In this case, the machine on which the program unit was compiled had a section size of *comp-section-size*, and the compiler produced code that could be run only on a machine with this same section size.

VECTOR(... SIZE(*comp-section-size*) ...)
: In this case, the compiler was directed to produce code that could be run only on a machine with a section size of *comp-section-size*.

In either case, because the compiled code was capable of running only on a machine with a section size of *comp-section-size*, it couldn't be run on the machine that had a section size of *mach-section-size*.

Programmer response: If you want the compiled code to be able on run on machines with various section sizes, then compile the program with the following VECTOR compile-time option, which has the IBM-supplied default for the SIZE suboption:

VECTOR(... SIZE(ANY) ...)

When the SIZE(ANY) suboption is used, the code sequences are not as efficient as they would be if the code were targeted only for machines with a specific section size.

If you can ensure that the compiled code will be run only on a machine with the a section size of *mach-section-size*, and if you want the code to be optimized for machines with that section size, then compile the program with the following VECTOR compile-time option:

VECTOR (... SIZE(*mach-section-size*) ...)

If you can ensure that the compiled code will be run only on a machine with the same section size as that on which it is compiled, and if you want the code to be optimized for machines with that section size, then compile the program with the following VECTOR compile-time option:

VECTOR(... SIZE(LOCAL) ...)

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2101

FOR2102S An internal table used to control the allocation of vector spill areas was corrupted and couldn't be used. VS FORTRAN Version 2 Error Number: AFB935I

Explanation: At entry to a program unit that was compiled with the VECTOR compile-time option, a Language Environment routine detected an inconsistency in a table that was supposed to control allocation of storage for vector spill areas, which are areas used to store the contents of vector registers. Most likely the table in virtual storage was overlaid by some routine (but not necessarily by the routine containing the table that was destroyed).

Programmer response: Determine and correct the cause of the overlaid table. In Fortran program units, this is often caused by:

- Using subscripts that reference virtual storage outside the declared bounds of an array
- Referring to variables that are in EQUIVALENCE statements when the variables are declared to overlay too much storage
- Referring to storage that's addressed through a pointer whose value isn't properly established
- In a CALL statement or function reference, providing actual arguments that are not consistent with the dummy arguments declared in the subprogram. The actual arguments could be of the wrong type, rank, or have the wrong array bounds. There could be an incorrect number of actual arguments

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2102

FOR2121C A suboption of the AUTOTASK run-time option was missing. VS FORTRAN Version 2 Error Number: AFB917I-1

Programmer response: If you want to use the Fortran multitasking facility (MTF), provide the AUTOTASK run-time option in the following format:

AUTOTASK(*loadmod*,*numtasks*)

where:

loadmod	Is the name of the parallel subroutine load module; that is, the name of the member in the data set referenced by the file definition with the ddname AUTOTASK.
numtasks	Is the number of tasks to be created by MTF.

If you didn't intend to use MTF, then either remove the AUTOTASK run-time option or specify the NOAUTOTASK run-time option.

System action: The condition is signaled, and the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2121

FOR2130C A construct of the parallel feature of VS FORTRAN Version 2 could not be completed. Parallel programs cannot be executed using Language Environment.

Explanation: The program was compiled with VS FORTRAN Version 2 Release 5 or 6 and was considered to be a parallel program for one or more of these reasons:

- The program contained parallel language constructs.
- The program invoked one of the parallel callable services (PEORIG, PEPOST, PEWAIT, PETERM, PLCOND, PLFREE, PLLOCK, PLORIG, or PLTERM).
- The program was compiled with the PARALLEL compile-time option.

You cannot run a parallel program if it has been link-edited with Language Environment.

Programmer response: If you want to link-edit and run the program with Language Environment, then compile it without the PARALLEL compile-time option, and remove the parallel language constructs and any calls to the parallel callable services.

If you want to continue to run the program as a parallel program, then link-edit it with the VS FORTRAN Version 2 Release 6 library. In this case, don't code anything in the program (including in any related subprograms) that makes use of any of the Language Environment features that aren't in VS FORTRAN Version 2 Release 6. You can then run the program either with the VS FORTRAN Version 2 Release 6 library or with Language Environment.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2130

FOR2131S The program *program-name* was compiled by VS FORTRAN Version 2 compiler with the EC option, which is not supported by Language Environment.

Explanation: The program was compiled with VS FORTRAN Version 2 Release 5 or 6 with the EC compile-time option. This option specified that certain common blocks were be treated as extended common blocks and that their virtual storage was to be allocated in data spaces. You cannot run such a program if it has been link-edited with Language Environment.

Programmer response: If you want to link-edit and run the program with Language Environment, then compile it without the EC compile-time option. Specify the common blocks as dynamic common blocks by giving their names as suboptions of the DC compile-time option. However, unless you can reduce the size of the extended common blocks, this approach won't work if the program and the common blocks won't fit in the primary address space.

If you want to continue to run the program with extended common blocks, then link-edit it with the VS FORTRAN Version 2 Release 6 library. In this case, don't code anything in the program (including in any related subprograms) that makes use of any of the Language Environment features that aren't in VS FORTRAN Version 2 Release 6. You can then run the program either with the VS FORTRAN Version 2 Release 6 library or with Language Environment.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2131

FOR2132S An invalid argument was provided to an INTEGER*8 simulation routine. VS FORTRAN Version 2 Error Number: AFB177I

Explanation: An INTEGER*8 simulation routine was implicitly referenced by the compiled code because of the use of an integer variable of length 8. One of the arguments to this routine had an unexpected value. Most likely the argument or argument list in virtual storage was overlaid by some routine (but not necessarily by the routine containing the argument that was destroyed).

Programmer response: Determine and correct the cause of the overlaid argument. In Fortran program units, this is often caused by:

- Using subscripts that reference virtual storage outside the declared bounds of an array

- Referring to variables that are in EQUIVALENCE statements when the variables are declared to overlay too much storage
- Referring to storage that's addressed through a pointer whose value is not properly established
- In a CALL statement or function reference, providing actual arguments that are not consistent with the dummy arguments declared in the subprogram. The actual arguments could be of the wrong type, rank, or have the wrong array bounds. There could be an incorrect number of actual arguments

If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2132

FOR2133S A program interruption occurred during the simulation of an INTEGER*8 instruction. VS FORTRAN Version 2 Error Number: AFB178I

Explanation: An INTEGER*8 simulation routine in the Fortran library portion of Language Environment was implicitly referenced by the compiled code because of the use of an integer variable of length 8. During the execution of this simulation routine, a program interruption occurred.

Programmer response: Examine the operands involved in any use of integer variables of length 8, and correct any errors that you find. Here are some errors that might have caused the program interruption:

- Using subscripts that reference virtual storage outside the declared bounds of an array
- Referring to variables that are in EQUIVALENCE statements when the variables are declared to overlay too much storage
- Referring to storage that's addressed through a pointer whose value isn't properly established
- In a CALL statement or function reference, providing actual arguments that are not consistent with the dummy arguments declared in the subprogram. The actual arguments could be of the wrong type, rank, or have the wrong array bounds. There could be an incorrect number of actual arguments

If you are unable to resolve the problem, seek assistance from your Language Environment support personnel.

System action: The condition is signaled. If the condition is unhandled, the application is terminated.

Qualifying Data: None

Permissible Resume Actions: None

Symbolic Feedback Code: FOR2133

Chapter 6. PL/I Run-Time Messages

The messages in this topic pertain to PL/I. Each message is followed by an explanation describing the condition that caused the message, a programmer response suggesting how you might prevent the message from occurring again, and a system action indicating how the system responds to the condition that caused the message.

The messages also contain a symbolic feedback code, which represents the first 8 bytes of a 12-byte condition token. You can think of the symbolic feedback code as the nickname for a condition. As such, the symbolic feedback code can be used in user-written condition handlers to screen for a given condition, even if it occurs at different locations in an application.

The messages in this topic contain alphabetic suffixes that have the following meaning:

I	Informational message
W	Warning message
E	Error message
S	Severe error message
C	Critical error message

IBM0004S The program terminated with user code= *user-code*

Explanation: The program terminated with a user code.

Programmer response: Check your program documentation or the program source to determine the reason the code was issued.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM004

IBM0005S The number of files, CONTROLLED variables, or fetched procedures exceeded the limit.

Explanation: The total length of the pseudoregister vector for the program was more than 4096 bytes. Four bytes are used for each file constant, controlled variable, and fetched procedure.

Programmer response: Modify the program so the pseudoregister vector does not exceed 4096 bytes by reducing the number of files or controlled variables used or by restructuring the program into several external procedures.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM005

IBM0006S The program could not be executed because it did not have a main procedure.

Explanation: There are two possible causes for this error:

- An attempt was made to run a program which contained one or more external PL/I procedures. None of the procedures had the MAIN or FETCHABLE option in the PROCEDURE statement.
- An attempt was made to FETCH a load module with entry PLISTART or CEESTART which contained one or more external PL/I procedures. None of the procedures had the MAIN or FETCHABLE option in the PROCEDURE statement.

Programmer response: Ensure that the first external PL/I procedure to be invoked has the MAIN or FETCHABLE option in the PROCEDURE statement. When fetching a load module, follow the instructions on “Link-Editing Fetchable Load Modules” in *z/OS Language Environment Programming Guide*.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM006

IBM0010W An invalid length or address was found in the PLICALLB argument list for ISA, HEAP, or TASKHEAP storage. The length or address was ignored.

Explanation: If the length or address of ISA, HEAP, or TASKHEAP storage is provided, it must be valid and for the length it must be a multiple of 8 bytes and for the address it must be on a double-word boundary.

Programmer response: Check the provided length and/or address to make sure it is valid and follows the rules.

System action: Execution continues.

Symbolic Feedback Code: IBM00A

IBM0011W The ISA or HEAP storage provided in the PLICALLB argument list was above the 16M line but the BELOW suboption of the STACK or HEAP run-time option was in effect. The provided storage was ignored.

Explanation: The location of the user-provided ISA or HEAP storage conflicts with the location in effect in the STACK or HEAP run-time option. The provided storage is ignored but the provided length is still used.

Programmer response: Make sure the location of the provided ISA or HEAP storage agrees with the location in the STACK or HEAP run-time option.

System action: Execution continues.

Symbolic Feedback Code: IBM00B

IBM0020S ONCODE=600. The CONVERSION condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise the CONVERSION condition for which there was no associated ON-unit.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the CONVERSION condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM00K

IBM0021S ONCODE=601. The CONVERSION condition was raised because of unknown source attributes on input.

Explanation: The CONVERSION condition was raised within a GET LIST or GET DATA statement with the FILE option. The attributes of the source data could not be determined.

Example:

```
DCL (A,B) CHAR(14);
GET LIST(A,B);
```

where the input stream contained 'PIG'C, 'DOG'. The condition will be raised when the first item is encountered. The value for ONSOURCE would be: "'PIG'C", and value of ONCHAR would be: "C". The ONCODE associated with this message is 601.

Programmer response: Include a suitable ON-unit in the program to monitor errors in the input data revealed by the CONVERSION condition. Use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to assign a valid value so the program can continue processing. Also, check the input data for correctness before rerunning the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM00L

IBM0022S **ONCODE=602. The CONVERSION condition was raised because of unknown source attributes on input after the TRANSMIT condition was detected.**

Explanation: The CONVERSION condition was raised after an error caused the TRANSMIT condition to be raised. For an example of the conversion error, refer to the explanation given for message IBM0021. The ONCODE associated with this message is 602.

Programmer response: Correct the transmit error. If the conversion error recurs after correcting the transmit error, refer to the steps for conversion errors in message IBM0021.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM00M

IBM0023S **ONCODE=***oncode-value* **The CONVERSION condition was raised because of unknown source attributes.**

Explanation: The CONVERSION condition was raised within a GET LIST STRING or GET DATA STRING statement. For an example of the conversion error, refer to the explanation for message IBM0021.

Programmer response: Follow the steps given for conversion errors in message IBM0021.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM00N

IBM0024S **ONCODE=***oncode-value* **The CONVERSION condition was raised because a conversion error occurred using F-format on input.**

Explanation: An invalid character was detected in an F-format input field. The ONCODEs associated with this message are:

- 603 - GET STRING statement
- 604 - GET FILE statement

Programmer response: Include a suitable ON-unit in the program to monitor errors in the input data that are revealed by the CONVERSION condition. Use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to assign a valid numeric value so the program can continue processing. Also, ensure all input is in the correct format before running the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM00O

IBM0025S **ONCODE=605. The CONVERSION condition was raised because a conversion error occurred using F-format on input after the TRANSMIT condition was detected.**

Explanation: An invalid character was detected in an F-format input field. A transmission error also occurred and may be the cause of the conversion error. The ONCODE associated with this message is 605.

Programmer response: Correct the transmit error. If the conversion error recurs after correcting the transmit error, refer to the steps given for message IBM0024.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM00P

IBM0027S **ONCODE=***oncode-value* **The CONVERSION condition was raised because a conversion error occurred using E-format on input.**

Explanation: An invalid character was detected in an E-format input field. The ONCODEs associated with this message are:

- 606 - GET STRING statement
- 607 - GET FILE statement

Programmer response: Refer to the steps for conversion errors in message IBM0024. Use the ONSOURCE and

ONCHAR built-in functions to identify the error, and the ONSOURCE and ONCHAR pseudovariables to assign a valid value so the program can continue processing.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM00R

IBM0028S ONCODE=608. The CONVERSION condition was raised because a conversion error occurred using E-format on input after the TRANSMIT condition was detected.

Explanation: An invalid character was detected in an E-format input field. A transmission error also occurred and may be the cause of the conversion error. The ONCODE associated with this message is 608.

Programmer response: Correct the transmission error. If the conversion error recurs after correcting the transmission error, refer to the steps for message IBM0024.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM00S

IBM0029S ONCODE=*oncode-value* The CONVERSION condition was raised because a conversion error occurred using B-format on input.

Explanation: An invalid character was detected in a B-format input field. The ONCODEs associated with this message are:

- 609 - GET STRING statement
- 610 - GET FILE statement

Programmer response: Include a suitable ON-unit in the program to monitor errors in the input data that are revealed by the CONVERSION condition. Use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to assign a valid bit character so the program can continue processing. Also, ensure all input is in the correct format before running the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM00T

IBM0031S ONCODE=611. The CONVERSION condition was raised because a conversion error occurred using B-format on input after the TRANSMIT condition was detected.

Explanation: An invalid character was detected in a B-format input field. A transmission error also occurred and may be the cause of the conversion error. The ONCODE associated with this message is 611.

Programmer response: Correct the transmission error. If the conversion error recurs after correcting the transmission error, refer to the steps for message IBM0029.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM00V

IBM0032S ONCODE=612. The CONVERSION condition was raised because a conversion error occurred when converting a character string to an arithmetic value.

Explanation: An invalid character was detected in a character string that was being converted to an arithmetic data type. The ONCODE associated with this message is 612.

Programmer response: If the error is in the conversion of a PL/I source program constant or in the conversion of a character string created while the program is running, correct the source program. Recompile and rerun the program. Use the ONSOURCE and ONCHAR built-in functions to identify the error, and the ONSOURCE and ONCHAR pseudovariables to assign a valid value so the program can continue processing.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM010

IBM0033S ONCODE=613. The CONVERSION condition was raised because a conversion error occurred when converting character to arithmetic on input or output.

Explanation: A character which is invalid for conversion to an arithmetic form was detected in one of the following:

- An arithmetic constant in a list-directed or data-directed item.
- A character constant being converted to an arithmetic form in a list-directed or data-directed item.
- An A-format input field being converted to an arithmetic form.

The ONCODE associated with this message is 613.

Programmer response: Refer to the steps for message IBM0024.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM011

IBM0034S ONCODE=614. The CONVERSION condition was raised because a conversion error occurred when converting from character on input after the TRANSMIT condition was detected.

Explanation: A character is invalid for conversion to an arithmetic form was detected in one of the following:

- An arithmetic constant in a list-directed or data-directed input item.
- A character constant being converted to an arithmetic form in a list-directed or data directed input item.
- An A-format input field being converted to an arithmetic form.

A transmission error also occurred and may be the cause of the conversion error. The ONCODE associated with this message is 614.

Programmer response: Correct the transmission error. If the conversion error recurs after correcting the transmission error, refer to the steps for message IBM0024.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM012

IBM0035S ONCODE=615. The CONVERSION condition was raised because a conversion error occurred when converting from character to bit.

Explanation: An invalid character was detected in a character string that was being converted to a bit string. The ONCODE associated with this message is 615.

Programmer response: If the error is in the conversion of a program constant or in the conversion of a character string created while the program is running, correct the source program. Recompile and rerun the program. Use the ONSOURCE and ONCHAR built-in functions to identify the error, and the ONSOURCE and ONCHAR pseudovariables to assign a valid value so the program can continue processing.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM013

IBM0036S ONCODE=616. The CONVERSION condition was raised because a conversion error occurred when converting character to bit on input or output.

Explanation: A character other than 0 or 1 appeared in one of the following:

- A bit constant in a list-directed or data-directed item
- A character constant being converted to bit form in a list-directed or data-directed item
- An A-format input field being converted to bit form
- A B-format input field (excluding any leading or trailing blanks)

The ONCODE associated with this message is 616.

Programmer response: Refer to the steps for message IBM0035.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM014

IBM0037S ONCODE=617. The CONVERSION condition was raised because a conversion error occurred when converting character to bit on input after the TRANSMIT condition was detected.

Explanation: A character other than 0 or 1 appeared in one of the following:

- A bit constant in a list-directed or data-directed input item
- A character constant being converted to bit form in a list-directed or data-directed input item
- An A-format input field being converted to bit form
- A B-format input field (excluding any leading or trailing blanks)

A transmission error also occurred and may have caused the conversion error. The ONCODE associated with this message is 617.

Programmer response: Correct the transmission error. If the conversion error recurs after correcting the transmission error, refer to the steps for message IBM0024.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM015

IBM0038S ONCODE=618. The CONVERSION condition was raised because a conversion error occurred when converting to a PICTURE character string.

Explanation: A character that did not match the picture specification was detected in a conversion to a PICTURE character string. The ONCODE associated with this message is 618.

Programmer response: Ensure the character string to be converted to a PICTURE character string matches the picture string specification. If necessary, use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to replace an erroneous character with a valid conversion character.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM016

IBM0039S ONCODE=619. The CONVERSION condition was raised because a conversion error occurred when converting to a PICTURE character string on input or output.

Explanation: A character that did not match the picture specification was detected in a STREAM-oriented item that required conversion to a PICTURE character string. The ONCODE associated with this message is 619.

Programmer response: Either ensure all input data to the program is in the correct format or refer to the steps for message IBM0038. These steps ensure the program has adequate error recovery facilities to process any invalid data found in its input and continue processing.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM017

IBM0040S ONCODE=620. The CONVERSION condition was raised because a conversion error occurred when converting to a PICTURE character string on input after the TRANSMIT condition was detected.

Explanation: A character that did not match the picture specification was detected in a STREAM-oriented input item that required conversion to a PICTURE character string. A transmission error also occurred and may be the source of the conversion error. The ONCODE associated with this message is 620.

Programmer response: Correct the transmission error. If the conversion error recurs after correcting the transmission error, refer to the steps for message IBM0039.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM018

IBM0042S **ONCODE=*oncode-value* The CONVERSION condition was raised because a conversion error occurred when converting from PICTURE format on input.**

Explanation: An edit-directed PICTURE format input item contained a character that did not match the picture specification. The ONCODEs associated with this message are:

- 621 - GET STRING statement
- 622 - GET FILE statement

Programmer response: Either ensure all input data to the program is in the correct format before running the program or use the program to check the data. If necessary, use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to replace an erroneous character with a character valid for conversion.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01A

IBM0043S **ONCODE=623. The CONVERSION condition was raised because a conversion error occurred when converting from a PICTURE format on input after the TRANSMIT condition was detected.**

Explanation: An invalid character was detected in a PICTURE format input field. A transmission error also occurred and may be the cause of conversion error. The ONCODE associated with this message is 623.

Programmer response: Correct the transmission error.

Programmer response: If the conversion error recurs after correcting the transmission error, refer to the steps for message IBM0042.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01B

IBM0045S **ONCODE=625. The CONVERSION condition was raised because a conversion error occurred when converting from PICTURE format on input.**

Explanation: An invalid character was detected in a PICTURE format input item. The ONCODE associated with this message is 625.

Programmer response: Either ensure all input data to the program is in the correct format before running the program or use the program to check the data. If necessary, use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to replace an erroneous character with a valid conversion character.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01D

IBM0046S **ONCODE=626. The CONVERSION condition was raised because a conversion error occurred when converting from PICTURE format on input after the TRANSMIT condition was detected.**

Explanation: An invalid character was detected in a PICTURE format input item. A transmission error also occurred and may be the cause of the conversion error. The ONCODE associated with this message is 626.

Programmer response: Correct the transmission error. If the conversion error recurs after correcting the transmission error, refer to the steps for message IBM0045.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01E

IBM0047S **ONCODE=627. The CONVERSION condition was raised because a graphic or mixed character string was encountered in a non-graphic environment.**

Explanation: A graphic ('G') or mixed ('M') string was used as a data value in the expression for the STRING option of a GET statement. The ONCODE associated with this message is 627.

Programmer response: Remove the graphic or mixed string from the expression.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01F

IBM0048S ONCODE=628. The CONVERSION condition was raised because a graphic or mixed character string was encountered in a non-graphic environment on input.

Explanation: A graphic ('G') or mixed ('M') string was detected in an input file that was not declared with the GRAPHIC option in the ENVIRONMENT attribute. The ONCODE associated with this message is 628.

Programmer response: Specify the GRAPHIC option for a file that contains graphic or mixed character strings.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01G

IBM0049S ONCODE=629. The CONVERSION condition was raised because a graphic or mixed character string was encountered in a non-graphic environment on input after the TRANSMIT condition was detected.

Explanation: The CONVERSION condition was raised after an error caused the TRANSMIT condition to be raised. For an example of the conversion error, refer to the explanation for message IBM0048. The ONCODE associated with this message is 629.

Programmer response: Correct the transmission error. If the conversion error recurs after correcting the transmission error, refer to the steps for message IBM0048.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01H

IBM0053S ONCODE=633 The CONVERSION condition was raised because an invalid character was detected in an X, BX, or GX string constant.

Explanation: A character other than a hexadecimal character was detected. Only hexadecimal characters (0-9,a-f,A-F) are allowed in X, BX, and GX string constants. The ONCODE associated with this message is 633.

Programmer response: Include a suitable ON-unit in the program to monitor errors in the input data that are revealed by the CONVERSION condition. Use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to assign a valid hexadecimal character so the program can continue processing. Also ensure all input is in the correct format before executing the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01L

IBM0054S ONCODE=634 The CONVERSION condition was raised because an invalid character was detected in an X, BX, or GX string constant on input.

Explanation: A character other than a hexadecimal character was detected. Only hexadecimal characters (0-9,a-f,A-F) are allowed in X, BX, and GX string constants. The ONCODE associated with this message is 634.

Programmer response: Include a suitable ON-unit in the program to monitor errors in the input data that are revealed by the CONVERSION condition. Use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to assign a valid hexadecimal character so the program can continue processing. Also, ensure all input is in the correct format before executing the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01M

IBM0055S ONCODE=635 The CONVERSION condition was raised because an invalid character was detected in an X, BX, or GX string constant on input after the TRANSMIT condition was detected.

Explanation: A character other than a hexadecimal character was detected. Only hexadecimal characters (0-9,a-f,A-F) are allowed in X, BX, and GX string constants. A transmission error also occurred and may be the source of the conversion error.

Programmer response: Correct the transmission error. If the conversion error recurs after correcting the transmission error, refer to the steps for message IBM0054.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01N

IBM0056S ONCODE=636 The CONVERSION condition was raised because a graphic string contained an invalid character.

Explanation: This condition was raised by the GRAPHIC built-in function. The source was a graphic (DBCS) string and a shift character was detected in it.

Programmer response: Remove the shift characters from the graphic (DBCS) string. ONSOURCE and ONCHAR pseudovariables cannot be used to assign a new value to the string. ERROR is raised if retry is attempted.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01O

IBM0059S ONCODE=639 The CONVERSION condition was raised because a mixed character string contained an invalid character.

Explanation: This condition was raised by the GRAPHIC built-in function. One of the following rules for mixed constants was broken:

- SBCS portions of the constant cannot contain a shift-in.
- Neither byte of a DBCS character can contain a shift code.

Note: In mixed character strings, a shift-in following a DBCS character or following a shift-out causes a transition to single-byte mode. It is impossible for the first byte of a DBCS character in a mixed character string to contain a shift-in.

Programmer response: Ensure mixed character strings contain balanced, unnested shift-out/shift-in pairs. The MPSTR built-in function can be used to check shift-out/shift-in pairs. ONSOURCE and ONCHAR pseudovariables cannot be used to assign a new value to the string. ERROR is raised if retry is attempted.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01R

IBM0060S ONCODE=667. The CONVERSION condition was raised because there was no SBCS equivalent in the GRAPHIC conversion to character.

Explanation: This condition is raised during an attempt to convert a GRAPHIC string, containing ASCII DBCS characters, that represents a character value. The string contained a DBCS character for which there is no equivalent SBCS character. The ONCODE associated with this message is 667.

Programmer response: Modify your program to ensure such strings contain only valid ASCII DBCS characters. Use the ONSOURCE pseudovariable to assign a valid GRAPHIC string to the ONSOURCE built-in function to allow the conversion to be retried.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01S

IBM0061S ONCODE=668. The CONVERSION condition was raised because there was no SBCS equivalent in the GRAPHIC conversion to character on input.

Explanation: This condition is raised during an attempt to convert a GRAPHIC string in an input file, containing ASCII DBCS character, that represents a character value. The string contained a DBCS character for which there is no equivalent SBCS character. The ONCODE associated with this message is 668.

Programmer response: Modify your program to ensure such strings contain only valid ASCII DBCS characters. Use the ONSOURCE pseudovariable to assign a valid GRAPHIC string to the ONSOURCE built-in function to allow the conversion to be retried.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01T

IBM0062S **ONCODE=669. The CONVERSION condition was raised because there was no SBCS equivalent in the GRAPHIC conversion to character on input after the TRANSMIT condition was detected.**

Explanation: The CONVERSION condition was raised after an error caused the TRANSMIT condition to be raised. For an example of the conversion error, see the explanation given for message IBM0061. The ONCODE associated with this message is 669.

Programmer response: If the conversion error recurs after eliminating the transmission error, take the steps given for message IBM0061.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM01U

IBM0092I **PL/I PLIDUMP was called with Traceback (T) option.**

Explanation: PLIDUMP was called with the T option.

Programmer response: No programmer response is necessary.

System action: No system action is performed.

Symbolic Feedback Code: IBM02S

IBM0100W **ONCODE=***oncode-value* **The NAME condition was raised by a SIGNAL statement (***FILE= or ONFILE= file-name***).**

Explanation: The program contained a SIGNAL statement to raise the NAME condition for which there was no associated ON-unit. The ONCODE associated with this message is 10.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the NAME condition in the program.

System action: Execution continues with the next sequential statement.

Symbolic Feedback Code: IBM034

IBM0101W **ONCODE=***oncode-value* **The NAME condition was raised because an invalid element-variable in a STREAM item was encountered during a GET FILE DATA statement (***FILE= or ONFILE= file-name***).**

Explanation: One of the following conditions was detected:

- An identifier in the input stream had no counterpart in the data list of the GET statement, or the GET statement had no data list and an unknown identifier was encountered in the stream.
- Invalid blank characters were found within an identifier in the input stream.
- The name field or part of a qualified name was omitted.
- There were more than 256 characters in a fully-qualified name.
- Blanks were found within an array subscript other than between the optional sign and the decimal digits.
- An array subscript was missing or indicated too many dimensions.
- A value in a subscript was not a decimal digit.
- The subscript was beyond the declared range of subscripts for a particular array.
- The left-parenthesis was missing after the name of an array.
- A character other than "=" or a blank was found after a right-parenthesis that delimits an array subscript in the input stream.
- The end-of-file or a nonblank delimiter was found before "=" in an item in the input stream.

Programmer response: Use the DATAFIELD built-in function in a NAME ON-unit to obtain the invalid data item.

System action: The incorrect data field is ignored and execution of the GET statement continues.

Symbolic Feedback Code: IBM035

IBM0120S ONCODE=*oncode-value* **The RECORD condition was raised by a SIGNAL statement. (***FILE= or ONFILE= file-name***).**

Explanation: The program contained a SIGNAL statement to raise the RECORD condition for which there was no associated ON-unit.

Programmer response: Supply an ON-unit for the RECORD condition or remove the SIGNAL statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM03O

IBM0121S ONCODE=*oncode-value* **The RECORD condition was raised because the length of the record variable was less than the record length (***FILE= or ONFILE= file-name***).**

Explanation: This message was produced for records that were longer than the associated PL/I variable.

1. For a READ statement, the record was truncated to the length of the variable in the INTO option.
2. For a LOCATE statement (F-format records only), a buffer was not allocated.
3. For a WRITE statement (F-format records only), the record was transmitted with the appropriate number of padding bytes added to equal the length of the record on the data set. The contents of the padding bytes were undefined.
4. For a REWRITE statement, the record was replaced by the shorter record with the appropriate number of padding bytes added to equal the length of the record on the data set. The contents of the padding bytes were undefined.

Programmer response: Either supply an ON-unit for the RECORD condition so the program can continue running, or modify the program to make the length of the record variable the same as the length of the records on the data set. Refer to the language reference manual for this compiler for details of how such records are handled when the RECORD condition is raised.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM03P

IBM0122S ONCODE=*oncode-value* **The RECORD condition was raised because the length of the record variable was greater than the record length (***FILE= or ONFILE= file-name***).**

Explanation: This message was produced for records that were shorter than the associated PL/I variable.

1. For the READ statement using F-format records and a fixed-length variable in the INTO option, the excess bytes in the variable were undefined.
2. For a LOCATE statement, where the maximum length of the records was less than the length of the PL/I variable, the buffer was not allocated.
3. For a WRITE statement, the variable in the FROM option was longer than the maximum length of the records, and was truncated to the maximum record length.
4. For a REWRITE statement, the variable in the FROM option was longer than the record it was to replace, and was truncated to the length of this record.

Programmer response: Either supply an ON-unit for the RECORD condition so the program can continue running, or modify the program to make the length of the record variable the same as the length of the records on the data set. Refer to the language reference manual for this compiler for details of how such records are handled when the RECORD condition is raised.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM03Q

IBM0123S ONCODE=*oncode-value* **The RECORD condition was raised because the WRITE or LOCATE variable had a zero length (***FILE= or ONFILE= file-name***).**

Explanation: A WRITE or REWRITE statement attempted to transmit a record variable of zero length, or a LOCATE statement attempted to obtain buffer space for a zero length record variable.

Programmer response: Ensure the varying-length string used as a record variable is not a null string when the WRITE, REWRITE or LOCATE statement is run.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM03R

IBM0124S **ONCODE=24 The RECORD condition was raised because a zero length record was read from a Regional data set (***FILE= or ONFILE= file-name***).**

Explanation: A record of zero length was read from a REGIONAL data set associated with a DIRECT file. A zero-length record on a direct-access device indicates the end of the data set. However, this message is generated only if the data set was created incorrectly. The ONCODE associated with this message is 24.

Programmer response: Ensure the data set is created correctly as a regional data set. If necessary, recreate the data set and ensure the record is accessed with a valid key.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM03S

IBM0125S **ONCODE=***oncode-value* **The RECORD condition was raised because a WRITE or LOCATE area was too short to contain the embedded string (***FILE= or ONFILE=file-name***).**

Explanation: A record variable was too short to contain the data set embedded key. Either a WRITE or REWRITE statement attempted to transmit the record variable or a LOCATE statement attempted to allocate buffer space for the record variable. For a WRITE or REWRITE statement, no transmission takes place. For a LOCATE statement, a buffer is not allocated.

Programmer response: Ensure the record variable is long enough to contain the data set embedded key and the key is valid.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM03T

IBM0140S **ONCODE=40. The TRANSMIT condition was raised by a SIGNAL statement (***FILE= or ONFILE= file-name***).**

Explanation: The program contained a SIGNAL statement to raise the TRANSMIT condition for which there was no associated ON-unit. The ONCODE associated with this message is 40.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the TRANSMIT condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM04C

IBM0141S **ONCODE=***oncode-value* **The TRANSMIT condition was raised because of an uncorrectable error in output (***FILE= or ONFILE=file-name***).**

Explanation: Data management routines detected an uncorrectable error while transmitting output data between main storage and an external storage device. The condition was raised on the completion of a WRITE, REWRITE or LOCATE statement. For BUFFERED files, this condition can be raised only after executing several I/O statements following the processing of an OUTPUT file. The outfile can not be associated with a unit record device. Processing of an UPDATE file can continue. For INDEXED data sets, the condition can occur while searching through the indexes or tracing an overflow record. The ONCODEs associated with this message are:

- 41 output data set
- 42 input data set

Programmer response: If the error recurs, obtain a dump of the input/output buffer areas by using PLIDUMP in a TRANSMIT ON-unit. Refer to *z/OS Language Environment Programming Guide* for details of PLIDUMP. The resultant output, together with all relevant listings and data sets, should be preserved for later study by IBM.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM04D

IBM0142S ONCODE=42. The TRANSMIT condition was raised because of an uncorrectable error in input (*FILE= or ONFILE= file-name*).

Explanation: Data management routines detected an uncorrectable error while transmitting input data between main storage and an external storage device. If the block contains VS-format records, the error is raised once only for the block. Otherwise, the condition is raised on the completion of a READ or REWRITE statement for each record in the block that contains the error and for every item transmitted by GET statements from a block that contains the error. The contents of the record or data item are undefined. However, processing of subsequent records in the input file can be continued. For INDEXED data sets, the condition can be raised while searching the indexes or tracing an overflow record. The ONCODE associated with this message is 42.

Programmer response: If the error recurs, obtain a dump of the input/output buffers by using PLIDUMP in a TRANSMIT ON-unit. Refer to *z/OS Language Environment Programming Guide* for details of PLIDUMP. Save the PLIDUMP output and all relevant listings and data sets for later study by IBM.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM04E

IBM0143S ONCODE=*oncode-value* The TRANSMIT condition was raised because of unreadable OMR data (*FILE= or ONFILE= file-name*).

Explanation: One or more OMR columns contained a marginal mark, weak mark or poor erasure that could not be read. The condition is raised on completion of the READ operation for the information. An X'3F' character is substituted for unreadable characters, and also put in the last byte of the record. The ONCODE associated with this message is 42.

Programmer response: Replace the information that caused the TRANSMIT condition to be raised. Ensure the data on the information is readable by the OMR.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM04F

IBM0144S ONCODE=*oncode-value* The TRANSMIT condition was raised because of a write error in the index set (*FILE= or ONFILE=file-name*).

Explanation: Data management detected a physical error while attempting to write on the index set of a VSAM KSDS. The condition is raised on the completion of a WRITE, REWRITE, LOCATE or DELETE statement. No further processing of an OUTPUT file can occur. Processing of an UPDATE file can continue. The ONCODE associated with this message is 43.

Programmer response: Check the DASD on which the data set is being written for errors.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM04G

IBM0145S ONCODE=*oncode-value* The TRANSMIT condition was raised because of a read error in the index set (*FILE= or ONFILE=file-name*).

Explanation: Data management detected a physical error while attempting to read from the index set of a VSAM KSDS. The condition is raised on the completion of a READ, WRITE, REWRITE, LOCATE or DELETE statement. No further processing of an OUTPUT file can occur. Processing of an UPDATE file can continue. If the error occurs on a READ statement, no data is transferred to the record variable. For sequential access, data set positioning can be lost, causing a subsequent READ without KEY to raise ERROR. Refer to message IBM0831 for information on sequential access errors. The ONCODE associated with this message is 44.

Programmer response: Check the DASD on which the data set resides for errors. If more research is required, consult with the system programmer.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM04H

IBM0146S ONCODE=*oncode-value* **The TRANSMIT condition was raised because of a write error in the sequence set (***FILE= or ONFILE= file-name***).**

Explanation: Data management detected a physical error while attempting to write on the sequence set of a VSAM KSDS. The condition is raised on the completion of a WRITE, REWRITE, LOCATE or DELETE statement. No further processing of an OUTPUT file can occur. Processing of an UPDATE file can continue. The ONCODE associated with this message is 45.

Programmer response: Check the DASD on which the data set is being written for error. Also, consult with the system programmer.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM04I

IBM0147S ONCODE=*oncode-value* **The TRANSMIT condition was raised because of a read error in the sequence set (***FILE= or ONFILE= file-name***).**

Explanation: Data management detected a physical error while attempting to read from the sequence set of a VSAM KSDS. The condition is raised on the completion of a READ, WRITE, REWRITE, LOCATE or DELETE statement. No further processing of an OUTPUT file can occur. Processing of an UPDATE file can continue. If the error occurs on a READ statement, no data is transferred to the record variable. For sequential access, data set positioning can be lost, causing a subsequent READ without KEY to raise ERROR. Refer to message IBM0831 for sequential access errors. The ONCODE associated with this message is 46.

Programmer response: Check the DASD on which the data set resides for errors. Also, consult with the system programmer.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM04J

IBM0160S ONCODE=*oncode-value* **The KEY condition was raised by a SIGNAL statement (***FILE= or ONFILE= file-name***).**

Explanation: The program contained a SIGNAL statement to raise the KEY condition for which there was no associated ON-unit. The ONCODE associated with this message is 50.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the KEY condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM050

IBM0161S ONCODE=*oncode-value* **The KEY condition was raised because the specified key could not be found (***FILE= or ONFILE= file-name***).**

Explanation: A READ, REWRITE or DELETE statement specified a recorded key which could not be found on the data set. In the case of an INDEXED data set, the key in error was either higher than the highest level index or the record was not in the prime area or the overflow areas of the data set. In the case of a DIRECT file associated with a data set with REGIONAL organization, the key in error was not in the specified region or within the search limit defined by the LIMCT subparameter of the DCB parameter. The ONCODE associated with this message is 51.

Programmer response: Determine why the key was incorrect and modify the program or the data set to correct the error. Use of the ONKEY built-in function in a KEY ON-unit will aid in determining the value of the erroneous key.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM051

IBM0162S **ONCODE=***oncode-value* **The KEY condition was raised because the specified key was already in use in data set (***FILE= or ONFILE= file-name***).**

Explanation: In the case of data set with INDEXED organization, an attempt was made to transmit a keyed record to a data set that already held a record with the same key. In the case of a data set with REGIONAL(1) or REGIONAL(2) organization that was being created sequentially, an attempt was made to transmit a record to a region that already contains a record. The ONCODE associated with this message is 52.

Programmer response: Either check the validity of the data that is being processed before running the program or use the program to check the data. Use of the ONKEY built-in function in a KEY ON-unit can aid in identifying an erroneous key, correcting it, and allowing processing to continue normally.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM052

IBM0163S **ONCODE=***oncode-value* **The KEY condition was raised because the specified key was less than the value of the previous key (***FILE= or ONFILE= file-name***).**

Explanation: A key with a value that was less than the value of the preceding key was detected during the creation or extension of an INDEXED or REGIONAL SEQUENTIAL data set. The ONCODE associated with this message is 53.

Programmer response: Ensure the records written onto an INDEXED or REGIONAL data set that is being created or extended are in the correct ascending key sequence order. Also, use a KEY ON-unit to comment on the error and, where possible, allow processing to continue normally.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM053

IBM0164S **ONCODE=***oncode-value* **The KEY condition was raised because the specified key could not be converted to valid data (***FILE= or ONFILE= file-name***).**

Explanation: A WRITE, READ, REWRITE, DELETE or LOCATE statement for a REGIONAL data set specified a key with a invalid character-string value. Invalid values consist entirely of blanks, contain characters other than 0-9, or a have blank as part of the region number. The ONCODE associated with this message is 54.

Programmer response: Ensure the key is in the correct format. If necessary, use the ONKEY built-in function in a KEY ON-unit to identify the erroneous key. The ON-unit can be used to report any such errors and allow processing to continue. Records associated with the erroneous keys can be transmitted in a subsequent run if the keys have been corrected.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM054

IBM0165S **ONCODE=***oncode-value* **The KEY condition was raised because the specified key was invalid (***FILE= or ONFILE= file-name***).**

Explanation: For an INDEXED data set, either the KEY or the KEYFROM expression was a null string or an attempt was made to rewrite a record with the embedded key of the replacement record not equal to the record to be overwritten. For a REGIONAL data set, the key specified was a null string or a string commencing with ’11111111’B. The ONCODE associated with this message is 55.

Programmer response: Refer to the steps for message IBM0165.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM055

IBM0166S ONCODE=*oncode-value* The KEY condition was raised because the key specifies a position outside the Regional data set (*FILE= or ONFILE= file-name*).

Explanation: A WRITE, READ, REWRITE or DELETE statement specified a key whose relative record or track value exceeded the number of records or tracks respectively for the REGIONAL data set. The ONCODE associated with this message is 56.

Programmer response: Refer to the steps for message IBM0164.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM056

IBM0167S ONCODE=*oncode-value* The KEY condition was raised because space was not available to add a keyed record (*FILE= or ONFILE= file-name*).

Explanation: For a SEQUENTIAL file associated with an INDEXED data set, an attempt was made to write or locate a record during the creation or extension of such a data set when the space allocated to the data set was full. For a DIRECT file associated with an INDEXED data set, space in overflow areas was unable to accept the overflow record. This was caused by the insertion of a new record by a WRITE statement. For a DIRECT file associated with a REGIONAL data set, space was unavailable to add the record in the specified limit of search as specified in the LIMCT subparameter of the DCB parameter. Note that the data set is not necessarily full. The ONCODE associated with this message is 57.

Programmer response: Use the ONKEY built-in function to identify the key value that caused the error. If the key is in error, correct it and continue the job from the point reached when the error occurred. If the key is correct, organize the data set so the rejected record can be accessed.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM057

IBM0168S ONCODE=*oncode-value* The KEY condition was raised because the KEYFROM value was outside the KEYRANGE(s) defined for the data set (*FILE= or ONFILE= file-name*).

Explanation: A WRITE or LOCATE statement specified a key with a value outside the key ranges defined for the data set (VSAM KSDS). The ONCODE associated with this message is 58.

Programmer response: Use the ONKEY built-in function to identify the key value that caused the error and correct the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM058

IBM0180S ONCODE=*oncode-value* The ENDFILE condition was raised by a SIGNAL statement (*FILE= or ONFILE= file-name*).

Explanation: The program contained a SIGNAL statement to raise the ENDFILE condition for which there was no associated ON-unit. The ONCODE associated with this message is 70.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the ENDFILE condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM05K

IBM0181S ONCODE=*oncode-value* The ENDFILE condition was raised (*FILE= or ONFILE= file-name*).

Explanation: The end of an input file was detected. The ONCODE associated with this message is 70.

Programmer response: Include an ON-unit for the ENDFILE condition for each input file in the program to handle the end-of-file processing.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM05L

IBM0182S ONCODE=*oncode-value* **The ENDFILE condition was raised because an end-of-file was previously encountered in STREAM input (***FILE= or ONFILE= file-name***).**

Explanation: The ENDFILE condition was raised when the file mark was encountered but an attempt was made to read beyond the end of the file. Either an ENDFILE ON-unit was run and an attempt was made to read the file or the end-of-file mark was encountered between items in the data list of the current GET statement. The ONCODE associated with this message is 70.

Programmer response: If the program contains an ENDFILE ON-unit, ensure the program does not attempt to read the file after the ENDFILE condition is raised. If the error occurred while a GET statement with two or more items in the data list is running, ensure the GET statement can complete by providing sufficient data items before the end-of-file mark is encountered.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM05M

IBM0190W The ENDPAGE condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise the ENDPAGE condition. The message for this condition is never issued by PL/I.

Programmer response: None.

System action: None.

Symbolic Feedback Code: IBM05U

IBM0191W The ENDPAGE condition was raised.

Explanation: A PUT statement resulted in an attempt to start a new line beyond the limit specified for the current page. The message for this condition is never issued by PL/I.

Programmer response: None.

System action: None.

Symbolic Feedback Code: IBM05V

IBM0195W The PENDING condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise the PENDING condition. The message for this condition is never issued by PL/I.

Programmer response: None.

System action: None.

Symbolic Feedback Code: IBM063

IBM0196W The PENDING condition was raised.

Explanation: An attempt was made to read a record for a TRANSIENT INPUT file that was temporarily unavailable. The message for this condition is never issued by PL/I.

Programmer response: None.

System action: None.

Symbolic Feedback Code: IBM064

IBM0200S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised by a SIGNAL statement (***FILE= or ONFILE= file-name***).**

Explanation: The program contained a SIGNAL statement to raise the UNDEFINEDFILE condition for which there was no associated ON-unit. The ONCODE associated with this message is 80.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the UNDEFINEDFILE condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM068

IBM0201S ONCODE=81 The UNDEFINEDFILE condition was raised because of conflicting DECLARE and OPEN attributes (*FILE= or ONFILE= file-name***).**

Explanation: An attribute in an OPEN statement conflicted with an attribute in a DECLARE statement. The attributes may have been written explicitly or implied by other attributes. For example, DIRECT implies KEYED. Also, some RECORD input/output statements imply file attributes in an implicit OPEN statement. For example, LOCATE implies RECORD OUTPUT BUFFERED SEQUENTIAL. Conflicting attributes are:

BACKWARDS	STREAM, OUTPUT/UPDATE, DIRECT, KEYED, EXCLUSIVE, PRINT, TRANSIENT
BUFFERED	STREAM, UNBUFFERED, PRINT
DIRECT	STREAM, SEQUENTIAL, BACKWARDS, PRINT, TRANSIENT
EXCLUSIVE	STREAM, INPUT/OUTPUT, SEQUENTIAL, BACKWARDS, PRINT, TRANSIENT
INPUT	OUTPUT/UPDATE, EXCLUSIVE, PRINT
KEYED	STREAM, BACKWARDS, PRINT
OUTPUT	INPUT/UPDATE, EXCLUSIVE, BACKWARDS
PRINT	RECORD, INPUT/UPDATE, DIRECT/SEQUENTIAL, BUFFERED/UNBUFFERED, KEYED, EXCLUSIVE, BACKWARDS, TRANSIENT
RECORD	STREAM, PRINT
SEQUENTIAL	STREAM, DIRECT, EXCLUSIVE, PRINT, TRANSIENT
STREAM	RECORD, UPDATE, DIRECT/SEQUENTIAL, BUFFERED/UNBUFFERED, KEYED, EXCLUSIVE, BACKWARDS, TRANSIENT
TRANSIENT	STREAM, UPDATE, DIRECT/SEQUENTIAL, EXCLUSIVE, BACKWARDS, PRINT
UNBUFFERED	STREAM, BUFFERED, PRINT
UPDATE	STREAM, INPUT/OUTPUT, BACKWARDS, PRINT, TRANSIENT

Programmer response: Ensure the attributes specified on the DECLARE statement are compatible with the attributes specified on the OPEN statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM069

IBM0202S ONCODE=*oncode-value*** The UNDEFINEDFILE condition was raised because the device type conflicted with file attributes (***FILE= or ONFILE= file-name***).**

Explanation: A conflict between the device type and the file attributes was detected. For example, a file with the UPDATE attribute cannot be associated with a paper tape reader, a printer, or a magnetic-tape device. The ONCODE associated with this message is 82.

Programmer response: Ensure the device type and the file attributes are compatible.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06A

IBM0203S ONCODE=*oncode-value*** The UNDEFINEDFILE condition was raised because the BLOCKSIZE was not specified (***FILE= or ONFILE= file-name***).**

Explanation: The blocksize for an output file was not specified. For an output file, the blocksize must be specified in either the ENVIRONMENT attribute or in the DCB parameter of the DD statement or CMS FILEDEF. The ONCODE associated with this message is 83.

Programmer response: For output files, ensure the block size is specified. For input files, ensure the block size is valid.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06B

IBM0204S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because a DD statement or CMS FILEDEF was not used in (***FILE= or ONFILE=file-name***).**

Explanation: The job stream for a file did not contain either a DD statement or a CMS FILEDEF. The job stream must contain a DD statement or a CMS FILEDEF with a ddname that is either the name of the file (if the TITLE option is not specified) or the name provided by the TITLE option. The ONCODE associated with this message is 84.

Programmer response: Specify a DD statement or CMS FILEDEF to associate the file with a physical data set.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06C

IBM0205S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because of an I/O error - the Regional data set could not be formatted (***FILE= or ONFILE=file-name***).**

Explanation: An I/O error prevented the data set from being formatted correctly. When a REGIONAL data set is opened for direct output, data management routines format the data set into specified regions by writing dummy or control records into the data set.

Example:

```
TF: PROC;
OPEN FILE(F) DIRECT OUTPUT;
END;
```

The ONCODE associated with this message is 85.

Programmer response: If the problem recurs, have the direct access device or storage medium checked by a customer engineer.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06D

IBM0206S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because a LINESIZE or PAGESIZE argument was outside the defined limits (***FILE= or ONFILE= file-name***).**

Explanation: The implementation-defined maximum or minimum for the LINESIZE option of the ENVIRONMENT attribute was exceeded. For F-format and U-format records, the maximum is 32,759. For V-format records, the maximum is 32,751. The minimum for V- and F-format records is 1. The minimum for V- format PRINT files is 9. The minimum for V-format non-PRINT files is 10. The ONCODE associated with this message is 86.

Programmer response: Ensure the argument to the LINESIZE option is within the prescribed limits. If the argument is a variable, verify it is a FIXED BINARY (31,0) STATIC variable that was correctly initialized before the file was opened.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06E

IBM0207S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because the key length was not specified (***FILE= or ONFILE= file-name***).**

Explanation: A key length was not specified in either the ENVIRONMENT attribute or the DCB parameter of the associated DD statement.

Programmer response: Specify the key length and rerun the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06F

IBM0208S **ONCODE=*oncode-value* The UNDEFINEDFILE condition was raised because the wrong BLOCKSIZE or record length was specified (*FILE= or ONFILE= file-name*).**

Explanation: One of the following conditions was detected:

1. Block size was less than record length.
2. For FB-format records, block size was not a multiple of record length.
3. For VS-format and VBS-format consecutive files:
   - LRECL=X was specified but RECSIZE was not specified or was invalid in the ENVIRONMENT attribute.
   - The file was opened for update with a specified logical record size exceeding 32,756.
4. For VS-format REGIONAL(3) files, logical record size was greater than block size minus four.
5. FUNC=EO was specified with a record length not equal to 80 or FUNC=CO was specified with a record size not equal to 160.
6. Column binary was specified with a record length not equal to 160 on an output file.
7. FUNC=I (punch interpret) was specified with a record length not equal to 80 (or 81 if control characters are in use).

The ONCODE associated with this message is 87.

Programmer response: The numbered responses below apply to the correspondingly numbered explanations above:

1. Check the block size and record length specified in the BLKSIZE and RECSIZE options of the ENVIRONMENT attribute. If LINESIZE was specified, ensure it is compatible with BLKSIZE.
2. If the argument of either option is a variable, ensure it is FIXED BINARY(31,0) STATIC and has been initialized.
3. To correct this error:
   a. Specify a record size in the ENVIRONMENT attribute or correct its value.
   b. Specify a record size less than 32,757.
4. Specify a record size less than or equal to the block size minus four.
5. If FUNC=EO is specified, ensure the record length is 80. If FUNC=CO is specified, ensure the record length is 160.
6. Ensure the record length is 160 when column binary is specified.
7. If FUNC=I is specified, ensure the record length is 80.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06G

IBM0209S **ONCODE=*oncode-value* The UNDEFINEDFILE condition was raised because of conflicting attributes and file organization specifications (*FILE= or ONFILE= file-name*).**

Explanation: The file organization conflicted with one or more explicit or implicit file attributes. Refer to Table 2 for a list of possible conflicts.

Table 2. File Organization and Conflicting Attributes

Organization	Conflicting Attributes
CONSECUTIVE	DIRECT, EXCLUSIVE, KEYED, TRANSIENT
INDEXED	STREAM, TRANSIENT, DIRECT OUTPUT, OUTPUT without KEYED
REGIONAL	STREAM, TRANSIENT, OUTPUT without KEYED
TP	Non-TRANSIENT

Table 2. File Organization and Conflicting Attributes (continued)

Organization	Conflicting Attributes
VSAM	STREAM, TRANSIENT, BACKWARDS, DIRECT OUTPUT, OUTPUT without KEYED(KSDS), KEYED(ESDS), DIRECT(ESDS), REUSE for other than OUTPUT file, DIRECT with NON-UNIQUE INDEXES
None	KEYED, TRANSIENT

The ONCODE associated with this message is 82.

Programmer response: Ensure the file attributes are compatible with the file organization.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06H

IBM0210S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because the record format was invalid for this file organization (***FILE= or ONFILE= file-name***).**

Explanation: The following combinations of file organization and record format are valid:

Organization	**Record Format**
CONSECUTIVE BUFFERED	All
CONSECUTIVE UNBUFFERED	F, FS, V, D, U
INDEXED	F, FB, V, VB
REGIONAL(1)	F
REGIONAL(2)	F
REGIONAL(3)	F, V, VS, U
TP(M), TP(R)	None

The ONCODE associated with this message is 87.

Programmer response: Change the file declaration so the record format is compatible with the file organization.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06I

IBM0211S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because the record format was not specified (***FILE= or ONFILE= file-name***).**

Explanation: The record format was not specified. A record format must be supplied for a file with the RECORD attribute in either the ENVIRONMENT attribute or in the data set label. The ONCODE associated with this message is 83.

Programmer response: Modify the program to include the record format for the file.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06J

IBM0212S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because the KEYLENGTH was negative or greater than 255 (***FILE= or ONFILE= file-name***).**

Explanation: The KEYLENGTH option of the ENVIRONMENT attribute for this file had an invalid key length greater than 255 or less than zero.

Programmer response: Check the argument of the KEYLENGTH option to ensure it is either a constant or a variable with the attributes FIXED BINARY (31,0) STATIC and value between zero and 255 when the file is opened. If the argument is a variable, ensure it is correctly initialized.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06K

IBM0213S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because an invalid KEYLOC value was detected (***FILE= or ONFILE=file-name***).**

Explanation: One of the following conditions was detected:

1. The offset of the key within a record was invalid. The sum of the KEYLOC value and the key length was greater than the record length.
2. For blocked ISAM files, either KEYLOC was not specified or KEYLOC(0) was specified. Both are invalid.

Programmer response: The two numbered responses below apply to the numbered explanations above.

1. Check the value of the argument to the KEYLOC option. If the argument is a variable, check that it is FIXED BINARY (31,0) STATIC and that it has been correctly initialized.
2. Specify a KEYLOC value that is greater than zero.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06L

IBM0214S **ONCODE=***oncode-value***. The UNDEFINEDFILE condition was raised because of conflicting or invalid environment options** *FILE= or ONFILE=file-name***).**

Explanation: There were conflicting environment options.

Programmer response: Ensure all environment options for the file are compatible. If there are invalid environment options specified, remove or correct them.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06M

IBM0215S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because an invalid BUFOFF value was detected (***FILE= or ONFILE= file-name***). ASCII input data set are in the range 0 thru 99.**

Programmer response: Ensure the value specified in the BUFOFF option is within the range of valid values. If the argument is a variable, also ensure if is correctly initialized.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06N

IBM0219S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because the MODE or FUNC option conflicts with the file attribute (***FILE= or ONFILE= file-name***).**

Explanation: The MODE or FUNC DCB subparameter conflicted with a file attribute. Refer to *z/OS Language Environment Programming Guide* for details of possible conflicts.

Programmer response: Remove the conflicting file attribute, or replace it with one that is compatible with the MODE or FUNC option values. For more information on the MODE and FUNC subparameters of the DCB parameter, refer to *z/OS MVS JCL User's Guide*.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06R

IBM0220S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because the MODE or FUNC option conflicted with the record format (***FILE= or ONFILE= file-name***).**

Explanation: OMR or RCE files, IBM 3525 print files, and IBM 3525 associated files can be F-format only. The ONCODE associated with this message is 88.

Programmer response: Ensure the MODE or FUNC option value is compatible with the record format of the file. For more information on the MODE and FUNC subparameters of the DCB parameter, refer to the *Job Control Language (JCL) User's Guide.*

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06S

IBM0221S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because the device type conflicted with the MODE option (***FILE= or ONFILE= file-name***).**

Explanation: OMR can be used only on an IBM 3505 and RCE on an IBM 3525 device. The ONCODE associated with this message is 88.

Programmer response: Ensure the device type and the MODE option value is compatible.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06T

IBM0222S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because the TOTAL option is invalid with an OMR or associated file (***FILE= or ONFILE= file-name***).**

Explanation: Either the OMR (MODE=EO or MODE=CO) was specified on a file with the TOTAL option, or a device association was specified on a file with the TOTAL option. The ONCODE associated with this message is 88.

Programmer response: Either remove the TOTAL option or modify the MODE option so it is compatible with a file with the TOTAL option.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06U

IBM0223S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because of a conflict between the MODE and FUNC options (***FILE= or ONFILE= file-name***).**

Explanation: Refer to *z/OS Language Environment Programming Guide* for details of possible conflicts. The ONCODE associated with this message is 88.

Programmer response: Ensure the values specified for the MODE and FUNC options are compatible. For more information, refer to *z/OS MVS JCL User's Guide*.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM06V

IBM0225S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because the value of the ENV option conflicted with the actual data set value (***FILE= or ONFILE= file-name***).**

Explanation: For VSAM data sets, the values of KEYLOC, KEYLENGTH and RECSIZE are specified when the data set is defined. If values are specified on any file declarations, they must match the defined values. The ONCODE associated with this message is 91.

Programmer response: Ensure the values of KEYLOC, KEYLENGTH and RECSIZE specified in the program match the defined values.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM071

IBM0226S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because the NCP or STRNO value was not 1 (***FILE= or ONFILE= file-name***).**

Explanation: Either an NCP value greater than one was specified in the ENV attribute or a STRNO value greater than one was specified in the AMP parameter in the DD statement. For VSAM files, only one outstanding operation is allowed. An operation with the EVENT option must be processed before another operation is started.

Programmer response: Ensure the NCP or STRNO value for a VSAM file is one if the EVENT option is involved, or modify the program to use operations without the EVENT option to allow concurrent operations on the data set.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM072

IBM0227S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because the TOTAL option is invalid for ESDS (***FILE= or ONFILE= file-name***).**

Explanation: The specification of TOTAL can cause the compiler to generate in-line code for I/O statements for CONSECUTIVE files. If the data set to be accessed is a VSAM Entry Sequenced Data set (ESDS) this code is invalid. The ONCODE associated with this message is 91.

Programmer response: Remove the TOTAL option from the file declaration.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM073

IBM0228S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because the password was invalid or was not specified (***FILE= or ONFILE= file-name***).**

Explanation: For VSAM data sets defined with a password, ENV (PASSWORD) and the password must be specified in the file declaration. If the password is incorrect or is not specified, a number of attempts will be given to specify the correct password. The number of retries allowed is specified when the data set is defined. If these attempts fail, UNDEFINEDFILE is raised.

Note: If the Authorized Program Facility (APF) is being used, the load module must be authorized.

The ONCODE associated with this message is 89.

Programmer response: Modify the program to include the ENV (PASSWORD) option and the correct password in the file declaration.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM074

IBM0229S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because an entry was not in the VSAM catalog for data set (***FILE= or ONFILE= file-name***).**

Explanation: The ENV(VSAM) was specified for a file, but the data set was not converted from ISAM to VSAM. Before using a VSAM data set, a catalog entry must be created and space allocated for the data set using the access method services DEFINE command. The catalog containing the data set must be specified in a JOBCAT or STEPCAT DD statement (unless it is the master catalog). The ONCODE associated with this message is 92.

Programmer response: Ensure the data set is catalogued and the right catalog is accessed. Also, ensure the data set is a valid VSAM data set.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM075

IBM0230S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because of an I/O error reading the catalog or the volume label (***FILE= or ONFILE= file-name***).**

Explanation: An I/O error prevented the reading of a VSAM catalog or a volume label. The ONCODE associated with this message is 92.

Programmer response: Consult with the system programmer.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM076

IBM0231S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because a timestamp mismatch was detected (***FILE= or ONFILE= file-name***).**

Explanation: For VSAM data sets, the index and data can be updated separately and the time of the latest update of each is recorded. If these times do not match, the integrity of the data is uncertain and an OPEN error will occur. Similarly, the timestamp in the data set catalog record might not match the timestamp on the volume containing the data set. This indicates the extent information in the catalog record might not agree with the extents indicated in the VTOC for the volume. Message IEC161 is displayed on the operator's console and will provide more detail. The ONCODE associated with this message is 92.

Programmer response: Resubmit the job. If the error recurs after resubmitting the job, use PLIDUMP to obtain a storage dump and save all the relevant information for study by IBM.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM077

IBM0232S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because the requested data set was not available (***FILE= or ONFILE= file-name***).**

| **Explanation:** The data set to be accessed either does not exist or was already being used by another program and
| could not be shared. Refer to *z/OS Language Environment Programming Guide* for further information.

Programmer response: Refer to *z/OS Language Environment Programming Guide* for more information on sharing data sets.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM078

IBM0233S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because the data set was not properly closed (***FILE= or ONFILE= file-name***).**

Explanation: The last time the data set was opened the close operation failed, leaving the data set in an unusable state. The ONCODE associated with this message is 92.

Programmer response: Use the access method services VERIFY command to restore the data set to a usable state. Refer to the MVS/DFP Access Method Services manual for details.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM079

IBM0234S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because the data set was never loaded (***FILE= or ONFILE= file-name***).**

Explanation: A file can not be opened for INPUT or UPDATE to access a VSAM data set until one or more records have been loaded into the data set using a SEQUENTIAL OUTPUT file. Once records are loaded into the data set, records can be added using a DIRECT UPDATE file even after all records have been deleted from the data set. The ONCODE associated with this message is 82.

Programmer response: Load the empty data set first. Then proceed with further update/input/delete activity.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM07A

IBM0235S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because of an unidentified error during VSAM open (***FILE= or ONFILE= file-name***). Subcode1=***sc1* **Subcode2=***sc2*

Explanation: The VSAM routines detected an error during the open process which PL/I did not recognize. Subcode1 and Subcode2 provide detailed VSAM diagnostic information. See message IBM0811S for an explanation of these fields. VSAM message IEC161 will also be displayed on the operator's console and will provide more detail.

Programmer response: Use the VSAM diagnostic messages to correct the cause of the error and resubmit the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM07B

IBM0236S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because the operating system was unable to OPEN the file Subcode1= sc1 Subcode2=sc2 (***FILE= or ONFILE= file-name***).**

Explanation: The operating system or access method encountered an error during the open process. Subcode1 indicates why the file could not be opened. Subcode2, if not zero, indicates the return code (in hexadecimal) given by the operating system or access method. Subcode2 information is mainly used by IBM support when diagnosing problems. The meaning of the Subcode1 values are as follows:

1. 50 - A non-existent ISAM file is being opened for input.
2. 51 - An unexpected error occurred when opening an ISAM file. Subcode2 gives the return code from ISAM.
3. 52, 53 - An unexpected error occurred when opening a native or REGIONAL(1) file.
4. 54 - A non-existent BTRIEVE file is being opened for input.
5. 55 - An unexpected error occurred when opening a BTRIEVE file. Subcode2 gives the return code from BTRIEVE.
6. 56 - An unexpected error occurred when opening a DDM file.
7. 57,58 - An unexpected error occurred when opening a DDM sequential, DDM relative or DDM indexed file. Subcode2 gives the return code from DDM.
8. 59 - An attempt was made to open a file that was already open.
9. 60 - A file of invalid type is being opened. An example of this is opening a VSAM file under UNIX System Services. VSAM files are not supported under UNIX System Services.
10. 66 - Open of a VSAM file failed. Subcode2 gives the feedback code.

The ONCODE associated with this message is 93.

Programmer response: For Subcodes 50 and 54, ensure the input file exists. For Subcode 60, ensure the file being opened has a file type that is supported by the operating system under which the program is being run. For all the other subcodes, call IBM Support for assistance.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM07C

IBM0241S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because the REUSE option was specified for a non-reusable data set (***FILE= or ONFILE= file-name***).**

Explanation: The ENVIRONMENT option REUSE can only be specified with VSAM data sets which have been defined as reusable during their creation by access method services. The ONCODE associated with this message is 94.

Programmer response: Remove the REUSE option.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM07H

IBM0242S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because the alternate index path was empty (***FILE= or ONFILE= file-name***).**

Explanation: An alternate index can be emptied by having all of its pointers deleted. An empty alternate index cannot be opened. The ONCODE associated with this message is 95.

Programmer response: Ensure the index is defined before it is built and the right alternate index is used.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM07I

IBM0243S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because an attempt to position the file at the last record failed (***FILE= or ONFILE= file-name***). Subcode1=** *sc1* **Subcode2=** *sc2*

Explanation: When the ENVIRONMENT option BKWD is specified for a file open, the file must be positioned at the last record. If an attempt to position at the last record fails, the file is closed and the UNDEFINEDFILE condition is raised with this message. Subcode1 and Subcode2 provide detailed VSAM diagnostic information. See message IBM0811S for an explanation of these fields. This message is also issued if the data set only consists of deleted records. For this case, the subcodes are zero.

Programmer response: Use the VSAM diagnostic information to correct the cause of the error and resubmit the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM07J

IBM0260S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because of an incorrect environment variable (***FILE= or ONFILE= file-name***).**

| **Explanation:** Either the DD environment variable defining the characteristics of the data set or the name in the
| TITLE option of the OPEN statement was entered incorrectly, contained an invalid option, or was too long. The
| ONCODE associated with this message is 96.

| **Programmer response:** Correct the SET DD command (OS/2 and Windows), the export DD command (AIX and
| UNIX System Services), or the filespec in the TITLE option of the OPEN statement and then rerun your program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM084

IBM0265S **ONCODE=***oncode-value***. The UNDEFINEDFILE condition was raised because the file could not be opened Subcode1=** *sc1* **Subcode2=***sc2* **(***FILE= or ONFILE= file-name***).**

Explanation: The file could not be opened. Subcode1 indicates why the file could not be opened and Subcode2, if not zero, indicates the return code (in hexadecimal) given by the operating system or DDM. Subcode2 information is mainly used by IBM support when diagnosing problems. The meaning of the Subcode1 values are as follows:

- 1, 2 - no RECCOUNT or RECSIZE values were given via the ENVIRONMENT option or the set DD or export DD environment variable.
- 3 - A positioning error occurred for a sequential output file.
- 4 - TYPE(FIXED) was specified for a native file, but the file size was not a multiple of RECSIZE.
- 5, 13 - A positioning error occurred for a regional(1) file.
- 6 to 12 - A positioning error occurred for an output file.
- 21 to 23 - AMTHD(DDM) was specified on the DD environment variable but the DDM loadable component (DUBRUN and DUBLDM on OS/2, or PLI_DDM on AIX) could not be found or could not be accessed on the system.
- 24 - Incorrect extended attribute existed on a DDM file.
- 25 - The ORGANIZATION option of the ENVIRONMENT attribute conflicted with the type of data set (DDM or native).
- 26 - Conflicts exist with the way the file is being used.
- 27 - A composite key was detected with a keyed-opening. Composite keys are acceptable only for non-keyed openings.
- 28 to 30 - A new DDM file could not be created.
- 31 - A positioning error occurred for a DDM file.
- 35 - AMTHD(BTRIEVE) was specified on the DD environment variable but the BTRIEVE loadable component (BTRCALLS) could not be found or could not be accessed on the system.
- 36 - Unexpected error occurred when opening a BTRIEVE file.
- 37 - A new BTRIEVE file could not be created.
- 38 - A positioning error occurred for a BTRIEVE file.

- 40 - AMTHD(ISAM) was specified on the DD environment variable but the ISAM non-multithreading loadable components(IBMOS20F and IBMOS20G on OS/2, or IBMWS20F and IBMWS20G on Windows) or the ISAM multithreading loadable components(IBMOM20F and IBMOM20G on OS/2, or IBMWM20F and IBMWM20G on Windows) could not be found or could not be accessed on the system.
- 41 - Unexpected error occurred when opening an ISAM file.
- 42 - A new ISAM file could not be created.
- 43 - A positioning error occurred for an ISAM file.
- 62 - Query for file information failed for a VSAM file under MVS batch.
- 63 - A non-VSAM file is being opened as a VSAM file under MVS batch.
- 64 - A VSAM file is being opened with an invalid type (that is, the file is not a KSDS, ESDS, or RRDS file).
- 65 - A VSAM file is being opened in a non-MVS batch environment. VSAM files are only supported under MVS batch.
- 67 - A VSAM file is being opened as a non-VSAM file under MVS batch.
- 68 - An invalid VSAM file is opened.
- 69 - Query for file information failed for a native file under MVS batch.
- 70 - Positioning for a VSAM file failed.

The ONCODE associated with this message is 99.

Programmer response: Re-issue the DD environment variable and use the information to correct the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM089

IBM0269S ONCODE=*oncode-value*. The UNDEFINEDFILE condition was raised because the file function conflicted with the DDM data set definition (*FILE= or ONFILE= file_name*).

Explanation: A conflict existed between the I/O functions intended for the file and the functions allowed on the data set. One of the following was detected when attempting to open a file to be accessed by the DDM access method:

- The file was being opened for INPUT but the data set was not **get capable**
- The file was being opened for UPDATE, but the data set was not **insert capable**, **get capable**, **modify capable**, or **delete capable**
- The file was being opened for OUTPUT, but the data set was not **insert capable**

Programmer response: Ensure the correct data set is being referenced and the data set is re-created with an appropriate set of capabilities.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM08D

IBM0280S The ERROR condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise the ERROR condition.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the ERROR condition in the program that transfers control out of the ON-unit with a GO TO statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM08O

IBM0281S A prior condition was promoted to the ERROR condition.

Explanation: This condition was raised by PL/I because the implicit action occurred for a PL/I condition that includes raising the ERROR condition as part of its implicit action.

The message for this condition is never issued, but it can appear in a dump. Note that the message for the prior condition was issued.

Programmer response: Investigate the prior condition that led to the ERROR condition. Remove the cause of that condition, or include an ON-unit for that condition or an ON-unit for the ERROR condition.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM08P

IBM0290S **ONCODE=*oncode-value*. The CONVERSION condition was raised because a conversion from PICTURE format contained an invalid character.**

Explanation: An invalid character was detected in a picture string that was being converted to an arithmetic data type.

Programmer response: If the error is in the conversion of a PL/I source program constant or in the conversion of a picture character string while the program is running, correct the source program, recompile it, and rerun the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM092

IBM0291S **ONCODE=*oncode-value*. The CONVERSION condition was raised because a conversion from PICTURE format contained an invalid character on input or output.**

Explanation: A picture character which was invalid for conversion to an arithmetic form was detected in one of the following:

- An arithmetic constant in a list-directed or data-directed item
- A picture character constant being converted to an arithmetic form in a list-directed or data-directed item
- A PICTURE format input field being converted to an arithmetic form

Programmer response: Include a suitable ON-unit in the program to monitor errors in the input data that are revealed by the CONVERSION condition. Use the ONSOURCE and ONCHAR built-in functions to identify the error, and the ONSOURCE and ONCHAR pseudovariables to assign a valid numeric value so the program can continue running normally. Otherwise, ensure all input is in the correct format before running the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM093

IBM0292S **ONCODE=*oncode-value*. The CONVERSION condition was raised because a conversion from PICTURE format contained an invalid character on input or output after the TRANSMIT condition was detected.**

Explanation: A picture character which was invalid for conversion to an arithmetic form was detected in one of the following:

- An arithmetic constant in a list-directed or data-directed item
- A picture character constant being converted to an arithmetic form in a list-directed or data-directed item
- A PICTURE format input field being converted to an arithmetic form

A transmission error also occurred and may have caused the conversion error.

Programmer response: Correct the transmission error. If the conversion error recurs after the transmission error is corrected, refer to the steps for message IBM0291.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM094

IBM0300S **ONCODE=320 The ZERODIVIDE condition was raised by a SIGNAL statement.**

Explanation: The program contained a SIGNAL statement to raise the ZERODIVIDE condition for which there was no associated ON-unit.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the ZERODIVIDE condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM09C

IBM0301S ONCODE=*oncode-value* The ZERODIVIDE condition was raised.

Explanation: The program attempted to execute a statement in which a value of zero was used as the divisor in a division operation. Alternatively, an overflow occurred during a convert to binary operation.

Programmer response: Either check the data that could produce a zero divisor (or overflow, if doing a convert to binary operation) before running the program or include an ON-unit for the ZERODIVIDE condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM09D

IBM0320W ONCODE=*oncode-value* The UNDERFLOW condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise the UNDERFLOW condition for which there was no associated ON-unit. The ONCODE associated with this message is 330.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the UNDERFLOW condition in the program.

System action: Execution continues with the next sequential statement.

Symbolic Feedback Code: IBM0A0

IBM0321W ONCODE=*oncode-value* The UNDERFLOW condition was raised.

Explanation: The magnitude of a floating-point number was smaller than the allowed minimum.

Programmer response: Either modify the program so that the magnitude of the floating-point number is higher than the minimum allowed, or include an ON-unit for the UNDERFLOW condition in the program.

System action: Execution continues from the point at which the condition was raised.

Symbolic Feedback Code: IBM0A1

IBM0330W The ATTENTION condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise the ATTENTION condition. The message for this condition is never issued by PL/I.

Programmer response: None.

System action: None.

Symbolic Feedback Code: IBM0AA

IBM0340S ONCODE=*oncode-value* The SIZE condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise the SIZE condition for which there was no associated ON-unit. The ONCODE associated with this message is 340.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the SIZE condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0AK

IBM0341S ONCODE=*oncode-value* The SIZE condition was raised in an I/O statement.

Explanation: The high-order (leftmost) significant binary or decimal digits were lost in an input/output operation where the size of the value being transmitted exceeded the declared (or default) size of the data item. The ONCODE associated with this message is 341.

Programmer response: Either modify the program so that the data item is large enough for the value being transmitted or include an ON-unit for the SIZE condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0AL

IBM0342S ONCODE=*oncode-value* The SIZE condition was raised.

Explanation: The high-order (leftmost) significant binary or decimal digits were lost in an assignment to a variable or temporary variable where the size of the value being assigned exceeded the declared (or default) size of the data item. The ONCODE associated with this message is 341.

Programmer response: Either modify the program so that the data item is large enough for the value being assigned to it or include an ON-unit for the SIZE condition to allow processing to continue when the SIZE condition is raised.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0AM

IBM0360W ONCODE=*oncode-value* The STRINGRANGE condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise the STRINGRANGE condition for which there was no associated ON-unit. The ONCODE associated with this message is 350.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the STRINGRANGE condition in the program.

System action: Execution continues with the next sequential statement.

Symbolic Feedback Code: IBM0B8

IBM0361W ONCODE=*oncode-value* The STRINGRANGE condition was raised.

Explanation: In the expression SUBSTR(S,I,J), the substring represented by starting position I for a length of J does not lie wholly within the string S.

Programmer response: Ensure that the values used for I and J are neither less than nor greater than the length of S.

System action: Execution continues with a revised SUBSTR reference. Refer to the Language Reference Manual for details regarding the value of the revised SUBSTR reference.

Symbolic Feedback Code: IBM0B9

IBM0365W The FINISH condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise the FINISH condition. The message for this condition is never issued by PL/I.

Programmer response: None.

System action: None.

Symbolic Feedback Code: IBM0BD

IBM0366S The FINISH condition was raised during a STOP statement.

Explanation: The program contained a STOP statement which caused the FINISH condition to be raised. The message for this condition is never issued by PL/I.

Programmer response: None.

System action: None.

Symbolic Feedback Code: IBM0BE

IBM0367S The FINISH condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise during an EXIT statement which caused the FINISH condition to be raised. The message for this condition is never issued by PL/I.

Programmer response: None.

System action: None.

Symbolic Feedback Code: IBM0BF

IBM0368W The FINISH condition was raised due to a RETURN or END statement in the main procedure.

Explanation: The program completed normally, and as a result the FINISH condition was raised. The message for this condition is never issued by PL/I.

Programmer response: None.

System action: None.

Symbolic Feedback Code: IBM0BG

IBM0369S The FINISH condition was raised after the ERROR condition.

Explanation: The FINISH condition was raised as the normal return action or implicit action for the ERROR condition. The message for this condition is never issued by PL/I.

Programmer response: None.

System action: None.

Symbolic Feedback Code: IBM0BH

IBM0380S ONCODE=*oncode-value* The AREA condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise the AREA condition for which there was no associated ON-unit. The ONCODE associated with this message is 362.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the AREA condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0BS

IBM0381S ONCODE=*oncode-value* The AREA condition was raised because the target area was too small for the AREA assignment.

Explanation: In an assignment of an area variable, the current extent of the area on the right-hand side of the assignment statement was greater than the size of the area to which it was to be assigned. The ONCODE associated with this message is 361.

Programmer response: Modify the program to ensure that the target area is large enough to contain the source area.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0BT

IBM0382S ONCODE=*oncode-value* The AREA condition was raised because of insufficient contiguous space in the area for allocation.

Explanation: Insufficient space was available in the specified area for the allocation. The ONCODE associated with this message is 360.

Programmer response: Provide an ON-unit to allow the allocation to be tried again. If necessary, change the value of the pointer qualifying the reference to the inadequate area so that it points to another area in which the allocation can be tried again.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0BU

IBM0400W ONCODE=*oncode-value* **The CONDITION condition was raised by a SIGNAL statement and the condition** *condition-name* **was signaled.**

Explanation: The program contained a SIGNAL statement to raise the CONDITION condition for which there was no associated ON-unit. The ONCODE associated with this message is 500.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the CONDITION condition in the program.

System action: Execution continues with the next sequential statement.

Symbolic Feedback Code: IBM0CG

IBM0420S ONCODE=*oncode-value* **The SUBSCRIPTRANGE condition was raised by a SIGNAL statement.**

Explanation: The program contained a SIGNAL statement to raise the SUBSCRIPTRANGE condition for which there was no associated ON-unit. The ONCODE associated with this message is 520.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the SUBSCRIPTRANGE condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0D4

IBM0421S ONCODE=*oncode-value***. The SUBSCRIPTRANGE condition was raised.**

Explanation: An array subscript exceeded the declared bound for the array.

Programmer response: In order to ensure that the program can continue processing after encountering a subscript range error, include an ON-unit for this condition which runs a GOTO statement to the appropriate place in the program. Also, recompile the program. Normal return from a SUBSCRIPTRANGE ON-unit will produce this message and raise the error condition. Note that array handling operations are made slower when SUBSCRIPTRANGE is enabled.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0D5

IBM0440W ONCODE=*oncode-value* **The STRINGSIZE condition was raised by a SIGNAL statement.**

Explanation: The program contained a SIGNAL statement to raise the STRINGSIZE condition for which there was no associated ON-unit. The ONCODE associated with this message is 150.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the STRINGSIZE condition in the program.

System action: Execution continues with the next sequential statement.

Symbolic Feedback Code: IBM0DO

IBM0441W ONCODE=*oncode-value* **The STRINGSIZE condition was raised.**

Explanation: A string was assigned to a shorter string, causing right-hand characters or bits in the source string to be truncated.

Programmer response: Determine whether or not truncation of the right-hand characters or bits in the source string is correct. Use an ON-unit to record the relevant data or modify the program as required. Note that string-handling operations are made slower when STRINGSIZE is enabled.

System action: Execution continues from the point at which the condition was raised.

Symbolic Feedback Code: IBM0DP

IBM0442W ONCODE=151 The STRINGSIZE condition was raised. The condition was detected during a mixed character string assignment.

Explanation: This condition was raised by one of the CHAR, GRAPHIC, or MPSTR built-in functions. The target was not long enough to contain the result. This target can be the actual target or a temporary target that is created by the program. This condition may have occurred also due to a mixed character assignment with STRINGSIZE enabled and CHARGRAPHIC in effect for the procedure or block. An MPSTR call is generated in this case.

Programmer response: Determine whether or not truncation of right-hand characters in the result is correct. Use an ON-unit to record the relevant data or modify the program as required.

System action: Execution continues from the point at which the condition was raised.

Symbolic Feedback Code: IBM0DQ

IBM0450S ONCODE=*oncode-value* The STORAGE condition was raised by a SIGNAL statement.

Explanation: The program contained a SIGNAL statement to raise the STORAGE condition for which there was no associated ON-unit.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the STORAGE condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0E2

IBM0451S ONCODE=*oncode-value* The STORAGE condition was raised.

Explanation: There was insufficient storage available to satisfy a request for additional storage. For a storage allocation for a BASED variable, the variable was not allocated and its associated pointer will be undefined. For a storage allocation for a CONTROLLED variable, the controlled variable's generation was not allocated. A reference to the controlled variable will result in the access of a previous generation of the controlled variable (if any).

Programmer response: Attempt to free the allocated storage through a FREE statement or within an ON-unit, or provide necessary steps in the ON-unit to terminate the program without losing pertinent information.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0E3

IBM0460S ONCODE=*oncode-value*. The OVERFLOW was raised by a SIGNAL statement.

Explanation: The OVERFLOW condition was raised by a SIGNAL statement. :xpl.The program contained a SIGNAL statement to raise the OVERFLOW condition for which there was no associated ON-unit. The ONCODE associated with this message is 300.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the OVERFLOW condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0EC

IBM0461S ONCODE=*oncode-value* The OVERFLOW condition was raised.

Explanation: The magnitude of a floating-point number exceeded the allowed maximum.

Programmer response: Modify the program to ensure that the condition does not recur or provide an ON-unit to handle the condition.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0ED

IBM0470S ONCODE=*oncode-value* **The INVALIDOP condition was raised by a SIGNAL statement.**

Explanation: The program contained a SIGNAL statement to raise the INVALIDOP condition for which there was no associated ON-unit.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the INVALIDOP condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0EM

IBM0472S ONCODE=*oncode-value***. The INVALIDOP condition was raised.**

Explanation: One of the following types of floating point processor exceptions occurred:

- Invalid floating point operation exceptions, including the following:
  – Subtraction of two infinities
  – Multiplication of infinity by 0
  – Division of two infinities
  – Division of zero by zero
- Floating point processor stack overflow exception
- Floating point processor stack underflow exception
- Denormalized operand exception
- Precision exception
- Other nonspecific floating point processor exceptions

Continuing execution after an INVALIDOP condition, with or without an INVALIDOP ON-unit, can result in further conditions being raised and termination of the program. Generally, the program should be fixed to prevent INVALIDOP conditions from occurring because the occurrence of the INVALIDOP condition indicates the program has fatal or near-fatal errors.

Programmer response: Either check the data or sequence of floating point instructions which could cause the INVALIDOP condition before running the program or insert an INVALIDOP ON-unit to handle the condition whenever it arises.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0EO

IBM0480S ONCODE=*oncode-value* **The FIXEDOVERFLOW condition was raised by a SIGNAL statement.**

Explanation: The program contained a SIGNAL statement to raise the FIXEDOVERFLOW condition for which there was no associated ON-unit. The ONCODE associated with this message is 310.

Programmer response: Either remove the SIGNAL statement or include an ON-unit for the FIXEDOVERFLOW condition in the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0F0

IBM0482S ONCODE=*oncode-value* **The FIXEDOVERFLOW condition was raised.**

Explanation: The length of the result of a fixed-point arithmetic operation exceeded the allowed maximum.

Programmer response: Modify the program to ensure that the condition does not recur or provide an ON-unit to handle the condition.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0F2

IBM0501S ONCODE=*oncode-value*. Greenwich Mean Time was not available for the RANDOM built-in function.

Explanation: Greenwich Mean Time was not set on the system. The ONCODE associated with this message is 2101.

Programmer response: Greenwich Mean Time needs to be set on the system. Use the OS/2 API DosSetDateTime service to set the time. Refer to the OS/2 Control Programming Reference for details.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0FL

IBM0502S ONCODE=*oncode-value*. An invalid seed value was detected in the RANDOM built-in function.

Explanation: The input seed value was not within the valid range of 0 to 2,147,483,646. The random number was set to —1. The ONCODE associated with this message is 2102.

Programmer response: Correct the seed value to be within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0FM

IBM0503S ONCODE=*oncode-value*. Local time was unavailable.

Explanation: The system clock was not set. The ONCODE associated with this message is 2103.

Programmer response: Set the system clock using the appropriate OS/2 commands or use a program that uses the OS/2 API DosSetDateTime service. Refer to the OS/2 Control Programming Reference for details.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0FN

IBM0504S ONCODE=*oncode-value* The value of Y in SECSTODATE(X,Y), DAYS(X,Y), DAYSTODATE(X,Y), or DATETIME(Y) contained an invalid PICTURE.

Explanation: The character string representing the desired format for the output datetime stamp contained an invalid picture string. The ONCODE associated with this message is 2104.

Programmer response: Correct the format.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0FO

IBM0505S ONCODE=*oncode-value* X in DAYS(X,(Y)) contained an invalid day value.

Explanation: The supplied value for the day parameter was not within the valid range of 15 October 1582 to 31 December 9999. The ONCODE associated with this message is 2105.

Programmer response: Correct the value for the day parameter to be within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0FP

IBM0506S ONCODE=*oncode-value* X in DAYS(X,(Y)) contained an invalid month value.

Explanation: The supplied value for the month parameter was not within the valid range of October 1582 to December 9999. The ONCODE associated with this message is 2106.

Programmer response: Correct the value for the month parameter to be within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBMOFQ

IBM0507S ONCODE=*oncode-value* X in DAYS(X,(Y)) contained an invalid year value.

Explanation: The supplied value for the year parameter was not within the valid range of 1582 to 9999. The ONCODE associated with this message is 2107.

Programmer response: Correct the value for the year parameter to be within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBMOFR

IBM0508S ONCODE=*oncode-value* X in DAYSTODATE(X,(Y)) was outside the supported range.

Explanation: X represents the number of days since 15 October 1582. The valid range is from 1 to 3,074,324. The ONCODE associated with this message is 2108.

Programmer response: Correct the value for X to be within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBMOFS

IBM0509S ONCODE=*oncode-value*. X in SECSTODATE(X,(Y)) was outside the supported range.

Explanation: X represents the number of seconds elapsed since 00:00:00 on 14 October 1582, with 00:00:00.000 15 October 1582 being the first supported date/time, and 23:59:59.999 31 December 9999 being the last supported date/time. The valid range is from 86,400 to 265,621,679,999.999. The ONCODE associated with this message is 2109.

Programmer response: Correct the value for X to be within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0FT

IBM0510S ONCODE=*oncode-value*. X in DAYSTODATE(X,Y) could not be converted to a valid Era.

Explanation: An Era was used in the picture string X specified in the DAYSTODATE reference, but X was outside the supported range of Eras. The ONCODE associated with this message is 2110.

Programmer response: Ensure X contains a valid Lilian day number within the range of supported Eras.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0FU

IBM0511S ONCODE=*oncode-value*. The offset from Greenwich Mean Time was unavailable.

Explanation: The difference between the current local time and the Greenwich Mean Time was not available from the system. The ONCODE associated with this message is 2111.

Programmer response: Ensure that both the Greenwich Mean Time and the local time are set on the system. Use the OS/2 API DosSetDateTime service to set the time. Refer to the OS/2 Control Programming Reference for details.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0FV

IBM0512S ONCODE=*oncode-value* X in SECS(X,Y) or DAYS(X,Y) was outside the supported range.

Explanation: The input date supplied was earlier than 15 October 1582 or later than 31 December 9999. The ONCODE associated with this message is 2112.

Programmer response: Correct the input date to be within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0G0

IBM0513S ONCODE=*oncode-value* X in SECS(X,Y) contained an invalid seconds value.

Explanation: The supplied value for the seconds parameter was not within the valid range of 0 to 59. The ONCODE associated with this message is 2113.

Programmer response: Correct the value for the seconds parameter to be within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0G1

IBM0514S ONCODE=*oncode-value* X in SECS(X,Y) contained an invalid minutes value.

Explanation: The supplied value for the minutes parameter was not within the valid range of 0 to 59. The ONCODE associated with this message is 2114.

Programmer response: Correct the value for the minutes parameter to be within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0G2

IBM0515S ONCODE=*oncode-value* X in SECS(X,Y) contained an invalid hour value.

Explanation: The valid range for the hour parameter is 0 to 23. If the ″AP″ field is present, the valid range is 0 to 12. The ONCODE associated with this message is 2115.

Programmer response: Correct the value for the hour parameter to be within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0G3

IBM0516S ONCODE=*oncode-value* X in DAYS(X,Y)did not match the picture specification.

Explanation: The value of X did not match the format described by the picture specification. For example, non-numeric characters appear where only numeric characters are expected. The ONCODE associated with this message is 2116.

Programmer response: Verify the format of the input data matches the picture string specification.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBMOG4

IBM0517S ONCODE=*oncode-value* X in SECS(X,Y) did not match the picture specification.

Explanation: The value of X did not match the format described by the picture specification. For example, non-numeric characters appear where only numeric characters are expected. The ONCODE associated with this message is 2117.

Programmer response: Verify the format of the input data matches the picture string specification.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0G5

IBM0518S ONCODE=*oncode-value* The date string returned by DAYSTODATE(X,Y) was truncated.

Explanation: The output string was not large enough to contain the formatted date value. The ONCODE associated with this message is 2118.

Programmer response: Ensure the output string is large enough to contain the entire formatted date.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBMOG6

IBM0519S ONCODE=*oncode-value* **The timestamp string returned by DATETIME(X) or SECSTODATE(X,Y) was truncated.**

Explanation: The output string was not large enough to contain the formatted data value. The ONCODE associated with this message is 2119.

Programmer response: Ensure the output string is large enough to contain the entire formatted date.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0G7

IBM0520S ONCODE=*oncode-value* **X in SECSTODATE(X,Y) or DATETIME(X) contained an invalid number-of-seconds value.**

Explanation: An Era was used in the picture string X specified in the SECSTODATE reference, but X was outside the supported range of Eras. The ONCODE associated with this message is 2120.

Programmer response: Ensure X contains a valid number-of-seconds value within the range of supported Eras.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0G8

IBM0521S ONCODE=*oncode-value* **Insufficient data was passed to the DAYS or SECS built-in function.**

Explanation: The picture string passed to the DAYS or SECS built-in function did not contain enough information. The minimum information required is either month, day, and year, or year and Julian day. The ONCODE associated with this message is 2121.

Programmer response: Ensure the input data contains, as a minimum, the year, month, and day, or the year and Julian day.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0G9

IBM0522S ONCODE=*oncode-value* **X in SECS(X,Y) or DAYS(X,Y) contained an invalid Era name.**

Explanation: X did not contain a supported Japanese or Republic of China Era name. The ONCODE associated with this message is 2122.

Programmer response: Ensure X is a valid DBCS string.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0GA

IBM0531S ONCODE=*oncode-value* **Operation exception.**

Explanation: A programmer-related hardware error was detected. The ONCODE associated with this message is 8091.

Programmer response: It is possible that an error in the program has caused part of the instructions that can be run to be overwritten by data. Other possible causes of an operation exception might be an attempt to invoke an external procedure or other routine that was not incorporated into the running program by the linkage editor, or running a branch instruction that is incorrect because a control block had previously been overwritten. Consequently, it is advisable to check the linkage editor diagnostics to ensure that all requested external procedures and subroutines have in fact been incorporated into the running program, and that any overlay phases do not overwrite any phases that are still active.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0GJ

IBM0532S ONCODE=*oncode-value* Privileged operation exception

Explanation: A programmer-related hardware error was detected. The ONCODE associated with this message is 8092.

Programmer response: If the error is not in a non-PL/I routine included in the running program, the PL/I program should be checked for an error that could cause the instructions that run to be overwritten by data that matches a privileged operation.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0GK

IBM0533S ONCODE=*oncode-value* EXECUTE exception

Explanation: A programmer-related hardware error was detected. The ONCODE associated with this message is 8093.

Programmer response: If the error is not in a non-PL/I routine included in the running program, the PL/I program should be checked for an error that could cause the running instruction to be overwritten by data that matches the operation code for the EXECUTE instruction on.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0GL

IBM0534S ONCODE=*oncode-value* Protection exception

Explanation: A programmer-related hardware error was detected. The ONCODE associated with this message is 8094.

Programmer response: If the error is not in a non-PL/I routine included in the running program, the PL/I program should be checked for an error that could cause the address used by the store instruction to be corrupted.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0GM

IBM0535S ONCODE=*oncode-value* Addressing exception

Explanation: A programmer-related hardware error was detected. The ONCODE associated with this message is 8095.

Programmer response: If the error is not in a non-PL/I routine included in the running program, the PL/I program should be checked for an error that could cause the address to be corrupted.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0GN

IBM0536S ONCODE=*oncode-value* Specification exception

Explanation: A programmer-related hardware error was detected. The ONCODE associated with this message is 8096.

Programmer response: If the error is not in a non-PL/I routine included in the running program, the PL/I program should be checked for an error that could cause the operand to be corrupted by overwriting control blocks or sections of running code.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0GO

IBM0537S ONCODE=*oncode-value* Data exception

Explanation: A programmer-related hardware error was detected. The ONCODE associated with this message is 8097.

Programmer response: The PL/I program should be checked for an error such as an operation on a FIXED DECIMAL data item before it has been initialized, or an error which could cause the data item to be overwritten.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0GP

IBM0541S ONCODE=*oncode-value* X in ASIN(X) or ACOS(X) was invalid

Explanation: One of the following conditions was detected:

- ABS(X) was greater than one.

The ONCODEs associated with this message are:

- For real short floating-point arguments:

  1518 Argument greater than one
- For real long floating-point arguments:

  1519 Argument greater than one
- For real extended floating-point arguments:

  1520 Argument greater than one

Programmer response: Ensure X is a real expression where ABS(X) is less than or equal to one.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0GT

IBM0542S ONCODE=*oncode-value* X in ATAN(X) or ATAND(X) was invalid.

Explanation: One of the following conditions was detected:

- The real and imaginary parts of X were equal to (0, +1i) or (0, –1i).

The ONCODEs associated with this message are:

- For complex short floating-point arguments:

  1558 Argument equal to (0,+1i) or (0,-1i)
- For complex long floating-point arguments:

  1559 Argument equal to (0,+1i) or (0,-1i)
- For complex extended floating-point arguments:

  1564 Argument equal to (0,+1i) or (0,-1i)

Programmer response: If X is complex, ensure X is not equal to +1i or –1i.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0GU

IBM0543S ONCODE=*oncode-value* X in ATANH(X) was invalid

Explanation: One of the following conditions occurred::

- ABS(X) was greater than one.

The ONCODEs associated with this message are:

- For real short floating-point arguments:

  1514 Argument greater than one

- For real long floating-point arguments:

  1515 Argument greater than one
- For real extended floating-point arguments:

  1516 Argument greater than one

Programmer response: If X is real, ensure ABS(X) is less than one. If X is complex, ensure X is not equal to +1i or —1i.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0GV

IBM0544S ONCODE=*oncode-value* X in SIN(X), COS(X), SIND(X) or COSD(X) was invalid.

Explanation: One of the following conditions occurred:
- ABS(X) was greater than or equal to K, where K=2**63 for short and long floating-point values, and K=2**64 for extended floating-point values.
- The absolute value of the real part of X was greater than or equal to K, where K=2**63 for complex short and long floating-point values, and K=2**64 for complex extended floating-point values.
- An overflow occurred because the absolute value of the imaginary part of X was greater than K, where K is as follows:
  - 89.76 for complex short floating-point arguments
  - 710.82 for complex long floating-point arguments
  - 11357.56 for complex extended floating-point arguments
- An overflow occurred because the absolute value of the imaginary part of X was greater than I but less than J, and the absolute value of the real part was out of range. The values for I and J are as follows:
  - I = 89.41 and J = 89.76 for complex short floating-point arguments
  - I = 710.47 and J = 710.82 for complex long floating-point arguments
  - I = 11357.21 and J = 11357.56 for complex extended floating-point arguments

The ONCODEs associated with this message are:
- For real short floating-point arguments:

  1506 Argument greater than or equal to limit

  2425 Argument equal to plus or minus limit
- For complex short floating-point arguments:

  1529 Absolute value of the real part of argument greater than or equal to limit
- For real long floating-point arguments:

  1507 Argument greater than or equal to limit

  2426 Argument equal to plus or minus limit
- For complex long floating-point arguments:

  1530 Absolute value of the real part of argument greater than or equal to limit
- For real extended floating-point arguments:

  1517 Argument greater than or equal to limit
- For complex extended floating-point arguments:

  1531 Absolute value of the real part of argument greater than or equal to limit

Programmer response: Ensure X is valid.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0H0

IBM0545S ONCODE=*oncode-value* X in SINH(X) or COSH(X) was invalid.

Explanation: One of the following conditions occurred:

- The absolute value of the imaginary part of X was greater than or equal to K, where K=2**63 for complex short and long floating-point values, and K=2**64 for complex extended floating-point values.
- ABS(X) was greater than 89.41 for X represented as a short floating-point value.
- ABS(X) was greater than or equal to K, where K=710.47 for long floating-point values and K=11357.22 for extended floating-point values.
- An overflow occurred because the absolute value of the real part of X was greater than K, where K is as follows:
  - 89.76 for complex short floating-point arguments
  - 710.82 for complex long floating-point arguments
  - 11357.56 for complex extended floating-point arguments
- An overflow occurred because the absolute value of the real part of X was greater than I but less than J, and the absolute value of the imaginary part was out of range. The values for I and J are as follows:
  - I = 89.41 and J = 89.76 for complex short floating-point arguments
  - I = 710.47 and J = 710.82 for complex long floating-point arguments
  - I = 11357.21 and J = 11357.56 for complex extended floating-point arguments

The ONCODEs associated with this message are:

- For real short floating-point arguments:

  1523 Absolute value of argument greater than limit
- For complex short floating-point arguments:

  1914 Absolute value of the imaginary part of argument greater than or equal to limit
- For real long floating-point arguments:

  1524 Absolute value of argument greater than or equal to limit
- For complex long floating-point arguments:

  1915 Absolute value of the imaginary part of argument greater than or equal to limit
- For real extended floating-point arguments:

  1525 Absolute value of argument greater than or equal to limit
- For complex extended floating-point arguments:

  1916 Absolute value of the imaginary part of argument greater than or equal to limit

Programmer response: Ensure X is valid.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0H1

IBM0547S ONCODE=*oncode-value* X in TAN(X) or TAND(X) was invalid.

Explanation: One of the following conditions occurred:

- ABS(X) was greater than or equal to K, where K=2**63 for short and long floating-point values, and K=2**64 for extended floating-point values.
- The absolute value of the real part of X was greater than or equal to K, where K=2**63 for complex short and long floating-point values, and K=2**64 for complex extended floating-point values.

The ONCODEs associated with this message are:

- For real short floating-point arguments:

  1508 Absolute value of argument greater than or equal to limit
- For complex short floating-point arguments:

  1853 Absolute value of the real part of argument greater than or equal to limit
- For real long floating-point arguments:

1509 Absolute value of argument greater than or equal to limit

- For complex long floating-point arguments:

1854 Absolute value of the real part of argument greater than or equal to limit

- For real extended floating-point arguments:

1522 Absolute value of argument greater than or equal to limit

- For complex extended floating-point arguments:

1855 Absolute value of the real part of argument greater than or equal to limit

Programmer response: Ensure X is valid.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0H3

IBM0548S ONCODE=*oncode-value* X in TANH(X) was invalid.

Explanation: One of the following conditions occurred:

- The absolute value of the imaginary part of X was greater than or equal to K, where K=2**63 for complex short and long floating-point values, and K=2**64 for complex extended floating-point values.
- An overflow occurred because the absolute value of the real part of X was greater than 11357.56.
- An overflow occurred because the absolute value of the real part of X was greater than 11357.21 but less than 11357.56, and the absolute value of the imaginary part was out of range.

The ONCODEs associated with this message are:

- For complex short floating-point arguments:

1574 Absolute value of the imaginary part of argument greater than or equal to limit

- For complex long floating-point arguments:

1575 Absolute value of the imaginary part of argument greater than or equal to limit

- For complex extended floating-point arguments:

1576 Absolute value of the imaginary part of argument greater than or equal to limit

Programmer response: Ensure X is valid.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0H4

IBM0549S ONCODE=*oncode-value* X in ERF(X) was invalid.

Explanation: X was not a valid number.

The ONCODEs associated with this message are:

2177 Real short floating-point arguments

2178 Real long floating-point arguments

2179 Real extended floating-point arguments

Programmer response: Ensure X is valid.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0H5

IBM0550S ONCODE=*oncode-value* X in EXP(X) was invalid.

Explanation: One of the following conditions occurred:

- X was less than K, where K is as follows:
  - –87.33 for short floating-point arguments

- –708.39 for long floating-point arguments
- –11355.13 for extended floating-point arguments

- The absolute value of the imaginary part of X was greater than or equal to K, where K=2**63 for complex short and long floating-point values, and K=2**64 for complex extended floating-point values. :li.An overflow occurred because the real part of X was greater than K, where K is as follows:
  - 89.06 for complex short floating-point arguments
  - 710.12 for complex long floating-point arguments
  - 11356.87 for complex extended floating-point arguments
- An overflow occurred because the real part of X was greater than I but less than J, and the imaginary part was out of range. The values for I and J are as follows:
  - I = 88.73 and J = 89.06 for complex short floating-point arguments
  - I = 709.79 and J = 710.12 for complex long floating-point arguments
  - I = 11357.53 and J = 11356.87 for complex extended floating-point arguments
- X was greater than or equal to K, where K is as follows:
  - 88.73 for short floating-point arguments
  - 709.79 for long floating-point arguments
  - 11356.53 for extended floating-point arguments

The ONCODEs associated with this message are:

- For real short floating-point arguments:

  1565 Argument less than limit

  1611 Argument greater than or equal to limit
- For complex short floating-point arguments:

  1568 Absolute value of the imaginary part of argument greater than or equal to limit
- For real long floating-point arguments:

  1566 Argument less than limit

  1612 Argument greater than or equal to limit
- For complex long floating-point arguments:

  1569 Absolute value of the imaginary part of argument greater than or equal to limit
- For real extended floating-point arguments:

  1567 Argument less than limit

  1613 Argument greater than or equal to limit
- For complex extended floating-point arguments:

  1570 Absolute value of the imaginary part of argument greater than or equal to limit

Programmer response: Ensure X is valid.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0H6

IBM0551S ONCODE=*oncode-value* X in GAMMA(X) or LOGGAMMA(X) was invalid.

Explanation: One of the following conditions occurred:

- X was less than K, where K is as follows:
  - for the built-in function GAMMA:
  - 35.04 for short floating-point arguments
  - 171.62 for long floating-point arguments
  - 1755.54 for extended floating-point arguments
  - for the built-in function LOGGAMMA:

- 4.085E+36 for short floating-point arguments
- 2.559E+305 for long floating-point arguments
- 1.048E+4928 for extended floating-point arguments

- For GAMMA(X), X was less than or equal to zero.
- For LOGGAMMA(X), X was less than zero.
- For GAMMA(X), the calculated result was greater in magnitude than the largest finite number representable in the result data type.

The ONCODEs associated with this message are:

- For real short floating-point arguments:

  1571 Argument greater than limit

  2165 Argument less than or equal to zero
- For real long floating-point arguments:

  1572 Argument greater than limit

  2166 Argument less than or equal to zero
- For real extended floating-point arguments:

  1573 Argument greater than limit

  2164 Argument less than zero

  2167 Argument equal to zero

  2403 Argument less than or equal to minus zero

  2404 Argument equal to zero

Programmer response: If X is numeric, ensure X is greater than zero.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0H7

IBM0552S ONCODE=*oncode-value* X in LOG(X), LOG10(X) or LOG2(X) was invalid.

Explanation: One of the following conditions occurred:

- X was less than or equal to zero.
- A floating point division by zero occurred because X was equal to (0,0i).

The ONCODEs associated with this message are:

- For real short floating-point arguments:

  1504 Argument less than zero

  1577 Argument equal to plur or minus zero
- For complex short floating-point arguments:

  2413 X equal to (0,0i)
- For real long floating-point arguments:

  1505 Argument less than zero

  1578 Argument equal to plus or minus zero
- For complex long floating-point arguments:

  2414 X equal to (0,0i)

Programmer response: If X is real, ensure X is greater than zero. If X is complex, ensure X is not equal to 0 + 0i.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0H8

IBM0553S ONCODE=*oncode-value* The ERFC(X) was invalid.

Explanation: One of the following conditions occurred:

- X was greater than K, where K is as follows:
  - 9.19 for short floating-point arguments
  - 26.54 for long floating-point arguments
  - 106.53 for extended floating-point arguments

The ONCODEs associated with this message are:

- For real short floating-point arguments:

  2171 Argument greater than limit
- For real long floating-point arguments:

  2172 Argument greater than limit
- For real extended floating-point arguments:

  2173 Argument greater than limit

Programmer response: Ensure X is greater than zero.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0H9

IBM0554S ONCODE=*oncode-value* X in SQRT(X) was invalid.

Explanation: One of the following conditions occurred:

- X was less than zero
- X was equal to minus zero.

The ONCODEs associated with this message are:

- For real short floating-point arguments:

  1500 Argument less than zero

  1960 Argument equal to limit
- For real long floating-point arguments:

  1501 Argument less than zero

  1962 Argument equal to limit

Programmer response: Ensure X is greater than zero.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HA

IBM0555S ONCODE=*oncode-value* X in ABS(X) was invalid.

Explanation: The calculated result was greater in magnitude than the largest finite number representable in the result data type.

Programmer response: Ensure X is valid.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HB

IBM0560S ONCODE=*oncode-value* The EVENT variable, as argument to the COMPLETION pseudovariable, was already in use with file *file-name*.

Explanation: The event variable used in this statement was already active and associated with an input/output operation on the named file. The ONCODE associated with this message is 3904.

Programmer response: Modify the program so that the COMPLETION pseudovariable refers to the event variable when it is inactive.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HG

IBM0561S ONCODE=*oncode-value* The TASK variable was already in use with entry *entry-name*.

Explanation: The task variable specified in a CALL statement is already associated with an active task. The named entry denotes the entry point of the task with which the variable is associated. The ONCODE associated with this message is 3901.

Programmer response: Modify the program so that the task variable is uniquely associated with each task in the application.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HH

IBM0562S ONCODE=*oncode-value* The EVENT variable, as argument to the COMPLETION pseudovariable, was already in use with a DISPLAY statement.

Explanation: The event variable used in this statement was already active and associated with a DISPLAY statement.

Programmer response: Modify the program so that the COMPLETION pseudovariable refers to the event variable when it is inactive.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HI

IBM0563S ONCODE=*oncode-value* The EVENT variable was already in use with file *file-name*.

Explanation: The event variable used in this statement was already active and associated with another input/output operation on the named file. The ONCODE associated with this message is 3907.

Programmer response: Modify the program so that the input/output operation refers to another event variable, or include a WAIT statement to prevent the statement from running until the active event is complete.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HJ

IBM0564S ONCODE=*oncode-value* The EVENT variable being assigned was already in use with file *file-name*.

Explanation: An attempt was made to assign a value to an event variable while it was still associated with an input/output operation.

Example:

```
DCL X FILE RECORD INPUT UNBUFFERED
ENV(BLKSIZE(80) RECSIZE(80) F);
DCL Y CHAR(80);
DCL (Z,Z1) EVENT;
READ FILE(X) INTO(Y) EVENT(Z);
Z = Z1;
```

The ONCODE associated with this message is 3906.

Programmer response: Modify the program so that the event variable used as the target in the assignment, or as the argument of the COMPLETION pseudovariable, is not the same event variable associated with an input/output operation. Alternatively, include a WAIT statement to prevent this statement from running until the active event is complete.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HK

IBM0566S **ONCODE=***oncode-value* **The task was not created because the total number of active tasks would exceed the allowable limit.**

Explanation: The request to create a task was not honored because otherwise the total number of concurrently active tasks would exceed the limit set either by the PLITASKCOUNT run-time option or the underlying UNIX System Services installation default for the maximum number of threads.

Programmer response: Increase the number of tasks allowed to be active concurrently or modify the program so that the existing number of tasks is not exceeded.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HM

IBM0567S **ONCODE=***oncode-value* **A WAIT occurred in the ON-unit for the I/O event required for the current task.**

Explanation: A WAIT statement specified an event variable. The completion of the event caused entry to an ON-unit for an I/O condition which contained another WAIT statement for the same event variable as in the original WAIT statement.

Example:

```
DCL F FILE RECORD OUTPUT UNBUFFERED
ENV(BLKSIZE(80) RECSIZE(80) F);
ON RECORD(F) BEGIN;
WAIT(E);
END;
WRITE FILE(F) FROM (X) EVENT(E);
WAIT(E);  (this statement raises the
record condition)
```

The ONCODE associated with this message is 3911.

Programmer response: Remove the WAIT statement from the ON-unit for the input/output condition.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HN

IBM0568S **ONCODE=***oncode-value* **The assigned EVENT variable was already in use with a DISPLAY statement.**

Explanation: The event variable specified as the argument of the COMPLETION built-in function, or used as the target in an assignment, was still associated with a DISPLAY statement.

Example:

```
DCL A CHAR, COMPLETION BUILTIN;
DISPLAY('MESSAGE TO OPERATOR')
REPLY(A) EVENT(E);
COMPLETION(E)='1'B;
```

ONCODEs associated with this message are:

- 3904—event variable as argument to the COMPLETION built-in function
- 3907—event variable is active

Programmer response: Modify the program so that the event variable used as the target in the assignment or as the argument of the COMPLETION pseudovariable is not the same event variable associated with the DISPLAY statement. Or include a WAIT statement to prevent this statement from running until the active event is complete.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HO

IBM0569S ONCODE=*oncode-value* **The assigned EVENT variable was already active and was used with entry** *entry-name***.**

Explanation: An active event variable was specified as the target of an event variable assignment.

Example:

```
DCL (E,E1) EVENT;
CALL P EVENT(E);
E=E1;
P:  PROC;
END;
```

ONCODEs associated with this message are:

3906	event assignment
3907	event variable is active

Programmer response: Either use another inactive event variable, or include a WAIT statement to ensure that this statement is not run until the active event is complete.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HP

IBM0570S ONCODE=*oncode-value* **The EVENT variable was active and was used with entry** *entry-name***.**

Explanation: An active event variable was specified in the EVENT option of an input/output statement.

Programmer response: Either insert a WAIT statement to ensure the event in question is inactive when this statement is run, or if the statement can be run correctly before the currently active event is complete, use another inactive event variable.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HQ

IBM0571S ONCODE=*oncode-value* **The EVENT variable was already used with a DISPLAY statement.**

Explanation: The event variable specified in the statement was already associated with a DISPLAY statement. The ONCODE associated with this message is 3907.

Programmer response: Either use a different event variable or insert a WAIT statement so that the DISPLAY statement is complete before this statement is run.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HR

IBM0572S ONCODE=*oncode-value* **A CALL statement with the TASK, EVENT, or PRIORITY option was found in a PUT FILE (SYSPRINT) statement.**

Explanation: A tasking CALL statement is not allowed from a PUT FILE (SYSPRINT) statement because no programs in the attempted new task via the tasking CALL statement can produce output on SYSPRINT while a PUT statement is running, and task interlock is likely to occur.

Example:

```
DCL X FIXED BIN(15);
PUT LIST(F(X));
F:  PROC(X);
CALL E TASK;
X=5;
```

```
RETURN (X);
END F;
E:  PROC;
END E;
```

The ONCODE associated with this message is 3912.

Programmer response: Remove the tasking CALL statement from the PUT FILE (SYSPRINT) statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HS

IBM0573S **ONCODE=***oncode-value* **The EVENT variable, as argument to the COMPLETION pseudovariable, was already used with entry** *entry-name***.**

Explanation: An active event variable was used as the argument to the COMPLETION pseudovariable. Event variables used as arguments to the COMPLETION pseudovariable must be inactive.

Programmer response: Either use a different event variable for the COMPLETION pseudovariable, or modify the program so that the COMPLETION pseudovariable refers to the event variable when it is inactive.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HT

IBM0575S **ONCODE=***oncode-value* **An attempt was made to invoke a Fortran or COBOL program while another active task had invoked a program of the same language.**

Explanation: If a Fortran or COBOL program has been invoked in a task, a program of the same language can not execute in any other task until the task used to invoke the Fortran or COBOL program terminates. The ONCODE associated with this message is 3914.

Programmer response: Make sure all Fortran or COBOL programs are invoked in one task or control the invocation sequence in such a way that the second Fortran or COBOL program invoked from a task waits until another task which has already invoked a program of the same language terminates.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0HV

IBM0576S **ONCODE=***oncode-value* **An attempt to use a CALL statement with the TASK, EVENT, or PRIORITY option was found in a non-tasking environment.**

Explanation: An attempt was made to create a task when the application was not linked with the multitasking library. The ONCODE associated with this message is 3915.

Programmer response: Remove the tasking option from the CALL statement, or if this is a multitasking application, relink it with the multitasking library.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0I0

IBM0577I **A PL/I multitasking application was found under CICS, IMS, DB2, CMS, pre-initialized environment, or a nested enclave.**

Explanation: PL/I Multitasking Facility is not supported under CICS, IMS, DB2, CMS, pre-initialized environment (both Language Environment-defined and PL/I-defined pre-initialized environment), and a nested enclave.

Programmer response: Do not use the PL/I Multitasking Facility in the above environment, or run the multitasking application under z/OS.

System action: The application is terminated with the 4093-12 Abend.

Symbolic Feedback Code: IBM0I1

IBM0579I **ONCODE=***oncode-value* **The callable service BPX1SYC for the installation default of the maximum number of threads was unsuccessful. The system return code was** *return_code***; the reason code was** *reason_code***.**

Explanation: The callable service BPX1SYC used to query the installation default of the maximum number of threads failed. The system return code and reason code were returned.

Programmer response: Look up the return code and reason code in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, and take the appropriate action. Consult with your system support personnel if necessary.

System action: The application is abnormally terminated with 4093-152 Abend.

Symbolic Feedback Code: IBM0I3

IBM0580S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because an attempt was made to OPEN the MSGFILE(SYSPRINT) file after a subtask had been created.**

Explanation: When the MSGFILE(SYSPRINT) run-time option is specified, you must ensure that the standard SYSPRINT file is opened in the major PL/I task before any subtask is ever created.

Programmer response: Ensure that the above rule has not been violated. One method is to add an OPEN statement for the SYSPRINT file at the start of the major PL/I task before an subtasks are created.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0I4

IBM0581I **The POSIX(ON) run-time option was in effect for a PL/I multitasking application.**

Explanation: The POSIX(ON) run-time option is not supported for a PL/I multitasking application.

Programmer response: Specify the POSIX(OFF) run-time option for a PL/I multitasking environment.

System action: The application is terminated with the 4093-52 Abend.

Symbolic Feedback Code: IBM0I5

IBM0583S **ONCODE=***oncode-value* **The callable service BPX1MPI (mvspauseinit) was unsuccessful. The return code was** *return_code* **and the reason code was** *reason_code***.**

Explanation: The callable service BPX1MPI (mvspauseinit) was called to initialize a wait for a PL/I WAIT or DISPLAY statement for a PL/I multi-tasking program. This service failed with the return code and reason code shown in the message text.

Programmer response: Look up the return code and reason code in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, and take the appropriate action. Consult with your system support personnel if necessary.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0I7

IBM0584S **ONCODE=***oncode-value* **The callable service BPX1MP (mvspause) was unsuccessful. The return code was** *return_code* **and the reason code was** *reason_code***.**

Explanation: The callable service BPX1MP (mvspause) was called to perform a wait for a PL/I WAIT or DISPLAY statement for a PL/I multi-tasking program. This service failed with the return code and reason code shown in the message text.

Programmer response: Look up the return code and reason code in the *z/OS UNIX System Services Programming: Assembler Callable Services Reference* and take the appropriate action. Consult with your system support personnel if necessary.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0I8

IBM0585S ONCODE=*oncode-value* **The callable service BPX1PTB for cancelling a PL/I subtask was unsuccessful. The system return code was** *return_code***, the reason code was** *reason_code***.**

Explanation: The callable service BPX1PTB used to cancel a PL/I subtask during abnormal termination failed. The system return code and reason code were returned.

Programmer response: Look up the return code and reason code in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, and take the appropriate action. Consult with your system support personnel if necessary.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0I9

IBM0586S ONCODE=*oncode-value* **The callable service BPX1ENV for supporting PL/I EXCLUSIVE files was unsuccessful. The system return code was** *return_code***, the reason code was** *reason_code***.**

Explanation: The callable service BPX1ENV used to support the PL/I EXCLUSIVE files failed. The system return code and reason code were returned.

Programmer response: Look up the return code and reason code in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, and take the appropriate action. Consult with your system support personnel if necessary.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0IA

IBM0590S ONCODE=*oncode-value* **The fetchable procedure with entry** *entry-name* **could not be found.**

Explanation: The libraries available when the program was run did not contain a member with a name or alias matching that used in the FETCH statement. The ONCODE associated with this message is 9250.

Programmer response: Ensure that the load module that is to be fetched is accessible at run-time, and that it is stored with the same name or alias as that used in the FETCH statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0IE

IBM0591S ONCODE=*oncode-value* **There was a permanent I/O error while fetching procedure with entry** *entry-name***.**

Explanation: A permanent I/O error occurred while trying to load the module named in the FETCH statement. The ONCODE associated with this message is 9251.

Programmer response: Ensure that the required load module has been incorporated into the appropriate library with proper data set/file attributes, and then rerun the job. If the problem recurs, inform your installation system programmer, who will take the appropriate action.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0IF

IBM0592S ONCODE=*oncode-value* **FETCH/RELEASE is not supported in CMS.**

Explanation: An attempt was made to FETCH or RELEASE another program from a PL/I module that was linked with PL/I Version 2 Release 3 or earlier. The FETCH/RELEASE facility under CMS is only available with Language Environment PL/I. The ERROR condition has been raised. The ONCODE associated with this message is 9252.

Programmer response: Remove the FETCH or RELEASE statement from the application and use the CALL statement instead. Or relink the program with Language Environment PL/I.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0IG

IBM0593W ONCODE=*oncode-value* The CALL PLITEST statement failed because the NOTEST compiler option was in effect.

Explanation: An attempt was made to execute a CALL PLITEST statement in a program that was compiled with the NOTEST option. The debugger can not be invoked when the NOTEST compiler option is in effect.

Programmer response: Re-compile the program with the TEST option, or remove the CALL PLITEST statement(s) from the program.

System action: Processing continues with the next sequential statement. The debugger is not invoked.

Symbolic Feedback Code: IBM0IH

IBM0594S ONCODE=*oncode-value* Under CICS, an attempt was made to FETCH a main procedure from a PL/I routine.

Explanation: Under CICS, using FETCH to dynamically load a PL/I main procedure from a PL/I routine is not supported. The ONCODE associated with this message is 9254.

Programmer response: Remove the FETCH statement and use the EXEC CICS LINK command to create a nested enclave.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0II

IBM0595S ONCODE=9255 An attempt was made to release a load module containing non-PL/I high level language programs.

Explanation: A load module containing non-PL/I high level language programs, such as C, COBOL, or FORTRAN, could not be released by a PL/I RELEASE statement. The load module will be released automatically during the enclave termination. A load module containing PL/I programs and/or Assembler programs only can be released by a PL/I RELEASE statement. The associated ONCODE is 9255.

Programmer response: Remove the RELEASE statement from the program as the load module will be released automatically during the enclave termination.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0IJ

IBM0596S ONCODE=9256 The fetchable procedure could not be released.

Explanation: Either the routine was not previously fetched, or the fetched part containing the routine was no longer in use but could not be released. The ONCODE associated with this message is 9256.

Programmer response: Ensure the name used in the RELEASE statement is correct, and that the routine has been previously fetched. Also, ensure the fetched part containing the routine to be released is accessible at run-time.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0IK

IBM0597S ONCODE=*oncode-value* A subroutine load module using the PLICALLA entry point was fetched.

Explanation: The PLICALLA entry point can only be used for a load module with a PL/I main routine. The ONCODE associated with this message is 9257.

Programmer response: Either Specify OPTIONS(MAIN) and recompile or don’t use the PLICALLA entry point.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0IL

IBM0600S ONCODE=*oncode-value* An E-format specification contained incorrect values in fields W, D, and S.

Explanation: An edit-directed input/output operation for an E-format item was specified incorrectly. The ONCODE associated with this message is 3000.

Programmer response: Correct the E-format item according to the language rules.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0IO

IBM0601S ONCODE=*oncode-value* The value of a W field in an F-format specification was too small.

Explanation: An edit-directed input/output operation for an F-format item was specified with a W-specification that was too small to allow room for the decimal-point when the number of fractional digits was specified as zero. The ONCODE associated with this message is 3001.

Programmer response: Correct the F-format item according to the language rules.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0IP

IBM0604S ONCODE=*oncode-value* An invalid assignment was made to a pictured character string.

Explanation: An attempt was made to assign an invalid data item to a pictured string. A data item which is not a character string cannot be assigned to a pictured character string because it does not match the declared characteristics of the pictured target variable. The ONCODE associated with this message is 3006.

Programmer response: Alter the characteristics either of the source variable or of the target variable so the data item assignment is possible.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0IS

IBM0611S ONCODE=*oncode-value* The F-factor in the PICTURE specification was outside the range of -128 to 127.

Explanation: The picture character F specifies a picture scaling factor for fixed-point decimal numbers. The number of digits following the V picture character minus the integer specified with F was required to be between -128 and 127.

Programmer response: Correct the integer specified with the picture scaling factor F.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0J3

IBM0612S ONCODE=*oncode-value* The PICTURE specification contained an invalid character.

Explanation: The PICTURE specification can contain only A X 9 for character data and only 9 V Z * , . / B S + - $ CR DB Y K E F < > for numeric data. The characters between the insertion characters < > are not affected by this rule.

Programmer response: Ensure the PICTURE specification contains valid data.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0J4

IBM0613S ONCODE=*oncode-value* An invalid character(s) appeared in the F scaling factor.

Explanation: .The picture character F specifies a picture scaling factor for fixed-point decimal numbers. The format is F(#) where # can be any signed integer between -128 and 127 inclusively.

Programmer response: Ensure the value specified for the scaling factor is a valid fixed-point decimal number that is within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0J5

IBM0614S ONCODE=*oncode-value* An invalid character PICTURE specification was used.

Explanation: The PICTURE specification can contain only A X 9 for character data. Other characters are not permitted.

Programmer response: Ensure the PICTURE specification contains valid data.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0J6

IBM0615S ONCODE=*oncode-value* An invalid precision value was specified. The length was corrected automatically.

Explanation: The number of digits for the precision field within a numeric data PICTURE specification must be between one and fifteen digits. The invalid precision specification is corrected automatically.

Programmer response: Ensure the value specified for the precision is within the supported range.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0J7

IBM0616S ONCODE=*oncode-value* The characters T, I or R appeared too often in the PICTURE specification.

Explanation: T, I, R are the overpunch characters in a PICTURE specification. Only one overpunch character can appear in the specification for a fixed point number. A floating-point specification can contain two overpunch characters, one in the mantissa field and one in the exponent field.

Programmer response: Ensure the above restrictions are followed.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0J8

IBM0617S ONCODE=*oncode-value* The precision in the numeric PICTURE specification was less than 1.

Explanation: The number of digits for the precision field within a numeric data PICTURE specification must be between one and fifteen digits.

Programmer response: Check the precision and modify the program accordingly.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0J9

IBM0618S ONCODE=*oncode-value* The precision in the fixed decimal PICTURE specification exceeded the limit.

Explanation: The precision in the fixed decimal PICTURE specification must not exceed the specified value in the LIMITS compiler option. The default maximum is 15.

Programmer response: Use the LIMITS compiler option to specify a maximum value of 29 or 31.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JA

IBM0619S ONCODE=*oncode-value* The value specified for the precision in the float decimal PICTURE specification exceeded the limit.

Explanation: The precision in the float decimal PICTURE specification is limited by the hardware to 18 digits.

Programmer response: Check and correct the precision.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JB

IBM0620S **ONCODE=*oncode-value* The PICTURE specification did not contain picture characters for character or numeric data.**

Explanation: The PICTURE specification must contain picture characters for either character or numeric data.

Programmer response: Check the PICTURE specification string.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JC

IBM0621S **ONCODE=*oncode-value* The exponent in the float PICTURE specification exceeded the 4-digit limit.**

Explanation: The number of digits in the exponent of the float decimal PICTURE specification is limited to 4 digits.

Programmer response: Ensure that the exponent does not exceed 4 digits.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JD

IBM0622S **ONCODE=*oncode-value* The exponent in the float PICTURE specification was missing.**

Explanation: The exponent in the float decimal PICTURE specification was missing. A value must be entered, even if it is zero.

Programmer response: Enter the missing exponent value.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JE

IBM0623S **ONCODE=*oncode-value* The exponent in the PICTURE specification contained a V character.**

Explanation: The character V was specified in the PICTURE specification. The character V specifies an implicit decimal point and is not permitted in the exponent field.

Programmer response: Remove the character V from the exponent field.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JF

IBM0624S **ONCODE=*oncode-value* The float PICTURE specification contained invalid characters CR, DB or F.**

Explanation: The float PICTURE specification contained invalid characters CR, DB or F. Credit (CR), Debit (DB), and Scale Factor (F) are only allowed in the fixed PICTURE specification.

Programmer response: Remove the invalid characters from the float PICTURE specification.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JG

IBM0625S **ONCODE=*oncode-value*The PICTURE specification exceeded the limit. Excessive characters were truncated on the right.**

Explanation: The compiler restricts the length of the PICTURE specification to:

- Fixed Decimal: 254
- Float Decimal: 253
- Character Data: 511

Programmer response: Correct the PICTURE specification length.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JH

IBM0626S ONCODE=*oncode-value* The PICTURE specification contained an invalid delimiter.

Explanation: The floating insertion string is delimited by < > characters. The string can contain any character with the exception of the delimiters themselves. In order to include the characters < and > in the floating insertion string, angle brackets must be used in an ″escaped″ format. << denotes character < in the floating insertion string. <> denotes character > in the floating insertion string. The leading < and ending > characters are delimiters.

Example

```
<aaa<<bbb<>ccc>  denotes the FIS  aaa<bbb>ccc
```

Programmer response: Correct the floating insertion string.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JI

IBM0630S ONCODE=3009 A mixed character string ended incorrectly.

Explanation: A mixed character string contained a shift-out character but did not contain a matching shift-in character.

Programmer response: Ensure that mixed character strings contain unnested pairs of shift-out/shift-in characters.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JM

IBM0631S ONCODE=3010 A mixed character string contained an invalid character.

Explanation: One of the following rules for mixed constants was broken:

- SBCS portions of the constant cannot contain a shift-in
- DBCS portions of the constant cannot contain a shift-out (Either byte of a DBCS character cannot contain a shift-out.)
- The second byte of a DBCS character cannot contain a shift-in

Programmer response: Ensure the mixed character string contains balanced, unnested pairs of shift-out/shift-in characters.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JN

IBM0632S ONCODE=3011 An invalid function string was specified for the MPSTR built-in function.

Explanation: For the MPSTR built-in function, a function string is invalid if it is null, contains only blanks, or contains characters other than ’V’, ’v’, ’S’, ’s’, or a blank.

Programmer response: Ensure the function string is valid.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JO

IBM0633S ONCODE=3012 A retry was attempted after a graphic conversion error.

Explanation: The use of the ONSOURCE or ONCHAR pseudovariable to attempt a conversion retry for a graphic (DBCS) conversion error is not allowed.

Programmer response: Remove the retry attempt from your program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JP

IBM0634S ONCODE=3013 An invalid graphic variable assignment was attempted.

Explanation: A graphic (DBCS) target of length greater than 16,383 was encountered. This target could have been an actual target or a temporary target created by the program. This condition was raised by the GRAPHIC built-in function. The maximum length of a graphic (DBCS) string is 16,383 characters (32,766 bytes).

Programmer response: Ensure that graphic (DBCS) strings are less than the maximum allowed length of 16,383.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JQ

IBM0635S ONCODE=3014 An invalid use of a shift code occurred.

Explanation: There are two possible errors:

- The STREAM input record could not be scanned due to an unmatched or nested shift code.
- A graphic (DBCS) constant in STREAM input contained a shift code or used shift codes improperly.

Programmer response: Verify that shift code pairs are matched and unnested, continuation rules are followed, and graphic (DBCS) constants are in one of the allowable forms.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JR

IBM0636S ONCODE=3015 An invalid number of digits was used in a X or GX constant.

Explanation: X constants must be specified in pairs. GX constants must be specified in groups of four.

Programmer response: Change the STREAM input data so that all X constants are specified in pairs and all GX constants are specified in groups of four.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JS

IBM0637S ONCODE=3016 A double-byte character was used incorrectly.

Explanation: A non-EBCDIC double-byte character was used incorrectly. These characters are only valid in DBCS names, graphic (DBCS) constants, and mixed character constants.

Programmer response: Verify that a bit, character or hexadecimal constant does not contain a non-EBCDIC double-byte character, or that such a character is not present outside a constant unless it is part of a name for a GET DATA statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JT

IBM0638S ONCODE=*oncode-value* **A STREAM output record could not be written correctly.**

Explanation: A record could not be written out because there was not enough room for a valid DBCS continuation sequence. As a consequence, the record cannot be read correctly as STREAM input. The ONCODE associated with this message is 3017.

Programmer response: Define the STREAM I/O data set to contain V- or U-type record formats.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0JU

IBM0648S ONCODE=3797 The assignment of a graphic character string caused an error.

Explanation: STREAM I/O issued this message because LIST, DATA, or EDIT input/output was attempted for a graphic (DBCS) string and the corresponding source or target string or file did not have the necessary graphic attribute. This error could also be issued when a null graphic constant appears as an element in the data list of a PUT for LIST or EDIT. Null graphic constants are restricted as elements in the data list of a PUT for LIST or EDIT.

Programmer response: Ensure that the source or target string in the data list is a valid graphic (DBCS) string and that it has been declared with the GRAPHIC attribute. If a null graphic constant caused the error, remove the null graphic constant from the data list of the PUT statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0K8

IBM0650S ONCODE=3799 The source was not modified in the CONVERSION ON-unit. Retry was not attempted.

Explanation: The CONVERSION condition was raised by the presence of an invalid character in the string to be converted. The character was not corrected in an ON-unit using the ONCHAR or ONSOURCE pseudovariable.

Programmer response: Use either the ONCHAR or the ONSOURCE pseudovariable in the CONVERSION ON-unit to assign a valid character to replace the invalid character in the source string.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0KA

IBM0670S ONCODE=*oncode-value* X was less than 0 in SQRT(X).

Explanation: The built-in function SQRT was invoked with an argument that is less than zero. ONCODEs associated with this message are:
- 1500 Short floating-point argument
- 1501 Long floating-point argument
- 1502 Extended floating-point argument

Programmer response: Modify the program so that the argument of the SQRT built-in function is never less than zero.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0KU

IBM0671S ONCODE=*oncode-value* X was less than or equal to 0 in LOG(X), LOG2(X) or LOG10(X).

Explanation: One of the built-in functions LOG, LOG2, or LOG10 was invoked with an argument less than or equal to zero. The invocation may have been direct or as part of the evaluation of an exponentiation calculation. ONCODEs associated with this message are:
- 1503 Extended floating-point argument
- 1504 Short floating-point argument
- 1505 Long floating-point argument

Programmer response: If the invocation is direct, modify the program so that the argument of the LOG, LOG2, or LOG10 built-in function is greater than zero. If the invocation is part of an exponentiation calculation, ensure that the argument is greater than zero.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0KV

IBM0672S ONCODE=*oncode-value* ABS(X) was too large in SIN(X), COS(X), SIND(X), COSD(X), TAN(X) or TAND(X).

Explanation: The error occurred during one of the following:
- The evaluation of SIN, SIND, COS, COSD, TAN, or TAND when invoked implicitly
- The evaluation of TAN, when invoked during the evaluation of TAN or TANH with a complex argument

- The evaluation of SIN or COS, when invoked during the evaluation of EXP, SIN, SINH, COS, COSH, TAN, or TANH with a complex argument
- The evaluation of a general exponentiation function with complex arguments

The argument passed to TAN, TAND, SIN, SIND, COS, or COSD exceeded the limit specified below.

Floating-Point Precision	*Limit*	
Binary p ≤ 21 Decimal p ≤ 6	x<(2**18)*K	where K = pi for x in radians (SIN, COS, or TAN)
Binary 21 < p ≤ 53 Decimal 6 < p ≤ 16	x<(2**50)*K	where K = 180 for x in degrees (SIND, COSD, TAND)
Binary 53 < p ≤ 109 Decimal 6 < p ≤ 33	x<(2**100)*K/pi	

ONCODEs associated with this message are:
- 1506 Short floating-point argument involving SIN, COS, SIND or COSD
- 1507 Long floating-point argument involving SIN, COS, SIND or COSD
- 1508 Short floating-point argument involving TAN or TAND
- 1509 Long floating-point argument involving TAN or TAND
- 1517 Extended floating-point argument involving SIN, COS, SIND or COSD
- 1522 Extended floating-point argument involving TAN or TAND

Programmer response: Ensure that X does not violate the limits as described above. If X is an expression, simplify X for easier problem diagnosis.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0L0

IBM0674S ONCODE=*oncode-value* Both X and Y were 0 in ATAN(Y,X) or ATAND(Y,X).

Explanation: Two arguments, both of value zero, were given for the ATAN or ATAND built-in function. ATAN or ATAND was invoked either directly with a real argument or indirectly in the evaluation of the LOG built-in function with a complex argument. ONCODEs associated with this message are:
- 1510 Short floating-point arguments
- 1511 Long floating-point arguments
- 1521 Extended floating-point arguments

Programmer response: Change the arguments of ATAN or ATAND to nonzero values.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0L2

IBM0675S ONCODE=*oncode-value* ABS(X) was greater than or equal to 1 in ATANH(X).

Explanation: The ATANH built-in function had a floating-point argument with an absolute value that equaled or exceeded 1. ONCODEs associated with this message are:
- 1514 Short floating-point argument
- 1515 Long floating-point argument
- 1516 Extended floating-point argument

Programmer response: Modify the ATANH built-in function so that the absolute value of a floating-point assignment does not equal or exceed 1.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0L3

IBM0676S ONCODE=*oncode-value* ABS(X) was greater than 1 in ASIN(X) or ACOS(X).

Explanation: The absolute value of the floating-point argument of the ASIN or ACOS built-in function exceeded 1. ONCODEs associated with this message are:

- 1518 Short floating-point argument
- 1519 Long floating-point argument
- 1520 Extended floating-point argument

Programmer response: Modify the program so that the ASIN or ACOS built-in function is never invoked with a floating-point argument whose absolute value exceeds 1.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0L4

IBM0682S ONCODE=*oncode-value* X in EXPONENT(X) was invalid.

Explanation: One of the following conditions occurred:

- For X**Y where X and Y are integers, X was equal to zero and Y was less than or equal to zero.
- For X**Y where X is a real value and Y is an integer, X was equal to zero and Y was less than or equal to zero.
- For X**Y where X and Y are integers, X was not equal to plus or minus one and Y was less than zero.
- For X**Y where X and Y are complex values, X was (0,0i) and Y was less than or equal to zero.
- For X**Y where X and Y are complex values, X exceeded the limit K, where K=2**63 for complex short and long arguments, and K=2**55 for complex extended arguments.
- For X**Y where X and Y are complex values, X was equal to (0,0i).
- For X**Y where X and Y are real values, X was equal to zero and Y was not an integer-float greater than zero.
- For X**Y where X and Y are real values, X was less than zero and Y was not an integer-float.

The ONCODEs associated with this message are:

- For integer base and integer exponent

  1673 X equal to zero and Y less than or equal to zero

  1674 X not equal to plus or minus one and less than zero
- For real short floating-point base with integer exponent

  1550 X equal to zero and Y less than or equal to zero
- For real long floating-point base with integer exponent

  1551 X equal to zero and Y less than or equal to zero
- For real extended floating-point base with integer exponent

  1560 X equal to zero and Y less than or equal to zero
- For complex short floating-point base with integer exponent

  1554 X equal to (0,0i) and Y less than or equal to zero
- For complex long floating-point base with integer exponent

  1555 X equal to (0,0i) and Y less than or equal to zero
- For complex extended floating-point base with integer exponent

  1562 X equal to (0,0i) and Y less than or equal to zero
- For real short floating-point base with real short floating-point exponent

  1552 X equal to zero and Y not a positive integer-float, or X less than zero and Y not an integer-float

  1729 X equal to (0,0i) and Y less than or equal to zero
- For real long floating-point base with real long floating-point exponent

  1553 X equal to zero and Y not a positive integer-float, or X less than zero and Y not an integer-float

  1730 X equal to (0,0i) and Y less than or equal to zero

- For real extended floating-point base with real extended floating-point exponent

  1561 X equal to zero and Y not a positive integer-float, or X less than zero and Y not an integer-float
- For complex short floating-point base with complex short floating-point exponent

  1556 Argument equal to (0,0i)

  1754 Argument exceeded limit
- For complex long floating-point base with complex long floating-point exponent

  1557 Argument equal to (0,0i)

  1755 Argument exceeded limit
- For complex extended floating-point base with complex extended floating-point exponent

  1563 Argument equal to (0,0i)

  1756 Argument exceeded limit

Programmer response: Ensure X is a valid floating-point number.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0LA

IBM0683S ONCODE=*oncode-value* **X or Y in ATAN(X,Y) or ATAND(X,Y) was invalid.**

Explanation: One of the following conditions occurred:
- X and Y were invalid.

The ONCODEs associated with this message are:
- For real short floating-point arguments:

  1510 Both arguments were invalid
- For real long floating-point arguments:

  1511 Both arguments were invalid
- For real extended floating-point arguments:

  1521 Both arguments were invalid

Programmer response: Ensure X and Y are both real values and that Y is not equal to zero.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0LB

IBM0700S ONCODE=*oncode-value* **An attempt to assign data to an unallocated CONTROLLED variable occurred during GET DATA for file** *file-name***.**

Explanation: A CONTROLLED variable in the stream was accessed by a GET FILE DATA statement, but there was no current allocation for the variable.

Example:

```
DCL X CONTROLLED FIXED BIN;
GET DATA(X);

(Input stream contains
X=5,.....)
```

The ONCODE associated with this message is 4001.

Programmer response: Either remove the data item from the input stream or insert an ALLOCATE statement for the variable before the GET FILE DATA statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0LS

IBM0701S ONCODE=*oncode-value* An attempt to assign data to an unallocated CONTROLLED variable occurred on a GET DATA statement.

Explanation: A CONTROLLED variable in the stream was accessed by a GET FILE DATA statement, but there was no current allocation for the variable.

Example:

```
DCL STR CHAR(4) INIT('X=5'),
X CONTROLLED FIXED BIN;
GET STRING(STR) DATA(X);
```

The ONCODE associated with this message is 4001.

Programmer response: Either remove the data item from the string or insert an ALLOCATE statement for the variable before the GET STRING DATA statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0LT

IBM0702S ONCODE=*oncode-value* An attempt to to output an unallocated CONTROLLED variable occurred on a PUT DATA statement.

Explanation: A CONTROLLED variable was being output to a file by a PUT FILE DATA statement, but there was no current allocation for the variable. The ONCODE associated with this message is 4002.

Programmer response: Insert an ALLOCATE statement for the variable before the PUT FILE DATA statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0LU

IBM0703S ONCODE=*oncode-value*An attempt to assign from an unallocated CONTROLLED variable occurred on a PUT DATA statement with the STRING option.

Explanation: A CONTROLLED variable was being accessed by a PUT STRING DATA statement, but there was no current allocation for the variable. The ONCODE associated with this message is 4003.

Programmer response: Ensure the CONTROLLED variable is allocated and initialized before the PUT DATA statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0LV

IBM0722S ONCODE=*oncode-value* X was assigned a value of 0 and Y was not assigned the value of a positive real number in XY.**

Explanation: In an exponentiation operation the floating-point base was zero and the exponent was not a positive real number. ONCODEs associated with this message are:

- 1550 Real short floating-point base with an integer exponent
- 1551 Real long floating-point base with an integer exponent
- 1552 Real short floating-point base with a floating-point or non-integer exponent
- 1553 Real long floating-point base with a floating-point or non-integer exponent
- 1554 Complex short floating-point base with an integer exponent
- 1555 Complex long floating-point base with an integer exponent
- 1556 Complex short floating-point base with a floating-point or non-integer exponent
- 1557 Complex long floating-point base with a floating-point or non-integer exponent
- 1560 Real extended floating-point base with an integer exponent
- 1561 Real extended floating-point base with a floating-point or non-integer exponent
- 1562 Complex extended floating-point base with an integer exponent
- 1563 Complex extended floating-point base with a floating-point or non-integer exponent

Programmer response: Modify the program so that the exponentiation operation involves a nonzero floating-point base or a positive real exponent.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0MI

IBM0724S ONCODE=*oncode-value* Z=+1i or Z=-1i in ATAN(Z) or Z=+1 or Z=-1 in ATANH(Z).

Explanation: Either the complex floating-point argument of the ATAN built-in function had the value of +1i or -1i, or the complex floating-point argument of the ATANH built-in function has the value +1 or -1. ONCODEs associated with this message are:

- 1558 Complex short floating-point argument
- 1559 Complex long floating-point argument
- 1564 Complex extended floating-point argument

Programmer response: Modify the program so the complex floating-point argument of ATAN never has the value of +1i or -1i, or the complex floating-point argument of the ATANH built-in function never has the value +1 or -1.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0MK

IBM0750S ONCODE=*oncode-value* A GOTO to an invalid block was attempted.

Explanation: A GOTO statement that transfers control to a label variable was invalid. The possible causes are:

- The generation of the block that was active when the label variable was assigned was no longer active when the GOTO statement was run.
- The label variable was uninitialized.
- The element of the label array, to which control is to be transferred, does not exist in the program.
- An attempt has been made to transfer control to a block that is not within the scope of this task.

Example:

```
DCL L LABEL;
BEGIN;
A:  L = A;
END;
GOTO L;
```

The ONCODE associated with this message is 9002.

Programmer response: Modify the program so that the GOTO statement transfers control to a label in an active block.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0NE

IBM0751S ONCODE=*oncode-value* A GOTO was attempted to an element of a label constant array, but the subscripts for the element were not those of any label in that array.

Explanation: The subscripts of an element in a GOTO statement must match the label in the specified array. This error occurs in the following code if n is 1, 3, 5 or 7:

Example:

```
dcl n fixed bin;
⋮
goto x(n);
⋮
x(0):
⋮
x(2):
```

```
⋮
x(4):
⋮
x(6):
⋮
x(8):
```

Note: This error will not occur if n is less than the lower bound for x or greater than the upper bound.

Programmer response: Correct your program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0NF

IBM0752S **ONCODE=***oncode-value* **A RETURN without an expression was attempted from a procedure that had been entered at an ENTRY that specified the RETURNS attribute.**

Explanation: A procedure can contain ENTRYs some of which have the RETURNS attribute and some of which do not, but if it is entered at an ENTRY that has the RETURNS attribute, it must be exited with a RETURN statement that specifies a return value.

Programmer response: Correct your program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0NG

IBM0753S **ONCODE=***oncode-value* **A RETURN without an expression was attempted from a procedure that had been entered at an ENTRY that does not specify the RETURNS attribute.**

Explanation: A procedure can contain ENTRYs some of which have the RETURNS attribute and some of which do not, but if it is entered at an ENTRY that has the RETURNS attribute, it must be exited with a RETURN statement that does not specify a return value.

Programmer response: Correct your program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0NH

IBM0780S **ONCODE=***oncode-value* **No WHEN clauses were satisfied and no OTHERWISE clause was available.**

Explanation: No WHEN clauses of a SELECT statement were selected and no OTHERWISE clause was present. The ONCODE associated with this message is 3.

Programmer response: Add an OTHERWISE clause to the SELECT group.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0OC

IBM0802S **ONCODE=***oncode-value* **The GET/PUT STRING exceeded the source string size.**

Explanation: For input, a GET statement attempted to access data that exceeded the length of the source string. For output, a PUT statement attempted to assign data that exceeded the target string. The ONCODE associated with this message is 1002.

Programmer response: For input, either extend the length attribute of the source string, or correct the data so that the length does not exceed the declared length of the source string. For output, either extend the length attribute of the target string, or correct the data so that the length does not exceed the declared length of the target string.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0P2

IBM0803S ONCODE=*oncode-value* **A prior condition on file** *file-name* **prevented further output.**

Explanation: A PL/I WRITE, LOCATE, or PUT statement was issued for a file to which a previous attempt to transmit a record caused the TRANSMIT condition to be raised immediately. If the EVENT option was specified to be stacked until the event was waited on, the data set was not a unit-record device and no further processing of the file was possible. The ONCODE associated with this message is 1003.

Programmer response: Correct the error that caused the TRANSMIT condition to be raised and rerun the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0P3

IBM0804S ONCODE=*oncode-value* **The PRINT option/format item was used with non-PRINT file** *file-name***.**

Explanation: An attempt was made to use one of the options PAGE or LINE for a file that was not a print file. The ONCODE associated with this message is 1004.

Programmer response: Either remove the PRINT option/format item from the non-print file, or specify the PRINT option for the print file.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0P4

IBM0805S ONCODE=*oncode-value* **A DISPLAY with REPLY option had a zero-length string.**

Explanation: The current length of the character string to be displayed, or the maximum length of the character string to which the reply was assigned, was zero. The ONCODE associated with this message is 1005.

Programmer response: Change length of the character string to be displayed, or to which the reply is to be assigned, to greater than zero.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0P5

IBM0807S ONCODE=*oncode-value* **The REWRITE or DELETE on file** *file-name* **occurred without a preceding READ SET or READ INTO statement.**

Explanation: A REWRITE or DELETE statement without the KEY option was run. The last input/output operation on the file was not a READ statement with the SET or INTO option or was a READ statement with the IGNORE option. The ONCODE associated with this message is 1007.

Programmer response: Modify the program so that the REWRITE or DELETE statement is either preceded by a READ statement or, in the case of a REWRITE statement, replaced by a WRITE statement, according to the requirements of the program. A preceding READ statement with the IGNORE option will also cause the message to be issued.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0P7

IBM0808S ONCODE=*oncode-value* **An invalid element was present in the string for a GET STRING DATA statement.**

Explanation: The identifier in the string named in the STRING option of a GET STRING DATA statement did not match the identifier in the data specification. Note that the DATAFIELD built-in function does not return a value in this case. The ONCODE associated with this message is 1008.

Programmer response: Modify the program so that the string contains the identifier in the data specification.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0P8

IBM0809S ONCODE=*oncode-value* An invalid file operation was attempted on file *file-name*.

Explanation: An attempt was made to perform an invalid operation on a file. For example, it is not possible to run a REWRITE statement on a STREAM file, read an output file, or write an input file. Refer to Table 3 for a list of operations and conflicting file organizations.

Table 3. Operations and Conflicting File Organizations

Statement/Option	**File Organization**
Any record I/O statement	STREAM
Any stream I/O statement	RECORD
READ	OUTPUT
READ SET	UNBUFFERED
READ EVENT	UNBUFFERED
READ KEY	REGIONAL SEQUENTIAL or CONSECUTIVE
READ IGNORE	DIRECT
READ NOLOCK	SEQUENTIAL or INPUT
WRITE	INPUT SEQUENTIAL UPDATE, INDEXED DIRECT NOWRITE, REGIONAL (not KEYED)
WRITE EVENT	BUFFERED
REWRITE	INPUT or OUTPUT
REWRITE (without FROM)	UNBUFFERED or DIRECT
REWRITE KEY	SEQUENTIAL
REWRITE EVENT	BUFFERED
LOCATE	INPUT or UPDATE, UNBUFFERED, DIRECT
LOCATE KEYFROM	INDEXED or REGIONAL (without KEYED)
DELETE	INPUT or OUTPUT, CONSECUTIVE, REGIONAL SEQUENTIAL, RKP=0 (blocked records), OPTCD=L not specified
DELETE KEY	SEQUENTIAL
UNLOCK	INPUT or OUTPUT, SEQUENTIAL
GET	OUTPUT
PUT	INPUT

The ONCODE associated with this message is 1009.

Programmer response: Ensure the file declaration and the input/output statements for the named file are compatible.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0P9

IBM0810S ONCODE=*oncode-value* A built-in function or pseudovariable referenced an unopened file or referenced a file with a contradicting attribute.

Explanation: An I/O built-in function or pseudovariable referenced a file that was not opened or referenced a file with an attribute that contradicted the function or pseudovariable. The functions/pseudovariables are :

- PAGENO - file not open or does not have the PRINT attribute
- SAMEKEY - file does not have the RECORD attribute
- ENDFILE - file not open
- FILEREAD - file not open or is not a TYPE(U) file

- FILEWRITE - file not open or is not a TYPE(U) file
- FILESEEK - file not open or is not a TYPE(U) file
- FILETELL - file not open or is not a TYPE(U) file
- FILEDDTEST - file not open or:
  - AMTHD - file not a native file
  - BKWD - file not a record file
  - DESCENDKEY - file not a record file
  - GENKEY - file not a record file
  - PRINT - file not a stream output file
- FILEDDINT - file not open or:
  - BUFSIZE - file not a native file
  - DELAY - file not a DDM, BTRIEVE or ISAM file
  - RETRY - file not a DDM, BTRIEVE or ISAM file
  - FILESIZE - file not a native file
  - KEYLEN - file not an indexed or a relative keyed file
  - KEYLOC - file not an indexed or a relative keyed file
- FILEDDWORD - file not open or:
  - TYPEF - file not a native file

Programmer response: Correct your program to use the built-in function or pseudovariable correctly.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PA

IBM0811S ONCODE=*oncode-value* **An I/O error occurred. Subcode1=** *sc1* **Subcode2=** *sc2*

Explanation: The data management routines detected an error during an input/output operation, which PL/I did not recognize. Subcode1 and Subcode2 provide VSAM diagnostic information; otherwise, they contain zeros. See the VSAM Macro Instruction manual for a description of the errors. Subcode1 indicates the I/O function involved:

- 0 - I/O function not identified
- 1 - GET
- 2 - PUT
- 3 - CHECK
- 4 - POINT
- 5 - ENDREQ
- 6 - ERASE
- 64 - OPEN
- 130 - GENCB of ACB
- 131 - GENCB of RPL
- 138 - SHOWCB of ACB
- 142 - TESTCB of ACB
- 146 - SHOWCB of block lengths

Subcode2 consists of 8 hexadecimal digits (xxxxyyyy). The meaning of Subcode2 varies depending on the PL/I product used.

For MVS and VM, the value of subcode1 determines the type of VSAM return code information provided.

- Subcode1 0: VSAM return code information is not provided.
- Subcode1 1-63: VSAM Request Parameter List Feedback Word (RPLFDBWD) = xxxxyyyy.
- Subcode1 64: Open register 15 = xxxx. Open reason code = yyyy.
- Subcode1 128-192: Register 15 = xxxx. Register 0 = yyyy.

For Enterprise PL/I for z/OS, Subcode2 gives the following information:

- Register 15 = xxxx. Reason code = yyyy.

Note that PL/I terminology translates to VSAM terms as follows:

- PL/I UPDATE OPEN is equivalent to VSAM IN and OUT OPEN.
- PL/I OUTPUT OPEN is equivalent to VSAM OUT OPEN, but only inserts or additions are allowed.
- PL/I READ results in VSAM POINT to key, if specified, followed by VSAM GET.
- PL/I WRITE or LOCATE results in VSAM PUT NUP. For PL/I LOCATE, the associated VSAM PUT NUP occurs on the next PL/I I/O request.
- PL/I REWRITE results in implied read, if needed, followed by VSAM PUT UPD.
- PL/I DELETE results in implied read, if needed, followed by VSAM ERASE.
- PL/I WAIT EVENT I/O results in VSAM CHECK.

Programmer response: Use the VSAM diagnostic information to correct the cause of the error and resubmit the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PB

IBM0812S ONCODE=*oncode-value* A READ SET or READ INTO statement did not precede a REWRITE request.

Explanation: A REWRITE statement with the INTO or SET option ran without a preceding READ statement. The ONCODE associated with this message is 1012.

Programmer response: Modify the program so that the REWRITE statement is either preceded by a READ statement or replaced by a WRITE statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PC

IBM0813S ONCODE=*oncode-value* The last READ statement before the last REWRITE or DELETE was incomplete.

Explanation: An attempt was made to run a REWRITE or DELETE statement before a preceding READ statement (with the EVENT option) for a file that had completed. The ONCODE associated with this message is 1013.

Programmer response: Insert a WAIT statement for the given event variable into the flow of control between the REWRITE or DELETE and READ statements. The REWRITE or DELETE statement should run after completion of the READ statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PD

IBM0814S ONCODE=*oncode-value* Excessive incomplete I/O operations occurred for file *file-name*.

Explanation: An attempt was made to initiate an input/output operation beyond the limit imposed by the NCP (number of channel programs) subparameter of the DCB parameter or option of the ENVIRONMENT attribute. For a file with the attributes SEQUENTIAL and UNBUFFERED, the default for NCP is one. The limit, for VSAM files, is specified either by the NCP option of the ENVIRONMENT attribute or by the STRNC sub-parameter of the AMP parameter in the DD statement. The limit is one for both SEQUENTIAL and DIRECT UNBUFFERED files except when using the ISAM compatibility interface. The ONCODE associated with this message is 1014.

Programmer response: Modify the program so that the input/output operation is not initiated until an incomplete input/output operation completes.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PE

IBM0816S ONCODE=*oncode-value* The implicit OPEN was unsuccessful for file *file-name*.

Explanation: An error occurred during the implicit opening of a file. The UNDEFINEDFILE condition was raised and a normal return was made from the associated ON-unit, but the file was still unopened. The ONCODE associated with this message is 1016.

Programmer response: Ensure that the file has been completely and correctly declared, and that the input/output statement that implicitly opens the file is not in conflict with the file declaration.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PG

IBM0818S ONCODE=*oncode-value* An unexpected end of file/string was detected in the STREAM input.

Explanation: The end of the file was detected before the completion of a GET FILE statement. The ONCODE associated with this message is 1018.

Programmer response: For edit-directed input, ensure that the last item of data in the stream has the same number of characters as specified in the associated format item. If the error occurs while an X-format is running, ensure that the same number of characters to be skipped are present before the last data item in the stream. For list-directed and data-directed input, ensure the last item of data in the data set that precedes the end-of-file character is terminated by a quote character for a string or a 'B' character for a bit-string.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PI

IBM0819S ONCODE=*oncode-value* An attempt was made to close a file in the wrong task.

Explanation: A file can only be closed by the task that opened it.

Programmer response: Change your program to insure the close is issued in the same task that opened the file.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PJ

IBM0820S ONCODE=*oncode-value* An attempt was made to access a locked record.

Explanation: In an exclusive environment, an attempt was made to read, rewrite, or delete a record when either the record or the data set was locked by another file in this task. The ONCODE associated with this message is 1021.

Programmer response: Ensure that all files accessing the data set have the EXCLUSIVE attribute. If a READ statement is involved, specify the NOLOCK option to suppress the locking mechanism.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PK

IBM0821S ONCODE=*oncode-value* An I/O statement occurred before a WAIT statement completed a previous READ.

Explanation: While an indexed sequential file was open for direct updating, an input/output statement was attempted before the completion of a previous READ statement with the EVENT option. The ONCODE associated with this message is 1020.

Programmer response: Include a WAIT statement so that the erroneous input/output statement cannot be run until the completion of the previous READ statement with the EVENT option.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PL

IBM0822S ONCODE=*oncode-value* Insufficient space was available for a record in the sequential output data set.

Explanation: The space allocated for the sequential output data set was full. The ONCODE associated with this message is 1040.

Programmer response: Increase the size of the data set, or check the logic of the application for possible looping.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PM

IBM0823S ONCODE=*oncode-value* An invalid control format item was detected during a GET/PUT STRING.

Explanation: An invalid control format item (PAGE, LINE, SKIP, or COL) was detected in a remote format list for a GET or PUT STRING statement.

Example:

```
DCL(A,B) CHAR(10),
C CHAR(80);
F:  FORMAT(A(10), SKIP,A(10));
A='FRED'; B='HARRY';
PUT STRING(C) EDIT(A,B) (R(F));
```

The ONCODE associated with this message is 1004.

Programmer response: Modify the source program so that GET or PUT STRING statements do not use the control format items PAGE, LINE, SKIP or COL.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PN

IBM0824S ONCODE=*oncode-value* Records were still locked in a subtask while attempting to close EXCLUSIVE file *file-name*.

Explanation: When an EXCLUSIVE file is closed by a task, no records should be locked by any subtasks.

Programmer response: Change your program to insure that the subtasks free any record locks before the file is closed.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PO

IBM0825S ONCODE=*oncode-value* The EVENT variable was already used.

Explanation: An input/output statement with an EVENT option was attempted while a previous input/output statement with an EVENT option that used the same event variable was still incomplete. The ONCODE associated with this message is 1015.

Programmer response: Either change the event variable used in the second EVENT option or insert a WAIT statement for the event variable between the two input/output statements.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PP

IBM0826S ONCODE=*oncode-value* The EVENT variable was already used with a DISPLAY statement.

Explanation: An input/output statement with an EVENT option was attempted while a previous DISPLAY statement with an EVENT option that used the same event variable was still incomplete.

Programmer response: Either change the event variable used in the second EVENT option or insert a WAIT statement for the event variable between the DISPLAY statement and the input/output statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PQ

IBM0827S ONCODE=*oncode-value* **The EVENT variable was already active and was used with entry** *entry-name***.**

Explanation: An event variable that was already used in the EVENT option in a CALL statement was still active when used again in the EVENT option of an input/output statement.

Programmer response: Either use a different event variable or insert a WAIT statement so that the input/output statement is not run until the event variable becomes inactive.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PR

IBM0828S ONCODE=*oncode-value* **An incorrect sequence of I/O operations was performed on an associated file.**

Explanation: Operations on a set of associated files were not carried out in the correct sequence, as follows:

1. Appropriate I/O operations were not carried out in the sequence Read-Punch-Print. Only the Print operation can be omitted or repeated.
2. An attempt was made to print more than the maximum number of lines on a card, using a print file that was associated with a read or punch file.

The ONCODE associated with this message is 1024.

Programmer response: Ensure that the above rules have not been violated.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PS

IBM0829S ONCODE=*oncode-value* **Insufficient virtual storage was available to VSAM.**

Explanation: During an OPEN/CLOSE or any other operation on a VSAM data set, insufficient storage was available for workspace and control blocks. The ONCODE associated with this message is 1025.

Programmer response: Increase the REGION size for the VSAM application.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PT

IBM0830S ONCODE=*oncode-value* **An I/O error occurred during a CLOSE operation.**

Explanation: An I/O error occurred while a VSAM close routine was either reading or writing a catalog record, or completing an outstanding I/O request.

Programmer response: If the problem is related to an insufficient amount of virtual storage available to VSAM, try running the job with a larger REGION size. The access method services VERIFY command can be used to obtain more information pertaining to the error. Refer to the MVS/DFP Access Method Services manual for details.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PU

IBM0831S ONCODE=*oncode-value* **A position was not established for a sequential READ statement.**

Explanation: A READ statement without the KEY option was attempted on a VSAM data set. This occurred after sequential positioning was lost as the result of a previous error during sequential processing (for example, read error on index set or failure to position to next highest key after a “key not found” condition). The ONCODE associated with this message is 1026.

Programmer response: Use the KEYTO option of the READ statement to obtain the keys of records read. Use this information to reposition a file for subsequent retrieval when positioning is lost.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0PV

IBM0832S ONCODE=*oncode-value* **Insufficient space was available for VSAM file** *file-name***.**

Explanation: VSAM was unable to allocate additional DASD space for the data set (ESDS or KSDS). This condition was raised during an attempt to write or locate a record during the sequential creation or extension of a data set and the space allocated to the data set was full. For a KSDS, the condition may have occurred when the associated PL/I file was opened for update and an attempt was made to write new records to the file or to increase the size of existing records using the WRITE and REWRITE statements respectively. An attempt to increase the size of a data set while processing with SHROPT=4 and DISP=SHR may also have raised this condition. The ONCODE associated with this message is 1022.

Programmer response: Use the access method services ALTER command to extend a data set provided secondary allocation was specified during data set definition. Refer to the MVS/DFP Access Method Services manual for details.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0Q0

IBM0833S ONCODE=*oncode-value* **A requested record was held in exclusive control.**

Explanation: The VSAM data set control interval containing the requested record was in the process of being updated by another file which used the same DD statement. The ONCODE associated with this message is 1027.

Programmer response: Retry the update after completion of the other file's data transmission, or avoid having two files associated with the same data set at one time.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0Q1

IBM0834S ONCODE=*oncode-value* **The requested record was stored on a non-mounted volume.**

Explanation: The requested record was stored on a non-mounted volume of a VSAM data set spanning several volumes. The ONCODE associated with this message is 1028.

Programmer response: Ensure that all volumes on which a VSAM data set resides are accessible at the time the application is run.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0Q2

IBM0835S ONCODE=*oncode-value* **An attempt to position the file for a sequential READ failed. Subcode1=** *sc1* **Subcode2=** *sc2*

Explanation: An attempt to reposition to the next highest key for subsequent sequential retrieval, after the 'key not found' condition, failed. If file processing is continued, the next I/O statement should specify the KEY option to effect positioning. Otherwise message IBM0831 may result. Subcode1 and Subcode2 provide detailed VSAM diagnostic information. See message IBM0811S for an explanation of these fields.

Programmer response: Use the VSAM diagnostic information to correct the cause of the error and resubmit the program. Alternatively, use the KEYTO option of the READ statement to obtain the keys of the records read, so that you can reposition the file yourself for sequential retrieval.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0Q3

IBM0836S ONCODE=*oncode-value* **The number of concurrent operations on a data set exceeded STRNO.**

Explanation: Several files accessed a VSAM data set by means of the same DD statement (that is, using the same title). The STRNO subparameter of the DD statement that specified the total number of operations on all files that can be active at the same time was less than the number of concurrent operations. The ONCODE associated with this message is 1014.

Programmer response: Ensure the concurrent operations are valid. Or, modify the STRNO parameter to reflect the correct number of allowed concurrent operations. A read-rewrite pair of operations on a sequential file counts as one operation. For example, if three sequential files are to update the same data set at the same time, 'STRNO=3' should be specified in the DD statement.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0Q4

IBM0837S ONCODE=*oncode-value* **An error occurred during an index upgrade. Subcode1=** *sc1* **Subcode2=** *sc2*

Explanation: A change to a base cluster could not be reflected in one of the indexes of the cluster's upgrade set. Subcode1 and Subcode2 provide detailed VSAM diagnostic information. See message IBM0811S for an explanation of these fields.

Programmer response: Run the job with a larger REGION size. The problem might be related to an insufficient amount of virtual storage available to VSAM.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0Q5

IBM0838S ONCODE=*oncode-value* **The maximum number of alternate index pointers was exceeded.**

Explanation: The maximum number of alternate index pointers exceeded 32767. The maximum number of pointers allowed in an alternate index for any given key is 32767. This message will also be issued when the RECORDSIZE specified for a VSAM alternate index, defined with NONUNIQUEKEY, is not large enough to hold all the base cluster key pointers for a given non-unique alternate key.

Programmer response: For alternate indices with non-unique keys, ensure the RECORDSIZE specified during the creation of the alternate index is large enough. For non-unique alternate indices, each alternate index record contains pointers to all the records that have the associated alternate index key. As a result, the index record can be quite large. If the number of alternate index pointers exceed the allowed maximum, then a different alternate key would need to be used. Refer to *z/OS Language Environment Programming Guide* for more information regarding the use of alternate index paths.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0Q6

IBM0839S ONCODE=*oncode-value* **An invalid alternate index pointer was used.**

Explanation: A pointer in the alternate index was invalid. This may have been caused by incorrect use of the alternate index as a Key Sequenced Data Set (KSDS).

Programmer response: Refer to *z/OS Language Environment Programming Guide* regarding a general description on the use of alternate index. For more information, refer to *z/OS Language Environment Programming Guide*.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0Q7

IBM0840S ONCODE=*oncode-value* **An invalid sequential WRITE was attempted.**

Explanation: A WRITE statement on a file associated with a Relative Record Data Set (RRDS) did not specify a relative record number. This resulted in an attempt to write in a slot already containing a record. The ONCODE associated with this message is 1031.

Programmer response: Modify the WRITE statement to include a relative record number (or key) by specifying the KEYFROM option. If a relative record number is used, ensure the record number is valid. For error diagnosis, the KEYTO option can be used to obtain the number of the key for each record written if previous sequential WRITE statements did not have the KEYFROM option specified.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0Q8

IBM0841S ONCODE=*oncode-value* A data set, open for output, used all available space.

Explanation: No more space on the disk. The ONCODE associated with this message is 1040.

Programmer response: Increase the size of the data set or check the logic of the program for possible looping.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0Q9

IBM0842S ONCODE=*oncode-value* An attempt was made to write a record containing record delimiter.

Explanation: An attempt was made to write a record containing a record delimiter (line feed character or carriage control and line feed character combination) to a native data set with the type(lf) or type(crlf) option applied.

Programmer response: Either change your program to let PL/I write the delimiter or use the type(fixed) option.

System action: The record is not transmitted to the data set.

Symbolic Feedback Code: IBM0QA

IBM0843S ONCODE=*oncode-value* A record in the data set was not properly delimited.

Explanation: While reading a native data set with TYPE(CRLF) applied, a record delimiter (carriage control and line feed character combination) was not found before the number of bytes specified by RECSIZE were read.

Programmer response: Increase the value of RECSIZE appropriately and re-run your program.

System action: The record is not assigned to the record variable.

Symbolic Feedback Code: IBM0QB

IBM0850S ONCODE=*oncode-value* The aggregate length exceeded the limit of 224 bytes.**

Explanation: The length of the structure or array to be mapped was greater than 2^{24}, thus exceeding the limits of addressability. The program was compiled with CMPAT(V1). The ONCODE associated with this message is 3800.

Programmer response: Reduce the size of the array or structure to a size that can be accommodated within the main storage available. If a variable is used to specify the dimension or length, check that it has been correctly initialized before the storage is allocated to the aggregate. Or, compile the program with the CMPAT(V2) option.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0QI

IBM0851S ONCODE=*oncode-value* The array structure element could not be mapped.

Explanation: The program was compiled with CMPAT(V1). Either the program contained a structure with:

- An adjustable element and an array element with extents that cause the relative virtual origin to exceed 2^{32}-1.
- A structure with an adjustable element and an array with a lower bound greater than the upper bound.

Example:

```
DCL 1 A CTL,
2 B CHAR(N),
2 C (15000:15001, 15000:15001,
15000:15001) CHAR(32700);
N=2;
ALLOCATE A;
```

The ONCODE associated with this message is 3801.

Programmer response: If possible, compile the program with the CMPAT(V2) option. If recompiling is not possible:

- Ensure aggregates with array elements remain within the limit of addressability (2^{32} -1), or
- Ensure the lower bound is not greater than the upper bound.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0QJ

IBM0852S ONCODE=*oncode-value* **The mapping of an aggregate to a COBOL form failed.**

Explanation: An attempt was made to pass to or obtain from a COBOL program a structure with more than three dimensions. The ONCODE associated with this message is 3808.

Programmer response: Ensure PL/I aggregates that are passed to or from COBOL programs are within the limits described above.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0QK

IBM0854S ONCODE=*oncode-value* **The maximum depth of iteration exceeded the limits during an array initialization.**

Explanation: The depth of iteration within the initial attribute on an AUTOMATIC array exceeded 12.

Programmer response: Change the depth of iteration to less than 12.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0QM

IBM0855S ONCODE=3809 The length of a data aggregate exceeded the maximum limit.

Explanation: The length of the structure to be mapped was greater than the allowable limit. Structures that do not contain any unaligned bit elements have a maximum size of 2**31-1 bytes. Structures with one or more unaligned bit elements have a maximum size of 2**28-1 bytes.

Programmer response: Reduce the size of the structure to less than the maximum allowed. If a variable is used to specify the dimension or length of an element, ensure the variable is correctly initialized before the storage is allocated to the aggregate.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0QN

IBM0856S ONCODE=3810 An extent of an array exceeded the maximum limit.

Explanation: During structure mapping, an array with an extent greater than the allowed maximum was encountered. The largest allowable extent (upper bound minus lower bound) of any dimension in an array is 2**31-1.

Programmer response: Reduce the extent of the array to less than the maximum allowed. If a variable is used to specify a bound, ensure the variable is correctly initialized.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0QO

IBM0860S ONCODE=*oncode-value* **The UNDEFINEDFILE condition was raised because the VSAM server was not available to perform the OPEN (***FILE= or ONFILE= file-name***).**

Explanation: VSAM Record Level Sharing (RLS) is supported by a VSAM server address space and data space. The VSAM server has failed and is unavailable for PL/I to complete the open function.

Programmer response: When the VSAM server becomes available, resubmit the program. See the VSAM publications for additional information.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0QS

IBM0861S **ONCODE=***oncode-value* **The UNDEFINEDFILE condition was raised because an attempt to position the file at the first record failed** *FILE= or ONFILE= file-name***). Subcode1=***sc1* **Subcode2=***sc2*

Explanation: For SEQUENTIAL INPUT or UPDATE, the file must be positioned at the first record. If an attempt to position at the first record fails, the file is closed and the UNDEFINEDFILE condition is raised with this message. Subcode1 and Subcode2 provide detailed VSAM diagnostic information. See message IBM0811S for an explanation of these fields.

Programmer response: Use the VSAM diagnostic information to correct the cause of the error and resubmit the program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0QT

IBM0862S **ONCODE=***oncode-value* **The VSAM server was not available to execute a VSAM I/O request**

Explanation: VSAM Record Level Sharing (RLS) is supported by a VSAM server address space and data space. The VSAM server has failed and is unavailable to perform VSAM I/O requests. The failing file must be CLOSEd, if an attempt is made to reopen the file and continue processing.

Programmer response: When the VSAM server becomes available, resubmit the program. See *z/OS Language Environment Programming Guide* for additional information.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0QU

IBM0863S **ONCODE=***oncode-value* **A deadlock was detected while attempting to lock a record.**

Explanation: The program has attempted to lock a record using VSAM Record Level Sharing (RLS). However, VSAM RLS processing has detected that a deadlock condition exists within its sysplex-wide set of lock owners and lock waiters. This program has been selected to receive the deadlock error so that the deadlock can be broken.

Programmer response: This program was found to be in deadlock with other programs. The system programmer will have SMSVSAM diagnostic tools and diagnostic information is available from CICS to determine what programs encountered the deadlock. The action for this error is to avoid running the same mix of programs. The program may also attempt to retry the PL/I request which encountered the deadlock error some number of times. However, depending of the mix of programs, this may or may not be successful.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0QV

IBM0865S **ONCODE=***oncode-value* **A retained lock reject has occurred while attempting to lock a record.**

Explanation: This program has attempted to lock a record using VSAM Record Level Sharing (RLS). However, VSAM RLS has rejected this request for the lock because of its retained lock status. That is, the lock is held by a failed CICS and until that CICS restarts and completes its backout of the record, the record is not available.

Programmer response: This error can occur on a READ statement for a file opened for INPUT when RLS=CR is used, but not if RLS=NRI is used. For sequential read, the program may wish to proceed to read the next available record. Or, the program can be resubmitted when the CICS restart is complete. See *z/OS Language Environment Programming Guide* for additional information about this failure.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0Q1

IBM0870S **ONCODE=***oncode-value* **The OS/VS COBOL program is not supported for interlanguage communication in Language Environment.**

Explanation: The OS/VS COBOL program is not supported for interlanguage communication in Language Environment.

Programmer response: Compile the OS/VS COBOL program with IBM SAA AD/Cycle COBOL/370 or don't run the application with Language Environment.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0R6

IBM0880S ONCODE=*oncode-value* A program check occurred in the SORT/MERGE program.

Explanation: An error occurred while the SORT/MERGE program was running after it was invoked from a PL/I program by use of the PL/I SORT interface facilities. As a result, the SORT program was unable to continue and control was passed to the PL/I error-handler. The ONCODE associated with this message is 9200.

Programmer response: Because the problem occurred while the SORT/MERGE program was running, refer to the appropriate SORT/MERGE program manual for an explanation of any SORT program messages and any other information that might be necessary to correct the error.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0RG

IBM0882S ONCODE=*oncode-value* The string RECORD TYPE was missing in the second argument of the call PLISRTx statement.

Explanation: The RECORD TYPE string must be given in the RECORD statement for calls to PLISRTx. It is used to specify the type of records in the file.

Programmer response: Ensure the RECORD TYPE is coded correctly in the RECORD statement and rerun the application.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0RI

IBM0883S ONCODE=*oncode-value* Incorrect record type was specified in the second argument of the call PLISRTx statement.

Explanation: The RECORD TYPE in the RECORD statement of PLISRTx takes F for fixed length and V for varying length EBCDIC. Characters other than F and V are invalid.

Programmer response: Code the correct record type in the RECORD statement and rerun the application.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0RJ

IBM0884S ONCODE=*oncode-value* The LENGTH= was not specified in the second argument of the call PLISRTx statement.

Explanation: The LENGTH specifier must be given for calls to PLISRTB, and PLISRTD. Use this specifier to indicate the length of the record to be sorted.

Programmer response: Ensure the LENGTH specifier is coded in the RECORD statement and rerun the application.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0RK

IBM0885S ONCODE=*oncode-value* The length specified in the LENGTH= parameter in the second argument of the call PLISRTx statement was not numeric.

Explanation: The length coded for LENGTH= in the RECORD statement of the PLISRTx call must be numerical.

Programmer response: Ensure numerical data is coded for LENGTH= in the RECORD statement and rerun the application.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0RL

IBM0886S **ONCODE=***oncode-value* **Incorrect return code** *rc* **received from user's E15 or E35 handling routine.**

Explanation: The allowed return code from the E15 input handling routine are 8, 12, and 16. The allowed return code from the E35 output handling routine are 4 and 16.

Programmer response: Ensure the return code returned by the PLIRETC built-in function is correct and rerun the application.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0RM

IBM0887S **ONCODE=***oncode-value* **dfsort failed with a return code of** *rc***.**

Explanation: The sort program returns an unsuccessful return code. For the explanation of the return code, refer to the message in the JES log.

Programmer response: Correct the program based on the information from the return code and the message and rerun the application.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0RN

IBM0888S **ONCODE=***oncode-value* **PLISRTx not supported in environments other than ADMVS.**

Explanation: The PL/I program calling the PLISRTx function must have the ADMVS running.

Programmer response: Take out the PLISRTx call and rerun the application.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0RO

IBM0900S **ONCODE=***oncode-value* **The WAIT statement would cause a permanent wait. The program has been terminated.**

Explanation: A WAIT statement that could never have been completed was encountered.

Example:

```
COMPLETION (E1) = '0'B;
WAIT(E1);
```

The event E1 is inactive and incomplete.

Programmer response: Modify the program so that the WAIT statement can never wait for an inactive or incomplete event.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0S4

IBM0913S **ONCODE=***oncode-value* **An error occurred on a FREE statement.**

Explanation: PL/I storage management detected an error during the processing of either a FREE statement or the PLIFREE built-in function.

Programmer response: Ensure the variable specified on the FREE statement is a controlled variable that has been allocated. Another suggestion is to acquire a storage report to check on the program's use of storage. A PLIDUMP should be obtained for later study by IBM.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0SH

IBM0914S ONCODE=*oncode-value* An abnormal termination has occurred in a linked PL/I program while running a CICS transaction.

Explanation: A PL/I program called through EXEC LINK or EXEC XCTL terminated abnormally.

Programmer response: Examine the linked PL/I program unit and correct the error that caused error.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0SI

IBM0915S ONCODE=*oncode-value* An internal error occurred in PL/I library.

Explanation: An error occurred within the PL/I library. The ONCODE associated with this message is 1104.

Programmer response: A PLIDUMP should be obtained for later study by IBM.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0SJ

IBM0916S ONCODE=*oncode-value* An object window was unable to be created.

Explanation: The Presentation Manager returned an error when an attempt was made to create an object window during the execution of a DISPLAY statement or I/O to a Presentation Manager Terminal (PMT).

Programmer response: The problem may be that too many windows have been created. Reduce the number of windows and re-run your program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0SK

IBM0917S ONCODE=*oncode-value* An internal error occurred in PL/I storage management.

Explanation: There was insufficient space available to satisfy a storage allocation request within PL/I storage management. The ONCODE associated with this message is 1106.

Programmer response: Acquire a storage report to check on the program's use of storage. A PLIDUMP should be obtained for later study by IBM.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0SL

IBM0924W Closing a file in the ON-unit caused errors in this statement.

Explanation: An ON-unit for an I/O condition was entered and the file associated with the ON-unit was closed in the ON-unit. A GOTO statement should have been used to exit from the ON-unit. The result of a normal return from an ON-unit is undefined.

Programmer response: Use a GOTO statement to exit from the ON-unit, or close the file outside of the ON-unit.

System action: No system action is performed.

Symbolic Feedback Code: IBM0SS

IBM0925W The PLIRETC value was reduced to 999.

Explanation: The value passed to the PLIRETC built-in procedure was greater than 999. 999 is the maximum allowed user value.

Programmer response: Ensure all PLIRETC values are below 999.

System action: Processing continues with the next sequential statement.

Symbolic Feedback Code: IBM0ST

IBM0926S The CHECKPOINT/RESTART facility is not supported in a CMS environment.

Explanation: An attempt was made to call the CHECKPOINT/RESTART facility from PL/I. CHECKPOINT/RESTART is not supported under CMS. The ERROR condition was raised.

Programmer response: Remove the call to the CHECKPOINT/RESTART facility. If this facility needs to be used, run the application under OS/390.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0SU

IBM0930S ONCODE=*oncode-value* An attempt was made to call a Checkout-compiled program in Language Environment.

Explanation: Checkout-compiled programs are not supported in Language Environment.

Programmer response: Remove the call to the Checkout-compiled program.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0T2

IBM0950S ONCODE=*oncode-value* A system error occurred in PL/I multithreading support for the WAIT statement.

Explanation: An uninitialized task variable may have been specified in the THREAD option. Another reason why an error may have occurred in WAIT is that the operating system may have run out of resources to satisfy the request or may have timed out.

Programmer response: Ensure that the tasking variable has been initialized to a valid value. The ATTACH statement with the THREAD option must be used to give a tasking variable a starting value. Ensure that there are enough resources for the operating system to acquire.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0TM

IBM0951S ONCODE=*oncode-value* A system error occurred in PL/I multithreading support for the DETACH statement.

Explanation: An uninitialized task variable may have been specified in the THREAD option.

Programmer response: Ensure that the tasking variable has been initialized to a valid value. The ATTACH statement with the THREAD option must be used to give a tasking variable a starting value.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0TN

IBM0952S ONCODE=*oncode-value* A system error occurred in PL/I multithreading support for the ATTACH statement.

Explanation: The operating system may have run out of resources (not enough memory, too many handles) to satisfy the request.

Programmer response: Ensure that there are enough resources for the operating system to acquire.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0TO

IBM0953S ONCODE=*oncode-value* A system error occurred in PL/I multithreading support for the STOP statement.

Explanation: An uninitialized task variable may have been specified in the THREAD option.

Programmer response: Ensure that the tasking variable has been initialized to a valid value. The ATTACH statement with the THREAD option must be used to give a tasking variable a starting value.

System action: The ERROR condition is raised.

Symbolic Feedback Code: IBM0TP

Chapter 7. COBOL Run-Time Messages

The messages in this topic pertain to COBOL. Each message is followed by an explanation describing the condition that caused the message, a programmer response suggesting how you might prevent the message from occurring again, and a system action indicating how the system responds to the condition that caused the message.

The messages also contain a symbolic feedback code, which represents the first 8 bytes of a 12-byte condition token. You can think of the symbolic feedback code as the nickname for a condition. As such, the symbolic feedback code can be used in user-written condition handlers to screen for a given condition, even if it occurs at different locations in an application.

The messages in this topic contain alphabetic suffixes that have the following meaning:

I	Informational message
W	Warning message
E	Error message
S	Severe error message
C	Critical error message

IGZ0002S *debugging-information*

Explanation: A SYNAD error has occurred on a QSAM file. The text was supplied by the system SYNADAF routine. Since the debugging information supplied in this message is system specific, the message format differs between CMS and MVS environments. The message issued under MVS consists of the following:

```
IGZ0002S job name, step name, unit address, device
         type, ddname, operation attempted, error
         description, actual track address and
         block number, access method.
```

The message issued under CMS is as follows:

```
IGZ0002S 120S operation
type ERROR nnn ON ddname,
```

Definitions requiring further explanation for the above message formats are:

120S	is the CMS message number for SYNAD errors
operation type	INPUT or OUTPUT
device type	UR for unit record device
	TA for magnetic tape device
	DA for direct access device
nnn	is the associated error code
ddname	is the DDNAME of the related file
operation attempted	actual operation

Programmer response: For more information regarding the CMS message number 120S and related error codes, see *VM/SP System Messages and Codes*. For information on the MVS text of this SYNADAF message, see *z/OS DFSMS Macro Instructions for Data Sets*, SC26-7408, and *z/OS DFSMS Using Data Sets*, SC26-7410.

System action: The application was terminated.

Symbolic Feedback Code: IGZ002

IGZ0003W A logic error occurred for file *file-name* **in program** *program-name* **at relative location** *relative-location***.**

Explanation: This error is usually caused by an I/O operation request that is not valid for the file—for example, a WRITE into a file opened for INPUT, or a START to a VSAM ESDS.

A file status clause was specified or an error declarative statement was active for the file.

Programmer response: Check the operation request and modify the program.

System action: No system action was taken.

Symbolic Feedback Code: IGZ003

IGZ0005S OS/VS COBOL programs in the application were found in multiple enclaves.

Explanation: OS/VS COBOL programs are restricted to one enclave within an application.

Programmer response: Modify the application so that the OS/VS COBOL programs appear in one enclave only.

System action: The application was terminated.

Symbolic Feedback Code: IGZ005

IGZ0006S The reference to table *table-name* **by verb number** *verb-number* **on line** *line-number* **addressed an area outside the region of the table.**

Explanation: When the SSRANGE option is in effect, this message is issued to indicate that a fixed-length table has been subscripted in a way that exceeds the defined size of the table, or, for variable-length tables, the maximum size of the table.

The range check was performed on the composite of the subscripts and resulted in an address outside the region of the table. For variable-length tables, the address is outside the region of the table defined when all OCCURS DEPENDING ON objects are at their maximum values; the ODO object's current value is not considered. The check was not performed on individual subscripts.

Programmer response: Ensure that the value of literal subscripts and/or the value of variable subscripts as evaluated at run-time do not exceed the subscripted dimensions for subscripted data in the failing statement.

System action: The application was terminated.

Symbolic Feedback Code: IGZ006

IGZ0007S The size of variable length group *group-name* **exceeded the maximum defined length of the group at the time of reference by verb number** *verb-number* **on line** *line-number***.**

Explanation: When the SSRANGE option is in effect, this message is issued to indicate that a variable-length group generated by OCCURS DEPENDING ON has a length that is less than zero, or is greater than the limits defined in the OCCURS DEPENDING ON clauses.

The range check was performed on the composite length of the group, and not on the individual OCCURS DEPENDING ON objects.

Programmer response: Ensure that OCCURS DEPENDING ON objects as evaluated at run-time do not exceed the maximum number of occurrences of the dimension for tables within the referenced group item.

System action: The application was terminated.

Symbolic Feedback Code: IGZ007

IGZ0009C A delete of module *module-name* **was unsuccessful.**

Explanation: An attempt to delete a module failed.

Programmer response: See your IBM service representative.

System action: The application was terminated.

Symbolic Feedback Code: IGZ009

IGZ0011C *module-name* **was not a proper module for this system environment.**

Explanation: A library subroutine that is system sensitive is inappropriate for the current system environment. For example, an OS environment specific module has been loaded under CICS. The likely causes are:
- Improper concatenation sequence of partitioned data sets that contain the subroutine library, either during run-time or during link-edit of the COBPAC.
- An attempt to use a function unsupported on the current system (for example, ACCEPT on CICS).

Programmer response: Check for the conditions stated above, and modify the environment or the application as needed.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00B

IGZ0012S There was an invalid attempt to end a sort or merge.

Explanation: A sort or merge initiated by a COBOL program was in progress and one of the following was attempted:
1. A STOP RUN was issued.
2. A GOBACK or an EXIT PROGRAM was issued within the input procedure or the output procedure of the COBOL program that initiated the sort or merge. Note that the GOBACK and EXIT PROGRAM statements are allowed in a program called by an input procedure or an output procedure.
3. A user handler associated with the program that initiated the sort or merge moved the condition handler resume cursor and resumed the application.

Programmer response: Change the application so that it does not use one of the above methods to end the sort or merge.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00C

IGZ0013S An error return code *return-code* **came from a CICS command** *CICS-command* **issued by library subroutine** *library-subroutine***.**

Explanation: An error was encountered when a run-time routine issued a CICS command. The error return code is from the field EIBRESP in the CICS EIB. For more information about the values for the field EIBRESP, see the *CICS/ESA Application Programmer's Reference* , SC33-0676.

Programmer response: Modify your application as required.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00D

IGZ0014W *module-name* **is no longer supported. Its content was ignored.**

Explanation: This message is issued when the run-time detects that IGZETUN or IGZEOPT is linked with the application. IGZETUN and IGZEOPT are ignored when running with Language Environment. CEEUOPT may be used in place of IGZETUN and IGZEOPT.

Programmer response: Remove the explicit INCLUDE of IGZEOPT or IGZETUN during the link-edit step.

System action: No system action was taken.

Symbolic Feedback Code: IGZ00E

IGZ0015S A recursive call was attempted to a program that was already active. The program name is *program-name***.**

Explanation: An illegal recursive entry to an active program is detected. For example, Program A has CALLed Program B, and Program B is CALLing Program A.

Programmer response: Remove the recursive call to *program-name* or specify the IS RECURSIVE phrase on the

PROGRAM-ID statement for the recursively CALLed program. Additionally, if the recursive program is called dynamically, link-edit it with REUS.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00F

IGZ0016W Program *program-name* **could not be deactivated by non-return exit of a routine. Subsequent reentry is not supported.**

Explanation: A COBOL program cannot normally be recursively entered. When non-return style procedure collapse processing is being performed for a COBOL program, an attempt is made to reset the program to a state where it can be recursively entered. This is not supported for certain combinations of function used within the program. After this message is issued, any attempt to reenter the program will result in message IGZ0015S and termination of the enclave.

Programmer response: Do not reenter the program or modify the program to allow it to be successfully reset.

System action: No system action was taken.

Symbolic Feedback Code: IGZ00G

IGZ0017S The open of DISPLAY or ACCEPT file with environment name *environment-name* **was unsuccessful.**

Explanation: An error occurred while opening the DISPLAY/ACCEPT file.

Programmer response: Check to make sure a ddname has been defined for the file.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00H

IGZ0018S On CICS, an attempt was made to run a COBOL program which is not reentrant. The program name is *program-name***.**

Explanation: COBOL programs running on CICS must be reentrant.

Programmer response: In order to make a COBOL program reentrant, compile the COBOL program with the RENT compile-time option.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00I

IGZ0019W A FUNCTION result used as a DELIMITED BY operand is larger than the UNSTRING object in program *program-name* **at displacement** *displacement***. The DELIMITED BY phrase is ignored.**

Explanation: A FUNCTION used as a DELIMITED BY operand was larger than the UNSTRING object.

Programmer response: Check the FUNCTION arguments to ensure that they are not larger than the UNSTRING object.

System action: No system action was taken.

Symbolic Feedback Code: IGZ00J

IGZ0020S A logic error occurred. Neither FILE STATUS nor a declarative was specified for file *file-name* **in program** *program-name* **at relative location** *relative-location***. The status code was** *status-code***.**

Explanation: This error is an I/O error, usually caused by an operation request that is not valid for the file, for example, a WRITE into a file opened for INPUT, or a START to a VSAM ESDS.

No file status clause was specified, and no error declarative was in effect.

Programmer response: Check operation request for the file.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00K

IGZ0021C *macro-name* **was unsuccessful for file** *file-name***.**

Explanation: The execution of an ENDREQ, GENCB, MODCB, SHOWCB, or TESTCB macro failed. This is the result of system or VSAM problems.

Programmer response: See your IBM service representative.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00L

IGZ0022W **File** *file-name* **in program** *program-name* **will return the maximum record length when read.**

Explanation: A VSAM RRDS with a varying record length has been opened for input. The maximum record length will be returned.

Programmer response: None

System action: No system action was taken.

Symbolic Feedback Code: IGZ00M

IGZ0023S **The dynamic allocation of file** *file-name* **was unsuccessful. The return code was** *return-code***. The reason code was** *reason-code***.**

Explanation: An attempt to dynamically allocate a file using DYNALLOC failed, resulting in the indicated return and reason codes.

Programmer response: Review the job stream or filedef to see if any DDNAMES are missing or misspelled. If you can not find any errors, resubmit the job with the CBLQDA(OFF) run-time option and check for any access method messages.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00N

IGZ0024S **An invalid separate sign character was detected in** *program-name* **at displacement** *displacement***.**

Explanation: An operation was attempted on data defined with a separate sign. The value in the sign position was not a plus (+) or a minus (-).

Programmer response: This error might have occurred because of a REDEFINES clause involving the sign position or a group move involving the sign position, or the position was never initialized. Check for these cases. The compiler formatting option TEST(), or equivalent, along with the ABTERMENC() run-time option, can be used to generate a formatted dump of the user data. This dump can then be used to identify the unacceptable data item contents.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00O

IGZ0026W **The SORT-RETURN special register was never referenced, but the current content indicated the sort or merge operation in program** *program-name* **on line number** *line-number* **was unsuccessful.**

Explanation: The COBOL source does not contain any references to the sort-return register. The compiler generates a test after each sort or merge verb. A nonzero return code has been passed back to the program by Sort/Merge.

Programmer response: Determine why the Sort/Merge was unsuccessful and fix the problem. Possible reasons why the Sort/Merge was unsuccessful include:
- There was an error detected by DFSORT. See the DFSORT messages for the reason for the error.
- The SORT-RETURN special register was set to a non-zero value by the application program while in an input procedure or an output procedure.

System action: No system action was taken.

Symbolic Feedback Code: IGZ00Q

IGZ0027W The sort control file could not be opened.

Explanation: An attempt to open the sort control file has failed. Possible reasons for the open failure include:
- A ddname for the sort control file was not provided.
- The IGZSRTCD ddname was provided, but the file associated with the ddname could not be found.

When the sort control file cannot be opened, user-supplied sort control cards will not be passed to Sort/Merge.

The sort control file is optional. On MVS, if you did not provide a ddname for the sort control file (the sort control file name is IGZSRTCD unless it is overridden by changing the value of the SORT-CONTROL special register) you will also get this message: IEC130I 'IGZSRTCD DD STATEMENT MISSING'. This message is informational only.

Programmer response: If you want to pass in sort control cards from the sort control file, verify that the ddname is specified and the file is available.

System action: No system action was taken.

Symbolic Feedback Code: IGZ00R

IGZ0028S An I/O error occurred in sort control file *file-name*.

Explanation: An I/O error was encountered while trying to read the sort control file. Some or all of the user-supplied sort control cards will not be passed to Sort/Merge.

Programmer response: For more information, look at the previous system message you received relating to this I/O error.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00S

IGZ0029S Argument-1 for function *function-name* in program *program-name* at line *line-number* was less than zero.

Explanation: An illegal value for argument-1 was used.

Programmer response: Ensure that argument-1 is greater than or equal to zero.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00T

IGZ0030S Argument-2 for function *function-name* in program *program* at line *line-number* was not a positive integer.

Explanation: An illegal value for argument-2 was used.

Programmer response: Ensure that argument-2 is a positive integer.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00U

IGZ0031S A restart was not possible since the checkpoint record *record-name* was taken while a sort or merge was in progress.

Explanation: An attempt was made to use the restart facility of checkpoint/restart to resume execution of a job from a checkpoint taken by a COBOL program because of a rerun clause during a Sort/Merge operation. Only checkpoints taken by the sort product can be used to restart from a point within the Sort/Merge operation.

The checkpoint record cannot be used for restart.

Programmer response: Use a different checkpoint record. If no other checkpoint records exist, the job cannot be restarted.

System action: The application was terminated.

Symbolic Feedback Code: IGZ00V

IGZ0032S A CANCEL was attempted on active program *program-name***.**

Explanation: An attempt was made to cancel an active program. For example, program A called program B; program B is trying to cancel program A.

Programmer response: Remove the failing CANCEL statement. In order to locate the failing CANCEL statement, rerun the application with TERMTHDACT(TRACE) or (ABEND). Review the traceback information to identify the program that issued the CANCEL.

System action: The application was terminated.

Symbolic Feedback Code: IGZ010

IGZ0033S An attempt was made to pass a parameter address above 16 megabytes to AMODE(24) program *program-name***.**

Explanation: An attempt was made to pass a parameter located above the 16-megabyte storage line to a program in AMODE(24). The called program will not be able to address the parameter.

Programmer response: If the calling program is compiled with the RENT option, the DATA(24) option may be used in the calling program to make sure that its data is located in storage accessible to an AMODE(24) program. If the calling program is compiled with the NORENT option, the RMODE(24) option may be used in the calling program to make sure that its data is located in storage accessible to an AMODE(24) program. Verify that no linkedit, binder or genmod overrides are responsible for this error.

System action: The application was terminated.

Symbolic Feedback Code: IGZ011

IGZ0034W The file with system-name *system-name* **could not be extended. Secondary extents were not specified or were not available. The last WRITE was at offset** *offset* **in program** *program-name***.**

Explanation: There is insufficient space available for an output file. There is no invalid key clause, file status, or user error declarative. This corresponds to the MVS X37 ABEND.

Programmer response: Check the file attributes and if necessary, reallocate the file. Also check data set allocations.

System action: No system action was taken.

Symbolic Feedback Code: IGZ012

IGZ0035S There was an unsuccessful OPEN or CLOSE of file *file-name* **in program** *program-name* **at relative location** *location***. Neither FILE STATUS nor an ERROR declarative were specified. The status code was** *status-code***.**

Explanation: An error has occurred while opening or closing the named file. No file status or user error declarative was specified.

Programmer response: Check to make sure there is a ddname defined for the indicated file.

System action: The application was terminated.

Symbolic Feedback Code: IGZ013

IGZ0036W Truncation of high order digit positions occurred in program *program-name* **on line number** *line-number***.**

Explanation: The generated code has truncated an intermediate result (that is, temporary storage used during an arithmetic calculation) to 30 digits; some of the truncated digits were not 0.

Programmer response: See *COBOL for OS/390 & VM Programming Guide* or *COBOL for MVS & VM Programming Guide* for a description of intermediate results.

System action: No system action was taken.

Symbolic Feedback Code: IGZ014

IGZ0037S The flow of control in program *program-name* proceeded beyond the last line of the program.

Explanation: The program did not have a terminator (STOP, GOBACK, or EXIT), and control fell through the last instruction.

Programmer response: Check the logic of the program. Sometimes this error occurs because of one of the following logic errors:

- The last paragraph in the program was only supposed to receive control as the result of a PERFORM statement, but due to a logic error it was branched to by a GO TO statement.
- The last paragraph in the program was executed as the result of a “fall-through” path, and there was no statement at the end of the paragraph to end the program.

System action: The application was terminated.

Symbolic Feedback Code: IGZ015

IGZ0038S A reference modification length value of *reference-modification-value* on line *line-number* which was not equal to 1 was found in a reference to data item *data-item* which was passed by value.

Explanation: The length value in a reference modification specification was not equal to 1. The length value must be equal to 1.

Programmer response: Check the indicated line number in the program to ensure that any reference modified length values are (or will resolve to) 1.

System action: The application was terminated.

Symbolic Feedback Code: IGZ016

IGZ0039S An invalid overpunched sign was detected in program *program-name* on line *line-number*.

Explanation: An operation was attempted on data defined with an overpunched sign. The value in the sign was not valid.

Programmer response: This error might have occurred because of a REDEFINES clause involving the sign position or a group move involving the sign position, or the position was never initialized. Check for the above cases.

System action: The application was terminated.

Symbolic Feedback Code: IGZ017

IGZ0040S An invalid separate sign was detected in program *program-name* on line *line-number*.

Explanation: An operation was attempted on data defined with a separate sign. The value in the sign position was not a plus (+) or a minus (-).

Programmer response: This error might have occurred because of a REDEFINES clause involving the sign position or a group move involving the sign position, or the position was never initialized. Check for the above cases.

System action: The application was terminated.

Symbolic Feedback Code: IGZ018

IGZ0041W The warning message limit was exceeded. Further warning messages were suppressed.

Explanation: The limit on warning messages is 256. This constraint on the number of warning messages prevents a looping program from flooding the system buffers.

Programmer response: Correct the situations causing the warning messages or correct the looping problem.

System action: No system action was taken.

Symbolic Feedback Code: IGZ019

IGZ0042C There was an attempt to use the IGZBRDGE macro, but the calling program was not COBOL.

Explanation: A non-COBOL program attempted to call a COBOL program using the IGZBRDGE interface. COBOL/370 could not find a COBOL environment.

Programmer response: Do not call an entry point specified via the IGZBRDGE macro from a non-COBOL program.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01A

IGZ0044S There was an attempt to call the COBOL main program *program-name* **that was not in initial state.**

Explanation: You will receive this message if you attempt to enter a NONREENTRANT COBOL/370, VS COBOL II, COBOL for MVS & VM, or COBOL for OS/390 & VM main program more than once. This is a nonstandard entry attempt.

Programmer response: Modify the application so that the non-reentrant COBOL main program won't be called more than once.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01C

IGZ0045S Unable to invoke method *method-name* **on line number** *line number* **in COBOL program** *program-name***.**

Explanation: The specific method is not supported for the class of the current object reference.

Programmer response: Check the indicated line number in the program to ensure that the class of the current object reference supports the method being invoked.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01D

IGZ0046W The value specified in the program for the *special-register* **special register was overridden by the corresponding value in the sort control file.**

Explanation: A nondefault value for the SORT special register specified in the message was used in a program, but a value in the SORT control file which corresponds to that SORT special register was found. The value in the SORT control file was used, and the value in the SORT special register was ignored.

Programmer response: See *COBOL for OS/390 & VM Programming Guide* or *COBOL for MVS & VM Programming Guide* for a description of SORT special registers and the SORT control file.

System action: No system action was taken.

Symbolic Feedback Code: IGZ01E

IGZ0047S Unable to invoke method *method-name* **on line number** *line number* **in COBOL class** *class-name***.**

Explanation: The specific method is not supported for the class of the current object reference.

Programmer response: Check the indicated line number in the class to ensure that the class of the current object reference supports the method being invoked.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01F

IGZ0048W A negative base was raised to a fractional power in an exponentiation expression in program *program-name* **at displacement** *displacement***. The absolute value of the base was used.**

Explanation: A negative number raised to a fractional power occurred in a library routine.

The value of a negative number raised to a fractional power is undefined in COBOL. If a SIZE ERROR clause had appeared on the statement in question, the SIZE ERROR imperative would have been used. However, no SIZE

ERROR clause was present, so the absolute value of the base was used in the exponentiation.

Programmer response: Ensure that the program variables in the failing statement have been set correctly.

System action: No system action was taken.

Symbolic Feedback Code: IGZ01G

IGZ0049W A zero base was raised to a zero power in an exponentiation expression in program *program-name* **at displacement** *displacement***. The result was set to one.**

Explanation: The value of zero raised to the power zero occurred in a library routine.

The value of zero raised to the power zero is undefined in COBOL. If a SIZE ERROR clause had appeared on the statement in question, the SIZE ERROR imperative would have been used. However, no SIZE ERROR clause was present, so the value returned was one.

Programmer response: Ensure that the program variables in the failing statement have been set correctly.

System action: No system action was taken.

Symbolic Feedback Code: IGZ01H

IGZ0050S A zero base was raised to a negative power in an exponentiation expression in program *program-name* **at displacement** *displacement***.**

Explanation: The value of zero raised to a negative power occurred in a library routine.

The value of zero raised to a negative number is not defined. If a SIZE ERROR clause had appeared on the statement in question, the SIZE ERROR imperative would have been used. However, no SIZE ERROR clause was present.

Programmer response: Ensure that the program variables in the failing statement have been set correctly.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01I

IGZ0051S An invalid EBCDIC digit string was detected on conversion to floating point in *program-name* **at displacement** *displacement***.**

Explanation: The input to the conversion routine contained invalid EBCDIC data.

Programmer response: Ensure that the program variables in the failing statement have been set correctly.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01J

IGZ0052C An internal error or invalid parameters were detected in the floating point conversion routine called from *program-name* **at displacement** *displacement***.**

Explanation: None

Programmer response: See your IBM service representative.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01K

IGZ0053S An overflow occurred on conversion to floating point in *program-name* **at displacement** *displacement***.**

Explanation: A number was generated in the program that is too large to be represented in floating point.

Programmer response: You need to modify the program appropriately to avoid an overflow.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01L

IGZ0054W An overflow occurred on conversion from floating point to fixed point in *program-name* at displacement *displacement*. The result was truncated.

Explanation: The result of a conversion to fixed point from floating point contains more digits than will fit in the fixed point receiver. The high order digits were truncated.

Programmer response: No action is necessary, although you may want to modify the program to avoid an overflow.

System action: No system action was taken.

Symbolic Feedback Code: IGZ01M

IGZ0055W An underflow occurred on conversion to floating point in *program-name* at displacement *displacement*. The result was set to zero.

Explanation: On conversion to floating point, the negative exponent exceeded the limit of the hardware. The floating point value was set to zero.

Programmer response: No action is necessary, although you may want to modify the program to avoid an underflow.

System action: No system action was taken.

Symbolic Feedback Code: IGZ01N

IGZ0056W One or more files were not closed by program *program-name* before program termination.

Explanation: The specified program has finished but has not closed all of the files it opened. COBOL attempts to clean up storage and closes any open files.

Programmer response: Check that all files are closed before the program terminates.

System action: No system action was taken.

Symbolic Feedback Code: IGZ01O

IGZ0057S There was an attempt to initialize a reusable environment through ILBOSTP0, but either the enclave was not the first enclave or COBOL was not the main program of the already established enclave.

Explanation: A request to establish a reusable environment through ILBOSTP0 can only occur at the beginning of the application. Examples when this error can occur:

- PL/I program calls ASSEMBLE program which calls ILBOSTP0.
- Language Environment enabled ASSEMBLER program calls ILBOSTP0.

Programmer response: Only invoke ILBOSTP0 before calling any program within the application that brings up Language Environment.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01P

IGZ0058S Exponent overflow occurred in program *program-name* at displacement *displacement*.

Explanation: Floating point exponent overflow occurred in a library routine.

Programmer response: Ensure that the program variables in the failing statement have been set correctly.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01Q

IGZ0059W An exponent with more than nine digits was truncated in program *program-name* **at displacement** *displacement***.**

Explanation: Exponents in fixed point exponentiations may not contain more than nine digits. The exponent was truncated back to nine digits; some of the truncated digits were not 0.

Programmer response: No action is necessary, although you may want to adjust the exponent in the failing statement.

System action: No system action was taken.

Symbolic Feedback Code: IGZ01R

IGZ0060W Truncation of high order digit positions occurred in program *program-name* **at displacement** *displacement***.**

Explanation: Code in a library routine has truncated an intermediate result (that is, temporary storage used during an arithmetic calculation) back to 30 digits; some of the truncated digits were not 0.

Programmer response: See *COBOL for OS/390 & VM Programming Guide* or *COBOL for MVS & VM Programming Guide* for a description of intermediate results.

System action: No system action was taken.

Symbolic Feedback Code: IGZ01S

IGZ0061S Division by zero occurred in program *program-name* **at displacement** *displacement***.**

Explanation: Division by zero occurred in a library routine. Division by zero is not defined. If a SIZE ERROR clause had appeared on the statement in question, the SIZE ERROR imperative would have been used. However, no SIZE ERROR clause was present.

Programmer response: Ensure that the program variables in the failing statement have been set correctly.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01T

IGZ0063S An invalid sign was detected in a numeric edited sending field in *program-name* **on line number** *line-number***.**

Explanation: An attempt has been made to move a signed numeric edited field to a signed numeric or numeric edited receiving field in a MOVE statement. However, the sign position in the sending field contained a character that was not a valid sign character for the corresponding PICTURE.

Programmer response: Ensure that the program variables in the failing statement have been set correctly.

System action: The application was terminated.

Symbolic Feedback Code: IGZ01V

IGZ0064S A recursive call to active program *program-name* **in compilation unit** *compilation-unit* **was attempted.**

Explanation: COBOL does not allow reinvocation of an internal program which has begun execution, but has not yet terminated. For example, if internal programs A and B are siblings of a containing program, and A calls B and B calls A, this message will be issued.

Programmer response: Examine your program to eliminate calls to active internal programs.

System action: The application was terminated.

Symbolic Feedback Code: IGZ020

IGZ0065S A CANCEL of active program *program-name* **in compilation unit** *compilation-unit* **was attempted.**

Explanation: An attempt was made to cancel an active internal program. For example, if internal programs A and B are siblings in a containing program and A calls B and B cancels A, this message will be issued.

Programmer response: Examine your program to eliminate cancellation of active internal programs.

System action: The application was terminated.

Symbolic Feedback Code: IGZ021

IGZ0066S The length of external data record *data-record* **in program** *program-name* **did not match the existing length of the record.**

Explanation: While processing External data records during program initialization, it was determined that an External data record was previously defined in another program in the run-unit, and the length of the record as specified in the current program was not the same as the previously defined length.

Programmer response: Examine the current file and ensure the External data records are specified correctly.

System action: The application was terminated.

Symbolic Feedback Code: IGZ022

IGZ0067S The NOEQUALS keyword in the sort control file *file-name* **conflicted with the specifications of the DUPLICATES phrase on the SORT statement.**

Explanation: A sort control file with an OPTION card specifying the NOEQUALS keyword was used for a SORT which had the DUPLICATES IN ORDER phrase specified. The NOEQUALS keyword and the DUPLICATES phrase conflict.

Programmer response: Either remove the NOEQUALS keyword from the sort control file or remove the DUPLICATES IN ORDER phrase from the SORT statement.

System action: The application was terminated.

Symbolic Feedback Code: IGZ023

IGZ0068W Duplicate characters were ignored in an INSPECT CONVERTING statement in program *program-name* **at displacement** *displacement***.**

Explanation: The same character appeared more than once in the identifier that contained the characters to be converted in an INSPECT CONVERTING statement. The first occurrence of the character, and the corresponding character in the replacement field, are used, and subsequent occurrences are not used.

Programmer response: Duplicate characters in the indicated INSPECT statement may be deleted; programmer action is not required.

System action: No system action was taken.

Symbolic Feedback Code: IGZ024

IGZ0069S On VM, file *file-name* **in program** *program-name* **attempted to use VSAM in XA or ESA mode. Using VSAM while in XA or ESA mode is not supported under the installed level of VM. The program was terminated.**

Explanation: VSAM can only operate in S/370 mode virtual machines on VM/SP XA and VM/ESA Release 1 ESA feature. The job was cancelled. Only on VM/ESA Release 1.1 (CMS8), and higher releases, can VSAM and VS COBOL II be used in XA-mode and XC-mode virtual machines.

Programmer response: See your systems programmer for assistance.

System action: The application was terminated.

Symbolic Feedback Code: IGZ025

IGZ0070S The FILEDEF command ″FILEDEF *ddname* DISK FILE *ddname* A4″ was unsuccessful.

Explanation: An attempt at dynamic allocation for CMS file *ddname* using the FILEDEF command has failed.

Programmer response: See your systems programmer for assistance.

System action: The application was terminated.

Symbolic Feedback Code: IGZ026

IGZ0071S ALL subscripted table reference to table *table-name* by verb number *verb-number* on line *line-number* had an ALL subscript specified for an OCCURS DEPENDING ON dimension, and the object was less than or equal to 0.

Explanation: When the SSRANGE option is in effect, this message is issued to indicate that there are 0 occurrences of dimension subscripted by ALL.

The check is performed against the current value of the OCCURS DEPENDING ON OBJECT.

Programmer response: Ensure that ODO object(s) of ALL-subscripted dimensions of any subscripted items in the indicated statement are positive.

System action: The application was terminated.

Symbolic Feedback Code: IGZ027

IGZ0072S A reference modification start position value of *reference-modification-value* on line *line-number* referenced an area outside the region of data item *data-item*.

Explanation: The value of the starting position in a reference modification specification was less than 1, or was greater than the current length of the data item that was being reference modified. The starting position value must be a positive integer less than or equal to the number of characters in the reference modified data item.

Programmer response: Check the value of the starting position in the reference modification specification.

System action: The application was terminated.

Symbolic Feedback Code: IGZ028

IGZ0073S A non-positive reference modification length value of *reference-modification-value* on line *line-number* was found in a reference to data item *data-item*.

Explanation: The length value in a reference modification specification was less than or equal to 0. The length value must be a positive integer.

Programmer response: Check the indicated line number in the program to ensure that any reference modified length values are (or will resolve to) positive integers.

System action: The application was terminated.

Symbolic Feedback Code: IGZ029

IGZ0074S A reference modification start position value of *reference-modification-value* and length value of *length* on line *line-number* caused reference to be made beyond the rightmost character of data item *data-item*.

Explanation: The starting position and length value in a reference modification specification combine to address an area beyond the end of the reference modified data item. The sum of the starting position and length value minus one must be less than or equal to the number of characters in the reference modified data item.

Programmer response: Check the indicated line number in the program to ensure that any reference modified start and length values are set such that a reference is not made beyond the rightmost character of the data item.

System action: The application was terminated.

Symbolic Feedback Code: IGZ02A

IGZ0075S **Inconsistencies were found in EXTERNAL file** *file-name* **in program** *program-name***. The following file attributes did not match those of the established external file:** *attribute-1 attribute-2 attribute-3 attribute-4 attribute-5 attribute-6 attribute-7*

Explanation: One or more attributes of an external file did not match between two programs that defined it.

Programmer response: Correct the external file. For a summary of file attributes which must match between definitions of the same external file, see *IBM COBOL Language Reference*

System action: The application was terminated.

Symbolic Feedback Code: IGZ02B

IGZ0076W **The number of characters in the INSPECT REPLACING CHARACTERS BY data-name in program** *program-name* **at displacement** *displacement* **was not equal to one. The first character was used.**

Explanation: A data item which appears in a CHARACTERS phrase within a REPLACING phrase in an INSPECT statement must be defined as being one character in length. Because of a reference modification specification for this data item, the resultant length value was not equal to one. The length value is assumed to be one.

Programmer response: You may correct the reference modification specifications in the failing INSPECT statement to ensure that the reference modification length is (or will resolve to) 1; programmer action is not required.

System action: No system action was taken.

Symbolic Feedback Code: IGZ02C

IGZ0077W **The lengths of the** *data-item* **items in program** *program-name* **at displacement** *displacement* **were not equal. The shorter length was used.**

Explanation: The two data items which appear in a REPLACING or CONVERTING phrase in an INSPECT statement must have equal lengths, except when the second such item is a figurative constant. Because of the reference modification for one or both of these data items, the resultant length values were not equal. The shorter length value is applied to both items, and execution proceeds.

Programmer response: You may adjust the operands of unequal length in the failing INSPECT statement; programmer action is not required.

System action: No system action was taken.

Symbolic Feedback Code: IGZ02D

IGZ0078S **ALL subscripted table reference to table** *table-name* **by verb number** *verb-number* **on line** *line-number* **will exceed the upper bound of the table.**

Explanation: When the SSRANGE option is in effect, this message is issued to indicate that a multi-dimension table with ALL specified as one or more of the subscripts will result in a reference beyond the upper limit of the table.

The range check was performed on the composite of the subscripts and the maximum occurrences for the ALL subscripted dimensions. For variable-length tables the address is outside the region of the table defined when all OCCURS DEPENDING ON objects are at their maximum values; the ODO object's current value is not considered. The check was not performed on individual subscripts.

Programmer response: Ensure that OCCURS DEPENDING ON objects as evaluated at run-time do not exceed the maximum number of occurrences of the dimension for table items referenced in the failing statement.

System action: The application was terminated.

Symbolic Feedback Code: IGZ02E

IGZ0079S **On CICS,** *program-lang* **program** *program-name* **attempted to call OS/VS COBOL program** *program-name***.**

Explanation: On CICS, a COBOL/370, VS COBOL II, COBOL for MVS & VM, or COBOL for OS/390 & VM program attempted to call an OS/VS COBOL program with the CALL statement. Using the CALL statement to perform calls between the following are not not supported on CICS:

- COBOL for OS/390 & VM programs and OS/VS COBOL programs

- COBOL for MVS & VM programs and OS/VS COBOL programs
- COBOL/370 programs and OS/VS COBOL programs
- VS COBOL II programs and OS/VS COBOL programs

Programmer response: If you need to invoke an OS/VS COBOL program from a COBOL/370, VS COBOL II, COBOL for MVS & VM, or COBOL for OS/390 & VM program use EXEC CICS LINK.

System action: The application was terminated.

Symbolic Feedback Code: IGZ02F

IGZ0080S A dynamic call to *module-name* failed because the program entry name *program-name* does not match.

Explanation: If a program compiled with the PGMNAME(LONGUPPER) or the PGMNAME(LONGMIXED) option is dynamically called, the program name must be identical to the name of the module that contains it. If an alternate entry name is called, the entry name must be identical to the ALIAS name representing that entry point. Note that the program entry name can not exceed 8 bytes and must be entirely upper-case.

Programmer response: The name of the program failing the dynamic call, must be modified to comply with the rules state aboved. Otherwise, only static calls to the program are permitted.

System action: The application was terminated.

Symbolic Feedback Code: IGZ02G

IGZ0096C A load of module *module-name* was unsuccessful.

Explanation: An attempt to load a module failed. The module was not available or a system load failure occurred.

Programmer response: See your systems programmer for assistance.

System action: The application was terminated.

Symbolic Feedback Code: IGZ030

IGZ0097S Argument-1 for function *function-name* in program *program-name* at displacement *displacement* contained no digits.

Explanation: Argument-1 for the indicated function must contain at least 1 digit.

Programmer response: Adjust the number of digits in Argument-1 in the failing statement.

System action: The application was terminated.

Symbolic Feedback Code: IGZ031

IGZ0098C The message text for message *message-number* was inaccessible to IGZCWTO.

Explanation: The message text module used by IGZCWTO did not contain message text for the indicated message number.

Programmer response: See your IBM service representative.

System action: The application was terminated.

Symbolic Feedback Code: IGZ032

IGZ0099C Internal error *error-code* was detected in module *module-name*.

Explanation: An unrecoverable error was detected in run-time module *module-name*.

When the module name in the message is IGZCXCC, the error-code indicates the error as described below:

Error-code	**Description**
1	The COBOL environment is not initialized. The COBOL environment must be initialized before calling IGZCXCC.

2	An invalid function code was passed to IGZCXCC.
3	An invalid name length was passed to IGZCXCC.
4	IGZCXCC detected that a nested enclave should be created.
5	IGZCXCC cannot be called when running on CICS.

When the module name in the message is IGZCLNC, IGZCLNK, or IGZCFCC, the error-code indicates the error as described below:

Error-code	**Description**
9	IGZCXCC is being used and an invalid cancel was attempted.

When the module name in the message is IGZEINI, the error-code indicates the error as described below:

Error-code	**Description**
101	There was an attempt to initialize a VS COBOL II or OS/VS COBOL program as a subprogram before the main program has run.
102	An OS/VS COBOL program is being initialized but the TGT address was not passed.

When the module name in the message is IGZCII1, the error-code indicates the error as described below:

Error-code	**Description**
NOTMAIN1	Subprogram initialization occurred when main program initialization was expected.
MAINCLAS	COBOL class initialization occurred when main program initialization was expected.
INVSTRC1	Invalid threading status.
INVST001	Invalid program initialization state.
INVST002	Invalid program initialization state.
INVST003	Invalid program initialization state.
PGMIIP01	Program initialization occurred when another program was in the process of being initialized.

When the module name in the message is IGZCII2, the error-code indicates the error as described below: :

Error-code	**Description**
INVSIG01	During class initialization, the initialization in-progress count is negative.
INVSIG02	During program initialization, the initialization in-progress count is not one.
INVSIG03	During program initialization, the initialization in-progress count is not zero.
INVST001	Invalid program initialization state.

Programmer response: See your IBM service representative.

System action: The application was terminated.

Symbolic Feedback Code: IGZ033

IGZ0100S **Argument-1 for function** *function* **in program** *program* **at displacement** *displacement* **was less than or equal to -1.**

Explanation: An illegal value was used for Argument-1.

Programmer response: Ensure that argument-1 is greater than -1.

System action: The application was terminated.

Symbolic Feedback Code: IGZ034

IGZ0108S The cancel of program *program-name* **failed because the module load point address was not provided when the program was loaded.**

Explanation: In a Language Environment/370 preinitialized environment users may specify their own load service routine. If this routine fails to provide the module load point address as an output parameter when loading a COBOL program, that program can not be cancelled using COBOL'S CANCEL statement.

Programmer response: Modify the user load service to provide the module load point address.

System action: The application was terminated.

Symbolic Feedback Code: IGZ03C

IGZ0151S Argument-1 for function *function-name* **in program** *program-name* **at displacement** *displacement* **contained more than 18 digits.**

Explanation: The total number of digits in argument-1 of the indicated function exceeded 18 digits.

Programmer response: Adjust the number of digits in argument-1 in the failing statement.

System action: The application was terminated.

Symbolic Feedback Code: IGZ04N

IGZ0152S Invalid character *character* **was found in column** *column-number* **in argument-1 for function** *function-name* **in program** *program-name* **at displacement** *program-displacement***.**

Explanation: A non-digit character other than a decimal point, comma, space or sign (+,-,CR,DB) was found in argument-1 for NUMVAL/NUMVAL-C function.

Programmer response: Correct argument-1 for NUMVAL or NUMVAL-C in the indicated statement.

System action: The application was terminated.

Symbolic Feedback Code: IGZ04O

IGZ0154S A procedure pointer was set to nested program *nested-program-name* **in program** *program-name* **at displacement** *displacement***.**

Explanation: Procedure pointers can not be set to a nested program.

Programmer response: Make sure that the procedure program is set to an external program.

System action: The application was terminated.

Symbolic Feedback Code: IGZ04Q

IGZ0155S Invalid character *character* **was found in column** *column-number* **in argument-2 for function** *function-name* **in program** *program-name* **at displacement** *program-displacement***.**

Explanation: Illegal character was found in argument-2 for NUMVAL-C function.

Programmer response: Check that the function argument does follow the syntax rules.

System action: The application was terminated.

Symbolic Feedback Code: IGZ04R

IGZ0156S Argument-1 for function *function-name* **in program** *program-name* **at line** *line-number* **was less than zero or greater than 28.**

Explanation: Input argument to function FACTORIAL is greater than 28 or less than 0.

Programmer response: Check that the function argument is only one byte long.

System action: The application was terminated.

Symbolic Feedback Code: IGZ04S

IGZ0157S The length of Argument-1 for function *function-name* **in program** *program-name* **at line** *line-number* **was not equal to 1.**

Explanation: The length of input argument to ORD function is not 1.

Programmer response: Check that the function argument is only one byte long.

System action: The application was terminated.

Symbolic Feedback Code: IGZ04T

IGZ0158S The length of Argument-1 for function *function-name* **in program** *program-name* **at displacement** *displacement* **was zero.**

Explanation: The length of the argument of the REVERSE, the UPPER-CASE or the LOWER-CASE function is zero.

Programmer response: Make sure that the length of the argument is greater than zero.

System action: The application was terminated.

Symbolic Feedback Code: IGZ04U

IGZ0159S Argument-1 for function *function-name* **in program** *program-name* **at line** *line-number* **was less than 1 or greater than 3067671.**

Explanation: The input argument to DATE-OF-INTEGER or DAY-OF-INTEGER function is less than 1 or greater than 3067671.

Programmer response: Check that the function argument is in the valid range.

System action: The application was terminated.

Symbolic Feedback Code: IGZ04V

IGZ0160S Argument-1 for function *function-name* **in program** *program-name* **at line** *line-number* **was less than 16010101 or greater than 99991231.**

Explanation: The input argument to function INTEGER-OF-DATE is less than 16010101 or greater than 99991231.

Programmer response: Check that the function argument is in the valid range.

System action: The application was terminated.

Symbolic Feedback Code: IGZ050

IGZ0161S Argument-1 for function *function-name* **in program** *program-name* **at line** *line-number* **was less than 1601001 or greater than 9999365.**

Explanation: The input argument to function INTEGER-OF-DAY is less than 1601001 or greater than 9999365.

Programmer response: Check that the function argument is in the valid range.

System action: The application was terminated.

Symbolic Feedback Code: IGZ051

IGZ0162S Argument-1 for function *function-name* **in program** *program-name* **at line** *line-number* **was less than 1 or greater than the number of positions in the program collating sequence.**

Explanation: The input argument to function CHAR is less than 1 or greater than the highest ordinal position in the program collating sequence.

Programmer response: Check that the function argument is in the valid range.

System action: The application was terminated.

Symbolic Feedback Code: IGZ052

IGZ0163S Argument-1 for function *function-name* in program *program-name* at line *line-number* was less than zero.

Explanation: The input argument to function RANDOM is less than 0.

Programmer response: Correct the argument for function RANDOM in the failing statement.

System action: The application was terminated.

Symbolic Feedback Code: IGZ053

IGZ0164C *module-name* was unable to get HEAP storage.

Explanation: The request made to obtain heap storage failed.

Programmer response: See your IBM service representative.

System action: The application was terminated.

Symbolic Feedback Code: IGZ054

IGZ0165S A reference modification start position value of *start-position-value* on line *line* referenced an area outside the region of the function result of *function-result*.

Explanation: The value of the starting position in a reference modification specification was less than 1, or was greater than the current length of the function result that was being reference modified. The starting position value must be a positive integer less than or equal to the number of characters in the reference modified function result.

Programmer response: Check the value of the starting position in the reference modification specification and the length of the actual function result.

System action: The application was terminated.

Symbolic Feedback Code: IGZ055

IGZ0166S A non-positive reference modification length value of *length* on line *line-number* was found in a reference to the function result of *function-result*.

Explanation: The length value in a reference modification specification for a function result was less than or equal to 0. The length value must be a positive integer.

Programmer response: Check the length value and make appropriate correction.

System action: The application was terminated.

Symbolic Feedback Code: IGZ056

IGZ0167S A reference modification start position value of *start-position* and length value of *length* on line *line-number* caused reference to be made beyond the rightmost character of the function result of *function-result*.

Explanation: The starting position and length value in a reference modification specification combine to address an area beyond the end of the reference modified function result. The sum of the starting position and length value minus one must be less than or equal to the number of characters in the reference modified function result.

Programmer response: Check the length of the reference modification specification against the actual length of the function result and make appropriate corrections.

System action: The application was terminated.

Symbolic Feedback Code: IGZ057

IGZ0168S The creation of a second enclave within a reusable environment was attempted. The first program of the second enclave was *program-name*.

Explanation: Reusable environment support is limited to a single enclave. The enclave must be the first enclave.

Programmer response: Modify the application so that it can run within a single enclave with the COBOL reusable environment. If the program name printed is ″????????″ then the first program of the second enclave is not COBOL.

System action: The application was terminated.

Symbolic Feedback Code: IGZ058

IGZ0169W External data *data-record* **was allocated within the 31-bit address range. The called program** *program-name* **contained a definition for this external data, and it was compiled with the DATA(24) option.**

Explanation: External data was allocated ANYWHERE within the 31-bit addressing range by a program. But a subsequently called program containing a definition for that same external data was compiled with the DATA(24) option. This was discovered while processing external data records during program initialization.

Programmer response: Re-compile program with the DATA(31) option if appropriate. If the external data needs to be allocated below 16M, then the FIRST program in the rununit that contains a definition of the external data must be compiled with the DATA(24) option.

System action: No system action was taken.

Symbolic Feedback Code: IGZ059

IGZ0170S One or more files were not closed by NORENT program *program-name* **and the program cannot be found in storage.**

Explanation: The specified NORENT program has not closed all of the files it opened and the program cannot be found in storage. COBOL is unable to close the files because the required control blocks which reside in the program are no longer available. Unpredictable results will occur when the system attempts to close the files. This error can occur if the application has an assembler program that loads and deletes the specified NORENT program.

Programmer response: Ensure that all files are closed by the NORENT program.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05A

IGZ0172W RTEREUS was specified, but ignored. A reusable run-time environment was not established because the first program in the application was not COBOL.

Explanation: A reusable environment can be established only when the main program of the first enclave is COBOL.

Programmer response: Ensure that RTEREUS is off. The performance benefits of using RTEREUS are available without the run-time option when the application is running under Language Environment.

System action: No system action is taken.

Symbolic Feedback Code: IGZ05C

IGZ0173S There was an invalid attempt to start a sort or merge.

Explanation: A sort or merge initiated by a COBOL program was already in progress when another sort or merge was attempted by another COBOL program. Only one sort or merge can be active at a time.

Programmer response: Change the application so that it does not initiate another sort or merge from within the COBOL sort exists.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05D

IGZ0174S A dynamic call to *module-name* **failed because the load module is a DLL.**

Explanation: A COBOL dynamic call cannot be made to a load module that is a DLL. A load module that is a DLL contains one or more of the following:

- A COBOL for OS/390 & VM program compiled with the DLL option and the EXPORTALL option.
- A C routine compiled with the DLL option that exports functions or variables.

- A C++ routine that exports functions or variables.

Programmer response: Change the dynamically called load module so that it does not contain routines that export functions or variables. If the load module contains COBOL for OS/390 & VM programs compiled with the DLL and the EXPORTALL options, recompile the programs with NOEXPORTALL.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05E

IGZ0175S A dynamic call to *module-name* failed because the entry point is a COBOL program compiled with the DLL compiler option.

Explanation: A COBOL dynamic call cannot be made to a COBOL for OS/390 & VM program that is compiled with the DLL compiler option.

Programmer response: Compile the COBOL for OS/390 & VM program with the NODLL compiler option.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05F

IGZ0176S A call from a COBOL program compiled with the DLL compiler option failed because the program *program-name* was previously dynamically called by a COBOL program compiled without the DLL compiler option.

Explanation: When dynamically calling a COBOL program, insure that the DLL compiler option is consistent between calling and called programs.

Programmer response: Compile both the calling and called COBOL for OS/390 & VM programs with either the DLL or the NODLL compiler option.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05G

IGZ0177S A CANCEL of DLL *program-name* is not allowed.

Explanation: The program was called with a CALL identifier statement from a COBOL program compiled with the DLL option. This caused the called program to be identified as a DLL. A DLL cannot be cancelled.

Programmer response: Do not request that a DLL be cancelled.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05H

IGZ0178S An attempt to find program *program-name* in DLL *module-name* was unsuccessful.

Explanation: An error during the load of a DLL or during a query DLL function request prevented an entry point address from being returned.

Programmer response: See the corresponding CEEnnnnI message for additional information and the details of the problem. If the CEEnnnn message is not found in the MSGFILE insure that the runtime option INFOMSGFILTER is OFF.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05I

IGZ0179S A dynamic call to *module-name* failed because the load module contains one or more routines with XPLINK linkage.

Explanation: A COBOL dynamic call cannot be made to a load module that contains routines with XPLINK linkage.

Programmer response: Change the dynamically called load module so that it does not contain routines that use XPLINK linkage.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05J

IGZ0180S An attempt was made to run a VS COBOL II or OS/VS COBOL program in a OS/390 UNIX process. The program name is *program-name*.

Explanation: VS COBOL II and OS/VS COBOL programs cannot be run in a z/OS UNIX process.

Programmer response: Compile the program with COBOL for MVS & VM or COBOL for OS/390 & VM.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05K

IGZ0181S An attempt was made to run a COBOL program that is not reentrant in a OS/390 UNIX process. The program name is *program-name*.

Explanation: COBOL programs running in a z/OS UNIX process must be reentrant.

Programmer response: In order to make a COBOL program reentrant, compile the COBOL program with the RENT compile-time option.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05L

IGZ0182W A fork() is not allowed when a COBOL reusable environment is active.

Explanation: A COBOL reusable environment is active and the fork() function was called. A COBOL reusable environment is established by doing one of the following:

- Using the RTEREUS run-time option
- Calling ILBOSTP0
- Calling IGZERRE

Programmer response: Change the application so that a COBOL reusable environment is not used.

System action: The fork() function is not performed.

Symbolic Feedback Code: IGZ05M

IGZ0183W A fork() is not allowed when an OS/VS COBOL program or a VS COBOL II program is in the environment.

Explanation: At least one OS/VS COBOL program or VS COBOL II program is in the environment and the fork() function was called.

Programmer response: Compile all OS/VS COBOL programs and the VS COBOL II programs with COBOL for MVS & VM or COBOL for OS/390 & VM.

System action: The fork() function is not performed.

Symbolic Feedback Code: IGZ05N

IGZ0184W A fork() is not allowed when a sort or merge is in progress.

Explanation: A SORT or MERGE statement is in progress and the fork() function was called.

Programmer response: Change the application to call fork() when sort or merge is not active.

System action: The fork() function is not performed.

Symbolic Feedback Code: IGZ05O

IGZ0185W A fork() is not allowed when a declarative in a COBOL program is active.

Explanation: A declarative in a COBOL program is active and the fork() function was called.

Programmer response: Change the application to call fork() when a declarative is not active.

System action: The fork() function is not performed.

Symbolic Feedback Code: IGZ05P

IGZ0186S An attempt was made to run a VS COBOL II program with the run-time option XPLINK(ON). The program name is *program-name*.

Explanation: Run-time option XPLINK(OFF) must be specified to run VS COBOL II programs.

Programmer response: Set the XPLINK run-time option to OFF and remove any load modules from the application that use XPLINK linkage, or compile the COBOL program with COBOL for MVS & VM or COBOL for OS/390 & VM.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05Q

IGZ0187S There was an attempt to establish a COBOL reusable environment with the run-time option XPLINK(ON).

Explanation: A COBOL reusable environment cannot be established when the XPLINK(ON) run-time option is specified. A COBOL reusable environment is established by doing one of the following:

- Using the RTEREUS run-time option
- Calling ILBOSTP0
- Calling IGZERRE

Programmer response: Set the XPLINK run-time option to OFF and remove any load modules from the application that use XPLINK linkage, or do not use a COBOL reusable environment.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05R

IGZ0188S Value *string* is invalid for environment variable _IGZ_SYSOUT.

Explanation: Allowable values for environment variable _IGZ_SYSOUT are ″stdout″ or ″stderr″. Value can be any combination of upper and lower case and must not contain leading or trailing spaces.

Programmer response: Change value to be either ″stdout″ or ″stderr″.

System action: The application was terminated.

Symbolic Feedback Code: IGZ05S

IGZ0193W Search argument *argument-number* in the WHEN phrase of the SEARCH ALL statement in program *program-name* on line number *line-number* was a signed item with a negative value. The corresponding table key was an unsigned item, and so the argument could never match the key in any of the table entries.

Explanation: The SEARCH ALL statement specified a search argument that was a signed item with a negative sign. The table key was an unsigned numeric item, and so the comparison of the argument and the keys would always be unequal. Hence the SEARCH ALL statement could never succeed in locating a matching table entry.

Programmer response: Ensure that the search argument is correctly initialized before issuing the SEARCH ALL statement. For an unsigned table key, a numeric argument must either be unsigned or have a positive sign for the search statement to have a possibility of successfully finding a matching table entry.

System action: No system action was taken.

Symbolic Feedback Code: IGZ061

IGZ0194W **Search argument** *argument-number* **in the WHEN phrase of the SEARCH ALL statement in program** *program-name* **on line number** *line-number* **was longer than the corresponding key. The excess digit or character positions of the argument were not zeros or spaces respectively, and so the argument could never match the key in any of the table entries.**

Explanation: The SEARCH ALL statement specified a search argument that was longer than the table key, and since the excess digits or characters were not zeros or spaces respectively, the comparison of the argument and the keys would always be unequal. Hence the SEARCH ALL statement could never succeed in locating a matching table entry.

Programmer response: Initialize the excess argument positions to zeros or blanks as appropriate before issuing the SEARCH ALL statement. Alternatively, use a MOVE statement to move the argument to a shorter temporary variable to truncate the excess argument positions, then use that temporary variable as the SEARCH ALL argument.

System action: No system action was taken.

Symbolic Feedback Code: IGZ062

IGZ0195S **A SORT or MERGE statement was attempted when running in an OS/390 UNIX process.**

Explanation: SORT and MERGE statements are not supported when running in an z/OS UNIX process.

Programmer response: Remove the SORT or MERGE statements from the application or run the program in an z/OS environment.

System action: The application was terminated.

Symbolic Feedback Code: IGZ063

IGZ0196S **Argument-1 for function** *function-name* **in program** *program-name* **at displacement** *displacement* **contained more than 31 digits.**

Explanation: The total number of digits in argument-1 of the indicated function exceeded 31 digits.

Programmer response: Adjust the number of digits in argument-1 in the failing statement.

System action: The application was terminated.

Symbolic Feedback Code: IGZ064

IGZ0197S **There was an unsuccessful READ or WRITE of file** *file-name* **in program** *program-name* **at relative location** *location* **Neither FILE STATUS nor an ERROR declarative were specified. The file status code was** *status-code***. The BPX return code was** *return-code***. The BPX reason code was** *reason-code***.**

Explanation: An error has occurred while reading or writing the named file. No file status or user error declarative was specified.

Programmer response: For additional information on the return-code and reason-code, see *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, SC23-3020, topics for read (BPX1RED) and write (BPX1WRT).

System action: The application was terminated.

Symbolic Feedback Code: IGZ065

IGZ0198W **File** *file-name* **in program** *program-name* **had a block size of** *block-size* **which exceeds the maximum supported block size.**

Explanation: The program file description specified a block size that exceeds the maximum supported block size. The OPEN statement failed.

Programmer response: Ensure that the block size specified in the BLOCK CONTAINS clause is supported for the file, for the device that the dataset resides on and for the operating system level being used.

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ066

IGZ0199S **An attempt was made to run a COBOL program that was compiled with the SEPARATE suboption of the TEST compiler option. This is not supported with this level of Language Environment or this level of Debug Tool. The program name is *program-name*.**

Explanation: COBOL programs running with TEST(,,SEPARATE) must run on levels of Language Environment and Debug Tool that support it. This error can occur with any of the following:

- running with a level of Language Environment that does not support the SEPARATE suboption
- running with a level of Language Environment that could support the SEPARATE suboption but does not have current maintenance applied
- running with a level of Debug Tool that could support the SEPARATE suboption but does not have current maintenance applied

Programmer response: Run the program under levels of Language Environment and Debug Tool that support programs compiled with TEST(,,SEPARATE) or recompile the COBOL program without the SEPARATE suboption of the TEST compiler option.

System action: The application was terminated.

Symbolic Feedback Code: IGZ067

IGZ0200W **A file attribute mismatch was detected. File *file-name* in program *program-name* was defined as a physical sequential file and the file specified in the ASSIGN clause was a VSAM data set.**

Explanation: The program file description specified that the file was a physical sequential file and the data set associated with the ASSIGN clause was found to be a VSAM file. The OPEN statement failed.

Programmer response: Check that the file description and the DD parameter associated with the ASSIGN clause are for the correct data set.

System action: If a file status was specified, no system action is performed. If a file status field was not specified, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ068

IGZ0201W **A file attribute mismatch was detected. File *file-name* in program *program-name* had a record length of *record-length-1* and the file specified in the ASSIGN clause had a record length of *record-length-2*.**

Explanation: The program file description specified a record length that did not match the record length of the data set associated with the ASSIGN clause. The OPEN statement failed.

Programmer response: For Format-V and Format-S files the maximum record length specified in your program must be exactly 4 bytes smaller than the length attribute of the data set. For Format-F files, the record length specified in your program must exactly match the length attribute of the data set. For Format-U files, the maximum record length specified in your program must exactly match the length attribute of the data set. If your file is a printer file, the compiler may add one byte to the file description for carriage control character, depending on the ADV compiler option and the COBOL statements used in your program. In which case, the added byte must be included in the data set length attribute. For VSAM files, the record length must not be greater than the maximum length attribute of the data set. For VSAM simulated RRDS (SIMVRD run-time option) the record length specified in the ASSIGN clause is incremented by 4 bytes before comparison with the length attribute of the data set.

System action: If a file status was specified, no system action is performed. If a file status field was not specified, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ069

IGZ0202W A file attribute mismatch was detected. File *file-name* in program *program-name* specified ASCII data and the file specified in the ASSIGN clause did not contain the ASCII data attribute.

Explanation: The CODE-SET clause was specified in the program file description and the data set associated with the ASSIGN clause did not contain ASCII data. The OPEN statement failed.

Programmer response: Check that the data set associated with the ASSIGN clause is the correct one, and if it is, check the data set for the ASCII attribute.

System action: If a file status was specified, no system action is performed. If a file status field was not specified, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ06A

IGZ0203W A file attribute mismatch was detected. File *file-name* in program *program-name* specified non-ASCII data and the file specified in the ASSIGN clause contained the ASCII data attribute.

Explanation: The data set associated with the ASSIGN clause contained ASCII type data and the file description in the program did not contain ASCII data. The OPEN statement failed.

Programmer response: Check that the data set associated with the ASSIGN clause is the correct one, and if it is, check the data set for the ASCII attribute.

System action: If a file status was specified, no system action is performed. If a file status field was not specified, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ06B

IGZ0204W A file attribute mismatch was detected. File *file-name* in program *program-name* was defined as RECORDING MODE *recording-mode* and the file specified in the ASSIGN clause did not contain the same attribute.

Explanation: The RECORDING MODE specified in the program file description did not match the data control block fields of the data set associated with the ASSIGN clause. The OPEN statement failed.

Programmer response: Check the data control block fields of the actual data set to verify that the RECORDING MODE matches. The most common cause of this error is conflicting fixed and variable record length data set attributes.

System action: If a file status was specified, no system action is performed. If a file status field was not specified, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ06C

IGZ0205W An OPEN failure occurred for file *file-name* in program *program-name* because the SMSVSAM server was not available. The file was closed.

Explanation: COBOL encountered a SMSVSAM server not available error return while performing OPEN, I/O, or control block testing of a VSAM data set in RLS mode. For this error condition VSAM requires that the file be closed, opened, and positioned before resubmitting requests. Look for possible VSAM error messages in the job log.

Programmer response: COBOL only performs a close of the file. Resolve the SMSVSAM server not available condition and resubmit the run or remove the RLS keyword specification from the DD statement.

System action: No system action is performed.

Symbolic Feedback Code: IGZ06D

IGZ0206W The AIXBLD run-time option was invalid for file *file-name* in program *program-name* because the file was opened in RLS mode. The file was closed.

Explanation: The AIXBLD option is only supported for VSAM data sets opened without RLS mode. VSAM data sets opened in RLS mode can be empty, but upgrades to empty paths are not supported. The alternate index path must be built before using RLS mode. The alternate index was not built and the file was closed.

Programmer response: If AIXBLD option is required, remove the RLS keyword specification from the DD statement for this file and resubmit the run.

System action: No system action is performed.

Symbolic Feedback Code: IGZ06E

IGZ0207W The SIMVRD run-time option was invalid for file *file-name* **in program** *program-name* **because the file was opened in RLS mode. The file was closed.**

Explanation: The SIMVRD option is not supported for VSAM data sets in RLS mode. The file was closed.

Programmer response: If SIMVRD option is required, remove the RLS keyword specification from the DD statement for this file and resubmit the run.

System action: No system action is performed.

Symbolic Feedback Code: IGZ06F

IGZ0210S There was an attempt to run an OS/VS COBOL program *program-name* **in a non-initial thread.**

Explanation: OS/VS COBOL programs can only run in the initial thread. For example, OS/VS COBOL programs can not run in a subtask created by a PL/I CALL statement with the TASK, EVENT, or PRIORITY option.

Programmer response: Compile the COBOL program with the COBOL for MVS & VM or COBOL for OS/390 & VM compiler.

System action: The application is terminated.

Symbolic Feedback Code: IGZ06I

IGZ0215S Argument —1 for function *function-name* **in program** *program-name* **at line** *line-number* **was less than 0 or greater than 99.**

Explanation: An illegal value was used for Argument-1.

Programmer response: Ensure that argument-1 is greater than, or equal to 0, and less than 100.

System action: The application is terminated.

Symbolic Feedback Code: IGZ06N

IGZ0216S Argument —1 for function *function-name* **in program** *program-name* **at line** *line-number* **was less than 0 or greater than 99366.**

Explanation: An illegal value was used for Argument-1.

Programmer response: Ensure that argument-1 is greater than, or equal to 0, and less than 99367.

System action: The application is terminated.

Symbolic Feedback Code: IGZ06O

IGZ0217S Argument —1 for function *function-name* **in program** *program-name* **at line** *line-number* **was less than 0 or greater than 991231.**

Explanation: An illegal value was used for Argument-1.

Programmer response: Ensure that argument-1 is greater than, or equal to 0, and less than 991231.

System action: The application is terminated.

Symbolic Feedback Code: IGZ06P

IGZ0218S The sum of the year at the time of execution and the value of argument —2 was less than 1700 or greater than 10000 for function *function-name* **in program** *program-name* **at line** *line-number***.**

Explanation: An illegal value was used for Argument-2.

Programmer response: Ensure that the sum of the year at the time of execution and the value of argument-2 is less than 1700 or greater than 10000.

System action: The application is terminated.

Symbolic Feedback Code: IGZ06Q

IGZ0219S **The base year for program** *program-name* **was outside the valid range of current year minus 99 through 1999. The sliding window value** *window-value* **resulted in a base year of** *base-year*.

Explanation: The current year was outside the 100-year fixed window specified by the YEARWINDOW compiler option value.

For example, if a COBOL program is compiled with YEARWINDOW(1920), the 100-year window for the program is 1920 and 2019. When the program is run in the year 2020, this error message would occur since the current year is not within the 100-year window.

Programmer response: Examine the application design to determine if it will support a change to the YEARWINDOW option value. If the application can run with a change to the YEARWINDOW option value, then compile the program with an appropriate YEARWINDOW option value. If the application cannot run with a change to the YEARWINDOW option value, then convert all date fields to expanded dates and compile the program with NODATEPROC.

System action: The application was terminated.

Symbolic Feedback Code: IGZ06R

IGZ0220S **The current year was outside the 100-year window,** *year-start* **through** *year-end* **for program** *program*.

Explanation: The current year was outside the 100-year fixed window specified by the YEARWINDOW compiler option value.

For example, if a COBOL program is compiled with YEARWINDOW(1920), the 100-year window for the program is 1920 through 2019. When the program is run in the year 2020, this error message would occur since the current year is not within the 100 year window.

Programmer response: Examine the application design to determine if it will support a change to the YEARWINDOW option value. If the application can run with a change to the YEARWINDOW option value, then compile the program with an appropriate YEARWINDOW option value. If the application cannot run with a change to the YEARWINDOW option value, then convert all date fields to expanded dates and compile the program with NODATEPROC.

System action: The application was terminated.

Symbolic Feedback Code: IGZ06S

IGZ0221W **The Y2PAST=** *y2past-value* **SORT option (from the YEARWINDOW compiler option) was overridden by the Y2PAST value in the sort control file.**

Explanation: A windowed date field was specified as a KEY in a SORT or MERGE, which resulted in the YEARWINDOW compiler option being converted into a SORT option Y2PAST value, but Y2PAST was also specified in the sort control file.

The value in the sort control file was used, and the Y2PAST value from the program was ignored.

Programmer response: See *COBOL for OS/390 & VM Programming Guide* or *COBOL for MVS & VM Programming Guide* for a description of using windowed date fields with SORT and MERGE.

System action: No system action was taken.

Symbolic Feedback Code: IGZ06T

IGZ0222S **No significant digits remain in a fixed-point exponentiation operation in program** *program* **at displacement** *displacement* **due to excessive decimal positions specified in the operands or receivers.**

Explanation: A fixed-point exponentiation operation that specifies a negative exponent could not be completed because all significant digits were lost after the operands were scaled. This condition is caused by excessive decimal positions being specified in the operands or receivers of the expression.

Programmer response: Simplify the arithmetic expression, specifying less decimal positions in the operands.

Do not use exponentiation of a base having 31 decimal positions, using a negative integral exponent. Rather, use an exponentiation specifying a positive exponent followed by an explicit division operation.

Alternatively, use a floating-point expression. To do this, specify at least one floating-point operand or receiver.

System action: The application was terminated.

Symbolic Feedback Code: IGZ06U

IGZ0223S **Argument-1 for function** *function-name* **in program** *program-name* **at line** *line-number* **was less than zero or greater than 29.**

Explanation: Input argument to function FACTORIAL is greater than 29 or less than 0.

Programmer response: Check that the function argument is in the valid range.

System action: The application was terminated.

Symbolic Feedback Code: IGZ06V

IGZ0224S **There was an attempt to run COBOL programs in more than one thread and all of the COBOL programs were not enabled for multithreading. The error was detected when attempting to run COBOL program** *program-name***.**

Explanation: There was an attempt to run COBOL programs that are not enabled for multithreading in more than one thread. In order to run COBOL programs in more than one thread, all of the COBOL programs in the application must be compiled with the Enterprise COBOL compiler using the THREAD compiler option.

COBOL programs compiled with the following compilers can only run in one thread at a time:
- Enterprise COBOL with the NOTHREAD compiler option
- COBOL for OS/390 & VM
- COBOL for MVS & VM
- COBOL/370
- VS COBOL II
- OS/VS COBOL

This condition can occur when PL/I multitasking is used or when POSIX(ON) is in effect.

If PL/I multitasking is used, here are examples that can cause this condition:
- If an Enterprise COBOL program compiled without the THREAD compiler option or a COBOL for OS/390 & VM program has been invoked in the main task, then any attempts to invoke a COBOL program in a subtask created by a PL/I statement with the TASK, EVENT or the PRIORITY option will cause this condition to be signaled.
- If an Enterprise COBOL program compiled without the THREAD compiler option or a COBOL for OS/390 & VM program has been invoked in a subtask, then any attempts to invoke a COBOL program in any other task will cause this condition to be signaled until the subtask is terminated.

If POSIX(ON) is in effect, here are examples that can cause this condition:
- If an Enterprise COBOL program compiled without the THREAD compiler option or a COBOL for OS/390 & VM program has been invoked in the initial thread, then any attempts to invoke a COBOL program in a non-initial thread will cause this condition to be signaled.
- If an Enterprise COBOL program compiled without the THREAD compiler option or a COBOL for OS/390 & VM program has been invoked in a non-initial thread, then any attempts to invoke a COBOL program in another thread will cause this condition to be signaled until the non-initial thread in which a COBOL program was invoked is terminated.

Programmer response: Compile all the COBOL programs with the Enterprise COBOL using the THREAD compiler option.

System action: The application is terminated.

Symbolic Feedback Code: IGZ070

IGZ0225S There was an attempt to run the thread enabled COBOL program *program-name* in a COBOL reusable environment.

Explanation: Enterprise COBOL programs compiled with the THREAD compiler option cannot run in a COBOL reusable environment.

A COBOL reusable environment is established by doing one of the following:
- Using the RTEREUS run-time option
- Calling ILBOSTP0
- Calling IGZERRE

Programmer response: If you want to continue to run with the COBOL reusable environment, compile the COBOL program with the NOTHREAD compiler option. If you want to run COBOL programs compiled with the THREAD option in a preinitialized environment, use the Language Environment preinitialization support provided by CEEPIPI.

System action: The application is terminated.

Symbolic Feedback Code: IGZ071

IGZ0226S On CICS, an attempt was made to run a COBOL program that contains object-oriented syntax. The program name is *program-name*.

Explanation: COBOL programs running on CICS cannot contain any object-oriented syntax.

Programmer response: Change the COBOL program so that it does not contain object-oriented class definitions, INVOKE statements, or references to the JNIEnvPtr special register.

System action: The application is terminated.

Symbolic Feedback Code: IGZ072

IGZ0227S There was an invalid attempt to end an XML PARSE statement.

Explanation: An XML PARSE statement initiated by a COBOL program was in progress and one of the following was attempted:
1. A GOBACK or an EXIT PROGRAM was issued within the COBOL program that initiated the XML PARSE.
2. A user handler associated with the program that initiated the XML PARSE moved the condition handler resume cursor and resumed the application.

Programmer response: Change the application so that it does not use one of the above methods to end the XML PARSE statement.

System action: The application is terminated.

Symbolic Feedback Code: IGZ073

IGZ0228S There was an invalid attempt to start an XML PARSE statement.

Explanation: An XML PARSE statement initiated by a COBOL program was already in progress when another XML PARSE statement was attempted by the same COBOL program. Only one XML PARSE statement can be active in a given invocation of a COBOL program.

Programmer response: Change the application so that it does not initiate another XML PARSE statement from within the same COBOL program.

System action: The application is terminated.

Symbolic Feedback Code: IGZ074

IGZ0229S Argument–2 for function *function-name* in program *program-name* at line *line-number* was less than 1 or greater than 65535.

Explanation: Argument-2, the CODEPAGE, for the indicated function was not within the range of 1 thru 65535.

Programmer response: Ensure the argument-2 for the indicated function is within the valid range of 1 thru 65535.

System action: The application is terminated.

Symbolic Feedback Code: IGC075

IGZ0251W **An invalid keyword** *keyword* **was found at position** *position* **in environment variable** *env-var* **while processing file** *file-name* **in program** *program-name***.**

Explanation: While processing the allocation of the file, the specified keyword was encountered at the indicated position. The keyword was not allowed in the environment variable. The OPEN statement failed with file status 98.

Programmer response: Remove the identified keyword from the environment variable. Also, make sure that all keywords are in uppercase. Additionally, for QSAM and VSAM files:

- make sure that the DSN or PATH keyword is specified first in the environment variable

and for line sequential files:

- make sure that the PATH keyword is the only keyword specified in the environment variable

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ07R

IGZ0252W **An invalid delimiter was found at position** *position* **in environment variable** *env-var* **while processing file** *file-name* **in program** *program-name***.**

Explanation: While processing the allocation of the file, the delimiter at the indicated position was invalid in the context used in the environment variable. The OPEN statement failed with file status 98.

Programmer response: Correct the identified delimiter in the environment variable.

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ07S

IGZ0253W **Keyword** *keyword1* **was found at position** *position* **in environment variable** *env-var* **while processing file** *file-name* **in program** *program-name* **and is mutually exclusive with keyword** *keyword2***.**

Explanation: While processing the allocation of the file, keyword1 was found at the indicated position in the environment variable but cannot be specified with keyword2 which was found earlier. The OPEN statement failed with file status 98.

Programmer response: Remove either keyword1 or keyword2 from the environment variable.

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ07T

IGZ0254W **Environment variable** *env-var* **is null or only contains blanks for file** *file-name* **in program** *program-name***.**

Explanation: The contents of the environment variable that was used for the allocation of the file were null or only contained blanks. The file cannot be allocated. The OPEN statement failed with file status 98.

Programmer response: For QSAM and VSAM files, make sure that the ddname is properly defined for the file by either:

- specifying a ddname for the file setting an environment variable with the same name as the ddname to identify the file

For line sequential files, make sure that the file is properly defined by:

- setting an environment variable with the same name as the file name to identify the file using an absolute pathname

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ07U

IGZ0255W **Dynamic allocation failed for ddname** *ddname* **while processing file** *file-name* **in program** *program-name***. The return code from the dynamic allocation was** *return-code***, error code** *error-code***, reason code** *reason-code***, and information code** *information-code***.**

Explanation: A error occurred issuing the dymanic allocation for the ddname. The OPEN statement failed with file status 98.

Programmer response: See the JOBLOG output for any additional messages from data management explaining the error. For additional information on the error codes, see *z/OS MVS Programming: Authorized Assembler Services Guide*, under *Interpreting DYNALLOC Return Codes*.

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ07V

IGZ0256W **Dynamic deallocation failed for ddname** *ddname* **while processing file** *file-name* **in program** *program-name***. The return code from the dynamic deallocation was** *return-code***, error code** *error-code***, reason code** *reason-code***, and information code** *information-code***.**

Explanation: An error occurred issuing the dynamic deallocation for the ddname. The CLOSE statement failed with file status 98.

Programmer response: See the JOBLOG output for any additional messages from data management explaining the error. For additional information on the error codes, see *z/OS MVS Programming: Authorized Assembler Services Guide*, *Interpreting DYNALLOC Return Codes*.

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ080

IGZ0257W **The environment variable** *env-var* **for file** *file-name* **in program** *program-name* **contains an invalid dataset name value.**

Explanation: While processing the allocation of the file, the dataset name specified in the DSN keyword of the environment variable was invalid. The OPEN statement failed with file status 98.

Programmer response: Correct the dataset name in the DSN keyword of the environment variable.

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ081

IGZ0258W **The environment variable** *env-var* **for file** *file-name* **in program** *program-name* **contains an invalid member name value.**

Explanation: While processing the allocation of the file, the member name specified in the DSN keyword of the environment variable was invalid. The OPEN statement failed with file status 98.

Programmer response: Correct the member name in the DSN keyword of the environment variable.

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ082

IGZ0259W The environment variable *env-var* for file *file-name* in program *program-name* contains an invalid path name value.

Explanation: While processing the allocation of the file, the path name specified in the PATH keyword of the environment variable was invalid. The OPEN statement failed with file status 98.

Programmer response: Correct the path name in the PATH keyword of the environment variable.

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ083

IGZ0260W Temporary dataset names cannot be used for dynamic allocation in environment variable *env-var* for file *file-name* in program *program-name*.

Explanation: The contents of the environment variable that was used for the allocation of the file specified a temporary dataset name in the DSN parameter. Temporary dataset names are not supported for dynamic allocation. The OPEN statement failed with file status 98.

Programmer response: Change the DSN parameter in the specified environment variable to specify a permanent dataset name.

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ084

IGZ0261W An absolute pathname was not specified in environment variable *env-var* for file *file-name* in program *program-name*.

Explanation: The contents of the environment variable that was used for the allocation of the file did not specify an absolute pathname in the PATH parameter. Only absolute pathnames are supported. The OPEN statement failed with file status 98.

Programmer response: Change the PATH parameter in the specified environment variable to specify an absolute pathname.

System action: If a file status or error declarative was specified for the file, no system action is performed. If neither a file status nor an error declarative was specified for the file, the program is terminated and message IGZ0035S is generated.

Symbolic Feedback Code: IGZ085

IGZ0270S In a multithreaded environment, COBOL program *program-name* was called during condition handling and the program cannot be initialized.

Explanation: In a multithreaded environment, COBOL programs cannot be called as a condition handler for conditions raised while performing COBOL program initialization.

Programmer response: Use a condition handler written in assembler or C to handle conditions raised during COBOL program initialization.

System action: The application is terminated.

Symbolic Feedback Code: IGZ08E

IGZ0271S There was an attempt to initialize COBOL in a nested enclave when the initial enclave is multithreaded or multitasking. The first program of the nested enclave was *program-name*.

Explanation: A COBOL program cannot run in a nested enclave when the initial enclave is multithreaded or multitasking.

Programmer response: Change the application so that the COBOL program is not running in a nested enclave or do not use multithreading or multitasking when using COBOL programs in nested enclaves. If the program name printed is ″????????″, then the first program of the nested enclave is not COBOL.

System action: The application is terminated.

Symbolic Feedback Code: IGZ08F

IGZ0272S Data conversion from CCSID *CCSID1* **to** *CCSID2* **was unsuccessful. The return code from the Unicode Conversion Service CUNLCNV was** *return-code* **and the reason code was** *reason-code***.**

Explanation: The data conversion for the CCSID pair failed as indicated by the return code and the reason code. The return code and the reason code values for conversion services are described in *z/OS Support for Unicode: Using Unicode Services*.

Programmer response: Follow the programmer responses indicated for specific return and reason codes in *z/OS Support for Unicode: Using Unicode Services*.

System action: The application is terminated.

Symbolic Feedback Code: IGZ08G

| **IGZ0273S A GOBACK, EXIT PROGRAM, or STOP RUN was attempted while an EXCEPTION/ERROR declarative was in control for a QSAM ABEND. The declarative in control is in program** *program-name* **for file** *file-name***.**

| **Explanation:** A GOBACK, EXIT PROGRAM, or STOP RUN statement cannot be used while an EXCEPTION/ ERROR declarative is in control due to a QSAM ABEND for a READ, REWRITE, or WRITE statement. When a QSAM abend occurs during a READ, WRITE, or REWRITE, the file status code can be file status 34 or file status 90.

| **Programmer response:** Change the program to not use a GOBACK, EXIT PROGRAM, or STOP RUN statement when the EXCEPTION/ERROR declarative is in control.

| **System action:** The application is terminated with ABEND 4043.

| **Symbolic Feedback Code:** IGZ08H

| **IGZ0274S A GOBACK, EXIT PROGRAM, or STOP RUN was attempted while a LABEL declarative was in control. The declarative in control is in program** *program-name* **for file** *file-name***.**

| **Explanation:** A GOBACK, EXIT PROGRAM, or STOP RUN statement cannot be used while a LABEL declarative is in control.

| **Programmer response:** Change the program to not use a GOBACK, EXIT PROGRAM, or STOP RUN statement when the LABEL declarative is active.

| **System action:** The application is terminated with ABEND 4043.

| **Symbolic Feedback Code:** IGZ08I

IGZ0280S XML to data structure conversion could not complete in program *program-name* **because an error return code of** *return-code* **was received from the XML PARSE statement. The error occurred at element** *element-name* **with the character content** *character-content***.**

Explanation: The XML to data structure conversion makes use of the COBOL XML PARSE statement. The return code provided is from the XML PARSE statement and is described in SC27-1412, ″Enterprise COBOL for z/OS and OS/390 Programming Guide Version 3″.

Programmer response: Use the return code to determine the error and correct the input XML message.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ08O

IGZ0281S XML to data structure conversion could not complete in program *program-name* **because the valid range of a repeating group or repeating data item was exceeded. The error occurred at element** *element-name***.**

Explanation: The XML converter may encounter errors while transforming an XML message to a data structure that are distinct from XML PARSE errors.

Programmer response: Correct the input XML message. Check that the input XML document has the correct number of occurrences of all entries that repeat.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ08P

IGZ0282S XML to data structure conversion could not complete in program *program-name* **because no element names in the XML document were recognized by the converter.**

Explanation: The XML converter may encounter errors while transforming an XML message to a data structure that are distinct from XML PARSE errors.

Programmer response: Supply the correct XML message corresponding to this converter.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ08Q

IGZ0283S XML to data structure conversion could not complete in program *program-name* **because the character content for element** *element-name* **was longer than the element's maximum of LIMIT characters.**

Explanation: The XML converter may encounter errors while transforming an XML message to a data structure that are distinct from XML PARSE errors.

Programmer response: Correct the input XML document.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ08R

IGZ0284S XML to data structure conversion could not complete in program *program-name* **because conversion of the character content of an element that is mapped as numeric failed. The error occurred at element** *element-name* **with the character content** *character-content***.**

Explanation: The XML converter may encounter errors while transforming an XML message to a data structure that are distinct from XML PARSE errors.

Programmer response: Correct the input XML message and try again. Check that the character content represents a valid numeric expression that is supported by this converter.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ08S

IGZ0285S XML to data structure conversion could not complete in program *program-name* **because the size of the input XML message was INPUTLEN characters, which exceeds the maximum of MAXIMUM characters for this converter.**

Explanation: The XML converter has a maximum message size that was set when the converter was generated in the XML Enablement tool.

Programmer response: Correct the input XML message, or regenerate the converter in the XML Enablement tool with a larger message size limit.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ08T

IGZ0286S **XML to data structure conversion could not complete in program** *program-name* **because the character content for element** *element-name* **had a length equal to or greater than INPUTLEN characters which was too long for the converter to process.**

Explanation: The XML converter maintains a buffer for character content that has a maximum size equal to 10 times the size of the largest item in the target data structure.

Programmer response: Correct the input XML message..

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ08U

IGZ0287S **Data structure to XML conversion could not complete in program** *program-name* **because the maximum output message size of SIZE characters was exceeded while creating the outbound XML document.**

Explanation: The outbound XML Converter does whitespace suppression and entity reference expansion before delivery of the XML document. Entity reference expansion makes a message larger and therefore has the potential to grow in relation to the number of special characters in the source data structure that need to be represented as one of the 5 predefined entity references in XML.

Programmer response: Re-generate the converter in the WSED XML-Enablement tool specifying a larger overall message size for the converter.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ08V

IGZ0288S **XML Conversion could not complete in program** *program-name* **because a non-zero return code was received from the Unicode Conversion Service CUNLCNV while converting from CCSID SOURCE-CCSID to CCSID TARGET-CCSID.**

Explanation: The XML Converter uses COBOL built-in support for Unicode when dealing with input XML in either UTF-16 or UTF-8.

Programmer response: Check that conversion services is properly installed, and ensure that the conversion attempted by the XML Converter is supported by the systems installation of conversion services.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ090

IGZ0289S **Data structure to XML conversion could not complete in program** *program-name* **because the content of non-numeric member** *member-name* **of data structure** *structure-name* **contained characters that are not legal in an XML document.**

Explanation: Certain data structure members permit storage of characters that are not legal in an XML document. The permitted characters are defined by the XML specification at http://www.w3c.org.

Programmer response: Ensure that the data structure member is properly initialized and does not contain any characters that are illegal in XML before attempting conversion.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ091

IGZ0290S **Data structure to XML conversion could not complete in program** *program-name* **because the content of numeric member** *member-name* **of data structure** *structure-name* **is invalid.**

Explanation: The converter has determined that the contents of the storage occupied by a numeric data structure member is invalid for the type.

Programmer response: Ensure that the numeric data structure member is properly initialized following the semantics of the language.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ092

IGZ0291S **XML to data structure conversion could not complete in program** *program-name* **because the maximum XML element nesting depth was exceeded. The error occurred at element** *element-name* **with character content** *character-content***.**

Explanation: The converter maintains an internal stack which represents the full qualification of the current element being processed in the XML document. If extraneous XML elements not described in the XML schema to which the converter is bound are present in the XML document, they may exceed the maximum supported element depth.

Programmer response: Either supply XML documents to the converter that validate against the bound XML schema or remove the extraneous element that causes the failure from the input XML document.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ093

IGZ0292S **XML conversion could not complete in program** *program-name* **because an attempt to register an exception handler failed with Language Environment error** *le-error***.**

Explanation: The converter uses Language Environment callable services to handle errors that may occur during conversion. An attempt by the converter to register a Language Environment condition handler failed.

Programmer response: Ensure that Language Environment is configured properly and that no conditions exist in the chain of execution leading up to the converter that would prevent proper operation.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ094

IGZ0293S **XML conversion could not complete in program** *program-name* **because an attempt to unregister an exception handler failed with Language Environment error** *le-error***.**

Explanation: The converter uses Language Environment callable services to handle errors that may occur during conversion. An attempt by the converter to unregister a Language Environment condition handler failed.

Programmer response: Ensure that Language Environment is configured properly and that no conditions exist in the chain of execution leading up to the converter that would prevent proper operation.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ095

IGZ0294S **XML conversion could not complete in program** *program-name* **because the address of one or more parameters was not valid.**

Explanation: The converter detected that one or more of the parameters that were passed to its main entry point do not have valid storage addresses to which the converter could refer and obtain the value.

Programmer response: Ensure that each parameter passed to the converter has a valid storage address unless a particular parameter can be null as defined by the converter call interface. For example, a null message address passed to the outbound converter will be interpreted as a request for the maximum outbound message size to be returned instead of performing a conversion.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ096

IGZ0295S XML to data structure conversion could not complete in program ″PROGRAM-NAME″ because a response code of ″RESPONSE-CODE″ with reason code ″REASON-CODE″ was received from the bidirectional data conversion module ″MODULE-NAME″ while processing the XML element ″ELEMENT-NAME″.

Explanation: XML to data structure conversion makes use of an external module to handle bidirectional data conversions.

Programmer response: Refer to the bidirectional data conversion module documentation for explanations of return codes.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ097

IGZ0296S Data structure to XML conversion could not complete in program ″PROGRAM-NAME″ because a return code of ″RETURN-CODE″ with reason code ″REASON-CODE″ was received from the bidirectional data conversion module ″MODULE-NAME″ while processing member ″MEMBER-NAME″ of data structure ″STRUCTURE-NAME″.

Explanation: Data structure to XML conversion makes use of an external module to handle bidirectional data conversions.

Programmer response: Refer to the bidirectional data conversion module documentation for explanations of return codes.

System action: The converter will either signal a Language Environment condition or, if the optional feedback code argument is provided, a condition token representing the condition will be placed in the argument.

Symbolic Feedback Code: IGZ098

Chapter 8. Language Environment Abend Codes

This topic lists the Language Environment abend codes with descriptions and programmer responses. The hexadecimal equivalent of the abend code is shown in parentheses. *Reason codes* are shown in hexadecimal with the decimal equivalent in parentheses.

UXXXX (≤ 4000)

Explanation: The assembler user exit could have forced an abend for an unhandled condition. These are user-specified abend codes.

Programmer response: Check the Language Environment message file for message output. This will tell you what the original abend was.

System action: Task terminated.

| **U4012(X'7015')**

| **Explanation:** An call to malloc() within __Throw() failed.

| **U4012(X'FAC') Not enough main storage available.**

| **Explanation:** Insufficient storage was available to allocate the writeable static area (WSA) and parameter list.

| **Programmer response:** Run the program in a larger region.

| **System action:** The routine terminates.

| **Symbolic Feedback Code:** X'70C1' (00)

U4034 (X'FC2')

Explanation: Language Environment condition handling was bypassed. This could result from an error condition being raised while Language Environment was dormant.

Programmer response: Follow appropriate problem determination procedures.

System action: Task terminated.

U4036 (X'FC4')

Explanation: A program check occurred and Language Environment determined that it could not turn the program check into a condition.

Reason codes:

X'01' (1)	A program check was detected when Language Environment condition handling was disabled. One cause for this abend is a program check during an Information Management System (IMS) call such as CEETDLI. The address of the EPIE is loaded into register 2 before the abend being issued. Another cause for this abend is a program check during a SORT or MERGE that has been initiated by a SORT or MERGE statement in an OS/VS COBOL program.
X'02' (2)	A program check was detected and Language Environment could not determine if the program check occurred in the current enclave. The address of the EPIE is loaded into register 2 before the abend being issued.
X'03' (3)	A program check was detected and the Language Environment run-time option for the enclave is TRAP(OFF). The address of the EPIE is loaded into register 2 before the abend being issued.
X'04' (4)	A program check was detected and the Language Environment run-time option for the enclave is TRAP(OFF) or TRAP(ON,NOSPIE). Language Environment expected to recover from this program check but was unable to do so. The address of the EPIE is not loaded into register 2 for this case.

X'05'(5)	A program check occurred due to a branch into unbacked storage, when the current XPLINK stack segment was almost full, and the TRAP(ON,SPIE) option was in effect. If the XPLINK stack were not almost full or TRAP(ON,NOSPIE) were in effect, normal CEE3204 condition handling would have occurred for this program check.

Programmer response: For reason codes 1–3:

- Use the contents of register 2 at the abend to find the EPIE. The EPIE is a system control block that has the value of the registers and the PSW at the time of the program check. The values in the EPIE can be used to start the problem determination process.

For reason code 4:

- This may be a secondary error. Check any messages and/or CEEDUMPs to diagnose the original error. If this program check is the only error then run the program with the run-time option TRAP(ON) (or TRAP(ON,SPIE)) to diagnose the original program check.

The EPIE has many formats:

Offset	Content
0	Eyecatcher: EPIE
8	Value of R0
C	Value of R1
10	Value of R2
14	Value of R3
18	Value of R4
1C	Value of R5
20	Value of R6
24	Value of R7
28	Value of R8
2C	Value of R9
30	Value of R10
34	Value of R11
38	Value of R12
3C	Value of R13
40	Value of R14
44	Value of R15
48	PSW bits 0–31
4C	PSW bits 32–63
50	Program interruption information: • ILC (Instruction Length Code) • Interruption code
54	Translation exception address if interruption code is a page fault interrupt code

System action: Task terminated.

U4038 (X'FC6')

Explanation: The enclave ended with an unhandled Language Environment software-raised or user-raised condition of severity 2 or greater, and the run-time option ABTERMENC(ABEND) was specified.

Programmer response: Check the Language Environment message file for message output.

System action: Enclave terminated.

U4039 (X’FC7’)

Explanation: Language Environment is requesting a system abend dump due to an unhandled severity 2, 3, or 4 condition. This does not necessarily indicate an error condition.

Programmer response: Refer to the original unhandled condition.

System action: Language Environment continues with termination.

U4041 (X’FC9’)

Explanation: Language Environment message processing tried to issue a dynamic allocation for a data set. The *return code* used by the ABEND macro is the same return code from SVC 99. For an explanation of SVC 99, see *TSO Extensions Version 2 Programming Services.*

Programmer response: Follow appropriate problem determination procedures.

System action: Enclave terminated.

U4042 (X’FCA’)

Explanation: User HEAP damage was found by the HEAPCHK run-time option.

Reason codes:

X'00' (00)	The error detected is in the heap.
X'01' (01)	The error detected is in a heap pool.
X'02' (02)	The error detected is in a heap pool.

Programmer response: Examine the Language Environment message file, looking for HEAPCHK messages that identify where the damage is located.

System action: The routine terminates.

U4043 (X’FCB’)

Explanation: Language Environment detected an unrecoverable error running a COBOL program. Before this abend is issued, Language Environment writes an IGZ message to the message file with the details of the error condition.

Programmer response: Read the IGZ message in the Language Environment message file to determine the error.

System action: Enclave terminated.

U4080 (X’FF0’)

Explanation: An error occurred, but the usual error message could not be displayed. This error occurred early during initialization or late in termination, when Language Environment could not display the message.

Reason codes:

CEEnnnnn	Corresponds to message number CEEnnnnX (X = I, W, E, S, C.)
EDCnnnnn	Corresponds to message number EDCnnnnX (X = I, W, E, S, C)
AAAnnnnn/BBBnnnnn	used for non-CEEnnnnX and non-EDCnnnnX messages. The Facility ID is not displayed in the ABEND reason code, but ’nnnnn’ is the message number.

For more information about the error condition, find the message number in one of the message topics in this information. Note that any variable message data is not available.

Programmer response: See the indicated message for the appropriate programmer response.

System action: See the indicated message for the appropriate system action.

U4082 (X’FF2’)

Explanation: A second malfunction occurred while handling a condition.

Reason codes:

X'01' (1)	A second malfunction occurred while trying to initialize a second math save area.
X'02' (2)	A condition was raised before the point where a second condition could be recorded.
X'03' (3)	A condition was raised while Language Environment was processing a current condition under CICS.

Programmer response: This condition can be fixed by correcting the initial condition.

System action: Enclave terminated.

U4083 (X’FF3’)

Explanation: The back chain was found in error. The reason code describes the most likely cause of the abend.

Reason codes:

X'01' (1)	A save area loop exists. The save area points to itself or another save area incorrectly points to a higher save area.
X'02' (2)	Traversal of the back chain resulted in a program check.
X'03' (3)	Under normal conditions, all save area chains should end with a save area pointed to by CEECAADDSA. In this case, the save area chain terminated with a back chain pointer of 0.
X'04' (4)	Under normal conditions, all save areas are presumed to be word-aligned. Either a linkage stack has been encountered with the character string of ″F1SA’ (X’C6F1E2C1’) in a backward pointer field of the save area chain, or a misaligned (non-word) boundary save area is in the chain. Examine the save area chain to determine which is the case.
X'05' (5)	A condition was raised before the allocation of the main stack frame, or after the main routine terminated.
X'06'(6)	The dsa format was not set to up or down.
X'07'(7)	The save area chain cannot be reached or is non-existent. The pointer referencing the back chain is set to 0 (R13).
X'08'(8)	A save area on the chain points to itself.
X'09'(9)	Under normal conditions, all XPLINK save areas are presumed to be quad-word-aligned (the starting address must end in **0**. Either a linkage stack has been encountered with the character string of FISA (X'C6F1E2C1' in a backward pointer field of the save area chain, or a misaligned (non-quad-word) boundary XPLINK save area is in the chain. Examine the save area chain to determine which is the case.
X'0F' (15)	The save area chain is not intact.

Programmer response: For applications that generate their own save areas, ensure the save areas are chained together correctly; all save areas must be addressable in AMODE(31). It may be helpful to generate a system dump of the original error by using run-time options TERMTHDACT(UAIMM) and TRAP(ON,NOSPIE).

For other types of applications, a storage overlay problem has probably occurred.

System action: Enclave terminated.

U4084 (X’FF4’)

Explanation: Thread terminated abnormally.

Reason code:

X'01' (1)	A shared resource associated with a member library-held mutex might have been corrupted.
X'02' (2)	Mutex prematurely released
X'03'(3)	Infinite loop detected while handling pthread_cond_timedwait()

Programmer response: This is an internal problem. Contact your service representative.

System action: Enclave terminated.

U4085 (X’FF5’)

Explanation: The GOTO routine encountered an error.

Reason code:

X'01' (1)	GOTO is already active.
X'02' (2)	The address of the stack frame could not be found on the save area chain, and no feedback code was provided.

Programmer response: Ensure the save areas are active.

System action: Enclave terminated.

U4086 (X’FF6’)

Explanation: A library routine could not be loaded.

Reason codes:

X'01' (1)	Not enough storage to load module.
X'02' (2)	Module not found.
X'03' (3)	Module not loaded.

Programmer response: System installation error.

System action: Process terminated.

U4087 (X’FF7’)

Explanation: A recursive error was detected. A condition was raised, causing the number of nested conditions to exceed the limit set by the DEPTHCONDLMT option. The reason code indicates which subcomponent or process was active when the exception was detected.

Reason codes:

X'00' (0)	Language Environment condition manager was in control at the time of the condition.
X'01'(1)	While enabling the language specific condition handlers a subsequent condition was raised.
X'02' (2)	A user handler routine (CEEHDLR) was processing a condition when a subsequent condition was raised.
X'03' (3)	A language-specific condition handler was processing a condition when a subsequent condition was raised.
X'04' (4)	During the Language Environment condition manager’s processing of the stack frame that precedes the stack frame for the first routine, a subsequent condition was raised.
X'05' (5)	While a language-specific event handling was being processed, a subsequent condition was raised.
X'06' (6)	A malfunction occurred while the Debug Tool was in control.
X'07' (7)	While Language Environment was trying to output a message, a subsequent condition was raised.
X'08' (8)	While attempting to output a dump, a subsequent condition was raised.
X'0A' (10)	An abnormal termination exit was in control and Language Environment detected one of the following: • A program check • An ABEND • A call to CEESGL to signal a condition • Invalid DCT under CICS

Programmer response: In the case of CEEHDLR routine, recursion can occur when you use the DEPTHCONDLMT run-time option. It may be helpful to generate a system dump of the original error by using run-time options TERMTHDACT(UAIMM) and TRAP(ON,NOSPIE).

For reason code 10, determine the error in the abnormal termination exit.

System action: Enclave terminated.

U4088 (X'FF8')

Explanation: A storage condition occurred during the processing of a storage condition. The reason code indicates the request type.

Reason codes:

X'5B' (91)	Stack pointer corrupted at location 1.
X'5C' (92)	Stack pointer corrupted at location 2.
X'5D' (93)	Stack pointer corrupted at location 3.
X'5E' (94)	Stack pointer corrupted at location 4.
X'5F' (95)	Stack pointer corrupted at location 5.
X'61' (97)	DSA not found in stack
X'62' (98)	Previous NAB not in stack
X'63' (99)	Stack segment owning the next-available-byte (NAB) could not be found or a DSA backchain pointer did not contain a valid 31-bit addressable address. A storage overlay problem has probably occurred.DSA backchain pointers must contain valid addresses that can be accessed as is while in 31-bit addressing mode. For instance, a 24-bit address that was obtained by using the BAL or BALR assembler instruction will contain the ILC, CC, and Program Mask in the uppermost byte of this address, thus making it an invalid address in 31-bit mode.
X'64' — X'74' (10x)	First free storage request terminated with return code x.
X'75'(117)	During a stack overflow on the Down stack, the stack pointer (R4) was not within the current Down stack segment.
X'76'(118)	The system service IARVSERV was invoked with the CHANGEACCESS,TARGET_VIEW=HIDDEN options to create a guard page for a Down stack segment. The service returned a non-zero return code.
X'77'(119)	During a stack overflow on the Down stack, the entry point of the routine that caused the overflow could not be found or a program check occurred while attempting to access data from the routine's entry point or PPAs.
X'78'(120)	The Get Next Available Byte service (CEEVGTUN) was invoked when the stack direction was down.
X'79'(121)	The storage size requested on a stack overflow is invalid (for example negative or too big).
X'7B' (123)	An invalid stack segment header was detected during stack increment processing.
X'C8' — X'D8' (20x)	Second free storage request terminated with return code x.
X'12C' (300)	A call to CEEVFSTR failed
X'190' (400)	There is not enough storage to initialize the MEMCHECK vhm tool.
X'191' (401)	There is no more storage to keep the MEMCHECK vhm tool working.
X'3E8' — X'3F8' (100x)	First get storage terminated with return code x, and reserve stack segment already in use. This indicates a storage condition was raised while handling the storage condition.
X'400'(1024)	An out-of-storage condition has occurred with no reserve stack allocated.
X'800'(2048)	SLE set stack error.

X'801'(2049)	SLE stack reset error.
X'802'(2050)	Stack overflow error.
X'803'(2051)	Reserve stack overflow error.
X'804'(2052)	Invalid stack overflow error.
X'805'(2053)	Reserve stack error.
X'806'(2054)	Reserve stack error.
X'807'(2055)	Stack full retry error.
X'808'(2056)	SLE IARV64 error.
X'BB8' — X'BC2' (3000-3010x)	Debug Tool storage manager control blocks corrupted.
nnn	Critical condition nnn was signaled, but CEESGL returned control to the signaller. The signaller does not support a retry of the operation, so the module terminated.

Programmer response: For reason codes 91–20x, probable internal malfunction or storage corruption. For code 1001 or 1004, increase the region size or check for infinite recursion. For reason code 1024, increase the region size or request a Reserve stack. Use the STORAGE run-time option to get more information about the environment leading up to the out-of-storage condition or allow a user handler to get control and respond to the condition.

System action: Enclave terminated.

U4089 (X’FF9’)

Explanation: During attention processing, a request to end the task was made.

Reason code:

X'01' (1)	A debugging tool was asked to interrupt the code sequence and process the CEE3250 condition.

Programmer response: Continue debugging the application using a debugging tool.

System action: Process terminated.

U4091 (X’FFB’)

Explanation: An unexpected condition occurred during the running of Language Environment condition management.

Reason codes:

X'01' (1)	A GOTO was made by an enablement routine.
X'02' (2)	Invalid return code from a language-specific event handler was received during enablement processing.
X'03' (3)	Language Environment condition management detected an implicit movement of the resume cursor. Either a GOTO or move resume cursor should have been used for resumption in a different stack frame.
X'04' (4)	Invalid return code from a language-specific event handler was received.
X'05' (5)	A program check was detected while Language Environment condition manager was in control.
X'06' (6)	A request to resume the application was not accepted. The Language Environment condition manager does not accept resumption requests with conditions, such as abends.
X'07' (7)	Invalid return code from a language-specific event handler was received during stack frame processing.
X'08' (8)	CEESGL callable service was attempting to signal a new condition. A control information block could not be allocated for that condition.
X'09' (9)	The CEEHDLR routine returned with an invalid feedback code.
X'0A' (10)	The Language Environment library was unable to find a free control information block for a new condition. This is a critical error.

X'0B' (11)	The error count specified in the ERRCOUNT run-time option has been exceeded.
X'0C' (12)	Language Environment signaled a condition that could not be resumed. After a resume took place, Language Environment again attempted to terminate by signaling an imminent termination. Another resume was attempted and caused an abend.
X'0D' (13)	An invalid return code from the Debug Tool was received.
X'0E' (14)	An invalid attempt to populate an ISI with qualifying data was detected.
X'0F' (15)	A condition token other than CEE000 was returned from a member event handler. The feedback token resulted from a new condition being raised.
X'10' (16)	A request to extend stack storage could not be honored. A fixed-size stack might currently be in use.
X'11' (17)	A request for library stack storage could not be completed.
X'12' (18)	Stack storage was requested when storage management services were not available.
X'13' (19)	A request to extend stack storage could not be honored.
X'14' (20)	After being informed of a new condition, the condition handler indicated an unrecoverable error.
X'15' (21)	The maximum depth of condition nesting specified in the DEPTHCONDLMT run-time option was exceeded.
X'16' (22)	The resume point was invalid.
X'17' (23)	BXITA requested ABEND.
X'18' (24)	ABEND without LIBVEC layer.
X'19' (25)	A CIBH pointer was expected in HCOM_CIBH=0, but the field contained 0.
X'1A' (26)	A PCQ pointer was expected in HCOM_PCQ=0, but the field contained 0.
X'1B' (27)	No matching PCIBH was found because there was a logic error or the Language Environment is corrupted.
X'1C' (28)	No storage was available for PCIBH.
X'1D' (29)	No storage was available for QDATA.
X'1E' (30)	No storage was available for SigRetData.
X'1F' (31)	An internal call to the MVS function BPX1SPM was not successful.
X'20' (32)	An internal call to the MVS function BPX1PTR was not successful.
X'21' (33)	An internal call to the MVS function BPX1SPB was not successful.
X'22' (34)	CSRL16J tried unsuccessfully to return to the interrupt point for the signal delivery.
X'23' (35)	There was a logic error in Sig safing or the Language Environment is corrupted.
X'24' (36)	There was an internal logic error or the Language Environment is corrupted.
X'25' (37)	The alternate signal stack supplied by the application is full. Automatic expansion is not available for alternal signal stacks.
X'26' (38)	CEERSN_EMTYCIBH. No Language Environment condition information block (CIB) was found to be in use.
X'28' (40)	CEERSN_NOCIBH. The chain of Language Environment condition information blocks (CIB) is empty.
X'29'(41)	The BPX1SIA callable service failed when Language Environment was trying to generate a signal.
X'2A' (42)	An Internal error occurred in Language Environment condition management when the OS/390 CSRL16J service failed to resume the interrupted program. The return code from CSRL16J is loaded into register 8 at the time the ABEND is declared. See *z/OS MVS Programming: Assembler Services Reference ABE-HSP* for a description of these return codes.
X'2B' (43)	An Internal error occurred in Language Environment condition management when the RP instruction failed to resume the interrupted program.
X'2C' (44)	The DBX user (or other user of BPX1PTR or PTRACE) tried to resume the application after a

	program check using an invalid value in register 4 or 13. When altering the resume registers after a program check using DBX, make sure that register 4 or 13 points to a valid stack frame for the current thread.
X'2D' (45)	An Internal error occurred in Language Environment condition management when trying to resume an application with an unknown DSA address in register 4 or 13.
X'2E'(46)	An internal error occurred in Language Environment when the ESTAE service was unable to establish the ESTAE routine required by the Quick Freeze Exit Routine. The return code from ESTAE (in R15) is loaded into register 8 at the time the ABEND is declared, and the reason code from ESTAE (in R10) is loaded into register 9. See *z/OS MVS Programming: Assembler Services Reference ABE-HSP* for a description of these return codes.
X'2F'(47)	ESTAE 0 MACRO failed in CEEOPQEX
X'30' (48)	HCOM_SAVED_REG14 was 0 when CEEOSIGR got control
X'31' (49)	HCOM_SAVED_REG7 was 0 when CEEOSIGX/Y got control
X'32' (50)	HCOM_SAVED_REG7 was 0 when CEEOSIGZ got control
X'35' (53)	An internal call to the MVS function BPX1SPM was not successful. BPX1SPM failed because it was called on the wrong RB (request block) or in the wrong RB state. This error can happen if the Language Environment ESPIE routine gets control after a program check occurs on a different RB than Language Environment is using (for example, an application does a LINK SVC to another program that then takes a program check). Applications that can program check or ABEND on non-Language Environment RBs need to establish their own ESPIE/ESTAE to make sure the Language Environment ESPIE or ESTAE do not get control on an unexpected RB.

Programmer response: If this abend was caused by a user-written condition handler, check the return codes provided to Language Environment condition manager.

Another source of this problem can be the use of CEEMRCR with a `type_of_move '1'` done for a condition handler that is invoked for another condition handler.

System action: Enclave terminated.

U4092 (X’FFC’)

Explanation: ESPIE or ESTAE issued this abend because control storage was overlaid. Language Environment condition manager could not proceed.

Reason codes:

X'00' (0)	SPIE/ESPIE routine was detected.
X'01' (1)	STAE/ESTAE routine was detected.
X'02' (2)	A CICS interface routine was detected.

Programmer response: Determine why storage was overlaid.

System action: Enclave terminated.

U4093 (X’FFD’)

Explanation: Abend issued during initialization when errors were detected.

Reason codes:

X'04' (4)	Storage management could not properly allocate the initial storage area.
X'08' (8)	Language Environment control blocks could not be set up properly.
X'0C' (12)	System not supported.
X'10' (16)	The application’s parameter list could not be processed correctly. The parameter list might be invalid.
X'14' (20)	Hardware not supported.

X'18' (24)	An error occurred when attempting to process the options specified in the application.
X'1C' (28)	Stack management could not allocate stack and/or heap storage.
X'20' (32)	Program management could not find a module that was to be loaded.
X'24' (36)	When trying to load a module, program management encountered a storage condition.
X'28' (40)	Program management could not be initialized properly.
X'2C' (44)	The Language Environment math library could not be initialized properly.
X'30' (48)	Condition management could not be initialized properly.
X'34' (52)	A language-specific event handler returned to initialization with a feedback code, causing immediate termination.
X'38' (56)	Vector initialization did not succeed.
X'3C' (60)	The initial fixed-size stack overflowed.
X'40' (64)	Process level ran out of storage.
X'44' (68)	Enclave level ran out of storage.
X'48' (72)	Thread level ran out of storage.
X'4C' (76)	CAA pointer became corrupted.
X'50' (80)	PCB pointer became corrupted.
X'54' (84)	Assembler user exit malfunctioned.
X'58' (88)	Get heap malfunctioned during initialization.
X'5C' (92)	Anchor setup malfunctioned.
X'60' (96)	The PLIST run-time option conflicts with the operating system type.
X'64' (100)	The Language Environment anchor support was unavailable.
X'6C' (108)	The routine was compiled with an unsupported release of a compiler.
X'70' (112)	A load module did not contain a main procedure/function and was invoked without Language Environment having been previously initialized.
X'74' (116)	The primary entry point routine of the root load module was found with Language Environment V1R2 CEESTART, but the rest of the routines in the load module were not linked with Language Environment V1R2 (or later) library.
X'7C' (124)	An unsupported parameter style was detected.
X'80' (128)	Too many files, fetched procedures, controlled variables in a PL/I routine, or assembler use of external dummy sections caused the total length of the PRV to exceed the maximum limit of 4096 bytes.
X'84' (132)	Library routines required for CICS support are not defined in the CICS CSD. See *z/OS Language Environment Customization* for the library routines required for CICS support. If running a PL/I application with the shared library, see *PL/I for MVS & VM Compiler and Run-Time Migration Guide* for instructions on enabling shared library support under Language Environment.
X'88' (136)	Reinitialization feature is not supported in PL/I-defined preinitialization support.
X'8C' (140)	MVS has not installed an anchor pointer; therefore, no anchor support is available.
X'90' (144)	Condition management for MVS could not be initialized.
X'94' (148)	A language-specific event handler returned to thread initialization with a return code, causing immediate termination.
X'98' (152)	A bad return code was received from the member thread initialization exit, causing immediate termination.

X'A0' (160)	Re-entry at the top of an existing Language Environment run-time environment from a non-Language Environment-conforming driver is attempted at a different Link Level than that in effect when the Language Environment run-time environment was first created. Link level is the count of Link or CMSCALL SVCs currently active within the task. Following is an example of when this can happen: 1. An assembler program calls IGZERRE with the initialization call. 2. The assembler program calls a COBOL program using LOAD and BALR. 3. The assembler program calls a COBOL program using a LINK SVC. When the LINK SVC is done to the COBOL program, abend U4093 will occur. This abend can also happen when the RTEREUS run-time option, or calls to ILBOSTP0 or IGZERRE , are used in an IMS message processing region and all of the programs that receive control from IMS are not preloaded.
X'A4' (164)	The service routine vector address was non-zero when using request modifier value 4 in the Extended Parameter List (EPL) for the INIT function of the Pre-Init Compatibility Interface (PICI). This is not supported.
X'A8' (168)	Pre-Init Compatibility Interface (PICI) initialization was attempted while a Unix System Services medium weight process was in effect. This is not supported.
X'AC' (172)	POSIX(ON) run-time option in a nested enclave is not supported.
X'B1' (177)	An attempt was made to run an XPLINK application under LRR when an XPLINK application is not permitted.
X'B2' (178)	An attempt was made to run a non-XPLINK application with the XPLINK(ON) run time option specified under LRR when an XPLINK application is not permitted.
X'B8' (184)	An XPLINK application cannot be started while SORT is active. There is no support for SORT in an XPLINK environment.
X'BC' (188)	An XPLINK application cannot be started while PIPI is active. There is no support for PIPI in an XPLINK environment.
X'C0' (192)	An XPLINK application cannot be started while CICS is active. There is no support for CICS in an XPLINK environment.
X'C4' (196)	An XPLINK application cannot be started in a nested child enclave whose parent enclave is a non-XPLINK environment.
X'C8' (200)	The entry point of the XPLINK-compiled implicit caller of a DLL function could not be found. This is an internal error.
X'CC' (204)	The Writeable Static Area (WSA) of the XPLINK-compiled implicit caller of a DLL function could not be found. This is an internal error.
X'D0' (208)	There was an internal error in the format of the Import Export Table (IET) of the XPLINK-compiled implicit caller of a DLL function.
X'D4' (212)	Storage management could not properly allocate an initial below storage area after ALL31(ON) was assumed.
X'D8' (216)	Storage management could not properly allocate the dummy library stack when PL/I was added to an enclave.
X'DC'(220)	The application consists of one or more languages that are not supported on this version of the operating system. This abend is issued when a fenced language is detected during batch/UNIX System Services application initialization, Language Environment Preinitialization, or nested enclave initialization. This abend is always issued when Language Environment's use of RTLS is detected.
X'E0' (224)	The application main routine is XPLINK or the XPLINK(ON) run-time option was specified while running on VM. XPLINK is not supported on VM.
X'E4' (228)	Language Environment detected that the Library Routine Retention (LRR) control block eye-catcher was damaged during initialization of the environment.

X'E8' (232)	Language Environment was unable to obtain storage for the heap pools trace table. This storage is obtained when the HEAPCHK(ON) run-time option in effect and suboption pool_call_level is greater than zero. The failure is most likely due to there being insufficient available storage in the address space. Ensure that the REGION (and MEMLIMIT for AMODE 64 environment) size(s) are sufficient to run the application.
X'EC' (236)	Allocation of a thread-specific data control block (enclave level) failed, causing immediate termination.
X'F0' (240)	Allocation of a thread-specific data control block (thread level) failed, causing immediate termination.
X'F4'(244)	The CEEBXITA user exit was not found.
X'200' (512)	Language Environment was unable to obtain the library storage (control blocks) during initialization of the AMODE 64 run-time environment. Ensure that the MEMLIMIT size is sufficient to run the application.
X'204' (516)	Language Environment detected a re-initialization request while the AMODE 64 environment was already initialized. This failure is most likely due to a request to create a nested enclave (main to main). Nested enclaves are not supported in the AMODE 64 environment.
X'208' (520)	The length of the parameter list passed to the AMODE 64 application exceeded the maximum allowable size of 64K.
X'20C' (524)	Language Environment was unable to obtain storage for the extended parameter list. When the length of the parameter list exceeds 256 bytes, a separate storage request is needed. The failure is most likely due to there being insufficient available storage in the address space. Ensure that the MEMLIMIT size is sufficient to run the application.
X'210' (528)	Language Environment was unable to obtain initial stack and heap for the AMODE 64 environment. The failure is most likely due to there being insufficient available storage in the address space. Ensure that the MEMLIMIT size is sufficient to run the application.
X'214' (532)	Language Environment failed unexpectedly. Possible problems include returning from AMODE 64 environment termination and failure to set the active CEELAA address in the PSA when preinitialized environments are used.
X'218' (536)	Language Environment initialization was requested for an AMODE 64 application that does not contain a main. Specifically, the CELQMAIN CSECT is not present in the program object or the CELQMAIN CSECT does not point to the main routine (the address of the main is zero). One possible cause is that the user attempted to invoke a DLL or fetchable routine as a main.
X'21C' (540)	Load of the AMODE 64 Language Environment library CELQLIB failed. One possible cause is that the SCEERUN2 data set was not in the program search order.
X'220' (544)	Language Environment was unable to perform initialization for a pthread.
X'224' (548)	The 64-bit virtual limit (MEMLIMIT) was 0. This setting prevents the initialization and execution of an AMODE 64 application under Language Environment since storage above 2GB cannot be allocated. SMFPRMxx parameter MEMLIMIT, JCL JOB statement keyword MEMLIMIT, and JCL EXEC statement MEMLIMIT keyword are some of the mechanisms for setting the amount of 64-bit virtual storage that can be allocated in the address space. See *z/OS MVS Programming: Extended Addressability Guide* for more information on setting MEMLIMIT and how the system determines what value to use.
X'228' (552)	Language Environment initialization failed attempting to parse PLIST(HOST) style input arguments.
X'22C' (556)	Language Environment was unable to obtain storage for a copy of the _CEE_HEAP_MANAGER environment variable while initializing the AMODE 64

	environment that is to be run with CELQMCHK, or other vendor heap manager, in control. Ensure that the MEMLIMIT size is sufficient to run the application.
X'230' (560)	Language Environment was unable to obtain storage for a copy of the storage management routine address while initializing the AMODE 64 environment that is to be run with CELQMCHK, or other vendor heap manager, in control. Ensure that the MEMLIMIT size is sufficient to run the application.
X'234' (564)	Language Environment was unable to obtain the 31-bit addressable storage needed during initialization of the AMODE 64 environment. This storage is used during initialization and later during execution for low-level I/O operations. Ensure that the REGION size is sufficient to run the application.
X'3E8'–X'4E7' (1000–1255)	Unable to load event handler for a *high-level language*. The last 3 digits of the decimal reason code indicate the facility ID of the component that did not load correctly. The name of the event handler for which the load was attempted is constructed as follows: • CEEEVxxx for non-XPLINK event handlers. • CELHVxxx for XPLINK event handlers. In both cases, xxx is the 3 digits described above. For example, CEEEV003 is the non-XPLINK C/C++ event handle, and CELHV003 is the XPLINK C/C++ event handler. If either of these cannot be loaded, the reason code will be X'3EB' (1003). The load may have failed because the correct data set containing the event handler was not found in the search concatenation. For example, CELHV003 is contained in the Language Environment SCEERUN2 data set, which typically is part of your STEPLIB or LINKLST. For CICS applications, make sure you have added the Language Environment program resource definitions for CICS using the CEECCSD member in the Language Environment SCEESAMP data set. For more information on setting up Language Environment, refer to *z/OS Language Environment Customization*.
X'7D0' (2000)	For a Fortran application, a call to CEEARLU that the Language Environment CAA does not exist.

Programmer response: Set the XPLINK keyword of the CEELRR macro to YES when initializing the LRR environment.

System action: Enclave terminated.

U4094 (X’FFE’)

Explanation: An abend was issued during termination, when errors were detected.

Reason codes:

X'04' (4)	An invalid parameter to termination services was discovered.
X'08' (8)	A language-specific event handler returned an invalid return code.
X'0C' (12)	A language-specific event handler returned to termination with a return code, causing immediate termination.
X'10' (16)	Condition management could not properly terminate.
X'14' (20)	Program management could not properly terminate.
X'18' (24)	Storage management could not properly free stack and/or heap storage. This might be due to writing beyond storage.

X'1C' (28)	Storage management could not properly free the initial storage allocation.
X'20' (32)	The user stack was unable to be collapsed using GOTO.
X'24' (36)	The fixed-size termination stack overflowed.
X'28' (40)	An unhandled condition of severity 2 or greater occurred in a created enclave with TRAP(OFF) set in the creating enclave. Under CMS, this abend is issued when a severity 2 or greater condition is unhandled in a nested enclave, or a debugging tool has terminated the enclave at the user’s request. In addition to TRAP(OFF), this abend can also result when a parent enclave save area chain cannot be located, even though two enclaves existed, thus causing an attempt to propagate the failing condition. When the parent enclave received control, the save area chain was not intact, and the ABEND was percolated. An example of this is a COBOL program that is invoked without a LINK SVC and with a reusable run-time environment. On return from the COBOL program, the Language Environment enclave still exists, because of the reusable environment. When a second COBOL program is invoked by a LINK SVC, any Language Environment attempt to create a second enclave does not succeed. In an attempt to propagate this error condition to the parent enclave, Language Environment issues an abend. When the first enclave is not in the current save area chain, Language Environment percolates this abend. See *z/OS Language Environment Programming Guide* for information about nested enclaves.
X'2C' (44)	Termination requested during termination.
X'30' (48)	Condition management for MVS could not properly terminate.
X'34' (52)	The MVS environment could not properly terminate.
X'38' (56)	A language-specific event handler returned to thread termination with a return code, causing immediate termination.
X'3C' (60)	An internal logic error occurred during recursive termination handling.
X'40' (64)	An internal logic error occurred during forced thread termination handling.
X'44' (68)	An internal logic error occurred because termination was not expected.
X'48' (72)	During termination, library latches were being held and could not be released, causing immediate termination.
X'4C' (76)	Library latch services have received an unrecognized latch request, causing immediate termination.
X'4E' (78)	A language-specific event handler returned to thread termination with a return code, causing immediate termination.
X'54' (84)	Storage management could not properly free the dummy library stack that was added when PL/I was active in an enclave.
X'58' (88)	This error occurred when attempting to use `malloc()`, `free()`, `calloc()`, or `realloc()` after vendor heap manager (VHM) terminated for the enclave and heap manager is active. This may be an IBM problem. Contact IBM support.
X'5C' (92)	Abend when heap latch locked. A 04E ABEND code occurred while a Language Environment heap latch was held. The code that was holding the latch was unexpectedly interrupted by this asynchronous abend, and the Language Environment heap is in an unknown state. The 04E ABEND probably occurred because of problems on some other task or in some other address space. Determine the underlying problem that caused the other task or address space to send the 04E ABEND to this thread. Language Environment terminates with an abend immediately with 4094-96 to prevent other problems that would occur if recovery was started with the Language Environment heap in an unknown state
X'60' (96)	04E abend and latch was locked. An 04E ABEND code occurred while a Language Environment library latch was held. The code that was holding the latch was unexpectedly interrupted by this asynchronous abend, and some Language Environment resource protected by the latch is in an unknown state. The 04E ABEND probably occurred because of problems on some other task or in some other address space. Determine the underlying problem that caused the other task or address space to send the 04E ABEND to this thread. Language Environment terminates with an abend

	immediately with 4094-96 to prevent other problems that would occur if recovery was started with the Language Environment resource is in an unknown state
X'64' (100)	Access failure occurred when attempting to cleanup mutexes on behalf of the application at termination. Please make sure all mutexes are destroyed before application termination. Or make sure all mutexes are valid, and have valid storage backing at termination.
X'68' (104)	Access failure occurred when attempting to cleanup mutexes on behalf of the application at termination. Please make sure all mutexes are destroyed before application termination. Or make sure all mutexes are valid, and have valid storage backing at termination.
X'6C' (108)	The application was trying to wait on a mutex that appeared to be locked after all other threads had ended. This wait would hang indefinitely, so it was converted to an abend to avoid the hang. Most probably, the thread that originally held the mutex ended abruptly with a 422 ABEND, and the mutex was not cleaned up. This abend can probably be ignored, since one or more other threads in the application have already been ended with 422 ABENDs.
X'70' (112)	The application was trying to wait on a Language Environment fast latch that appeared to be locked after all other threads had ended. This wait would hang indefinitely, so it was converted to an abend to avoid the hang. Most probably, the thread that originally held the fast latch ended abruptly with a 422 ABEND, and the fast latch was not cleaned up. This abend can probably be ignored, since one or more other threads in the application have already been ended with 422 ABENDs.
X'74'(116)	A library mutex was being held when an asynchronous abend or unhandled condition caused termination. The mutex was marked as released prematurely and any attempt to lock the mutex caused termination to be entered again. After several attempts to terminate, this abend is issued to prevent an infinite loop.
X'200' (512)	An error occurred while releasing storage at termination of an AMODE 64 application. Capture a system dump of the failure.
X'204' (516)	An error occurred while terminating the C runtime library of an AMODE 64 application. Capture a system dump of the failure.
X'208' (520)	An error occurred while terminating the C runtime library of an AMODE 64 application. Capture a system dump of the failure.
X'20C' (524)	An error occurred while deleting the Language Environment ESPIE, ESTAE, or both when terminating an AMODE 64 application. Capture a system dump of the failure.
X'210' (528)	An error occurred while deleting the debugger when terminating an AMODE 64 application. Capture a system dump of the failure.
X'214' (532)	An error occurred while deleting the profiler when terminating an AMODE 64 application. Capture a system dump of the failure.
X'218' (536)	An error occurred during termination of an AMODE 64 application. Capture a system dump of the failure.

Programmer response: See system programmer.

System action: Enclave terminated.

U4095 (X'FFF')

Explanation: An abend was issued as a response to the fatal return code of a Language Environment-conforming language.

Reason codes:

<X'12C' (<300)	The reason code is set to the Language Environment-conforming language ID.
X'12C' (300)	The condition was provoked from a user handler attention routine.
X'12D' (301)	The condition was provoked from a user handler routine.

Programmer response: See system programmer.

System action: Enclave terminated.

Chapter 9. Language Environment Errno2 Values

Please refer to the errno2 information link found under the Assistance Center on the Language Environment website for a list of errno2 values and their corresponding information. The Language Environment Assistance Center website is located at the following address: http://www-03.ibm.com/servers/eserver/zseries/zos/le/assist/assist.html.

Chapter 10. SPC Messages, Abend and Reason Codes

This topic is divided into three small topics. The first topic lists the System Programming C abend codes and explanations. The hexadecimal equivalents of the abend codes are shown in parentheses. The second topic lists the System Programming reason codes and explanations. Reason codes are shown in hexadecimal with the decimal equivalent in parentheses. The final topic lists System Programming C Messages.

SPC Abend Codes

U2052 (X'804')

Explanation: The system programming application that is accessing the C run–time library is running AMODE=24. However, the C run–time library was installed above the 16M line, which the application cannot address.

Programmer response: Ensure that the AMODE of the application matches that of the C run-time library. Language Environment no longer supports C applications in AMODE=24. Relink the application to have AMODE=31.

System action: The application terminates.

U2100 (X'834')

Explanation: An internal request for more storage was unsuccessful.

Programmer response: Enlarge the *address space* to provide more storage.

System action: The routine terminates.

U2101 (X'835')

Explanation: An internal request to free storage was unsuccessful, probably as a result of corrupted storage.

Programmer response: Search for possible causes of corrupted storage.

System action: The routine terminates.

U2102 (X'836')

Explanation: The stack's home segment could not be found, indicating a corrupted stack.

Programmer response: Search for the cause of the corrupted storage in the user routine.

System action: The routine terminates.

U2103 (X'837')

Explanation: An error occurred when attempting to load the C library.

Programmer response: Ensure the C run-time library is available to your routine. Make sure that the modules CEEEV003 or EDCZV2 are available for the routine. The system programmer or the person who installed the product should be able to provide the location of the library and your routine can access it.

System action: The routine terminates and on MVS, a CSV code and message appears in the job. Check the message to see which C library module was not available.

U2104 (X'838')

Explanation: An error occurred during heap allocation. Using EDCXSTRX, a heap was supplied with a size smaller than the specified minimum.

Programmer response: Correct the heap size in the calling routine.

System action: The routine terminates.

U2105 (X’839’)

Explanation: An error occurred when CMSCALL or SVC 202 was issued.

Programmer response: Consult with your system programmer and correct the problem as a VM/CMS or System Programming Facilities problem.

System action: The routine terminates.

U2106 (X’83A’)

Explanation: A routine used with EDCXSTRT was compiled with the RENT option, but EDCRCINT was not included in the load module. Initialization of writable static was unsuccessful.

System action: The routine terminates.

U2107 (X’83B’)

Explanation: TRAP(ON) was requested through `#pragma runopts` but EDCXABRT was not included in the load module.

Programmer response: Rebuild the load module with EDCXABRT.

System action: The routine terminates.

U2108 (X’83C’)

Explanation: A routine built with EDCXSTRX attempted to terminate normally, but the termination routines discovered the heap needed to be extended earlier. All heap storage needs to be supplied by the caller of EDCXSTRX.

Programmer response: Correct the calling routine to provide sufficient heap storage to EDCXSTRX.

System action: The routine terminates.

U4000 (X’FA0’)

Explanation: An abend occurred during the handling of a prior abend.

Programmer response: Specify TRAP(OFF) in `#pragma runopts`, recompile and rerun the routine to isolate the cause of the original abend, and correct the cause of the original abend.

System action: The routine terminates.

U4012 (X’FAC’)

Explanation: Not enough main storage available.

Programmer response: Run the program in a larger region.

System action: The routine terminates.

SPC Reason Codes

X’7011’ (28689)

Explanation: A failure occurred during the CMS PIPE command issued to initialize the environment variables from GLOBALV. This is most likely caused because the application being initialized was invoked by a stage of the CMS PIPE command and illegal recursion occurred.

X’7012’ (28690)

Explanation: An error occurred while initializing the environment variables from GLOBALV. This may have occurred because the array of environment variable pointers was corrupted.

X’7201’ (29185)

Explanation: An error occurred during initialization.

X’7202’ (29186)

Explanation: An error occurred during termination.

X’7203’ (29187)

Explanation: An error occurred while extending the stack.

X’7204’ (29188)

Explanation: An error occurred during longjmp/setjmp.

X’7205’ (29189)

Explanation: Initialization of writable static had not been performed. The routine EDCRCINT must be included in your module if you use the RENT compiler option.

X’7206’ (29190)

Explanation: The EDCXABRT module was not explicitly included at link-edit time.

X’7207’ (29191)

Explanation: A heap was required, but the initialization had been requested without initial heap.

X’7501’ (29953)

Explanation: An H, D, or DD floating point length or type qualifier was found in a format string, indicating that a Decimal Floating Point number is being formatted or scanned. The hardware does not have the Decimal Floating Point Facility installed, so the formatting or scanning could not proceed.

SPC Messages

The System Programming C (SPC) messages have the following format:

Message Format: EDCKxxx text

EDCK	Indicates message is generated by the C library when running under CICS
xxx	Error message number

The messages that can be issued are as follows:

EDCK001 ABEND=8091 operation exception.

Explanation: An attempt has been made to execute an instruction with an invalid operation code. The operation code could be unassigned, or the instruction with that operation code might not be installed on the CPU.

Programmer response: Determine the reason for the operation exception in the user code and correct.

System action: The program terminates.

EDCK002 ABEND=8092 privileged operation exception.

Explanation: An attempt had been made to execute a privileged instruction in the problem state.

Programmer response: Determine the reason for the privileged operation exception in the user code and correct.

System action: The program terminates.

EDCK003 ABEND=8093 execute exception.

Explanation: An attempt had been made to execute an `EXECUTE` instruction.

Programmer response: Determine the reason for the execute exception in the user code and correct.

System action: The program terminates.

EDCK004 ABEND=8094 protection exception.

Explanation: An attempt had been made to access data that was protected against this type of reference or to store data in protected storage, such as a low address `0 - 511.`

Programmer response: Determine the reason for the protection exception in the user code and correct.

System action: The program terminates.

EDCK005 ABEND=8095 addressing exception.

Explanation: An attempt was made to reference a main storage location that is not available in the configuration.

Programmer response: Determine the reason for the addressing exception in the user code and correct.

System action: The program terminates.

EDCK006 ABEND=8096 specification exception.

Explanation: An alignment error in the operands of an instruction or an error in the specification of the operands has occurred (that is, an odd-numbered register was specified when an even-numbered register was expected).

Programmer response: Determine the reason for the specification exception in the user code and correct.

System action: The program terminates.

EDCK007 ABEND=8097 data exception.

Explanation: An attempt had been made to process packed decimal data that is not in the correct format.

Programmer response: Determine the reason for the data exception in the user code and correct.

System action: The program terminates.

EDCK008 ABEND=0220 zero divide.

Explanation: An attempt had been made to execute an instruction in which the value of zero has been used as the divisor of a division operation, or an overflow condition has occurred during a conversion to binary.

Programmer response: Determine the reason for the zero divide in the user code and correct.

System action: The program terminates.

EDCK009 ABEND=0620 overflow.

Explanation: The OVERFLOW condition occurred when the magnitude of a floating-point number exceeded the supported maximum.

Programmer response: Determine the reason for the overflow in the user code and correct.

System action: The program terminates.

EDCK010 The signal SIGFPE has been raised.

Explanation: The routine issued a `raise(SIGFPE)` under default conditions.

Programmer response: None.

System action: The program terminates.

EDCK011 The signal SIGILL has been raised.

Explanation: The routine issued a `raise(SIGILL)` under default conditions.

Programmer response: None.

System action: The program terminates.

EDCK012 The signal SIGSEGV has been raised.

Explanation: The routine issued a `raise(SIGSEGV)` under default conditions.

Programmer response: None.

System action: The program terminates.

EDCK017 ABEND=0320 fixed or decimal overflow.

Explanation: The overflow condition occurred when the magnitude of a fixed or decimal number exceeds the supported maximum.

Programmer response: Determine the reason for the fixed or decimal overflow in the user code and correct.

System action: The program terminates.

Chapter 11. Return Codes to CICS

When Language Environment detects an error and Language Environment is not fully initialized or unable to generate a message, the component of Language Environment in charge generates a return code. The return code passes from the Language Environment component to CICS. CICS returns the return code to the system console. The COBOL component also sends a message that precedes the return code to the system console.

Language Environment Return Codes

11000

Explanation: Invalid parameters passed from CICS to Language Environment for the partition (region) initialization call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS continues system initialization with Language Environment inactive.

11010

Explanation: Storage could not be acquired by Language Environment to initialize in the CICS region.

Programmer response: Increase the size of the CICS region using the DSASIZE SIT parameter.

System action: CICS continues system initialization with Language Environment inactive.

11020

Explanation: Unable to load Language Environment modules in order to initialize Language Environment for the CICS region.

Programmer response: Make sure the CSD definitions are correct for Language Environment. Also, make sure that the CICS region size is large enough to run Language Environment.

System action: CICS continues system initialization with Language Environment inactive.

11030

Explanation: Language Environment partition initialization did not succeed in a language support module.

Programmer response: Language Environment can write other messages to the operators console explaining the cause of the malfunction. If there are none, this is more than likely an internal error in Language Environment.

System action: CICS continues system initialization with Language Environment inactive.

11040

Explanation: An internal abend has occurred during Language Environment initialization for the CICS region.

Programmer response: There should be a CEE1000S message written to the operators console describing the abend code and reason code for the abend. See Chapter 8, “Language Environment Abend Codes,” on page 501 for more information.

System action: CICS continues system initialization with Language Environment inactive.

11100

Explanation: Invalid parameters passed from CICS to Language Environment for the partition (region) termination call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS continues system termination.

11110

Explanation: Unable to release Language Environment modules during partition (region) termination.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS continues system termination.

11120

Explanation: Unable to free storage acquired at partition (region) initialization during partition (region) termination.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS continues system termination.

11130

Explanation: Partition termination did not succeed in a language support module.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS continues system termination.

11140

Explanation: Language Environment could not release a CEEEVnnn module during partition (region) termination.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS continues system termination.

Explanation: Invalid anchor vector.

Programmer response: Most likely an internal error in CICS or Language Environment.

System action: CICS initialization will fail.

11150

Explanation: An internal abend occurred during Language Environment termination for the CICS region.

Programmer response: There should be a CEE1000S message in the operators console describing the abend code and reason code for the abend. See Chapter 8, “Language Environment Abend Codes,” on page 501 for more information.

System action: CICS continues system termination.

12000

Explanation: Invalid parameters passed from CICS to Language Environment for the thread initialization call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC7.

12020

Explanation: Preallocated storage was expected by Language Environment from CICS for the thread work area, but was not supplied.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC7.

12030

Explanation: Thread initialization did not succeed in a language support module.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC7.

12100

Explanation: Invalid parameters passed from CICS to Language Environment for the thread termination call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS continues with termination of the transaction.

12110

Explanation: Thread termination was called before all run units in the thread were terminated by calls to run unit termination.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS continues with termination of the transaction.

12120

Explanation: An error occurred while trying to free storage for language thread work areas during thread termination.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS continues with termination of the transaction.

12130

Explanation: Thread termination did not succeed in a language support module.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS continues with termination of the transaction.

13000

Explanation: Invalid parameters passed from CICS to Language Environment for the run unit initialization call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC8.

13010

Explanation: There was not enough preallocated storage by CICS to Language Environment to complete initialization for all languages in the application routine.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC8.

13020

Explanation: The mix of languages in the application load module is not supported by this release of Language Environment.

Programmer response: See *z/OS Language Environment Programming Guide* for information on supported languages and ILC.

System action: CICS abnormally terminates the transaction with abend code AEC8.

13030

Explanation: Run unit initialization did not succeed in a language support module.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC8.

13040

Explanation: An invalid application routine argument list passed by CICS to Language Environment during run unit initialization.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC8.

13050

Explanation: A member language support module is not available for a language in the application. Initialization cannot be performed.

Programmer response: Make sure the CEEEVnnn language support modules of Language Environment are defined in the CSD for all languages in the application programs.

System action: CICS abnormally terminates the transaction with abend code AEC8.

13060

Explanation: Allocation of storage for a language thread work area did not succeed.

Programmer response: Increase the size of the CICS region using the DSASIZE SIT parameter.

System action: CICS abnormally terminates the transaction with abend code AEC8.

13100

Explanation: Invalid parameters passed from CICS to Language Environment for the run unit termination call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS continues with termination of the application.

13110

Explanation: The thread token passed by CICS to Language Environment for run unit termination is invalid.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS continues with termination of the application.

13130

Explanation: Run unit termination did not succeed in a language support module.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS continues with termination of the application.

13140

Explanation: Unable to free storage for Language Environment control blocks.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS continues with termination of the application.

13200

Explanation: Invalid parameters passed from CICS to Language Environment for the run unit invocation call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC9.

13210

Explanation: Preallocated storage was expected by Language Environment from CICS for the run unit work area, but was not supplied.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC9.

13220

Explanation: Spool files for standard in, standard out, and standard error could not be opened.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC9.

13230

Explanation: Run unit invocation did not succeed in a language support module.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC9.

13240

Explanation: An invalid application routine argument list passed by CICS during run unit invocation.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS abnormally terminates the transaction with abend code AEC9.

13250

Explanation: A language support module was not available in order to invoke the application.

Programmer response: Make sure the CEEEVnnn language support modules of Language Environment are defined in the CSD for all languages in the application routines.

System action: CICS abnormally terminates the transaction with abend code AEC9.

13300

Explanation: Invalid parameters passed from CICS to Language Environment for the run unit end invocation call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS continues with termination of the application.

13310

Explanation: CICS passed an invalid thread token during run unit end invocation.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS continues with termination of the application.

13320

Explanation: CICS passed an invalid routine termination block during run unit end invocation.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS continues with termination of the application.

13330

Explanation: Unable to close spool files for standard in, standard out, and standard error.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS continues with termination of the application.

15000

Explanation: Invalid parameters passed from CICS to Language Environment for the establish ownership call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS abnormally terminates the transaction with abend code APCS.

15010

Explanation: Initialization could not be performed for a routine because the language-specific initialization routines of Language Environment were not available for the language.

Programmer response: Make sure the CEEEVnnn language support modules of Language Environment are defined in the CSD for all languages in the application routines.

System action: CICS abnormally terminates the transaction with abend code APCS.

15020

Explanation: The language of the main routine could not be determined. Initialization could not be performed for the routine.

Programmer response: The routine is probably link-edited incorrectly.

System action: CICS abnormally terminates the transaction with abend code APCS.

15030

Explanation: Language Environment establish ownership did not succeed in a language support module.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS abnormally terminates the transaction with abend code APCS.

15040

Explanation: The application load module does not contain a main routine.

Programmer response: Make sure there is a main routine in the application load module.

System action: CICS abnormally terminates the transaction with abend code APCS.

15050

Explanation: The AMODE of the routine is 24, but the routine contains C routines that must run with AMODE(31).

Programmer response: Relink-edit the routine AMODE(31).

System action: CICS abnormally terminates the transaction with the abend code APCS.

15060

Explanation: The application provided is a program object with deferred classes which can not be supported with the current level of CICS.

Programmer response: Build the application using the Language Environment Prelinker Utility.

System action: CICS abnormally terminates the transaction with the abend code APCS.

15070

Explanation: The application provided was compiled with the XPLINK compiler option and XPLINK is not supported with the current level of CICS.

Programmer response: Recompile the application program with the NOXPLINK compiler option since the level of CICS you are running does not support XPLINK.

System action: CICS abnormally terminates the transaction with the abend code APCS.

16000

Explanation: Invalid parameters passed from CICS to Language Environment for the determine working storage call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS continues running the transaction under Execution Diagnostic Facility (EDF).

16030

Explanation: Determine working storage call did not succeed in a language support module.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS continues running the transaction under Execution Diagnostic Facility (EDF).

16040

Explanation: Language Environment could not determine the working storage address and length for a routine.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS continues running the transaction under Execution Diagnostic Facility (EDF).

17000

Explanation: Invalid parameters passed from CICS to Language Environment for the perform goto call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS abnormally terminates the transaction with abend code APC2.

17030

Explanation: Perform goto cannot be completed because a goto-out-of-block is not supported for the language.

The application routine is a mix of languages. One routine is performing an EXEC CICS HANDLE with the LABEL option and calling another routine that is written in a language that does not support EXEC CICS HANDLE with LABEL. A condition occurred that caused CICS to try to branch to the handle label in the caller.

Programmer response: Change the logic of the routine. Try using EXEC CICS HANDLE with the PROGRAM option instead of EXEC CICS HANDLE with the LABEL option.

System action: CICS abnormally terminates the transaction with abend code APC2.

17040

Explanation: Errors occurred while trying to perform the goto-out-of-block on behalf of the perform goto call by CICS.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS abnormally terminates the transaction with abend code APC2.

17060

Explanation: An invalid stack frame chain was detected while trying to perform a goto-out-of-block on behalf of the perform goto call by CICS.

Programmer response: This is most likely an internal error in Language Environment.

System action: CICS abnormally terminates the transaction with abend code APC2.

18000

Explanation: Invalid parameters passed from CICS to Language Environment for the short on storage alert call.

Programmer response: This is most likely an internal error in CICS or Language Environment.

System action: CICS continues to attempt to free storage in response to the short of storage condition.

C Return Codes

31923

Explanation: Run units terminated out of sequence.

Programmer response: If this problem persists, it is most likely an internal error. Contact IBM service personnel.

System action: CICS abnormally terminates the transaction.

32112

Explanation: All run units have not been terminated before process termination.

Programmer response: If this problem persists, it is most likely an internal error. Contact IBM service personnel.

System action: CICS abnormally terminates the transaction.

32820

Explanation: Mixed-language module unsupported for pre-Language Environment C applications.

Programmer response: Recompile using IBM C compiler.

System action: CICS abnormally terminates the transaction.

32821

Explanation: C not present in language signature.

Programmer response: Recompile using IBM C compiler.

System action: CICS abnormally terminates the transaction.

32822

Explanation: Identify module entry point (event 28) was issued for a Language Environment conforming entry point when a non-Language Environment conforming entry point was expected.

Programmer response: If this problem persists, it is most likely an internal error. Contact IBM service personnel.

System action: CICS abnormally terminates the transaction.

COBOL Return Codes

51401

Explanation: Control returned from the application to the COBOL interface routine.

Programmer response: Contact IBM service personnel.

System action: CICS continues with termination of the application.

52801

Explanation: A VS COBOL II program does not have the required CSECTs link-edited with the load module.

Programmer response: Make sure that the VS COBOL II program has been link-edited correctly with no unresolved references for IGZEBST. Additionally, if the VS COBOL II program was link-edited with Language Environment, there must be no unresolved references for CEESTART or CEEBETBL.

System action: CICS abnormally terminates the transaction.

PL/I Return Codes

101010

Explanation: CICS GETMAIN command did not succeed during PL/I partition initialization.

Programmer response: Contact IBM service personnel.

System action: CICS continues system initialization with Language Environment PL/I inactive.

101020

Explanation: CICS LOAD for IBMRSAP did not succeed during PL/I partition initialization.

Programmer response: Make sure the module IBMRSAP is defined in CSD. Make sure the module IBMRSAP is located in DFHRPL.

System action: CICS continues system initialization with Language Environment PL/I inactive.

101030

Explanation: CICS LOAD did not succeed when loading one of the following shared library modules, IBMBPSMA or IBMBPSLA.

Programmer response: If the shared library compatibility support is requested, both IBMBPSMA and IBMBPSLA should be defined in CSD and located in DFHRPL.

System action: CICS continues system initialization with Language Environment PL/I inactive.

101110

Explanation: CICS FREEMAIN command did not succeed during PL/I partition termination.

Programmer response: Contact IBM service personnel.

System action: CICS continues system termination.

101120

Explanation: CICS RELEASE for IBMRSAP did not succeed during PL/I partition termination.

Programmer response: This is most likely an internal error in CICS or Language Environment. Contact IBM service personnel.

System action: CICS continues system termination.

105210

Explanation: Total length of PRV exceeded the specified maximum 4096 bytes.

Programmer response: Too many files, fetched procedures, CONTROLLED variables, or assembler use of PRV caused the total length of PRV to exceed the maximum limit of 4096 bytes. Try to reduce PRV usage.

System action: CICS abnormally terminates the transaction with abend code APCS.

Appendix. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS™ enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

Bibliography

This section lists the books in the Language Environment library and other publications that may be helpful when using Language Environment.

Language Products Publications

z/OS Language Environment
- *z/OS Language Environment Concepts Guide*, SA22-7567
- *z/OS Language Environment Programming Guide*, SA22-7561
- *z/OS Language Environment Programming Reference*, SA22-7562
- *z/OS Language Environment Customization*, SA22-7564
- *z/OS Language Environment Debugging Guide*, GA22-7560
- *z/OS Language Environment Writing Interlanguage Communication Applications*, SA22-7563
- *z/OS Language Environment Run-Time Messages*, SA22-7566
- *z/OS Language Environment Vendor Interfaces*, SA22-7568
- *z/OS Language Environment Programming Guide for 64-bit Virtual Addressing Mode*, SA22-7569

z/OS XL C/C++
- *z/OS XL C/C++ Language Reference*, SC09-4815
- *z/OS XL C/C++ Compiler and Run-Time Migration Guide for the Application Programmer*, GC09-4913
- *z/OS XL C/C++ Programming Guide*, SC09-4765
- *z/OS XL C/C++ User's Guide*, SC09-4767
- *z/OS XL C/C++ Run-Time Library Reference*, SA22-7821
- *z/OS XL C/C++ Messages*, GC09-4819
- *Standard C++ Library Reference*, SC09-4949, SC09-4949
- *IBM Open Class Library Transition Guide*, SC09-4948, SC09-4948

z/OS Run-Time Library Extensions
- *C/C++ Legacy Class Libraries Reference*, SC09-7652, SC09-7652

z/OS Metal C Runtime Library
- *z/OS Metal C Programming Guide and Reference*, SA23-2225, SA23-2225

Enterprise COBOL for z/OS
- *Enterprise COBOL for z/OS Licensed Program Specifications*, GC27-1411
- *Enterprise COBOL for z/OS Customization*, GC27-1410
- *Enterprise COBOL for z/OS Language Reference*, SC27-1408
- *Enterprise COBOL for z/OS Programming Guide*, SC27-1412
- *Enterprise COBOL for z/OS Migration Guide*, GC27-1409

COBOL for OS/390 & VM
- *COBOL for OS/390 & VM Licensed Program Specifications*, GC26-9044
- *COBOL for OS/390 & VM Customization under OS/390*, GC26-9045
- *COBOL for OS/390 & VM Language Reference*, SC26-9046
- *COBOL for OS/390 & VM Programming Guide*, SC26-9049
- *COBOL for OS/390 & VM Compiler and Run-Time Migration Guide*, GC26-4764

COBOL for MVS & VM (Release 2)
- *Licensed Program Specifications*, GC26-4761
- *Programming Guide*, SC26-4767
- *Language Reference*, SC26-4769

- *Compiler and Run-Time Migration Guide*, GC26-4764
- *Installation and Customization under MVS*, SC26-4766
- *Reference Summary*, SX26-3788
- *Diagnosis Guide*, SC26-3138

VS COBOL II

VS COBOL II Application Programming Guide for MVS and CMS, SC26-4045

Debug Tool

- Debug Tool documentation is available at: `http://www.ibm.com/software/ad/debugtool/library/`

VS FORTRAN Version 2

- *Language Environment Fortran Run-Time Migration Guide*, SC26-8499
- *Language and Library Reference*, SC26-4221
- *Programming Guide for CMS and MVS*, SC26-4222

Enterprise PL/I for z/OS

- *Enterprise PL/I for z/OS Licensed Program Specifications*, GC27-1456
- *Enterprise PL/I for z/OS Programming Guide*, SC27-1457
- *Enterprise PL/I for z/OS Language Reference*, SC27-1460
- *Enterprise PL/I for z/OS Migration Guide*, GC27-1458
- *Enterprise PL/I for z/OS Messages and Codes*, SC27-1461

PL/I for MVS & VM

- *PL/I for MVS & VM Licensed Program Specifications*, GC26-3116
- *PL/I for MVS & VM Programming Guide*, SC26-3113
- *PL/I for MVS & VM Language Reference*, SC26-3114
- *PL/I for MVS & VM Reference Summary*, SX26-3821
- *PL/I for MVS & VM Compiler and Run-Time Migration Guide*, SC26-3118
- *PL/I for MVS & VM Installation and Customization under MVS*, SC26-3119
- *PL/I for MVS & VM Compile-Time Messages and Codes*, SC26-3229
- *PL/I for MVS & VM Diagnosis Guide*, SC26-3149

High Level Assembler for MVS & VM & VSE

- *Programmer's Guide, MVS & VM Edition*, SC26-4941

Related Publications

CICS

- *CICS Transaction Server for z/OS Installation Guide*, GC34-6224
- *CICS Operations and Utilities Guide*, SC34-6229
- *CICS Problem Determination Guide*, SC34-6239
- *CICS Resource Definition Guide*, SC34-6228
- *CICS Data Areas*, LY33-6103
- *CICS Application Programming Guide*, SC34-6231
- *CICS Application Programming Reference*, SC34-6232
- *CICS System Definition Guide*, SC34-6226

DB2

Database 2 Application Programming and SQL Guide, SC26-4377

DFSMS/MVS

z/OS MVS Program Management: User's Guide and Reference, SA22-7643
z/OS DFSMS DFM Guide and Reference, SC26-7395

IPCS
- *z/OS MVS IPCS User's Guide*, SA22-7596
- *z/OS MVS IPCS Commands*, SA22-7594
- *z/OS MVS IPCS Customization*, SA22-7595

DFSORT

z/OS DFSORT Application Programming Guide, SC26-7523

IMS/ESA

IMS Version 8: Application Programming: Design Guide, SC27-1287
IMS Version 8: Application Programming: Database Manager, SC27-1286
IMS Version 8: Application Programming: Transaction Manager, SC27-1289
IMS Version 8: Application Programming: EXEC DLI Commands for CICS and IMS Version 8, SC27-1288

z/OS
- *z/OS Introduction and Release Guide*, GA22-7502
- *z/OS Program Directory*, GI10-0670
- *z/OS and z/OS.e Planning for Installation*, GA22-7504
- *z/OS Information Roadmap*, SA22-7500
- *z/OS Hot Topics Newsletter*, GA22-7501
- *z/OS Licensed Program Specifications*, GA22-7503
-
- *z/OS ISPF Dialog Tag Language Guide and Reference*, SC34-4824
- *z/OS ISPF Planning and Customizing*, GC34-4814
- *z/OS ISPF Dialog Developer's Guide and Reference*, SC34-4821
-
- *z/OS UNIX System Services User's Guide*, SA22-7801
- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803
- *z/OS UNIX System Services Planning*, GA22-7800
-
- *z/OS TSO/E Customization*, SA22-7783
- *z/OS TSO/E Programming Services*, SA22-7789
- *z/OS TSO/E System Programming Command Reference*, SA22-7793

Softcopy Publications

z/OS Collection, SK3T-4269

Index for Migration

Program Number: 5694-A01

Printed in USA

SA22-7566-08