IBM Language Environment for VSE/ESA

Debugging Guide and Run-Time Messages

Version 1 Release 4 Modification Level 4

SC33-6681-06

IBM Language Environment for VSE/ESA

Debugging Guide and Run-Time Messages

Version 1 Release 4 Modification Level 4

SC33-6681-06

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page xi.

Seventh Edition (March 2005)

This edition applies to Version 1 Release 4 Modification Level 4 of IBM Language Environment for VSE/ESA, 5686-CF7, and to any subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the addresses given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

```
IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany
```

You may also send your comments by FAX or via the Internet:

```
Internet: s390id@de.ibm.com
FAX (Germany): 07031-16-3456
FAX (other countries): (+49)+7031-16-3456
```

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

Any pointers in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this publication or accessed through an IBM Web site that is mentioned in this publication.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

```
IBM Deutschland Informationssysteme GmbH
Department 0215
Pascalstr. 100
70569 Stuttgart
Germany
```

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

Trademarks and Service Marks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX	IBMLink	System/370
AT	Integrated Language Environment	VSE/ESA
C/370	Language Environment	z/OS
CICS	MVS	zSeries
CICS/VSE	OS/390	z/VM
COBOL/370	OS/400	z/VSE
DFSORT	S/370	
IBM	S/390	

Microsoft, Windows, the Windows 95 logo, and Windows NT, are trademarks or registered trademarks of Microsoft Corporation.

About this Book

z/VSE is the successor to IBM's VSE/ESA product. Many products and functions supported on z/VSE may continue to use VSE/ESA in their names.

z/VSE can execute in 31-bit mode only. It does not implement z/Architecture, and specifically does not implement 64-bit mode capabilities.

z/VSE is designed to exploit select features of IBM eServer zSeries hardware.

This book provides assistance with detecting and locating programming errors that occur during run time while using the IBM Language Environment for VSE/ESA (LE/VSE) licensed program.

It is designed to help you establish a debugging process that replaces the work of guessing where an error may have occurred with a process of analyzing data and narrowing the scope and location of the problem.

This book is for application programmers interested in techniques for debugging run-time programs. To use this book, you should be familiar with:
- The LE/VSE product
- An LE/VSE-conforming language such as C, COBOL, or PL/I
- Program storage concepts

What Is LE/VSE?

LE/VSE is a set of common services and language-specific routines that provide a single run-time environment for applications written in *LE/VSE-conforming* versions of the C, COBOL, and PL/I high level languages (HLLs), and for many applications written in previous versions of COBOL. (For a list of LE/VSE-conforming languages, and a description of compatibility with previous versions of COBOL, see "LE/VSE-Conforming Languages" on page xiv.) LE/VSE also supports applications written in assembler language using LE/VSE-provided macros and assembled using High Level Assembler (HLASM).

Prior to LE/VSE, each programming language provided its own separate run-time environment. LE/VSE combines essential and commonly-used run-time services—such as message handling, condition handling, storage management, date and time services, and math functions—and makes them available through a set of interfaces that are consistent across programming languages. With LE/VSE, you can use one run-time environment for your applications, regardless of the application's programming language or system resource needs, because most system dependencies have been removed.

Services that work with only one language are available within language-specific portions of LE/VSE.

LE/VSE consists of:

- Basic routines for starting and stopping programs, allocating storage, communicating with programs written in different languages, and indicating and handling error conditions.
- Common library services, such as math services and date and time services, that are commonly needed by programs running on the system. These functions are supported through a library of callable services.
- Language-specific portions of the common run-time library.

LE/VSE is the implementation of Language Environment on the VSE platform. Language Environment is also offered on platforms z/OS and VM, and on OS/400 as Integrated Language Environment.

LE/VSE-Conforming Languages

An LE/VSE-conforming language is any HLL that adheres to the LE/VSE common interface. Table 1 lists the LE/VSE-conforming language compiler products you can use to generate applications that run with LE/VSE Release 4.

Table 1. LE/VSE-Conforming Languages

Language	LE/VSE-Conforming Language	Minimum Release
C	IBM C for VSE/ESA	Release 1
COBOL	IBM COBOL for VSE/ESA	Release 1
PL/I	IBM PL/I for VSE/ESA	Release 1

Any HLL not listed in Table 1 is known as a *non-LE/VSE-conforming* or, alternatively, a *pre-LE/VSE-conforming* language. Some examples of non-LE/VSE-conforming languages are:
- C/370
- DOS/VS COBOL
- VS COBOL II
- DOS PL/I
- DOS/VS RPG II

Only the following products can generate applications that run with LE/VSE:
- LE/VSE-conforming languages
- HLASM using LE/VSE-provided macros (for details, see *LE/VSE Programming Guide*)
- DOS/VS COBOL and VS COBOL II, with some restrictions (see LE/VSE Compatibility with Previous Versions of COBOL below).

LE/VSE Compatibility with Previous Versions of COBOL

Although DOS/VS COBOL and VS COBOL II are non-LE/VSE-conforming languages, many applications generated with these compilers can run with LE/VSE without recompiling. For details about compatibility, see *LE/VSE Run-Time Migration Guide*.

However relinking under LE/VSE is the *minimum* effort in order to migrate run-time, and involve LE/VSE COBOL-compatibility routines (rather than the old and unsupported library routines of non-LE/VSE conforming COBOL compilers). This particularly applies to NORES-compiled units or applications that involve former initialization techniques such as ILBDSET0. There are even restrictions with this approach, such as:

- No use of 4-digit dates.
- No exploitation of LE/VSE functionality.
- Interlanguage communication capabilities, and so on.

Therefore you are *strongly recommended* to carry out a (subsequent) full migration to a higher ANSI standard and LE/VSE-conforming COBOL compiler (COBOL for VSE/ESA).

VS COBOL II can also dynamically call some LE/VSE date and time callable services. For details, see *LE/VSE Programming Reference*.

Terms Used in This Book

Unless otherwise stated, the following terms are used in this book to refer to the specified languages:

Term...	**Refers to the language supported by...**
C	The IBM C for VSE/ESA compiler
COBOL	The IBM COBOL for VSE/ESA and VS COBOL II compilers
PL/I	The IBM PL/I for VSE/ESA compilers

For a list of LE/VSE-conforming language compilers, see “LE/VSE-Conforming Languages” on page xiv.

Where to Find More Information

These are the manuals that describe LE/VSE:

Table 2. LE/VSE Publications

Publication	Form Number
LE/VSE Fact Sheet	GC33-6679
LE/VSE Concepts Guide	GC33-6680
LE/VSE Customization Guide	SC33-6682
LE/VSE Programming Guide	SC33-6684
LE/VSE Programming Reference	SC33-6685
LE/VSE C Run-Time Programming Guide	SC33-6688
LE/VSE C Run-Time Library Reference	SC33-6689
LE/VSE Debugging Guide and Run-Time Messages	SC33-6681
LE/VSE Writing Interlanguage Communication Applications	SC33-6686
LE/VSE Run-Time Migration Guide	SC33-6687
LE/VSE Licensed Program Specifications	GC33-6683

These are the z/VSE manuals to which you might need to refer:

Table 3. z/VSE Publications

Publication	Form Number
z/VSE Administration	SC33-8224
z/VSE Messages and Codes, Volume 1	SC33-8226
z/VSE Messages and Codes, Volume 2	SC33-8227
z/VSE Messages and Codes, Volume 3	SC33-8228
z/VSE Planning	SC33-8221
z/VSE System Control Statements	SC33-8225
z/VSE System Macros Reference	SC33-8230
z/VSE System Macros User's Guide	SC33-8236
z/VSE System Upgrade and Service	SC33-8223
VSE/VSAM User's Guide and Application Programming	SC33-8246
VSE/VSAM Commands	SC33-8245
TCP/IP for VSE/ESA IBM Program Setup and Supplementary Information	SC33-6601

These are the manuals that describe IBM C for VSE/ESA:

Table 4. IBM C for VSE/ESA Publications

Publication	Form Number
Licensed Program Specifications	GC09-2421
Installation and Customization Guide	GC09-2422
Migration Guide	SC09-2423

Table 4. IBM C for VSE/ESA Publications (continued)

Publication	**Form Number**
User's Guide	SC09-2424
Language Reference	SC09-2425
Diagnosis Guide	GC09-2426

These are the manuals that describe IBM COBOL for VSE/ESA:

Table 5. IBM COBOL for VSE/ESA Publications

Publication	**Form Number**
General Information	GC33-6679
Licensed Program Specifications	GC33-6680
Migration Guide	SC33-6682
Installation and Customization Guide	GC33-6680
Programming Guide	SC33-6684
Language Reference	SC33-6685
Diagnosis Guide	SC33-6684
Reference Summary	SX26-3834

These are the manuals that describe IBM PL/I for VSE/ESA:

Table 6. IBM PL/I for VSE/ESA Publications

Publication	**Form Number**
Fact Sheet	GC26-8052
Programming Guide	SC26-8053
Language Reference	SC26-8054
Licensed Program Specifications	GC26-8055
Migration Guide	SC33-6684
Installation and Customization Guide	SC26-8057
Diagnosis Guide	SC26-8058
Compile-Time Messages and Codes	SC26-8059
Reference Summary	SX26-3836

These are the manuals that describe Debug Tool for VSE/ESA:

Table 7. Debug Tool for VSE/ESA Publications

Publication	**Form Number**
User's Guide and Reference	SC26-8797
Installation and Customization Guide	SC26-8798
Fact Sheet	GC26-8925

You might also refer to the ...

z/VSE Home Page

z/VSE has a home page on the World Wide Web, which offers up-to-date information about VSE-related products and services, new z/VSE functions, and other items of interest to VSE users.

You can find the z/VSE home page at:

`http://www.ibm.com/servers/eserver/zseries/zvse/`

Softcopy Publications

The following collection kit contains the LE/VSE and LE/VSE-conforming language product publications:

VSE Collection, SK2T-0060

Summary Of Changes

This section describes the changes introduced with the seventh edition and the previous three editions of the manual.

Changes Introduced with Seventh Edition (March 2005)

This section describes the changes that have been introduced with LE/VSE 1.4.4:
- The name **VSE/ESA** has now changed to **z/VSE**. However, the names of many features and programs related to z/VSE remain unchanged (such as IBM Language Environment for VSE/ESA, IBM COBOL for VSE/ESA, or Debug Tool for VSE/ESA).
- Information has been included describing how you can use *attention routine* (AR) commands to display Language Environment information. See "Using Attention Routine Interface Commands" on page 29.
- The example of an options report produced by LE/VSE run-time option RPTOPTS(ON) has been updated to show the use of the TERMTHDACT LSTQ sub-option. See Figure 1 on page 9.
- A short description of the OLPD function is provided, which you can use for online (CICS) debugging purposes. See Table 36 on page 155.
- Messages CEE0814S, CEE3558I, CEE3907I, CEE3908I, and CEE3909I are new. See Chapter 8, "LE/VSE Run-Time Messages," on page 163.
- Message CEE3492S has been modified. See Chapter 8, "LE/VSE Run-Time Messages," on page 163.
- To improve usability, the messages relating to the LE/VSE support of VSE/POWER LSTQ have been moved to their own section. See "VSE/POWER LSTQ Support Run-Time Messages" on page 210. In this section:
  - Messages CEEL000S, CEEL002S, CEEL003S, and CEEL004S have been modified.
  - Message CEEL005I is new.
- A section has been created containing run-time messages relating to the LE/VSE support of attention routine (AR) commands. See "Attention Routine Support Run-Time Messages" on page 212. In this section, messages CEL4000 to CEL4041 are new.
- A section has been created containing a run-time message relating to the LE/VSE support of the CICS CLER transaction. See "CLER CICS Transaction Support Message" on page 216. In this section, message CEL4101E has replaced CEL4001E.
- Messages EDC4057E and EDC4058E are new. See "Prelinker Messages" on page 218.
- Messages EDC5519 to EDC5522 are new. See "DSECT Utility Messages" on page 240. Also note that in this section, message EDC5519 has been renamed to EDC5599 (to avoid a conflict of message numbers).
- Message U4083 (X'FF3') has been modified. See Chapter 13, "LE/VSE Abend Codes," on page 329.

Changes Introduced with Sixth Edition (March 2004)

This section describes the changes that were introduced by APAR PQ74201:

- Six messages (IGZ0199S, IGZ0235W, IGZ0236I, IGZ0237I, IGZ0238I, IGZ0239I) were new, and might be generated when attempting to run a COBOL/VSE program compiled with the SEPARATE suboption of the TEST compiler option. These COBOL run-time messages are described in Chapter 12, “COBOL Run-Time Messages,” on page 307.

Changes Introduced with Fifth Edition (March 2003)

This section describes the changes that were introduced with LE/VSE 1.4.3:

- Short descriptions of the CICS transactions CLER, NEWC, and RPOC were included. See *page 9* onwards.
- Information was included on how to use the HEAPCHK run-time option to determine if a storage leakage exists. See page “Diagnosing Storage Leak Problems” on page 58.
- These were the changes to the chapter describing run-time messages (Chapter 8, “LE/VSE Run-Time Messages,” on page 163):
  - The description and programmer-response of message CEE0374C were changed. This message might be generated while processing a previously-unhandled condition.
  - Message CEE3556W was new, and might be generated if an alternate CICS Transient Data Queue (which is referred-to via the LE/VSE MSGFILE run-time option) is either not available or has not been defined correctly.
  - Messages CEE3557W and CEEL011S were new, and might be generated when sending an LE/VSE dump to the LSTQ destination.
  - Three messages (CEE3520, CEE3817, CEE3818) were new, and might be generated when using the CEEFETCH and CEERELES macros.
  - Message CEL4001E was new, and might be generated when using the CICS CLER transaction.

    Note: In LE/VSE 1.4.4, this message has been replaced with CEL4101E.
- Message IGZ0198S was new, and might be generated when a corrupt external data file is found while a program is running. See Chapter 12, “COBOL Run-Time Messages,” on page 307.
- Reason code X'52' was new for abend code U4093. See Chapter 13, “LE/VSE Abend Codes,” on page 329.

Part 1. Introduction to Debugging in LE/VSE

This part of the book provides information about options and features you can use to prepare your routine for debugging, describes some common errors that occur in routines, and offers methods for identifying errors and obtaining the information you need to debug your routine.

Chapter 1. Preparing Your Routine for Debugging

This chapter describes options and features that you can use to prepare your routine for debugging. The following topics are covered:

- Compile-time options for C, COBOL, and PL/I
- LE/VSE run-time options
- Use of storage in routines
- Options for modifying *condition handling* [1]
- User-created messages
- LE/VSE *feedback codes* and *condition tokens*

Setting Compile-Time Options

The following sections discuss language-specific compile-time options important for debugging routines in LE/VSE. These sections cover only the compile-time options that are important for debugging. For a complete list of compile-time options, refer to the appropriate language publications.

Each LE/VSE-conforming language offers options you can set during application development to assist you as you debug your application. You must set these options before you compile, and often you remove them before delivering the application.

Note: The use of some compile-time options (TEST, for example) can affect the performance of your routine. In some cases, you might need to remove the option and recompile your routine.

1. Terms are used in this book that might be unfamiliar to you. Each such word is highlighted (like *condition handling*) the first time it appears, and is defined in "Language Environment Glossary" on page 355.

C Compile-Time Options

When using C, set the ALL *suboption*, of the TEST compile-time option; this is equivalent to setting the suboptions SYM, BLOCK, and LINE. Table 8 lists TEST suboptions that you can use to simplify run-time debugging.

Table 8. TEST Suboptions

Suboption	Function
ALL	Sets all of the TEST suboptions.

SYM	Generates symbol table information and enables LE/VSE to generate a dump at run time.
	
	When you specify the SYM, you also get the value and type of variables displayed in the Local Variables section of the dump. For example, if in block 4 the variable *x* is a signed integer of 12 and in block 2 the variable *x* is a signed integer of 1, the following output appears in the Local Variables section of the dump:
	
	`%BLOCK4:>x   signed int        12`
	`%BLOCK2:>x   signed int         1`
	
	If a nonzero optimization level is used, variables do not appear in the dump.

BLOCK	Generates symbol information for nested blocks.
LINE	Generates line number hooks and allows a debugging tool to generate a symbolic dump.
GONUMBER	Generates line number tables, offset values, and statement numbers. Specifying the TEST option implicitly turns on GONUMBER.

Table 9 lists C compile-time options that you can use to simplify run-time debugging.

Table 9. C Compile-Time Options Useful in Debugging

Option	Function
AGGREGATE	Specifies that a layout for `struct` and `union` type variables appear in the listing.
CHECKOUT	Provides informational messages indicating possible programming errors.
FLAG	Specifies the minimum severity level that is tolerated.
GONUMBER	Generates line number tables corresponding to the input source file. This option is turned on when the TEST option is used. This option is needed to show statement numbers in dump output.
LIST	Includes the object routine in the compiler listing.
OFFSET	Displays the offset addresses relative to the entry point of each function.
SOURCE	Includes source input statements and diagnostic messages in the listing.
TEST	Generates information for the debugging interface. This also generates symbol tables needed for symbolic variables in the dump.
XREF	Includes a cross-reference table of names used in the routine and line numbers in the source listing.

For more detail on C compile-time options, see *LE/VSE C Run-Time Programming Guide*.

COBOL Compile-Time Options

When using COBOL, set the SYM suboption of the TEST compile-time option. The SYM suboption of TEST causes the compiler to add debugging information into the object module to resolve user names in the routine and to generate a symbolic dump of the *data division*. With this suboption specified, statement numbers can also be used in the dump output along with *offset* values.

Several compile-time options affect the type of listings generated for your routine at compile time. They are LIST, MAP, SOURCE, XREF, VBREF, and OFFSET.

To simplify debugging, set optimization to NOOPT. Higher optimization levels can change the location where *parameters* and instructions appear in the dump output.

Table 10 lists the COBOL compile-time options you can use to prepare your routine for run-time debugging.

Table 10. COBOL Compile-Time Options Useful in Debugging

Option	**Function**
ADATA	Produces a file containing information about the compiled program.
LIST	Produces a listing of the assembler expansion of your source code and global tables, literal pools, information about working storage, and size of routine's working storage.
MAP	Produces lists of items in the data division including the data division map, global tables, literal pools, nested program structure map and attributes.
OFFSET	Produces a condensed procedure division listing containing line numbers, statement references, and location of the first instruction generated for each statement.
OUTDD	Specifies the destination of DISPLAY statement messages.
SOURCE	Produces a listing of your source routine with any statements embedded by PROCESS or COPY statements.
TEST	Specifies whether dictionary tables are to be generated in the object module. The *hook*-location suboption of the TEST compiler option is provided for compile-time option compatibility with COBOL/370, and has no effect on the behavior of your application. When specified with any of the hook-location suboption values except NONE, this option forces the NOOPTIMIZE option. SYM suboption includes statement numbers in the LE/VSE dump and produces a symbolic dump. Use TEST only for debugging.
VBREF	Produces a cross-reference of all verb types used in the source routine and a summary of how many times each verb is used.
XREF	Creates a sorted cross-reference listing.

For more detail on COBOL compile-time options, see *IBM COBOL for VSE/ESA Programming Guide*

PL/I Compile-Time Options

When using PL/I, specify the TEST compile-time option to control the level of testing capability that is generated as part of the *object code*. Suboptions of the TEST option such as SYM, BLOCK, STMT, and PATH control the location of test hooks and specify whether or not a symbol table is generated. For more information about TEST, its suboptions, and the placement of test hooks, see *IBM PL/I for VSE/ESA Programming Guide*

To simplify debugging and decrease compile time, set optimization to NOOPTIMIZE or OPTIMIZE(0). Higher optimization levels can change the location where parameters and instructions appear in the dump output.

Table 11 lists some compile-time options that you can use to prepare PL/I routines for debugging.

Table 11. PL/I Compile-Time Options Useful in Debugging

Option	Function
AGGREGATE	Specifies that a layout for *arrays* and major structures appears in the listing.
ESD	Includes the external symbol dictionary in the listing.
GONUMBER	Tells the compiler to produce additional information specifying that line numbers from the source routine can be included in run-time messages and in the LE/VSE dump.
GOSTMT	Specifies that statement numbers are included in run-time messages and in the LE/VSE dump.
LIST	Produces a listing of the assembler expansion of source code and global tables, literal pools, information about working storage, and size of routine's working storage.
LMESSAGE	Tells the compiler to produce run-time messages in a long form. If the cause of a run-time malfunction is a programmer's understanding of language semantics, specifying LMESSAGE could better explain warnings or other information generated by the compiler.
MAP	Tells the compiler to produce tables showing how the variables are mapped in the static internal control section and in the *stack frames*, thus enabling static internal and automatic variables to be found in the LE/VSE dump. If LIST is also specified, the MAP option also produces tables showing constants, control blocks, and INITIAL variable values.
OFFSET	Specifies that the compiler prints a table of statement or line numbers for each procedure with their offset addresses relative to the primary *entry point* of the procedure.
SOURCE	Specifies that the compiler includes a listing of the source routine in the listing.
STORAGE	Includes a table of the main storage requirements for the object module in the listing.
TEST	Specifies the level of testing capability that is generated as part of the object code. TEST also controls the location of test hooks and whether or not the symbol table is generated. Because the TEST option increases the size of the object code and can affect performance, limit the number and placement of hooks.
XREF and ATTRIBUTES	Creates a sorted cross-reference listing with attributes.

For more detail on PL/I compile-time options, see *IBM PL/I for VSE/ESA Programming Guide*

Using LE/VSE Run-Time Options

There are numerous run-time options that affect debugging in LE/VSE. The ABPERC, CHECK, DEPTHCONDLMT, ERRCOUNT, TERMTHDACT, and TRAP options, for example, affect condition handling. The ABTERMENC option affects how an application ends (that is, with an *abend* or with a *return code* and *reason code*) when an *unhandled condition* of severity 2 or greater occurs.

Table 12 shows LE/VSE run-time options that affect debugging.

Table 12. LE/VSE Run-Time Options Useful in Debugging

Option	Function
ABPERC	Specifies that the indicated VSE cancel code, program-interruption code, or user-abend code bypasses the *condition handler*.
ABTERMENC	Specifies *enclave* termination behavior for an enclave ending with an unhandled condition of severity 2 or greater.
CHECK	Determines whether run-time checking is performed.
DEBUG	Controls the COBOL USE FOR DEBUGGING declarative.
DEPTHCONDLMT	Specifies the limit for the depth of nested conditions in *user-written condition handlers*.
ERRCOUNT	Specifies the number of conditions of severity 2 or greater tolerated.
MSGFILE	Specifies the *filename* of the LE/VSE message file.
MSGQ	Specifies the number of *instance specific information (ISI)* blocks that are allocated on a per-*thread* basis for use by an application. Located within the LE/VSE condition token, the ISI contains information used by the *condition manager* to identify and react to a specific occurrence of a condition.
STORAGE	Specifies that LE/VSE initializes all *heap* and *stack* storage to a user-specified value.
TERMTHDACT	Controls response when an enclave terminates due to an unhandled condition of severity 2 or greater. From LE/VSE 1.4.2 onwards, TERMTHDACT allows you to specify the dump output destination in a CICS environment.
TEST	Specifies the conditions under which a debugging tool assumes control.
TRACE	Activates LE/VSE run-time library tracing and controls the size of the trace table, the type of trace, and whether the trace table should be dumped unconditionally upon termination of the application.
TRAP	Specifies how the LE/VSE routines handle abends and program interrupts. LE/VSE expects TRAP(ON,MIN) to be in effect for successful execution of the application.
USRHDLR	Specifies the name of a user condition handler, if any, to be registered at stack frame 0.
XUFLOW	Specifies whether or not an exponent underflow causes a routine interrupt.

For a more detailed discussion of these run-time options, see the *LE/VSE Programming Guide*.

Determining Run-Time Options in Effect

The run-time options in effect at the time the routine is run can affect routine behavior. You can use the run-time option RPTOPTS(ON) to generate a report that lists run-time options and indicates where they were set.

With RPTOPTS set to ON, an options report is written to the LE/VSE message file when your routine terminates. If an abend occurs, however, the RPTOPTS output is not produced. Figure 1 on page 9 shows a sample options report.

Note: When TERMTHDACT is set with the LSTQ suboption, an LSTQ Options Report will be produced that shows the settings that were active when producing the LSTQ entries. When you set the LE/VSE default run-time options, you can change these settings using the CEELOPT macro that is supplied with the sample CEEDOPT/CEECOPT. Figure 1 on page 9 is an example of a sample LSTQ Options Report.

```
Options Report for Enclave CEEWIVP2 PHASE*** 12/19/03 12:49:15 PM
Language Environment for VSE/ESA V1 R4.4

LAST WHERE SET          OPTION
-------------------------------------------------------------------------------
Installation default      ABPERC(NONE)
Installation default      ABTERMENC(ABEND)
Installation default    NOAIXBLD
Installation default      ALL31(OFF)
Installation default      ANYHEAP(16384,8192,ANYWHERE,FREE)
Installation default      BELOWHEAP(8192,4096,FREE)
Installation default      CBLOPTS(ON)
Installation default      CBLPSHPOP(OFF)
Installation default      CHECK(OFF)
Installation default      COUNTRY(US)
Installation default    NODEBUG
Installation default      DEPTHCONDLMT(10)
Installation default      ENVAR("")
Installation default      ERRCOUNT(20)
Installation default      HEAP(32768,32768,ANYWHERE,KEEP,8192,4096)
Installation default      HEAPCHK(OFF,1,0)
Installation default      LIBSTACK(12288,4096,FREE)
Installation default      MSGFILE(SYSLST)
Installation default      MSGQ(15)
Installation default      NATLANG(UEN)
Installation default      RETZERO(OFF)
Invocation command        RPTOPTS(ON)
Installation default      RPTSTG(OFF)
Installation default    NORTEREUS
Installation default      STACK(131072,131072,BELOW,KEEP)
Installation default      STORAGE(00,NONE,NONE,32768)
Invocation command        TERMTHDACT(TRACE,LSTQ,0)
Installation default    NOTEST(ALL,"*","PROMPT","")
Installation default      TRACE(OFF,4096,DUMP,LE=0)
Installation default      TRAP(ON,MAX)
Installation default      UPSI(00000000)
Installation default    NOUSRHDLR()
Installation default      XUFLOW(AUTO)

LSTQ Options Report
-------------------------------------------------------------------------------
LSTQ Class Setting         L
LSTQ Disposition Setting   D
LSTQ Remote Node ID        LETEST
LSTQ Remote User ID        LETEST

1S55I  LAST RETURN CODE WAS 0000
EOJ CEEWIVP2  MAX.RETURN CODE=0002
```

Figure 1. Options Report Produced by LE/VSE Run-Time Option RPTOPTS(ON)

Using the CLER CICS Transaction to Display and Set Run-Time Options

The **CLER** CICS transaction allows you to interactively display and modify LE/VSE CICS-wide default runtime options, while your CICS system is running. It uses a CICS Basic Mapping Support (BMS) panel.

For details, refer to the section "CLER: Interactively Process CICS-Wide Run-Time Options" in the *LE/VSE Customization Guide*.

Activating Changed CICS-Wide Run-Time Options

The **NEWC** CICS transaction allows you to activate changed LE/VSE CICS-wide default runtime options, while your CICS system is running.

For details, refer to the section “NEWC: Activate Changed CICS-Wide Run-Time Options” in the *LE/VSE Customization Guide*.

Printing CICS-Wide Run-Time Options to the Console

The **ROPC** CICS transaction allows you to print LE/VSE CICS-wide run-time options to your z/VSE console.

For details, refer to the section “ROPC: Print CICS-Wide Run-Time Options to Console” in the *LE/VSE Customization Guide*.

Controlling Storage Allocation

This section provides information about storage report statistics to help you control storage allocation.

Controlling Storage Allocation

The following run-time options control storage allocation: STACK, LIBSTACK, HEAP, ANYHEAP, and BELOWHEAP. Ensure that these options are tuned appropriately to avoid performance problems.

To generate a report of the storage that a routine (or more specifically, an enclave) has used during its run, specify the RPTSTG(ON) run-time option. The storage report, generated during enclave termination, provides statistics that can help you understand how space is being consumed as the enclave runs. If storage management tuning is desired, the statistics can help you set the corresponding storage-related run-time options for future runs. The output is written to the LE/VSE message file.

Neither the storage report nor the corresponding run-time options include the storage that LE/VSE acquires during early initialization, before run-time options processing, and before the start of space management monitoring.

Figure 2 on page 11 shows a sample storage report, and the sections that follow describe the contents of the report.

```
Storage Report for Enclave main 04/23/96 11:01:38 AM

    STACK statistics:
      Initial size:                                     131072
      Increment size:                                   131072
      Total stack storage used (sugg. initial size)       9880
      Number of segments allocated:                          1
      Number of segments freed:                              0
    LIBSTACK statistics:
      Initial size:                                       8192
      Increment size:                                     4096
      Total stack storage used (sugg. initial size)        784
      Number of segments allocated:                          1
      Number of segments freed:                              0
    HEAP statistics:
      Initial size:                                      32768
      Increment size:                                    32768
      Total heap storage used (sugg. initial size):      20520
      Successful Get Heap requests:                         51
      Successful Free Heap requests:                        27
      Number of segments allocated:                          1
      Number of segments freed:                              0
    ANYHEAP statistics:
      Initial size:                                      16384
      Increment size:                                     8192
      Total heap storage used (sugg. initial size):       6208
      Successful Get Heap requests:                         43
      Successful Free Heap requests:                        12
      Number of segments allocated:                          1
      Number of segments freed:                              0
    BELOWHEAP statistics:
      Initial size:                                       8192
      Increment size:                                     4096
      Total heap storage used (sugg. initial size):      89952
      Successful Get Heap requests:                          8
      Successful Free Heap requests:                         3
      Number of segments allocated:                          4
      Number of segments freed:                              3
    Additional Heap statistics:
      Successful Create Heap requests:                       0
      Successful Discard Heap requests:                      0
      Total heap storage used:                               0
      Successful Get Heap requests:                          0
      Successful Free Heap requests:                         0
      Number of segments allocated:                          0
      Number of segments freed:                              0
End of Storage Report
```

Figure 2. Storage Report Produced by LE/VSE Run-Time Option RPTSTG(ON)

The statistics for initial and incremental allocations of storage types that have a corresponding run-time option differ from the run-time option settings when their values have been rounded up by the implementation, or when allocations larger than the amounts specified were required during execution. All of the following are rounded up to an integral number of double-words:

Initial STACK allocations
Initial allocations of all types of heap
Incremental allocations of all types of stack and heap

Stack Storage Statistics

LE/VSE stack storage is managed at the thread level—each thread has its own stack-type resources.

Note: LE/VSE Version 1 Release 4 supports only one thread within an enclave.

STACK and LIBSTACK Statistics:

Table 13. STACK and LIBSTACK Statistics

Statistic	Description
Initial size	The actual size of the initial segment assigned to each thread.
Increment size	The size of each incremental segment acquired, as determined by the increment portion of the corresponding run-time option.
Total stack storage used	The largest amount used by the thread at any one time.
Number of segments allocated	The number of incremental segments allocated.
Number of segments freed	The number of incremental segments freed.

Note: The number of incremental segments freed may be less than the number allocated if any of the segments were not freed individually during the life of the thread, but rather were freed implicitly in the course of thread termination.

Allocating Stack Storage: Another type of stack, called the reserve stack, is allocated for each thread and used to handle out-of-storage conditions. Its size is controlled by the 4th subparameter of the STORAGE run-time option, but its usage is neither tracked nor reported in the storage report.

LE/VSE acquires some initial storage, which is neither stack nor heap, at thread creation time; this storage is allocated from BELOWHEAP if ALL31(OFF) is in effect, or from ANYHEAP if ALL31(ON) is in effect.

Heap Storage Statistics

LE/VSE heap storage, is managed at the enclave level—each enclave has its own heap-type resources, which are shared by the threads that execute within the enclave.

HEAP, ANYHEAP, and BELOWHEAP Statistics:

Table 14. HEAP, ANYHEAP, and BELOWHEAP Statistics

Statistic	Description
Initial size	The default initial allocation, as specified by the corresponding run-time option.
Increment size	The minimum incremental allocation, as specified by the corresponding run-time option.

HEAP, ANYHEAP, BELOWHEAP, and Additional Heap Statistics:

Table 15. HEAP, ANYHEAP, BELOWHEAP, and Additional Heap Statistics

Statistic	Description
Total heap storage used	The largest total amount used by the enclave at any one time.

Table 15. HEAP, ANYHEAP, BELOWHEAP, and Additional Heap Statistics (continued)

Statistic	Description
Successful Get Heap requests	The number of Get Heap requests.
Successful Free Heap requests	The number of Free Heap requests.
Number of segments allocated	The number of incremental segments allocated.
Number of segments freed	The number of incremental segments individually freed.

Note: The number of Free Heap requests may be less than the number of Get Heap requests if the pieces of heap storage acquired by individual Get Heap requests were not freed individually, but rather were freed implicitly in the course of enclave termination.

The number of incremental segments individually freed may be less than the number allocated if the segments were not freed individually, but rather were freed implicitly in the course of enclave termination.

These statistics, in all cases, specify totals for the whole enclave.

Additional Heap Statistics: Besides the fixed types of heap, additional types of heap can be created, each with its own heap id. You can create and discard these additional types of heap by using the CEECRHP and CEEDSHP callable services.

Table 16. Additional Heap Statistics

Statistic	Description
Successful Create Heap requests	The number of successful Create Heap requests.
Successful Discard Heap requests	The number of successful Discard Heap requests.

Note: The number of Discard Heap requests may be less than the number of Create Heap requests if the special heaps allocated by individual Create Heap requests were not freed individually, but rather were freed implicitly in the course of enclave termination.

For more information about stack and heap storage, see *LE/VSE Programming Guide*, SC33-6684.

Modifying Condition Handling Behavior

Setting the condition handling behavior of your routine affects the response that occurs when the routine encounters an error.

You can modify condition handling behavior in the following ways:
- Callable services
- Run-time options
- User-written condition handlers

LE/VSE Callable Services

The callable services listed and described below can also be used to modify condition handling.

CEE5ABD
: You can use the CEE5ABD callable service to terminate an enclave using an abend.

CEEMRCR
: The CEEMRCR callable service moves the resume cursor relative to the current position of the handle cursor.

CEE5SPM
: The CEE5SPM callable service specifies the settings of the routine mask. The routine mask controls:
- Fixed overflow
- Decimal overflow
- Exponent underflow
- Significance

Note: Modifications to the LE/VSE hardware conditions can affect the behavior of your routine. Use only this service to modify any hardware conditions.

See *LE/VSE Programming Reference* for more information about callable services.

LE/VSE Run-Time Options

Some of the LE/VSE run-time options listed earlier can affect your routine's condition handling behavior. The following text describes these options:

ABPERC
: The ABPERC run-time option specifies a VSE cancel code, a program-interruption code, or a user-specified abend code that is exempted from condition handling when the LE/VSE condition handler is enabled. Normal condition handling activities are performed for everything except the specified cancel code, program-interruption code, or user-abend code. VSE cancel codes are specified as S*hh*, where *hh* is a hexadecimal cancel code. Program-interruption codes are specified as I*hh*, where *hh* is a hexadecimal interruption code. User abends are specified as U*dddd*, where *dddd* is a decimal user abend code. Any other 4-character *EBCDIC* string, such as NONE, can also be specified as a user-specified abend code. You can specify only one VSE cancel code, program-interruption code, or abend code with this option. This option assumes the use of TRAP(ON). ABPERC is not supported in CICS.

LE/VSE ignores ABPERC(S20). In this instance, VSE cancel code 20, which represents a program check, is not exempted from condition handling, and

LE/VSE condition handling semantics are in effect. You can, however, specify one program check interruption code, in the form I*hh*, to be exempted from LE/VSE condition handling.

CHECK (COBOL Only)
The CHECK run-time option specifies that “checking errors” within a COBOL application are detected.

DEPTHCONDLMT
The DEPTHCONDLMT run-time option limits the extent to which conditions can be nested in a user-written condition handler. For example, with a 5 value the initial condition and four nested conditions are processed. If the limit is exceeded, the application terminates with abend code 4091 and reason code 21 (X'15').

ERRCOUNT
The ERRCOUNT run-time option specifies the number of conditions of severity 2, 3, and 4 that are tolerated before the enclave terminates abnormally. If you specify 0 an unlimited number of conditions is tolerated.

TERMTHDACT
The TERMTHDACT run-time option sets the level of information that is produced when a condition of severity 2 or greater remains unhandled within the enclave. There are five possible parameter settings for different levels of information:
- QUIET for no information
- MSG for message only
- TRACE for message and a *traceback*
- DUMP for message, traceback, and LE/VSE dump
- UADUMP for message, traceback, LE/VSE dump, and system dump

Notes:
1. TERMTHDACT provides sub-options:
   - MSGFL and LSTQ that utilize `VSE POWER LST QUEUE` as LE/VSE dump destination under CICS.
   - reg_stor_amount that controls the amount of storage to be dumped around registers.

   For details, refer to the description of TERMTHDACT in the *LE/VSE Customization Guide*.
2. The LSTQ option can also be used to tailor LE/VSE dump destinations under batch.

TRAP(OFF)
The TRAP(OFF) run-time option disables the LE/VSE condition handler. In the batch environment, neither STXIT AB nor STXIT PC is issued. This causes abends and program interruptions to be ignored by the *run-time environment* and handled by the operating system. However, with TRAP(OFF) it is still possible to invoke the condition handler through the CEESGL callable service and pass conditions to registered user-written condition handlers. Be careful when using TRAP(OFF). Doing so can cause unpredictable results. For more information about the effects of using TRAP(OFF), see the TRAP run-time option in *LE/VSE Programming Reference*. You are recommended to use TRAP(ON,MIN) instead of TRAP(OFF) as this will minimise the involvement of the LE/VSE condition handler and circumvent any possible unexpected failures that could occur when using TRAP(OFF).

TRAP(ON)
: The TRAP(ON) run-time option enables the LE/VSE condition handler. This causes the LE/VSE condition handler to intercept abends and program interruptions. During normal processing, LE/VSE expects at least TRAP(ON,MIN) to be in effect for sucessful running of the application.

USRHDLR( ) or NOUSRHDLR( )
: The USRHDLR run-time option registers a user condition handler at stack frame 0, allowing you to register a user condition handler without having to include a call to CEEHDLR in your application, and then to recompile the application. The NOUSRHDLR run-time option (the default) does not register a user condition handler without recompiling the application to include a call to CEEHDLR. For both USRHDLR and NOUSRHDLR, you specify a phase name that contains the user condition handler that is to be registered at stack frame 0. For further details, refer to the *LE/VSE Customization Guide*, SC33-6682.

XUFLOW
: Specifies whether an exponent underflow causes a routine interrupt.

User-Written Condition Handlers

User-written condition handlers permit you to customize condition handling for certain conditions. You can *register* user-written condition handlers for the current stack frame by using the CEEHDLR callable service. When the LE/VSE condition manager encounters the condition, it requests the most recent condition handler associated with the current frame to handle the condition. If the condition is not handled, the LE/VSE condition manager percolates the condition to the next condition handler associated with the stack frame. If the condition is not handled by any of the condition handlers associated with the stack frame, the LE/VSE condition manager *percolates* the condition to the next (earlier) stack frame, and so forth to earlier stack frames until the condition has been handled or the earliest stack frame has been reached. Conditions that remain unhandled after the first (earliest) stack frame has been reached are presented to the LE/VSE condition handler. One of the following LE/VSE default actions is then taken, depending on the severity of the condition:
- Resume
- Percolate
- Promote

See *LE/VSE Programming Guide* for more information about user-written condition handlers and the LE/VSE condition manager.

Using the Assembler User Exit

For debugging purposes, the CEEBXITA *assembler user exit* can be invoked during:
- Enclave initialization
- Enclave termination
- *Process* termination

The functions of the CEEBXITA user exit depend on when the user exit is invoked and whether it is application-specific or installation-wide. Application-specific user exits must be linked with the application phase and run only when that application runs. Installation-wide user exits must be linked with the LE/VSE initialization/termination library routines and run with all LE/VSE library routines. Because an application-specific user exit has priority over any installation-wide user exit, you can customize a user exit for a particular application without affecting the user exit for any other applications.

At enclave initialization, the CEEBXITA user exit runs prior to the enclave establishment. Thus you can modify the environment in which your application runs in the following ways:
- Specify run-time options
- List VSE cancel codes, program-interruption codes, and abend codes to be exempted from LE/VSE condition handling
- Check the values of routine *arguments*

At enclave termination, the CEEBXITA user exit runs prior to the termination activity. Thus, you can request an abend and perform specified actions based on received return and reason codes. (This does not apply when LE/VSE terminates with an abend.)

At process termination, the CEEBXITA user exit runs after the enclave termination activity completes. Thus you can request an abend.

The assembler user exit must have an entry point of CEEBXITA, must be *reentrant*, and must be capable of running in *AMODE*(ANY) and *RMODE*(ANY).

See *LE/VSE Programming Guide* for more information on the assembler user exit.

Establishing Enclave Termination Behavior for Unhandled Conditions

You can establish enclave termination behavior when an unhandled condition of severity 2 or greater occurs by using one of the following methods:
- The assembler user exit
- The ABTERMENC run-time option

The assembler user exit and the ABTERMENC run-time option are briefly discussed in the following sections. For additional information, see “Using the Assembler User Exit” on page 16 and *LE/VSE Programming Guide*.

The Assembler User Exit

You can use the assembler user exit to establish enclave termination behavior for an enclave ending with an unhandled condition of severity 2 or greater in the following ways:

- If you do not request an abend in the assembler user exit for the enclave termination call, LE/VSE honors the setting of the ABTERMENC option to determine how to end the enclave.
- If you request an abend in the assembler user exit for the enclave termination call, LE/VSE issues an abend to end the enclave.

See *LE/VSE Programming Guide* for more information on the assembler user exit.

Using the ABTERMENC Run-Time Option

The ABTERMENC run-time option sets the enclave termination behavior for an enclave ending with an unhandled condition of severity 2 or greater.

If you specify the IBM-supplied *default* suboption RETCODE, LE/VSE uses the CEEAUE_ABND flag value set by the assembler user exit (which is called for enclave termination) to determine whether or not to issue an abend to end the enclave when an unhandled condition of severity 2 or greater occurs.

If you specify the ABEND suboption, LE/VSE issues an abend to end the enclave regardless of the setting of the CEEAUE_ABND flag. Additionally, the assembler user exit can alter the abend code, abend reason code, and abend dump attribute.

See *LE/VSE Programming Reference* for more information on using ABTERMENC.

Using Messages in Your Routine

You can create messages and use them in your routine to indicate the status and progress of the routine during *run time*, and to display variable values. The process of creating messages and using them requires that you create a message source file, and convert the source file into a loadable phase for use in your routine. For a complete explanation of how to create messages and use them in your routine, see *LE/VSE Programming Guide*.

You can use the LE/VSE callable service CEEMOUT to direct user-created message output to the LE/VSE message file. To direct the message output to another destination, use the LE/VSE MSGFILE run-time option to specify the filename of the file.

Each LE/VSE-conforming language also provides ways to display both user-created and run-time messages. (See "Interpreting Run-Time Messages" on page 25 for an explanation of LE/VSE run-time messages.)

For C routines, output from the `printf()` function is directed to `stdout`, which is associated with `SYSLST`. All C run-time messages and `perror()` messages are directed to `stderr`. `stderr` corresponds to the filename associated with the LE/VSE MSGFILE run-time option. The destination of the `printf()` function output can be changed by using the redirection `1>&2` at routine invocation to redirect `stdout` to the `stderr` destination. Both streams can then be controlled by the MSGFILE run-time option.

For COBOL routines, you can use the DISPLAY statement to display messages. Output from the DISPLAY statement is directed to SYSLST. SYSLST is the IBM-supplied default for the LE/VSE message file. The OUTDD compile-time option can be used to change the destination of the DISPLAY messages.

Under PL/I, run-time messages are directed to the file specified in the LE/VSE MSGFILE run-time option, instead of the PL/I SYSPRINT STREAM PRINT file. User-specified output is still directed to the PL/I SYSPRINT STREAM PRINT file. To direct this output to the LE/VSE MSGFILE file, specify the run-time option MSGFILE(SYSPRINT).

For a more detailed explanation of handling message output in C, COBOL, and PL/I routines, see *LE/VSE Programming Guide*.

Using Condition Information

If a condition that might require attention occurs while an application is running, LE/VSE builds a condition token. The condition token contains 12 *bytes* of information about the condition that LE/VSE or your routines can use to respond appropriately. Each condition is associated with a single LE/VSE run-time message.

You can use this condition information in two main ways:

- Specify the feedback code parameter when calling LE/VSE services (see “Using the Feedback Code”).
- Code a *symbolic feedback code* in a user-written condition handler (see “Using the Symbolic Feedback Code” on page 21).

Using the Feedback Code

The feedback code is a parameter of the LE/VSE callable services. The feedback code can be optional or required, depending on the programming language you use. [2]

When you provide the feedback code (*fc*) parameter, the callable service in which the condition occurs sets the feedback code to a specific value called a condition token.

When you do not provide the *fc* parameter, any nonzero condition is signaled and processed by LE/VSE condition handling routines. If you have registered a user-written condition handler, LE/VSE passes control to the handler, which determines the next action to take. If the condition remains unhandled, LE/VSE writes a message to the LE/VSE message file. The message is the translation of the condition token into English (or another supported national language).

LE/VSE provides callable services that can be used to convert condition tokens to routine variables, messages, or signaled conditions. Table 17 lists these callable services and their functions.

Table 17. LE/VSE Callable Services Used to Process Nonzero Condition Tokens

Callable Service	Function
CEEMSG	Transforms the condition token into a message and writes the message to the message file.
CEEMGET	Transforms the condition token into a message and stores the message in a buffer.
CEEDCOD	Decodes the condition token; that is, separates it into distinct user-supplied variables.
CEESGL	Signals the condition. This passes control to any registered user-written condition handlers. If a user-written condition handler does not exist, or the condition is not handled, LE/VSE by default writes the corresponding message to the message file and terminates the routine for severity 2 or higher. For severity 0, LE/VSE continues without writing a message. For severity 1, LE/VSE continues. If the severity 1 condition is associated with a COBOL routine, a message is issued. If the severity 1 condition is associated with a non-COBOL routine, a message is not issued. For details, see *LE/VSE Programming Guide*.

There are two types of condition tokens. Case 1 condition tokens contain condition information, including the LE/VSE message number. All LE/VSE callable services and most application routines use case 1 condition tokens. Case 2 condition tokens contain condition information and a user-specified class and cause code. Application routines, user-written condition handlers, assembler user exits, and some operating systems can use case 2 condition tokens.

2. For COBOL routines, you must provide the *fc* parameter in each call to an LE/VSE callable service. For C and PL/I routines, this parameter is optional. See *LE/VSE Programming Guide* for more information about *fc* and condition tokens.

Figure 3 illustrates the structure of the condition token.

0 - 31	32 - 33	34 - 36	37 - 39	40 - 63	64 - 95
Condition_ID	Case Number	Severity Number	Control Code	Facility_ID	ISI

```
For Case 1 condition tokens,
Condition_ID =
```

0 - 15	16 - 31
Severity Number	Message Number

```
A symbolic feedback code represents the first 8 bytes of a condition
token.
It contains the Severity, Message Number, Case Number, Severity,
Control Code, and Facility_ID.
```

Figure 3. LE/VSE Condition Token

For example, in the condition token: `X'0003032D 59C3C5C5 00000000'`

- X'0003' is severity.
- X'032D' is message number 813.
- X'59' are hexadecimal flags for case, severity, and control.
- X'C3C5C5' is the CEE facility ID.
- X'00000000' is the ISI (In this case, no ISI was provided.)

If an LE/VSE traceback or dump is generated while a condition token is being processed or when a condition exists, LE/VSE writes the run-time message to the condition section of the traceback or dump. If a condition is detected when a callable service is invoked without a feedback code, the condition token is passed to the LE/VSE condition manager. The condition manager polls active condition handlers for a response. If a condition of severity 0 or 1 remains unhandled, LE/VSE resumes. For a severity 0 condition, a message is not issued. For a severity 1 condition, if the condition is associated with a COBOL routine, a message is issued. If the severity 1 condition is associated with a non-COBOL routine, a message is not issued. For unhandled conditions of severity 2 or greater, LE/VSE issues a message and terminates. See Part 3, “Run-Time Messages and Codes,” on page 161 for a list of LE/VSE run-time messages and corrective information.

If a second condition is raised while LE/VSE is attempting to handle a condition, the message `CEE0374C CONDITION = <message no.>`. can be displayed using a write-to-operator (WTO). The message number in the CEE0374C message indicates the original condition that was being handled when the second condition was raised. This can happen when a critical error is signaled (for example, when internal control blocks are damaged).

If the output for this error message appears several times in sequence, the conditions appear in order of occurrence. Correcting the earliest condition can cause your application to run successfully.

Figure 4 shows the error message and associated output for a nested condition.

```
CEE0374C CONDITION = CEE3207S  TOKEN = 00030C87 59C3C5C5 00000000
         WHILE RUNNING PROGRAM PROCB WHICH STARTS AT  00020584
         AT THE TIME OF INTERRUPT
         PSW     FFE40007 D002040E
         GPR 0-3 00000000 00000001 00000002 00000003
         GPR 4-7 00000004 00000005 00000006 00000007
         GPR 8-B 00000008 00000009 0000000A 00020280
         GPR C-F 00362308 002D1240 500203E0 00000000
```

Figure 4. Message and Output for Nested Conditions

The token is the hexadecimal condition token for the original condition. The routine address is the location of the routine in which the original error occurred. The last *word* of the *program status word (PSW)* indicates the location where the original error occurred. The GPRs are the hexadecimal values in the registers at the time of the original error. If CEE3250 is the original condition, the abend code can be found in GPR1 and the reason code for the abend can be found in GPR15.

Using the Symbolic Feedback Code

The symbolic feedback code 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.

For more details on symbolic feedback codes, see *LE/VSE Programming Guide*.

Chapter 2. Classifying Errors

This chapter describes errors that commonly occur in LE/VSE routines, and explains how to use run-time messages and abend codes to obtain information about errors in your routine. It consists of these main sections:
- "Identifying Problems in Routines"
- "Interpreting Run-Time Messages" on page 25
- "Understanding Abend Codes and VSE Cancel Codes" on page 26

Identifying Problems in Routines

The following sections describe how you can identify errors in LE/VSE routines. Included are common error symptoms and solutions.

LE/VSE Module Names

You can identify LE/VSE-supplied module elements by any of the following three-character prefixes:
- CEE (LE/VSE)
- EDC (C)
- IGZ (COBOL)
- IBM (PL/I)

Module elements with other prefixes are not part of the LE/VSE product.

Common Errors in Routines

Table 18 lists some common errors with simple solutions.

Table 18. Common Errors in Routines

Error	Solution
Lack of virtual storage [3]	Increase your *partition* GETVIS size or decrease your storage usage (stack size) by using the storage-related run-time options and callable services.
Lack of disk space	Increase your disk allocation.
Unavailable executable phases	Check your LIBDEF statements.

If one of the items listed above is not the cause of error, examine your routine or routines for changes since the last successful run. If there have been changes, review these changes for errors that might be causing the problem. One way to isolate the problem is to branch around or comment out recent changes and rerun the routine. If the run is successful, the error can be narrowed to the scope of the changes. Also, review changes to files that can affect routine behavior.

Changes in levels of optimization, *addressing modes*, and input/output file formats can also cause unanticipated problems in your routine.

In most cases, run-time messages or generated condition tokens point to the nature of the error and the most efficient corrective action. To help you classify the error

3. See "Controlling Storage Allocation" on page 10 for information about using storage in routines.

and determine the most useful method to fix the problem, Table 19 lists types of errors, possible symptoms, and possible corrections.

Table 19. Common Error Symptoms, Possible Causes, and Programmer Responses

Error Symptom	Possible Cause	Programmer Response
Numbered run-time message appears	Condition raised in routine	For any messages you receive, read the Programmer Response for the messages listed in Part 3, "Run-Time Messages and Codes," on page 161. For information about message structure, see "Interpreting Run-Time Messages" on page 25.
User abend code ≥ 4000	a) LE/VSE detected an error and could not proceed b) An unhandled software-raised condition occurred and ABTERMENC(ABEND) was in effect c) The assembler user exit requested an abend for an unhandled condition of severity 4	For any abends you receive, read the appropriate Explanation listed in Chapter 13, "LE/VSE Abend Codes," on page 329.
User abend code < 4000	a) A non-LE/VSE abend occurred b) The assembler user exit requested an abend for an unhandled condition of severity ≥2	See Chapter 13, "LE/VSE Abend Codes," on page 329. Check for a *subsystem*-generated abend or a user-specified abend.
System abend with TRAP(OFF)	Cause depends on type of malfunction	Respond appropriately. Refer to the messages and codes book of the operating system.
System abend with TRAP(ON)	System detected error	Refer to the messages and codes book of the operating system.
No response (wait/loop)	Application logic failure	Check routine logic. **Note:** Ensure ERRCOUNT and DEPTHCONDLMT run-time options are set to a nonzero value.
Unexpected message (message received was not from most recent service)	Condition caused by something related to current service	Generate a traceback using CEE5DMP and see Part 3, "Run-Time Messages and Codes," on page 161.
Incorrect output	Incorrect file definitions, storage overlay, incorrect routine mask setting, references to uninitialized variables, data input errors, or application routine logic error	Correct the appropriate parameters.
No output	Check filename, file definitions, and message file setting	Correct the appropriate parameters.
Nonzero return code from enclave	Unhandled condition of severity 2, 3, or 4, or the return code was issued by the application routine	Check the LE/VSE message file for run-time messages.

Interpreting Run-Time Messages

The first step in debugging your routine is to look up any run-time messages that appear in the traceback. Run-time messages are written to the LE/VSE message file (default filename is SYSLST). The message file filename can be changed with the MSGFILE run-time option. For more information about displaying run-time messages for C, COBOL, and PL/I routines, see *LE/VSE Programming Guide*.

Run-time messages provide users with additional information about a condition and possible solutions for any errors that occurred. Run-time messages can be issued by LE/VSE common routines or language-specific run-time routines. Run-time messages contain a message prefix, message number, *severity code*, and descriptive text. The message prefix indicates the LE/VSE component that generated the message. The message number identifies the condition raised and references additional condition and programmer response information. Severity codes indicate the severity of the condition that occurred. The message text provides a brief explanation of the condition. In the following example LE/VSE message:

```
CEE3206S The system detected a specification exception.
```

- The message prefix is CEE.
- The message number is 3206.
- The severity code is S.
- The message text is "The system detected a specification exception".

The message prefix is the first three characters of the message number and is also the facility ID in the condition token. Table 20 provides additional information about LE/VSE run-time messages.

Table 20. Message Prefixes and Associated LE/VSE Components

Message Prefix	LE/VSE Component	For a List, See
CEE	Common run-time	Chapter 8, "LE/VSE Run-Time Messages," on page 163
EDC	C run-time	Chapter 10, "C Run-Time Messages," on page 243
IGZ	COBOL run-time	Chapter 12, "COBOL Run-Time Messages," on page 307
IBM	PL/I run-time	Chapter 11, "PL/I Run-Time Messages," on page 261

The message number is the 4-digit number following the message prefix. Leading zeros are inserted if the message number is less than four digits.

The severity code is the letter following the message number. Severity codes indicate the level of attention called for by the condition. Messages with severity of I are informational messages and do not usually require any corrective action. In general, if more than one run-time message appears, the first noninformational message indicates the problem. For a complete list of severity codes, severity values, condition information, and default actions, see *LE/VSE Programming Guide*.

Note: LE/VSE messages can appear even though you made no explicit calls to LE/VSE services. This is because C, COBOL, and PL/I run-time library routines commonly use the LE/VSE services.

Understanding Abend Codes and VSE Cancel Codes

When using LE/VSE, an abnormal termination is usually accompanied by an LE/VSE abnormal termination message containing a user abend code or a VSE cancel code, or a VSE system abend message. User abend codes have the format *dddd*, where *dddd* is a decimal user abend code. VSE cancel codes have the format *hh*, where *hh* is a hexadecimal cancel code. LE/VSE abend codes are usually in the range of 4000 to 4095. User-specified abends use the range of 0 to 3999.

Examples of abnormal termination messages containing user abend codes are:

User (LE/VSE) abend code: U4082

```
CEE3322C EXECUTION ABNORMALLY TERMINATED WITH USER-ABEND CODE 4082
         AND REASON CODE 00
```

User-specified abend code: U0005

```
CEE3322C EXECUTION ABNORMALLY TERMINATED WITH USER-ABEND CODE 0005
         AND REASON CODE 00
```

An example of an abnormal termination message containing a VSE cancel code is:

```
CEE3321C EXECUTION ABNORMALLY TERMINATED WITH VSE CANCEL CODE 20
         AND INTERRUPTION CODE 05
```

An example of a VSE system abend message is:

```
0S03I PROGRAM CHECK INTERRUPTION - HEX LOCATION 0006D2D0 -
CONDITION CODE 04 - PROTECTION EXCEPTION
```

User Abends

If you receive an LE/VSE abend code, see Chapter 13, "LE/VSE Abend Codes," on page 329 for a list of abend codes, error descriptions, and programmer responses.

The LE/VSE callable service CEE5ABD terminates your application with an abend. You can set the *clean-up* parameter value to determine how the abend is processed and how LE/VSE handles the raised condition. For more information about CEE5ABD and *clean-up*, see *LE/VSE Programming Reference*.

The TRAP(OFF) run-time option disables LE/VSE condition handling for program checks and abends. With TRAP(OFF), program checks such as addressing exceptions, data exceptions, and decimal divide exceptions result in VSE system abends rather than LE/VSE conditions.

Note: The use of TRAP(OFF) does not affect the LE/VSE callable service CEESGL, which signals a condition to the LE/VSE condition manager.

You can specify the ABTERMENC run-time option to determine what action is taken when an unhandled condition of severity 2 or greater occurs. For more information on ABTERMENC, see "Using the ABTERMENC Run-Time Option" on page 17 and *LE/VSE Programming Reference*.

User abends, such as LE/VSE 4xxx abends or abends raised by a call to the CEE5ABD service, can cause the generation of a system dump. Although system dumps are sometimes required for debugging complex error situations, it is usually better to generate an LE/VSE formatted dump. To request an LE/VSE dump whenever an unhandled condition is raised, specify both TRAP(ON) and TERMTHDACT(DUMP) run-time options. Your routine can also produce an LE/VSE dump at any time by calling the CEE5DMP callable service. The TRAP(ON) run-time option causes the LE/VSE condition handler to attempt to handle the abend. For a detailed explanation of the run-time options and callable services discussed in this section, see *LE/VSE Programming Guide*.

VSE Cancel Codes and VSE System Abends

If you receive an LE/VSE abnormal termination message containing a VSE cancel code, look up the code and the corresponding information in *z/VSE Messages and Codes*

If you receive a VSE system abend message, look up the message and the corresponding information in *z/VSE Messages and Codes*

When a VSE system abend occurs, the operating system can generate a system dump. System dumps are written to SYSLST, or to a dump sublibrary if one is defined. See "Generating a System Dump" on page 54 for more information about system dumps.

Chapter 3. Using LE/VSE Debugging Facilities

This chapter describes methods of debugging routines in LE/VSE. Currently, most problems in LE/VSE and member language routines can be determined through information provided in the LE/VSE dump. It consists of these main sections:
- "Using the Debug Tool for VSE/ESA"
- "Using Attention Routine Interface Commands"
- "Using the LE/VSE Dump Service" on page 31
- "Multiple Enclave Dumps" on page 52
- "Generating a System Dump" on page 54
- "Requesting an LE/VSE Trace for Debugging" on page 55
- "Diagnosing Storage Leak Problems" on page 58

Using the Debug Tool for VSE/ESA

Debugging tools are designed to help you detect errors early in your routine. IBM offers the *Debug Tool for VSE/ESA*, which you can use to examine, monitor, and control how your routines run, and debug your routines interactively or in batch mode. Debug Tool for VSE/ESA also provides facilities for setting breakpoints and altering the contents and values of variables. LE/VSE run-time options can be used with Debug Tool for VSE/ESA to debug or analyze your routine. Refer to the Debug Tool for VSE/ESA publications for a detailed explanation of how to invoke and run Debug Tool for VSE/ESA. For a list of Debug Tool for VSE/ESA publications, see "Where to Find More Information" on page xvii. If you are using another debugging facility, refer to the appropriate documentation.

Using Attention Routine Interface Commands

The attention routine interface is pre-customized and activated in z/VSE. You can use the attention routine (AR) commands described below, to display Language Environment information.

Command **Description**

D CEE,CEEDOPT

Displays a report of the current BATCH environment system-wide default Run-Time option settings. This can be useful for assisting in the debugging of application programs and for application-storage tuning. Before you can use this command, the sample job CEL$OPT$ (member CEEWOPTJ.Z) must be:
- Resident in the VSE/POWER reader queue.
- Configured to execute in a VSE/POWER class that has an *available partition*, each time this command is issued.

Note: CEEUOPT members that might be included in application programs will not affect the options report produced by this command.

Here is an example of the type of report that is produced by the D CEE,CEEDOPT command:

```
1Q47I   C1 CEL$OPT$ 02945 FROM (SYSA) , TIME=11:17:11
// JOB CEL4OPTS - PRODUCE BATCH INSTALLATION-WIDE OPTIONS REPORT
       DATE 11/18/2004, CLOCK 11/17/11
1QF0I  DATA FILE 090% FULL - QUEUE FILE 039% FULL
CEE3558I LE/VSE LSTQ Support Unavailable.
```

```
Options Report for Enclave CEL4OPTS 11/18/04 11:17:11 AM
Language Environment for VSE/ESA V1 R4.4

LAST WHERE SET          OPTION
-----------------------------------------------------------------
Installation default      ABPERC(NONE)
Installation default      ABTERMENC(ABEND)
Installation default    NOAIXBLD
Installation default      ALL31(OFF)
Installation default      ANYHEAP(16384,8192,ANYWHERE,FREE)
Installation default      BELOWHEAP(8192,4096,FREE)
Installation default      CBLOPTS(ON)
Installation default      CBLPSHPOP(OFF)
Installation default      CHECK(OFF)
Installation default      COUNTRY(US)
Installation default    NODEBUG
Installation default      DEPTHCONDLMT(10)
Installation default      ENVAR("")
Installation default      ERRCOUNT(20)
Installation default      HEAP(32768,32768,ANYWHERE,KEEP,8192,4096)
Installation default      HEAPCHK(OFF,1,0)
Installation default      LIBSTACK(12288,4096,FREE)
Installation default      MSGFILE(LSTQ)
Installation default      MSGQ(15)
Installation default      NATLANG(UEN)
Installation default      RETZERO(OFF)
Installation default      RPTOPTS(OFF)
Installation default      RPTSTG(OFF)
Installation default    NORTEREUS
Installation default      STACK(131072,131072,BELOW,KEEP)
Installation default      STORAGE(00,NONE,NONE,32768)
Installation default      TERMTHDACT(DUMP,LSTQ,0)
Installation default    NOTEST(ALL,"*","PROMPT","")
Installation default      TRACE(OFF,4096,DUMP,LE=0)
Installation default      TRAP(ON,MAX)
Installation default      UPSI(00000000)
Installation default    NOUSRHDLR()
Installation default      XUFLOW(AUTO)

EOJ CEL4OPTS  MAX.RETURN CODE=0000
       DATE 11/18/2004, CLOCK 11/17/12, DURATION   00/00/01
```

D CEE,CEEEXIT

Produces a report of all installed Language Environment exits, and the status of these exits. The report only includes exits that are *currently resident* in the SVA. Version information for the assembler user exit is only available for IBM-supplied or IBM-sample-tailored exit routines. OEM versions may not conform to the standard entry code requirements, and so version information may not be available.

Note: Any assembler user exits are not included in the report if they are included in specific applications during the link-edit.

Here is an example of the type of report that is produced by the D CEE,CEEEXIT command:

```
CEL4001I Currently defined Lang. Env. exits are

CEL4002I Exit Name  Exit Type     Dump Cntrl
CEL4007I CEEBNATX   Abnorm Term.  Postdump
```

```
CEL4000I Exit Name  Exit Type     Exit Status   Version
CEL4010I CEEBXITA   Asm User.     Default       V1.R4.M4
  Lang. Env. for VSE command processing completed
1I40I  READY
```

D CEE,CEESTAT

Displays all the available Language Environment system information. These types of detail are displayed:

- Current Language Environment level.
- Latest Language Environment APAR installed.
- Languages that are installed.
- Latest APAR levels of the installed languages.
- Information about certain core Language Environment load modules (which might be required for Level 2 support and analysis).

Note: If maintenance is applied to any of the Language Environment components, a refresh of the attention routine interface is required using the IBM-supplied sample CEEWARC. For details, refer to the section “Planning to Activate LE/VSE Attention Routine Commands” in the manual *LE/VSE Customization Guide*, SC33-6682.

Here is an example of the type of report that is produced by the D CEE,CEESTAT command:

```
CEL4014I Current Language Environment for VSE Status Information
-----------------------------------------------------------------
CEL4011I Language Environment Version :  01.04.04  BASELVL
CEL4018I Report Produced by CEL4CMDR at  83B09310  BASELVL

CEL4012I Language  Installed  SVA Resident  SVA Load Addr  APAR
CEL4013I LE/COBOL      YES         NO         Unknown     BASELVL
CEL4013I LE/PLI        YES         NO         Unknown     BASELVL
CEL4013I C/Locale      YES         YES        03B50AF0    Unknown
CEL4013I LE/C          YES         YES        039DCF80    BASELVL

CEL4015I  Module   Installed  SVA Resident  SVA Load Addr
CEL4013I CEEBINIT      YES         YES        001BFB60
CEL4013I CEEPLPKA      YES         YES        03A5E700
CEL4013I CEECCICS      YES         YES        001D1DA0
  Lang. Env. for VSE command processing completed
1I40I  READY
```

Each line of the console reports described above has a message number associated with it. These message numbers are described later in this manual. By referring to these messages, you can obtain further details about a report.

Using the LE/VSE Dump Service

The following sections provide information about using the LE/VSE dump service, and describe the contents of the LE/VSE dump.

There are three ways to invoke the LE/VSE dump service:

- CEE5DMP callable service
- TERMTHDACT run-time option
- Language-specific functions

Generating a Dump with CEE5DMP

The CEE5DMP callable service generates a dump of LE/VSE and the loaded member language library routines. CEE5DMP can be called directly from an application routine, and generates a dump of the run-time environment at the point of the CEE5DMP call.

The dump can contain information about conditions, tracebacks, variables, control blocks, stack and heap storage, file status and attributes, and language-specific information, depending on the options you specify. No conditions are necessary for this callable service to be used.

All output from CEE5DMP is written to the default filename CEEDUMP, unless you use the FNAME option to specify another filename. If you do not provide run-time *JCL* for the file CEEDUMP (or any other file you specify using the FNAME option), all output from CEE5DMP is written to SYSLST. The syntax for CEE5DMP is shown below.

Syntax

```
►►──CEE5DMP──(──title──,──options──,──fc──)──────────────────────────────►◄
```

title
An 80-byte fixed-length character string that contains a title to be printed at the top of each page of the dump.

options
A 255-byte fixed-length character string that contains options describing the type, format, and destination of dump information. The options are declared as a string of keywords separated by blanks or commas. Some options also have suboptions that follow the option keyword, and are contained in parentheses. The last option declaration is honored if there is a conflict between it and any preceding options.

fc (output)
A 12-byte feedback token code that indicates the result of a call to CEE5DMP. If specified as an argument, then feedback information, in the form of a condition token, is returned to the calling routine. If not specified, and the requested operation was not successfully completed, the condition is signaled to the condition manager.

You can use options with CEE5DMP to specify what information is included in the dump. Table 21 lists dump options and related information.

Table 21. CEE5DMP Options and Related Information

Dump Options	Abbreviation	Action Taken
ENCLAVE(ALL)	ENCL	Dumps all enclaves associated with the current process.[1] This is the default setting for ENCLAVE.
ENCLAVE(CURRENT)	ENCL(CUR)	Dumps the current enclave.[2]
ENCLAVE(n)	ENCL(n)	Dumps a fixed number of enclaves, indicated by *n*.
THREAD(CURRENT)	THR(CUR)	Dumps the current thread in this enclave.
TRACEBACK	TRACE	Includes a traceback of all *active routines*. The traceback shows transfer of control from either calls or *exceptions*. Calls include PL/I transfers of control from BEGIN-END blocks or ON-units.

Table 21. CEE5DMP Options and Related Information (continued)

Dump Options	Abbreviation	Action Taken
NOTRACEBACK	NOTRACE	Does not include a traceback of all active routines.
FILES	FILE	Includes attributes of all open files. File control blocks are included when the BLOCKS option is also specified. File buffers are included when the STORAGE option is specified.
NOFILES	NOFILE	Does not include file attributes.
VARIABLES	VAR	Includes a symbolic dump of all variables, arguments, and registers.
NOVARIABLES	NOVAR	Does not include variables, arguments, and registers.
BLOCKS	BLOCK	Dumps control blocks from LE/VSE and member language libraries.
NOBLOCKS	NOBLOCK	Does not include control blocks.
STORAGE	STOR	Dumps the storage used by the routine. The number of routines dumped is controlled by the STACKFRAME option.
NOSTORAGE	NOSTOR	Suppresses storage dumps.
STACKFRAME(ALL)	SF(ALL)	Dumps all stack frames from the *call chain*. This is the default setting for STACKFRAME.
STACKFRAME(n)	SF(n)	Dumps a fixed number of stack frames, indicated by *n*, from the call chain. The specific information dumped for each stack frame depends on the VARIABLES, BLOCK, and STORAGE options declarations. The first stack frame dumped is the *caller* of CEE5DMP, followed by its caller, and proceeding backward up the call chain.
PAGESIZE(n)	PAGE(n)	Specifies the number of lines on each page of the dump.
FNAME(s)	FNAME(s)	Specifies the filename of the file to which the dump is written.
CONDITION	COND	Dumps condition information for each condition active on the call chain.
NOCONDITION	NOCOND	For each condition active on the call chain, does not dump condition information.
ENTRY	ENT	Includes a description of the *program unit* that called CEE5DMP and the registers on entry to CEE5DMP.
NOENTRY	NOENT	Does not include a description of the program unit that called CEE5DMP or registers on entry to CEE5DMP.
REGSTOR(n)	REGST(n)	Controls the amount of storage to be dumped around registers. `n` must be in the range 0 to 256, and indicates the storage (in bytes) to be dumped around each register. `n` is rounded up to the nearest multiple of 32. If you do not require a dump of storage around registers (regardless of the `reg_stor_amount` value on the TERMTHDACT runtime option), you must specify `REGSTor(0)` as a CEE5DMP option.

Note:

1. In ILC applications in which a C routine calls another member language routine, and that routine in turn calls CEE5DMP, traceback information for the C routine is not provided in the dump.
2. On CICS, only (CURRENT) and (1) values are supported for the ENCLAVE dump option.

   In the batch environment, all values for ENCLAVE are supported.

The IBM-supplied default settings for CEE5DMP are:

```
TRACEBACK THREAD(CURRENT) FILES VARIABLES NOBLOCKS
NOSTORAGE STACKFRAME(ALL) PAGESIZE(60)
FNAME(CEEDUMP) CONDITION ENTRY
```

For additional information about the CEE5DMP callable service and dump options, see *LE/VSE Programming Reference*. For an example of an LE/VSE dump, see "Understanding an LE/VSE Dump" on page 36.

Generating a Dump with TERMTHDACT

You can use the TERMTHDACT run-time option to produce a traceback or dump upon abnormal termination of a thread due to an unhandled condition of severity 2 or greater (when this occurs, LE/VSE terminates the enclave). See *LE/VSE Programming Guide* for information on enclave termination.

The TERMTHDACT run-time option calls CEE5DMP to produce a dump during program checks, abnormal terminations, or calls to the CEESGL service. Enclave- and process-related storage could have changed from the time the dump request was issued.

The TERMTHDACT suboptions QUIET, MSG, TRACE, DUMP, and UADUMP control the level of information available. Table 22 lists these suboptions, the levels of information produced, and the destination of each.

Table 22. Suboptions for TERMTHDACT

Suboption	Level of Information	Destination
QUIET	No information	No destination.
MSG	Message	Filename specified in MSGFILE run-time option.
TRACE	Message and dump containing only a traceback	Message goes to filename specified in MSGFILE run-time option. Traceback goes to CEEDUMP file or SYSLST.
DUMP	Message and complete dump	Message goes to filename specified in MSGFILE run-time option. Dump goes to CEEDUMP file or SYSLST.
UADUMP	Message, complete dump, and VSE system dump.	Message goes to filename specified in MSGFILE run-time option. Dump goes to CEEDUMP file or SYSLST. System dump goes to dump sublibrary or SYSLST.

Note: From VSE/ESA 2.6 onwards, TERMTHDACT provides:
- Additional sub-options MSGFL and LSTQ that utilize `VSE POWER LST QUEUE` as LE/VSE dump destination under CICS.
- Additional sub-option reg_stor_amount that controls the amount of storage to be dumped around registers.

For details, refer to the description of TERMTHDACT in the *LE/VSE Customization Guide*.

Although you can modify CEE5DMP options, you cannot change options for a traceback or dump produced by TERMTHDACT.

The TRACE suboption of TERMTHDACT uses the following dump options: NOENTRY, CONDITION, TRACEBACK, THREAD(ALL), NOBLOCKS, NOSTORAGE, VARIABLES, FILES, STACKFRAME(ALL), PAGESIZE(60), and FNAME(CEEDUMP).

The DUMP and UADUMP suboptions of TERMTHDACT use the following dump options: NOENTRY, CONDITION, TRACEBACK, THREAD(ALL), BLOCKS, STORAGE, VARIABLES, FILES, STACKFRAME(ALL), PAGESIZE(60), and FNAME(CEEDUMP).

See *LE/VSE Customization Guide* for more information about the TERMTHDACT run-time option.

Generating a Dump with Language-Specific Functions

In addition to the CEE5DMP callable service and the TERMTHDACT run-time option, you can use language-specific conventions such as C functions and the PL/I PLIDUMP service to generate a dump.

C routines can use the functions `cdump()`, `csnap()`, and `ctrace()` to produce an LE/VSE dump. All three functions call the CEE5DMP callable service, and each function includes an options string, consisting of different CEE5DMP options, that you can use to control the information contained in the dump. For more information on these functions, see “Generating an LE/VSE Dump of a C Routine” on page 77.

PL/I routines can call PLIDUMP instead of CEE5DMP to produce a dump. PLIDUMP includes options that you can specify to obtain a variety of information in the dump. For a detailed explanation about PLIDUMP, see “Generating an LE/VSE Dump of a PL/I Routine” on page 139.

Understanding an LE/VSE Dump

The LE/VSE dump service generates output of data and storage from the LE/VSE run-time environment on an enclave basis. Figure 5 illustrates a dump for enclave DRVHAND, and “Sections of the LE/VSE Dump” on page 38 describes the information contained in the dump.

Note: This example assumes full use of the CEE5DMP dump options. Not all information appears in dumps generated by unhandled conditions. Ellipses are used to summarize some sections of the dump.

For easy reference, the sections of the dump are numbered to correspond with the description of each section that follows.

```
CEE5DMP V1 R4.2: Sample LE/VSE dump   1                           07/27/01 11:45:54 AM               PAGE:    1

CEE5DMP CALLED BY PROGRAM UNIT HANDLER AT OFFSET +0000035C.   2

REGISTERS ON ENTRY TO CEE5DMP:   3

  PM....... 0000
  GPR0..... 0057476C  GPR1..... 00574788  GPR2..... 0051F7EC  GPR3..... 00578250
  GPR4..... 00578150  GPR5..... 0054F018  GPR6..... 0054F608  GPR7..... 0054F540
  GPR8..... 0054F614  GPR9..... 00578100  GPR10.... 00500754  GPR11.... 00500898
  GPR12.... 00519E48  GPR13.... 00574620  GPR14.... 80541EEE  GPR15.... 83A01D68
  FPR0..... 40404040  40404040            FPR2..... 40404040  40404040
  FPR4..... 40404040  40404040            FPR6..... 40404040  40404040
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (00000000)
        +0000 00000000    INACCESSIBLE STORAGE.
        +0020 00000020    INACCESSIBLE STORAGE.
        +0040 00000040    INACCESSIBLE STORAGE.
      STORAGE AROUND GPR 1 (00515608)
        -0020 005155E8  0054F7D8 00020518 0054F528 C5000000  00000000 00000002 00000000 00000000  |..7Q......5.E...................|
        +0000 00515608  0054F540 0054F608 0054F614 005159FC  00000000 00000000 0054F604 0054F528  |..5 ..6...6..&.#169;......6...5.|
        +0020 00515628  0054F614 005159FC 00000001 010C0002  40254000 00000000 00000020 00000009  |..6..&.#169;...................|
⋮
      STORAGE AROUND GPR15 (00500658)
        -0020 00500638  58602008 50605134 D20F5138 3116D203  51483126 D20F5150 312A98EC D00C07FE  |.-..&-&.#169;.K.169;..K.169;....|
        +0000 00500658  47F0F028 00C3C5C5 00000000 00000014  47F0F001 4ACEAC00 00500704 00000000  |.00..CEE.........00......&......|
        +0020 00500678  00000000 00000000 90ECD00C 4110F038  98EFF04C 07FF0000 00500658 0050074C  |.............0.q.0<.....&...&.<|

INFORMATION FOR ENCLAVE DRVHAND   4

  INFORMATION FOR THREAD 8000000000000000   5

  REGISTERS ON ENTRY TO CEE5DMP:
    PM....... 0000
    GPR0..... 0057476C  GPR1..... 00574788  GPR2..... 0051F7EC  GPR3..... 00578250
    GPR4..... 00578150  GPR5..... 0054F018  GPR6..... 0054F608  GPR7..... 0054F540
    GPR8..... 0054F614  GPR9..... 00578100  GPR10.... 00500754  GPR11.... 00500898
    GPR12.... 00519E48  GPR13.... 00574620  GPR14.... 80541EEE  GPR15.... 83A01D68
    FPR0..... 40404040  40404040            FPR2..... 40404040  40404040
    FPR4..... 40404040  40404040            FPR6..... 40404040  40404040
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (0057476C)
        -0020 0057474C  00000000 005780C8 00578100 00000000  0054F608 0054F614 0054F540 005159FC  |.......H..a.......6...6..5 #.169|
        +0000 0057476C  00000000 005008C8 005008C8 C0000000  00000000 00000000 00000000 00578100  |.....&.H.&.H..................a.|
        +0020 0057478C  00578150 80578250 00000000 00574620  005780C8 00578100 00000000 00574000  |..a&..b&...........H..a....... .|
      STORAGE AROUND GPR 1 (00574788)
        -0020 00574768  005159FC 00000000 005008C8 005008C8  C0000000 00000000 00000000 00000000  |.&-#169;.......&.H.&.H..........|
        +0000 00574788  00578100 00578150 80578250 00000000  00574620 005780C8 00578100 00000000  |..a...a&..b.&..........H..a.....|
        +0020 005747A8  00574000 000002D8 FB000000 FFFFFFFF  0000003C C3C5C5C4 E4D4D740 FFFFFFFF  |.. ....Q...........CEEDUMP ....|
⋮
```

Figure 5. Example Dump Using CEE5DMP (Part 1 of 3)

```
    STORAGE AROUND GPR15 (83A01D68)
      -0020 03A01D48  00000014 03A01318 F2F0F0F1 F0F7F1F0  F1F2F2F6 F0F0F0F1 F0F4F0F2 00000778  |........2001071012260001040 2....|
      +0000 03A01D68  47F0F014 00C3C5C5 00000B48 00004C84  47F0F001 90ECD00C 18BF41A0 BFFF4190  |.00..CEE......&.d.00............|
      +0020 03A01D88  AFFF4180 9FFF4170 8FFF4160 7FFF5800  7CDC5810 D04C1E01 5500C00C 47D0B048  |...........-"...@....<..........|
    STORAGE AROUND GPR15 (83A01D68) (24-BIT)
      -0020 00A01D48  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
      +0000 00A01D68 - +00003F 00A01DA7             SAME AS ABOVE

  TRACEBACK:    6
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY         E ADDR    E  OFFSET   STATEMENT  STATUS
    00574620  HANDLER       00500658  +0000035C  HANDLER       00500658  +0000035C              CALL
    0054F018  CEEHDSP       03A58D38  +000017A0  CEEHDSP       03A58D38  +000017A0              CALL
    00574038  DRVHAND       00500078  +00000350  DRVHAND       00500078  +00000350              EXCEPTION

  CONDITION INFORMATION FOR ACTIVE ROUTINES    7
    CONDITION INFORMATION FOR DRVHAND (DSA ADDRESS 00574038)
      CIB ADDRESS: 0054F528
      CURRENT CONDITION:
        CEE3209S THE SYSTEM DETECTED A FIXED-POINT DIVIDE EXCEPTION.
      LOCATION:
        PROGRAM UNIT: DRVHAND ENTRY: DRVHAND STATEMENT:  OFFSET: +00000350
      MACHINE STATE:
        ILC..... 0004    INTERRUPTION CODE..... 0009
        PSW..... 071D2000 805003CC
        GPR0..... 00574178  GPR1..... 00574198  GPR2..... 00000000  GPR3..... 00000001
        GPR4..... 005000B0  GPR5..... 00572188  GPR6..... 0051F1AC  GPR7..... 00000000
        GPR8..... 0051FA70  GPR9..... 00578070  GPR10.... 00500178  GPR11.... 005002C4
        GPR12.... 0050016C  GPR13.... 00574038  GPR14.... 80500386  GPR15.... 00000000
    STORAGE DUMP NEAR CONDITION, BEGINNING AT LOCATION: 005003B8
      +000000 005003B8  A014D203 9010A010 58209010 8E200020  5D209008 50309018 5820D05C 58F0202C  |..K.............)...&......*.0..|

  PARAMETERS, REGISTERS, AND VARIABLES FOR ACTIVE ROUTINES:    8
    HANDLER (DSA ADDRESS 00574620):
      SAVED REGISTERS:
        GPR0..... 0057476C  GPR1..... 00574788  GPR2..... 0051F7EC  GPR3..... 00578250
        GPR4..... 00578150  GPR5..... 0054F018  GPR6..... 0054F608  GPR7..... 0054F540
        GPR8..... 0054F614  GPR9..... 00578100  GPR10.... 00500754  GPR11.... 00500898
        GPR12.... 00519E48  GPR13.... 00574620  GPR14.... 80541EEE  GPR15.... 83A01D68
:
      LOCAL VARIABLES:
:
  CONTROL BLOCKS FOR ACTIVE ROUTINES:    9
    DSA FOR HANDLER: 00574620
      +000000  FLAGS.... 0010      member... 8001      BKC...... 0054F018  FWC...... 00552018  R14...... 80541EEE
      +000010  R15...... 83A01D68  R0....... 0057476C  R1....... 00574788  R2....... 0051F7EC  R3....... 00578250
      +000024  R4....... 00578150  R5....... 0054F018  R6....... 0054F608  R7....... 0054F540  R8....... 0054F614
      +000038  R9....... 00578100  R10...... 00500754  R11...... 00500898  R12...... 00519E48  reserved. F3E3C7E3
      +00004C  NAB...... 00552018  PNAB..... 03000000  reserved. 41030220  00572188  0051F7EC  00574798
      +000064  reserved. 00000000  reserved. 00000151  MODE..... 805009B6  reserved. 00000000  005780B8
      +000078  reserved. 00000000  reserved. 00000000
:
  STORAGE FOR ACTIVE ROUTINES:    10
    HANDLER:
:
  CONTROL BLOCKS ASSOCIATED WITH THE THREAD:    11
    CAA: 00519E48
      +000000 00519E48  00000800 00000000 00552000 00572000  00000000 00000000 00000000 00000000  |................................|
      +000020 00519E68  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
      +000040 00519E88 - +00005F 00519EA7             SAME AS ABOVE
      +000060 00519EA8  00000000 00000000 00000000 00000000  00000000 80204A90 00000000 00000000  |................................|
      +000080 00519EC8  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
      +0000A0 00519EE8 - +00011F 00519F67             SAME AS ABOVE
      +000120 00519F68  00513EF8 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |.&.#169;.8......................|
      +000140 00519F88  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
:
```

Figure 5. Example Dump Using CEE5DMP (Part 2 of 3)

```
    DUMMY DSA: 0051A608
      +000000  FLAGS.... 0000      member... 0000      BKC...... 00438000  FWC...... 00000000  R14...... 00572488
      +000010  R15...... 00500078  R0....... 00000000  R1....... 001F9160  R2....... 00000000  R3....... 00000000
      +000024  R4....... 00000000  R5....... 00000000  R6....... 00000000  R7....... 00000000  R8....... 00000000
      +000038  R9....... 00000000  R10...... 00000000  R11...... 00000000  R12...... 00519E48  reserved. 00000000
      +00004C  NAB...... 00552018  PNAB..... 00552018  reserved. 00000000  00000000  00000000  00000000
      +000064  reserved. 00000000  reserved. 00000000  MODE..... 00000000  reserved. 00000000  00000000
      +000078  reserved. 00000000  reserved. 00000000

  ENCLAVE CONTROL BLOCKS:    12
    EDB: 00519060
      +000000 00519060  C3C5C5C5 C4C24040 C0000001 00519D08  005195F8 00000000 00000000 00000000  |CEEEDB  .....&.#169....&.#169.n8|
      +000020 00519080  005194B0 005194E0 00541038 00518D40  00000000 00000000 001F9160 00008000  |.&.#169.m..169.m................|
      +000040 005190A0  00000000 00000000 00438000 00000000  00000000 00000000 00519120 005190B8  |........................169.169|
      +000060 005190C0  00000000 00000000 00000000 00000000  00000000 00000000 03A892F8 00519E48  |........................yk8.169|
      +000080 005190E0  70000000 00000000 00000000 00000000  00000001 00000000 00000000 00000000  |................................|
    MEML: 00519D08
      +000000 00519D08  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
      +000020 00519D28 - +00003F 00519D47            SAME AS ABOVE
      +000040 00519D48  00000000 00000000 03A41338 00000000  00000000 00572188 8051A800 00000000  |.........u.............h.169y...|
      +000060 00519D68  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
      +000080 00519D88 - +00011F 00519E27            SAME AS ABOVE
:
  ENCLAVE STORAGE:    13
    Initial (User) Heap
      +000000 00578000  C8C1D5C3 00519480 00519480 00000000  00578000 00578258 00008000 00007DA8  |HANC.&.169m..169m...........b.'y|
      +000020 00578020  00578000 00000090 00000088 00000000  00000000 00000000 00000000 00000000  |...........h....................|
:
    LE Anywhere Heap
      +000000 00574000  C8C1D5C3 005194B0 005194B0 005194B0  00574000 00575530 00004000 00002AD0  |HANC.&.169m..169m..169m.........|
      +000020 00574020  00574000 00000198 00000190 00000000  005741C0 005723F8 00108001 0051A608  |.. ....q...............8....169.|
:
    LE Below Heap
      +000000 00572000  C8C1D5C3 005194E0 005194E0 005194E0  80572000 00572758 00002000 000018A8  |HANC.&169m..169m..169m.........y|
      +000020 00572020  00572000 00000040 C8E7E2E3 001FEFF0  001FE2E8 0051E800 00000000 00000000  |....... HXST...0..SY.169Y.......|
:
PROCESS CONTROL BLOCKS:    14

  PCB: 00518D40
    +000000 00518D40  C3C5C5D7 C3C24040 04030290 00000000  00000000 00000000 00518F38 03A8A6B0  |CEEPCB  .................169...yw.|
    +000020 00518D60  03A86838 03A89690 03A891C8 005089C8  00518118 00000000 00000000 00519060  |.y...yo..yjH.iH..a.........169...-|
    +000040 00518D80  03A894E8 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |.ymY.............................|

  MEML: 00518F38
    +000000 00518F38  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
    +000020 00518F58 - +00003F 00518F77            SAME AS ABOVE
    +000040 00518F78  00000000 00000000 03A41338 00000000  00000000 0051FA70 8051A800 00000000  |.........u...........169...16y..|
    +000060 00518F98  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
    +000080 00518FB8 - +00011F 00519057            SAME AS ABOVE
:
```

Figure 5. Example Dump Using CEE5DMP (Part 3 of 3)

Sections of the LE/VSE Dump

The sections of the dump listed here appear independently of the LE/VSE-conforming languages used. Each conforming language adds language-specific storage and file information to the dump.

For a detailed explanation of language-specific dump output:
- For C routines, see "Finding C Information in an LE/VSE Dump" on page 86.
- For COBOL routines, see "Finding COBOL Information in a Dump" on page 112.
- For PL/I routines, see "Finding PL/I Information in a Dump" on page 141.

1 Page Heading

The page heading section appears on the top of each page of the dump and contains:

- CEE5DMP identifier
- Title

  Note: For dumps generated as a result of an unhandled condition, the title is `CONDITION PROCESSING RESULTED IN THE UNHANDLED CONDITION.`
- Product abbreviation of LE/VSE
- Version number
- Release number
- Date
- Time
- Page number

2 Caller Program Unit and Offset

This information identifies the routine name and offset in the calling routine of the call to the dump service.

3 Registers on Entry to CEE5DMP

This section of the dump shows data at the time of the call to the dump service.

- Program mask

  The *program mask* contains the bits for the fixed-point overflow mask, decimal overflow mask, exponent underflow mask, and significance mask.
- General purpose registers (GPRs) 0–15

  On entry to CEE5DMP, the GPRs contain:

  GPR 0 Working register
  GPR 1 Pointer to the argument list
  GPR 2–11
  Working registers
  GPR 12
  Address of CAA
  GPR 13
  Pointer to caller’s stack frame
  GPR 14
  Address of next instruction to run if ALL31(ON) is specified
  GPR 15
  Entry point of CEE5DMP
- Floating point registers (FPRs) 0, 2, 4, 6

4 Enclave Identifier

This statement names the enclave for which information in the dump is provided. If multiple enclaves exist, the dump service generates data and storage information for the most current enclave, followed by previous enclaves in a last-in-first-out (LIFO) order. For more information about dumps for multiple enclaves, see “Multiple Enclave Dumps” on page 52.

5 Information for thread

This section shows the system identifier for the thread. (Note that LE/VSE Version 1 Release 4 supports only a single thread in an enclave.)

6 Traceback

For all active routines, the traceback section shows:

- Stack frame address
- Program unit

  The primary entry point of the external procedure. For COBOL routines, this is the routine unit name. For C and PL/I routines, this is the compile unit name. For LE/VSE-conforming assemblers, this is the *EPNAME* = value on the CEEPPA macro.
- Program unit address

  The starting address of the program unit.
- Program unit offset

  The offset of the last instruction to run in the routine. If the offset is a negative number, zero, or a very large positive number, the routine associated with the offset probably did not allocate a *save area*. Adding the program unit address to the offset gives you the location of the current instruction in the routine. This offset is from the starting address of the routine.
- Entry

  For COBOL and PL/I routines, this is the entry point name. For C routines, this is the function name.
- Entry point address
- Entry point offset
- Statement number

  The statement number of the last statement to run in the routine. The statement number appears only if your routine was compiled with the options required to generate statement numbers.
- Status

  Routine status can be call, exception, or running.

7 Condition Information for Active Routines

This section displays the following information for all conditions currently active on the call chain:

- Statement showing failing routine and stack frame address of routine
- Condition information block (CIB) address
- Current condition, in the form of an LE/VSE message for the condition raised or an LE/VSE abend code, if the condition was caused by an abend.
- Location. For the failing routine, this is the program unit, entry routine, statement number, and offset.
- Machine state, which shows:

  Instruction length counter
  Interruption code
  Program status word (PSW)
  Contents of GPRs 0–15

Notes:

1. These values are the current values at the time the condition was raised.
2. This area of the dump also displays the storage area near to the point where the condition occurred.

8 Parameters, Registers, and Variables for Active Routines

For each active routine, this section shows:

- Routine name and stack frame address
- Parameters

  For C, parameters are shown here rather than with the local variables. For COBOL, parameters are shown as part of local variables For PL/I, parameters are not displayed in the LE/VSE dump.
- Saved registers

  This lists the contents of GPRs 0–15 at the time the routine transferred control.
- Local variables

  This section displays the local variables and arguments for the routine. This section also shows the variable type. Variables are displayed only if the symbol tables are available. To generate a symbol table and display variables, use the following compile-time options:
  – For C, use TEST(ALL,SYM).
  – For COBOL, use TEST(ALL,SYM).
  – For PL/I, arguments and variables are not displayed.

9 Control Blocks for Active Routines

For each active routine controlled by the STACKFRAME option, this section lists contents of related control blocks. The LE/VSE-conforming language determines which language-specific control blocks appear. The possible control blocks are:

- Stack frame
- Condition information block
- Language-specific control blocks

10 Storage for Active Routines

This displays local storage for each active routine. The storage is dumped in hexadecimal, with EBCDIC translations on the right side of the page. There can be other information, depending on the language used. For C routines, this is the stack frame storage. For COBOL routines, this is language-specific information and WORKING-STORAGE.

11 Control Blocks Associated with the Thread

This section lists the contents of the LE/VSE *common anchor area (CAA)* and dummy stack frame.

Note: Other language-specific control blocks can appear in this section.

12 Enclave Control Blocks

This section lists the contents of the LE/VSE enclave data block (EDB), enclave member list (MEML), and the language control blocks for the enclave level.

Note: Other language-specific control blocks can appear in this section.

13 Enclave Storage

This section shows the LE/VSE heap storage. For C and PL/I routines, heap storage is the dynamically allocated storage. For COBOL routines, it is the *run unit* storage.

14 Process Control Blocks

This section lists the contents for the LE/VSE process control block (PCB), process member list (MEML), and the language control blocks for the process level.

Debugging with Tracebacks, Condition Information, and Data Values

The CEE5DMP call with dump options TRACEBACK, CONDITION, and VARIABLES generates run-time output that contains a traceback, information about any conditions, and a list of arguments, registers, and variables. This output contains the information needed to debug most basic routine errors.

The traceback, condition, and variable information provided in the LE/VSE dump can help you determine the location and context of the error without any additional information. The traceback section includes a sequential list for all active routines and the routine name, statement number, and offset where the exception occurred. The condition information section displays a message describing the condition and the address of the condition information block. The arguments, variables, and registers section shows the values of your arrays, structures, arguments, and data during the sequence of calls in your application. Thus, you can check the values passed to each routine. Static data values do not appear. Single quotes indicate character fields.

These sections of the dump are shown in Figure 5 on page 36.

Debugging with the Stack Frame

You might find it helpful to check the contents of the stack frame. The stack frame for each active routine is listed in the full dump. Stack frames are also called *dynamic storage areas (DSA)*.

A stack frame chain is associated with each thread in the run-time environment. A stack frame is acquired every time a separately compiled procedure or block is entered. A stack frame is also allocated for each call to an LE/VSE service. All stack frames are back-chained with a stopping stack frame (also called a dummy DSA) as the first stack frame on the stack. Register 13 addresses the recently active stack frame or a standard *register save area (RSA)*. The standard save area back chain must be initialized, and it holds the address of the previous save area. Not all LE/VSE-conforming compilers set the forward chain; thus, it cannot be guaranteed in all instances. Calling routines establish the member-defined fields.

When a routine makes a call, registers 0–15 contain the following values:
- R0 is used by COBOL *static call*.
- R1 is a pointer to parameter list or 0 if no parameter list passed.
- R2–R11 is unreferenced by LE/VSE. Caller’s values are passed transparently.
- R12 is the pointer to the CAA if entry to an external routine.
- R13 is the pointer to caller’s stack frame.
- R14 is the return address.
- R15 is the address of the called entry point.

With an optimization level other than 0, C routines save only the registers used during the running of the current routine. Non-LE/VSE RSAs can be in the save area chain. The length of the save area and the saved register contents do not always conform to LE/VSE conventions. For a detailed description of LE/VSE storage management, see *LE/VSE Programming Guide*. Figure 6 shows the format of the stack frame.

hex	(dec)		
00	(00)	Flags	Member-defined
04	(04)	CEEDSABWC - Standard Save Area Back Chain	
08	(08)	CEEDSAFWC - Standard Save Area Forward Chain	
0C	(12)	≈ CEEDSARn - GPRs (n=14, 15, 0-12) ≈	
48	(72)	Member-defined	
4C	(76)	CEEDSANAB - Current *Next Available Byte (NAB)* in Stack	
50	(80)	CEEDSAPNAB - End of Prolog NAB	
54	(84)	Member-defined	
58	(88)	Member-defined	
5C	(92)	Member-defined	
60	(96)	Member-defined	
64	(100)	CEEDSARENT - Program reentry address - IPAT	
68	(104)	Member-defined	
6C	(108)	CEEDSAMODE - Return Address of the Module That Caused the Last Mode Switch	
70	(112)	Member-defined	
74	(116)	Member-defined	
78	(120)	CEEDSARMR - Address of language-specific exception handler	
7C	(124)	Reserved for Future Use	

Figure 6. Stack Frame

Debugging with the Common Anchor Area

Each thread is represented by a common anchor area (CAA), which is the central communication area for LE/VSE. All thread- and enclave-related resources are anchored, provided for, or can be obtained through the CAA. The CAA is generated during thread initialization and deleted during thread termination. When calling LE/VSE-conforming routines, register 12 points to the address of the CAA.

Use CAA fields as described. Do not modify fields and do not use routine addresses as entry points, except as specified. Fields marked 'Reserved' exist for migration of specific languages, or internal use by LE/VSE. LE/VSE defines their location in the CAA, but not their use. Do not use or reference them except as specified by the language that defines them.

Figure 7 on page 44 and Figure 8 on page 45 show the format of the LE/VSE CAA.

hex	(dec)				
-00018	(-24)	CEECAAEYE CL8'CEECAA			
-0000C	(-12)	Reserved			
000000	(00)	CEECAAFLAG0	Reserved	CEECAALANGP	Reserved
000004	(04)	Reserved			
000008	(08)	CEECAABOS - Start of Current Storage Segment			
00000C	(12)	CEECAAEOS - End of Current Storage Segment			
000010	(16)	≈ Reserved - 10 thru 11F ≈			
000120	(288)	CEECAAATTN - Reserved			
000124	(292)	≈ Reserved - 124 through 1A7 ≈			
0001A8	(424)	≈ CEECAAHOOKS - Execute Hooks - 18 4-Byte Hooks ≈			
0001F0	(496)	≈ Reserved - 1F0 through 2AB ≈			
0002AC	(684)	CEECAASYSTM	CEECAAHRDWR	CEECAASBSYS	CEECAAFLAG2
0002B0	(688)	CEECAALEVEL CAA Level ID	Reserved		
0002B4	(692)	CEECAAGETLS - Addr OF CEL Library Stack Mgr			
0002B8	(696)	CEECAACELV - Addr of CEL LIBVEC			
0002BC	(700)	CEECAAGETS - Addr of CEL Get Stack Stg Rtn			
0002C0	(704)	CEECAALBOS - Start of Library Stack Stg Seg			
0002C4	(708)	CEECAALEOS - End of Library Stack Stg Seg			
0002C8	(712)	CEECAALNAB - Next Available Byte of Lib Stg			
0002CC	(716)	CEECAADMC - Addr of Shunt Routine			
0002D0	(720)	CEECAAABCODE - Reserved			
0002D4	(724)	CEECAARSNCODE - Reserved			
0002D8	(728)	CEECAAERR - Addr of the Current Condition Information Block			
0002DC	(732)	CEECAAGETSX - Addr of CEL Stack Stg Extender			
0002E0	(736)	CEECAADDSA - Addr of the Dummy DSA			
0002E4	(740)	CEECAASECTSIZ - Reserved			
0002E8	(744)	CEECAAPARTSUM - Reserved			
0002EC	(748)	CEECAASSEXPNT - Reserved			
0002F0	(752)	CEECAAEDB - Addr of the EDB			
0002F4	(756)	CEECAAPCB - Addr of the PCB			
0002F8	(760)	CEECAAEYEPTR - Addr of CAA Eyecatcher			
0002FC	(764)	CEECAAPTR - Addr of this CAA			

Figure 7. Common Anchor Area (Part 1)

hex	(dec)		
000300	(768)	CEECAAGETS1 - Stack Overflow for Non-DSA Save Area	
000304	(772)	CEECAASHAB - Addr of abend shunt routine	
000308	(776)	CEECAAPRGCK - Program Interrupt Code for CEECAADMC	
00030C	(780)	CEECAAFLAG1	Reserved
000310	(784)	CEECAAURC - Thread Level Return Code	
000314	(788)	CEECAARSRV1 - Reserved	
≈			≈
000324	(804)	CEECAAPICICB - Reserved	
000328	(808)	CEECAARSRV2 - Reserved	
00032C	(812)	CEECAAGOSMR - Reserved	Reserved

Figure 8. Common Anchor Area (Part 2)

Table 23 lists the fields of the CAA and explanations.

Table 23. Fields of CAA

CAA Field	Explanation
CEECAAFLAG0	CAA flag bits. The bits are defined as follows: **Bit** **Description** **0 - 5** Reserved. **6** CEECAAXHDL. A flag used by the condition handler. If the flag is set to 1, the application requires immediate return/percolation to the system on any interrupt or condition handler event. **7** Reserved.
CEECAALANGP	PL/I language compatibility flags external to LE/VSE. The bits are defined as follows: **Bit** **Description** **0 - 3** Reserved. **4** CEECAATHFN. A flag set by PL/I to indicate a PL/I FINISH ON-unit is active. If the flag is set to 1, no PL/I FINISH ON-unit is active. If the flag is set to 0, a PL/I FINISH ON-unit could be active. **5 - 7** Reserved.
CEECAABOS	Start of the current storage segment. This field is initially set during thread initialization. It indicates the start of the current stack storage segment. It is altered when the current stack storage segment is changed.
CEECAAEOS	End of the current storage segment. This field is initially set during thread initialization. It indicates the end of the current stack storage segment. It is altered when the current stack storage segment is changed.
CEECAAATTN	Reserved.

Table 23. Fields of CAA (continued)

CAA Field	Explanation
CEECAAHOOKS	Hook area. This is the start of 18 fullword execute hooks. LE/VSE initializes each fullword to X'07000000'. The hooks can be altered to support various debugging hook mechanisms.
CEECAASYSTM	The value indicates the operating system supporting the active environment. **Value** **Operating System** **0** Undefined. This value should not appear after LE/VSE is initialized. **1** Unsupported. **2** Unsupported. **3** Unsupported. **4** VSE
CEECAAHRDWR	This value indicates the type of hardware on which the routine is running. **Value** **Hardware** **0** Undefined. This value should not appear after LE/VSE is initialized. **1** Unsupported. **2** System/370, non-XA. **3** System/370, XA. **4** System/370, ESA.
CEECAASBSYS	This value indicates the subsystem (if any) on which the routine is running. **Value** **Subsystem** **0** Undefined. This value should not occur after LE/VSE is initialized. **1** Unsupported. **2** The routine is not running under an LE/VSE-recognized subsystem. **3** Unsupported. **4** Unsupported. **5** CICS.
CEECAAFLAG2	**Bit** **Description** **0** Bimodal addressing is available.
CEECAALEVEL	LE/VSE level identifier. This contains a unique value that identifies each release of LE/VSE. This number is incremented for each new release of LE/VSE. Its value is 2 for LE/VSE Version 1 Release 4.
CEECAAGETLS	Address of stack overflow for library routines.
CEECAACELV	Address of the LE/VSE library vector. This field is used to locate dynamically loaded LE/VSE routines.

Table 23. Fields of CAA (continued)

CAA Field	Explanation
CEECAAGETS	Address of the LE/VSE prolog stack overflow routine. The address of the LE/VSE get stack storage routine is included in prolog code for fast reference.
CEECAALBOS	Start of the library stack storage segment. This field is initially set during thread initialization. It indicates the start of the library stack storage segment. It is altered when the library stack storage segment is changed.
CEECAALEOS	End of the library stack storage segment. This field is initially set during thread initialization. It indicates the end of the library stack storage segment. It is altered when the library stack storage segment is changed.
CEECAALNAB	Next available library stack storage byte. This contains the address of the next available byte of storage on the library stack. It is modified when library stack storage is obtained or released.
CEECAADMC	LE/VSE shunt routine address. Its value is initially set to 0 during thread initialization. If it is nonzero, this is the address of a routine used in specialized exception processing.
CEECAAABCODE	Reserved
CEECAARSNCODE	Reserved
CEECAAERR	Address of the current condition information block. After completion of initialization, this always points to a condition information block. During exception processing, the current condition information block contains information about the current exception being processed. Otherwise, it indicates no exception being processed.
CEECAAGETSX	Address of the user stack extender routine. This routine is called to extend the current stack frame in the user stack. Its address is in the CEECAA for performance reasons.
CEECAADDSA	Address of the LE/VSE dummy DSA. This address determines whether a stack frame is the dummy DSA, also known as the zeroth DSA.
CEECAASECTSIZ	Reserved.
CEECAAPARTSUM	Reserved.
CEECAASSEXPNT	Reserved.
CEECAAEDB	Address of the LE/VSE EDB. This field points to the encompassing EDB.
CEECAAPCB	Address of the LE/VSE PCB. This field points to the encompassing PCB.
CEECAAEYEPTR	Address of the CAA eye catcher. The CAA eye catcher is CEECAA. This field can be used for validation of the CAA.
CEECAAPTR	Address of the CAA. This field points to the CAA itself and can be used in validation of the CAA.
CEECAAGETS1	Non-DSA stack overflow. This field is the address of a stack overflow routine, which cannot guarantee that the current register 13 is pointing at a stack frame. Register 13 must point, at a minimum, to a save area.

Table 23. Fields of CAA (continued)

CAA Field	Explanation
CEECAASHAB	LE/VSE abend shunt routine address. Its value is initially set to 0 during thread initialization. If it is nonzero, this is the address of a routine used in specialized abend processing.
CEECAAPRGCK	Routine interrupt code for CEECAADMC. If CEECAADMC is nonzero, and a routine interrupt occurs, this field is set to the routine interrupt code and control is passed to the address in CEECAADMC.
CEECAAFLAG1	CAA flag bits. The bits are defined as follows: **Bit** **Description** / **0** CEECAASORT. A call to the sort program is active. / **1 - 7** Reserved.
CEECAAURC	Thread level return code. This is the common place for members to set the return codes for subroutine-to-subroutine return code processing.
CEECAARSRV1	Reserved.
CEECAAPICICB	Reserved.
CEECAARSRV2	Reserved.
CEECAAGOSMR	Reserved.

Debugging with the Condition Information Block

The LE/VSE condition manager creates a *condition information block (CIB)* for each condition encountered in the LE/VSE environment. The CIB holds data required by the condition handling facilities and pointers to locations of other data. The address of the current CIB is located in the CAA.

LE/VSE provides members that map the CIB to language-specific structures:
- `_ceecib` structure in leawi.h (for C)
- CEEIGZCI.C (for COBOL)
- CEEIBMCI.P (for PL/I)
- CEECIB.A (for Assembler)

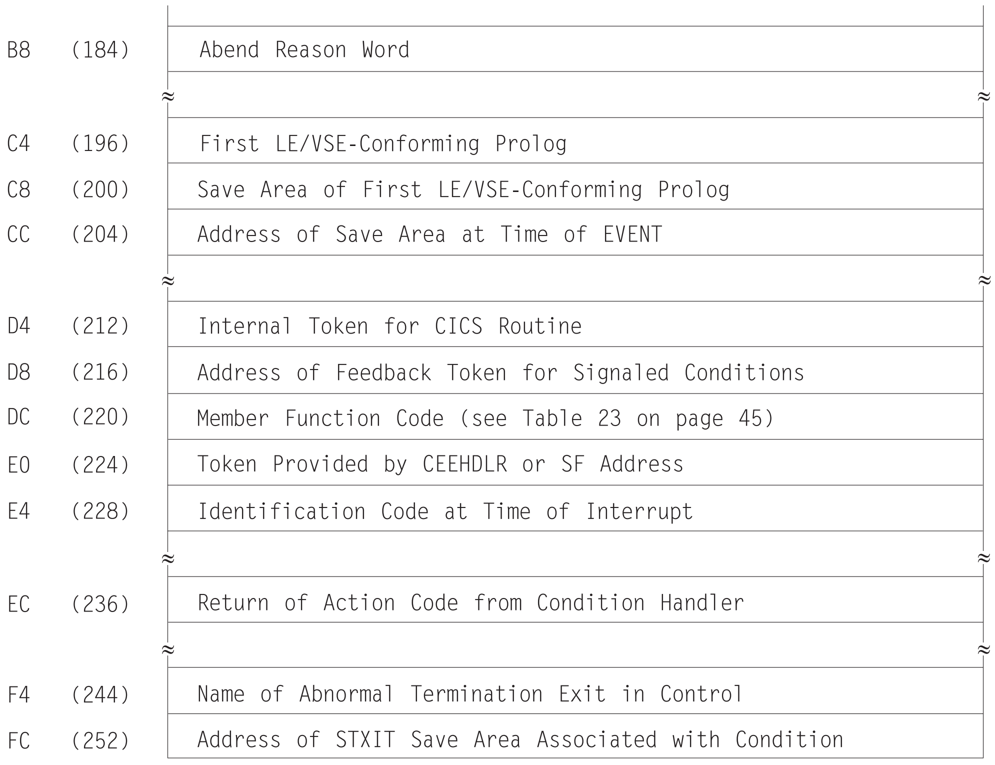
If LE/VSE has been installed in the default sublibrary, you can find these members in the PRD2.SCEEBASE sublibrary.

Figure 9 on page 49 and Figure 10 on page 50 show the condition information block.

hex	(dec)		
00	(00)	Condition Information Block Eye Catcher	
04	(04)	Previous Condition Information Block	
08	(08)	Most Recent Condition Information Block	
0C	(12)	Size of CIB	Version of CIB
10	(16)	Platform Identifier - 5 = LE/VSE	
		≈	
18	(24)	Current LE/VSE Condition	
24	(36)	Address of Machine State at Time of Interrupt	
28	(40)	Previous LE/VSE Condition	
		≈	
37	(55)	Condition Flags	
38	(56)	Handle Cursor	
		≈	
44	(68)	Resume Cursor	
		≈	
B0	(176)	Status Flag 5	
B1	(177)	Status Flag 6	
B2	(178)	Status Flag 7	
B4	(180)	Abend Code Word	

Figure 9. Condition Information Block (Part 1 of 2)

Note: For an explanation of Status Flag 5, see Table 25 on page 50. For an explanation of Status Flag 6, see Table 26 on page 51. For an explanation of Status Flag 7, see Table 27 on page 51.



```
B8   (184)   Abend Reason Word
             ≈                                                     ≈
C4   (196)   First LE/VSE-Conforming Prolog
C8   (200)   Save Area of First LE/VSE-Conforming Prolog
CC   (204)   Address of Save Area at Time of EVENT
             ≈                                                     ≈
D4   (212)   Internal Token for CICS Routine
D8   (216)   Address of Feedback Token for Signaled Conditions
DC   (220)   Member Function Code (see Table 23 on page 45)
E0   (224)   Token Provided by CEEHDLR or SF Address
E4   (228)   Identification Code at Time of Interrupt
             ≈                                                     ≈
EC   (236)   Return of Action Code from Condition Handler
             ≈                                                     ≈
F4   (244)   Name of Abnormal Termination Exit in Control
FC   (252)   Address of STXIT Save Area Associated with Condition
```

Figure 10. Condition Information Block (Part 2 of 2)

Note: For an explanation of Member Function Code, see Table 28 on page 51.

Table 24 shows the flags for Condition Flag 4.

Table 24. Condition Flag 4

Value	Description
2	The resume cursor has been moved.
4	Message service has processed the condition.
8	The resume cursor has been moved explicitly.

Table 25 shows the flags for Status Flag 5, LE/VSE events.

Table 25. Status Flag 5, LE/VSE Events

Value	Description
2	Caused by a signaled condition
4	Caused by a promoted condition
8	Caused by a condition management raised TIU
64	Caused by a program check
128	Caused by an abend

Table 26 shows the flags for Status Flag 6, LE/VSE actions.

Table 26. Status Flag 6, LE/VSE Actions

Value	Description
2	Doing stack frame zero scan
4	H-cursor pointing to owning SF
8	Enable only pass (no condition pass)
16	MRC type 1
32	Resume allowed
64	Math service condition
128	Abend reason code valid

Table 27 shows the flags for Status Flag 7, LE/VSE named cond itions

Table 27. Status Flag 7, LE/VSE Named Conditions

Value	Description
64	STXIT save area associated with condition
128	Storage condition

Table 28 shows the language-specific function codes for the CIB.

Table 28. Language-Specific Function Codes

Value	Description
X'1'	For condition procedure
X'2'	For enablement
X'3'	For stack frame zero conditions

Using the Machine State Information Block

The LE/VSE machine state information block contains condition informat ion pertaining to the hardware state at the time of the error. The address of the machine state information block is in the CIB at offset 36 (X'00000024'). Figure 11 shows the machine state information block.

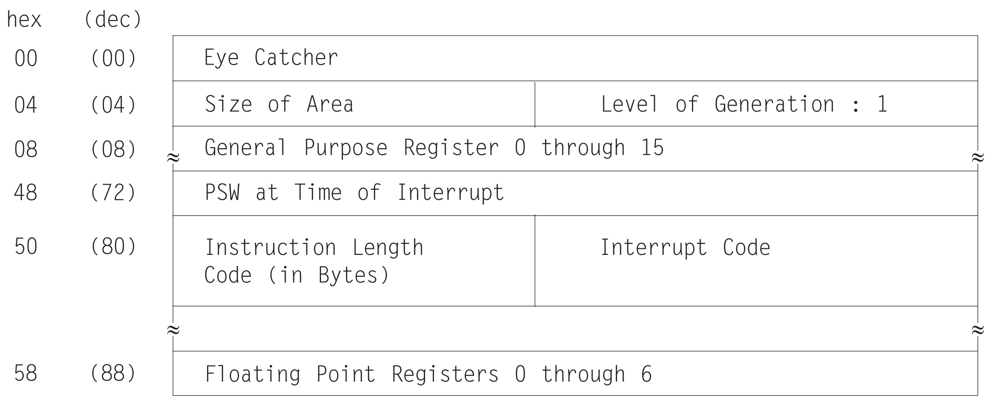


Figure 11. Machine State Information Block

Using the STXIT Save Area

If Status Flag 7 indicates a STXIT save area is associated with the condition, offset 252 (X'000000FC') in the CIB contains the address of an extended-format STXIT save area. For a description of the save area, see *z/VSE System Macros Reference*

Multiple Enclave Dumps

If multiple enclaves are used, the dump service generates data and storage information from the most current enclave and moves up the chain of enclaves to the starting enclave. For example, if two enclaves are used, the dump service first generates output for the most current enclave. Then the service creates output for the previous enclave.

Figure 12 on page 53 illustrates the information available in the LE/VSE dump and the order of information for multiple enclaves.

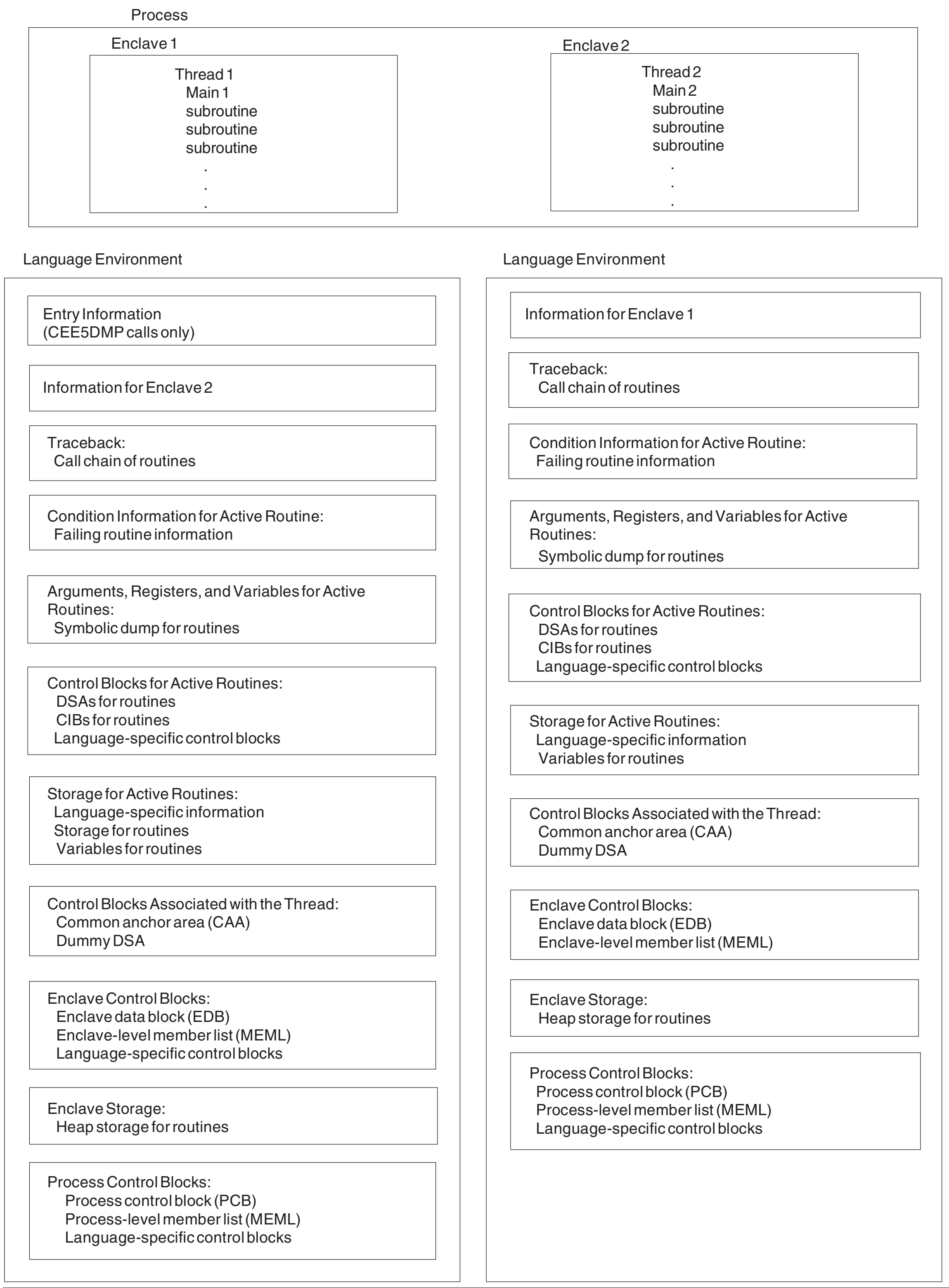

Process
Enclave 1
Thread 1
Main 1
subroutine
subroutine
subroutine
Enclave 2
Thread 2
Main 2
subroutine
subroutine
subroutine
Language Environment
Entry Information (CEE5DMP calls only)
Information for Enclave 2
Traceback: Call chain of routines
Condition Information for Active Routine: Failing routine information
Arguments, Registers, and Variables for Active Routines: Symbolic dump for routines
Control Blocks for Active Routines: DSAs for routines, CIBs for routines, Language-specific control blocks
Storage for Active Routines: Language-specific information, Storage for routines, Variables for routines
Control Blocks Associated with the Thread: Common anchor area (CAA), Dummy DSA
Enclave Control Blocks: Enclave data block (EDB), Enclave-level member list (MEML), Language-specific control blocks
Enclave Storage: Heap storage for routines
Process Control Blocks: Process control block (PCB), Process-level member list (MEML), Language-specific control blocks
Language Environment
Information for Enclave 1
Traceback: Call chain of routines
Condition Information for Active Routine: Failing routine information
Arguments, Registers, and Variables for Active Routines: Symbolic dump for routines
Control Blocks for Active Routines: DSAs for routines, CIBs for routines, Language-specific control blocks
Storage for Active Routines: Language-specific information, Variables for routines
Control Blocks Associated with the Thread: Common anchor area (CAA), Dummy DSA
Enclave Control Blocks: Enclave data block (EDB), Enclave-level member list (MEML)
Enclave Storage: Heap storage for routines
Process Control Blocks: Process control block (PCB), Process-level member list (MEML), Language-specific control blocks

Figure 12. LE/VSE Dump of Multiple Enclaves

Generating a System Dump

A system dump contains storage information for the current system task or job step and serves as a map to the active routines. Refer to system or subsystem documentation for detailed system dump information.

When a program abends, a system dump is written (either to the dump sublibrary or SYSLST, depending on whether the VSE SYSDUMP or NOSYSDUMP option is specified) if the VSE DUMP or PARTDUMP option is specified, and if one of the following is true:

- One of the following LE/VSE options is used:

  TRAP(OFF)
  : If you set the run-time option TRAP to OFF, LE/VSE does not handle program exceptions and abends. Instead, the operating system usually handles exceptions or abends and generates a system dump.

    If you rerun your application with TRAP set to OFF and a new condition appears, the new condition might be one that LE/VSE condition handling normally corrects when TRAP is set to ON.

    The recommended methods of generating a system dump are:

    1. TRAP(ON,MAX),TERMTHDACT(UADUMP). This will generate a CEE5DMP formatted dump as well as a system dump.
    2. TRAP(ON,MIN). Will only generate a system dump. Conditions that are to be handled normally will not terminate the environment or produce a system dump.

  ABPERC(abcode)
  : The ABPERC run-time option enables you to specify one VSE cancel code, program-interruption code, or user abend code to be exempted from LE/VSE condition handling. When a program abends with the specified VSE cancel code, program-interruption code, or user abend code, LE/VSE uses operating system services to produce a system dump before terminating.

    ABPERC is ignored under CICS.
- The assembler user exit has set the CEEAUE_ABND and CEEAUE_DUMP flags to ON. See *LE/VSE Programming Guide* for more information about assembler user exits.
- The abend is an LE/VSE-generated abend. When LE/VSE generates an abend, a system dump is produced regardless of the TRAP option that is specified.
- The VSE cancel code, program-interruption code, or user abend code is specified in the assembler user exit. When a program abends with a VSE cancel code, program-interruption code, or user abend code listed in the initialization assembler user exit (CEEBXITA), the LE/VSE condition handler uses operating system services to produce a system dump before terminating.
- The CEE5ABD callable service is used to cause an abend to be handled by the operating system.
- An unhandled condition of severity 2 or greater occurs, causing LE/VSE to invoke a batch abnormal termination exit that generates a system dump.

  Note: The default batch abnormal termination exit, CEEBDATX, does not generate a system dump; it is a null module that immediately returns to the caller when invoked. For information on customizing the batch abnormal exit to generate a system dump, see *LE/VSE Customization Guide*.

Requesting an LE/VSE Trace for Debugging

LE/VSE provides an in-storage, wrapping trace facility that can reconstruct the events leading to the point where a dump is taken. The events recorded by the trace facility are entry and exit library calls. LE/VSE produces a trace table in its dump report under any of the following conditions:

- The CEE5DMP callable service parameter BLOCKS is specified.
- The TRACE run-time option is set to NODUMP and the TERMTHDACT run-time option is set to DUMP.
- The TRACE run-time option is set to DUMP (the default).

For more information about the CEE5DMP callable service, the TERMTHDACT run-time option, or the TRACE run-time option, see *LE/VSE Programming Reference*.

The TRACE run-time option activates LE/VSE run-time library tracing and controls the size of the trace buffer, the type of trace events to record, and it determines whether a dump containing only the trace table should be unconditionally taken when the application (enclave) terminates.

The contents of the dump depend on the values set in the TERMTHDACT run-time option. Under abnormal termination, the following dump contents are generated:

- TERMTHDACT(QUIET)—generates a dump containing the trace table only
- TERMTHDACT(MSG)—generates a dump containing the trace table only
- TERMTHDACT(TRACE)—generates a dump containing the trace table and the traceback
- TERMTHDACT(DUMP)—generates a dump containing thread/enclave/process storage and control blocks (the trace table is included as an enclave control block)
- TERMTHDACT(UADUMP)—generates a CEE5DMP, which is the same as TERMTHDACT(DUMP), as well as producing a complete VSE system dump.

Under normal termination, independent of the TERMTHDACT setting, LE/VSE generates a dump containing the trace table only.

Locating the Trace Dump

If your application calls CEE5DMP, the LE/VSE dump is written to the file specified in the FNAME parameter of CEE5DMP (the default is CEEDUMP).

Understanding the Trace Table Entry (TTE)

The LE/VSE trace table is established unconditionally at enclave initialization time if the TRACE sub-option LE= is set to 0.

Each trace table entry is a fixed-length record consisting of a fixed-format portion (containing such items as the timestamp, thread ID, and member ID) and a member-specific portion. The member-specific portion has a fixed length, of which some (or all) can be unused. For information about how participating products use the trace table entry, refer to the product-specific documentation. Figure 13 on page 56 shows the trace table entry format.

Time of Day	Thread ID	Member ID and Flags	Member Entry Type	Mbr-specific info up to a maximum of 104 bytes
Char(8)	Char(8)	Char(4)	Char(4)	Char(104)

Figure 13. Trace Table Entry Format

Each field is defined below:

Time The 64-bit value obtained from a store clock instruction (STCK).

Thread ID
The 8-byte thread ID of the thread that is adding the trace table entry.

Member ID and Flags
Contains 2 fields:

Field	**Meaning**
Member ID	The 1-byte member ID of the member making the trace table entry.
Flags	24 flags reserved for internal use.

Member Entry Type
A number that indicates the type of the member-specific trace information that follows the field.

Note: To uniquely identify the information contained in a specific TTE, you must consider Member ID as well as Member Entry Type.

Member Specific Information
Based on the member ID and the member entry type, this field contains the specific information for the entry, up to 104 bytes.

Sample Dump for the Trace Table Entry

The following is an example of a dump of the trace table when you specify the LE=1 suboption (the library call/return trace):

```
 ⋮
LANGUAGE ENVIRONMENT TRACE TABLE:

  MOST RECENT TRACE ENTRY IS AT DISPLACEMENT: 000480

      DISPLACEMENT                        TRACE ENTRY IN HEXADECIMAL                                      TRACE ENTRY IN EBCDIC
      ------------ ----------------------------------------------------------------------    --------------------------------
        +000000    TIME...... ACAAC0AB0FC0C801    THREAD ID......... 8000000000000000
        +000010    MEMBER ID.... 03    FLAGS..... 000000    ENTRY TYPE..... 00000001
```

Figure 14. Trace Table in Dump Output (Part 1 of 2)

```
+000018    F001183F 58F0C31C 184E05EF 00000000  47F0303A 90E7D00C 58E0D04C 4100E098  |0....0C..+.......0...X.....<...q|
+000038    60606E4D F0F7F95D 40869697 85954D5D  40404040 40404040 40404040 40404040  |-->(079) fopen()                |
+000058    40404040 40404040 40404040 40404040  40404040 40404040 40000000 00000000  |                          ......|
+000078    00000000 00000000                                                         |........                        |

+000080    TIME...... ACAAC0AB0FDD5601    THREAD ID......... 8000000000000000
+000090    MEMBER ID.... 03    FLAGS..... 000000    ENTRY TYPE..... 00000002
+000098    4C60604D F0F7F95D 40D9F1F5 7EF0F1C3  C2F7F6F6 C340C5D9 D9D5D67E F0F0F0F0  |<--(079) R15=01CB766C ERRNO=0000|
+0000B8    F0F0F0F0 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |0000............................|
+0000D8    00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
+0000F8    00000000 00000000                                                         |........                        |

+000100    TIME...... ACAAC0AB0FDE4301    THREAD ID......... 8000000000000000
+000110    MEMBER ID.... 03    FLAGS..... 000000    ENTRY TYPE..... 00000001
+000118    F001183F 58F0C31C 184E05EF 00000000  47F0303A 90E7D00C 58E0D04C 4100E098  |0....0C..+.......0...X.....<...q|
+000138    60606E4D F0F7F95D 40869697 85954D5D  40404040 40404040 40404040 40404040  |-->(079) fopen()                |
+000158    40404040 40404040 40404040 40404040  40404040 40404040 40000000 00000000  |                          ......|
+000178    00000000 00000000                                                         |........                        |

+000180    TIME...... ACAAC0AB0FF1EF01    THREAD ID......... 8000000000000000
+000190    MEMBER ID.... 03    FLAGS..... 000000    ENTRY TYPE..... 00000002
+000198    4C60604D F0F7F95D 40D9F1F5 7EF0F1C3  C2F7F9C2 C340C5D9 D9D5D67E F0F0F0F0  |<--(079) R15=01CB79BC ERRNO=0000|
+0001B8    F0F0F0F0 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |0000............................|
+0001D8    00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
+0001F8    00000000 00000000                                                         |........                        |

+000200    TIME...... ACAAC0AB0FF28A01    THREAD ID......... 8000000000000000
+000210    MEMBER ID.... 03    FLAGS..... 000000    ENTRY TYPE..... 00000001
+000218    F001183F 58F0C31C 184E05EF 00000000  47F0303A 90E7D00C 58E0D04C 4100E098  |0....0C..+.......0...X.....<...q|
+000238    60606E4D F0F8F35D 40869799 8995A386  4D5D4040 40404040 40404040 40404040  |-->(083) fprintf()              |
+000258    40404040 40404040 40404040 40404040  40404040 40404040 40000000 00000000  |                          ......|
+000278    00000000 00000000                                                         |........                        |

+000280    TIME...... ACAAC0AB0FFF0001    THREAD ID......... 8000000000000000
+000290    MEMBER ID.... 03    FLAGS..... 000000    ENTRY TYPE..... 00000002
+000298    4C60604D F0F8F35D 40D9F1F5 7EF0F0F0  F0F0F0F2 F540C5D9 D9D5D67E F0F0F0F0  |<--(083) R15=00000025 ERRNO=0000|
+0002B8    F0F0F0F0 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |0000............................|
+0002D8    00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
+0002F8    00000000 00000000                                                         |........                        |
 +000300    TIME...... ACAAC0AB0FFF9101    THREAD ID......... 8000000000000000
+000310    MEMBER ID.... 03    FLAGS..... 000000    ENTRY TYPE..... 00000001
+000318    F001183F 58F0C31C 184E05EF 00000000  47F0303A 90E7D00C 58E0D04C 4100E098  |0....0C..+.......0...X.....<...q|
+000338    60606E4D F0F8F35D 40869799 8995A386  4D5D4040 40404040 40404040 40404040  |-->(083) fprintf()              |
+000358    40404040 40404040 40404040 40404040  40404040 40404040 40000000 00000000  |                          ......|
+000378    00000000 00000000                                                         |........                        |

+000380    TIME...... ACAAC0AB10027701    THREAD ID......... 8000000000000000
+000390    MEMBER ID.... 03    FLAGS..... 000000    ENTRY TYPE..... 00000002
+000398    4C60604D F0F8F35D 40D9F1F5 7EF0F0F0  F0F0F0F2 F640C5D9 D9D5D67E F0F0F0F0  |<--(083) R15=00000026 ERRNO=0000|
+0003B8    F0F0F0F0 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |0000............................|
+0003D8    00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
+0003F8    00000000 00000000                                                         |........                        |

+000400    TIME...... ACAAC0AB10030101    THREAD ID......... 8000000000000000
+000410    MEMBER ID.... 03    FLAGS..... 000000    ENTRY TYPE..... 00000001
+000418    F001183F 58F0C31C 184E05EF 00000000  47F0303A 90E7D00C 58E0D04C 4100E098  |0....0C..+.......0...X.....<...q|
+000438    60606E4D F0F7F75D 40868393 96A2854D  5D404040 40404040 40404040 40404040  |-->(077) fclose()               |
+000458    40404040 40404040 40404040 40404040  40404040 40404040 40000000 00000000  |                          ......|
+000478    00000000 00000000                                                         |........                        |
+000480    TIME...... ACAAC0AB10067801    THREAD ID......... 8000000000000000
+000490    MEMBER ID.... 03    FLAGS..... 000000    ENTRY TYPE..... 00000002
+000498    4C60604D F0F7F75D 40D9F1F5 7EF0F0F0  F0F0F0F0 F040C5D9 D9D5D67E F0F0F0F0  |<--(077) R15=00000000 ERRNO=0000|
+0004B8    F0F0F0F0 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |0000............................|
+0004D8    00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
+0004F8    00000000 00000000                                                         |........                        |
  ⋮
```

Figure 14. Trace Table in Dump Output (Part 2 of 2)

Diagnosing Storage Leak Problems

A *storage leak* occurs when a program does not return storage back to the heap, after it has finished using it. To determine if a storage leak exists, you should set the HEAPCHK run-time option to HEAPCHK(ON,*n*,*n*). A report will then be produced in the CEEDUMP output.

HEAPCHK identifies storage that is still allocated (that is, storage not freed) and lists such storage, together with a corresponding traceback, in the CEEDUMP output. For more information about setting the HEAPCHK run-time option, refer to the *LE/VSE Customization Guide*.

Figure 15 provides an example of the CEEDUMP output that includes storage not freed, as well as the calls involved in allocating this storage.

```
1 HEAP STORAGE DIAGNOSTICS
   STG ADDR  ID        LENGTH      ENTRY         E ADDR    E OFFSET    LOAD MOD
   007EF820  007EB094  00000048    CEEV#GTS      005855A8  +00000000   CEEPLPKA
                                   CEEVGTST      0052D7E8  +00000072   CEEBINIT
                                   identify_cu   006029B8  +00602BD4   CEEEV003
                                   SetDumpVars   0060DC88  +0060DEB2   CEEEV003
                                   __zdmpd       006402B8  +00640552   CEEEV003
                                   @@ZDMPS       00644948  +000000EA   CEEEV003
                                   CEEKDUMP      007C7378  +00002472   CEEKDS
                                   CEEHDSP       00562898  +000029FA   CEEEV003
                                   main          005200F8  +000000DE   CVSEPRG3
                                   @@MNINV       00649D3E  +000000B4   CEEEV003
 3 007EF868  007EB094  00000028    CEEV#GTS      005855A8  +00000000   CEEPLPKA
                                   CEEVGTST      0052D7E8  +00000072   CEEBINIT
                                   identify_cu   006029B8  +00602D34   CEEEV003
                                   SetDumpVars   0060DC88  +0060DEB2   CEEEV003
                                   __zdmpd       006402B8  +00640552   CEEEV003
                                   @@ZDMPS       00644948  +000000EA   CEEEV003
                                   CEEKDUMP      007C7378  +00002472   CEEKDS
                                   CEEHDSP       00562898  +000029FA   CEEEV003
                                 2 main          005200F8  +000000DE   CVSEPRG3
                                   @@MNINV       00649D3E  +000000B4   CEEEV003
```

Figure 15. Sections of the HEAPCHK Diagnostics Report Showing a Leak Problem

The numbers in Figure 15 refer to these parts of the CEEDUMP output:

1 Heap storage diagnostics report heading.

2 The name of the user application which has been executed. This is identifiable by the load module name (CVSEPRG3) and function name (main). This traceback shows that function `main` at offset `+X'DE'` caused `X'28'` bytes of storage to be acquired which was not released at program termination time. If the application terminated abnormally, this may be acceptable. However, if the application did not terminate abnormally, then this `X'28'` bytes of storage was never "freed".

Note: `@@MNINV` is the LE/C function name used to invoke the C/VSE user application, and so will always appear first in the traceback report.

3 This line of the traceback report shows the start address (STG ADDR) and length of the storage acquired.

Note: Some traceback reports may show the LE run-time with storage remaining. This is because the run-time requires more permanent storage areas that must be available for the entire duration of the application. In most cases these permanent areas will be freed, but only after the diagnostics report has been produced.

Part 2. Debugging Language-Specific Routines

This part of the book provides specific information for debugging applications written in C, COBOL, and PL/I. It also discusses techniques for debugging under CICS.

Chapter 4. Debugging C Routines

This chapter provides specific information to help you debug applications that contain one or more C routines. It includes the following topics:
- Debugging C I/O routines
- Using C compiler listings
- Generating an LE/VSE dump of a C routine
- Finding C information in an LE/VSE dump
- Debugging example of C routines

There are several debugging features that are unique to C routines. Before examining the C techniques to find errors, you might want to consider the following areas of potential problems:
- If you suspect that you are using uninitialized storage, you may want to use the STORAGE run-time option.
- If you are using the `fetch()` function, see *LE/VSE C Run-Time Programming Guide* to ensure that you are creating the fetchable phase correctly.
- Ensure that the entry point of the phase is CEESTART.
- You should avoid:
  - Incorrect casting
  - Referencing an *array* element with a subscript outside the declared bounds
  - Copying a string to a target with a shorter length than the source string
  - Declaring but not initializing a pointer variable, or using a pointer to allocated storage that has already been freed

  If a routine exception is the problem and you need more information than the condition handler provided, run your routine with the TRAP(OFF) run-time option. Note, however, that using TRAP(OFF) can cause unpredictable results and should be used only in special debugging circumstances. For more information about the effects of running with TRAP(OFF), see *LE/VSE Programming Reference*.

Debugging C Input/Output Programs

You can use C conventions such as `__amrc` and `perror()` when you debug I/O operations.

Using the `__amrc` and `__amrc2` Structures

`__amrc`, a structure defined in `stdio.h`, can help you determine the cause of errors resulting from an I/O operation, because it contains diagnostic information (for example, the return code from a failed VSAM operation).

There are two structures: `__amrc` and `__amrc2`, which are defined by types `__amrc_type` and `__amrc2_type`, respectively. The `__amrc2_type` structure contains secondary information that C can provide.

Because any I/O function calls, such as `printf()`, can change the value of `__amrc` or `__amrc2`, make sure you save the contents into temporary structures of `__amrc_type` and `__amrc2_type` respectively, before dumping them.

Figure 16 shows the `__amrc` structure as it appears in `stdio.h`. Figure 17 shows the `__amrc2` structure as it appears in `stdio.h`. A description of information contained in both structures follows.

```
typedef struct __amrctype {

   union {      1
      long int __error;     2


      struct {
        unsigned short __syscode,
                       __rc;
      } __abend;     3
      struct {
         unsigned char __fdbk_fill,
                       __rc,
                       __ftncd,
                       __fdbk;
      } __feedback;     4
      struct {
         unsigned short __svc99_info,
                        __svc99_error;
      } __alloc;      5
   } __code;
   unsigned long __RBA;     6

   unsigned int      __last_op;     7
   struct {
    unsigned long  __len_fill;
    unsigned long  __len;
    char           __str[120];
    unsigned long  __parmr0;
    unsigned long  __parmr1;
    unsigned long  __fill2[2];
    char           __str2[64];
   } __msg;     8
} __amrc_type;
```

Figure 16. `__amrc` Structure

```
struct {
    long int       __error2;     9                    */
    FILE          *__fileptr;    10                   */
    long int       __reserved[6];
}
```

Figure 17. `__amrc2` Structure

1 __code
The error or warning value from an I/O operation is in either `__error`, `__abend`, `__feedback`, or `__alloc`. You must look at `__last_op` to determine how to interpret the `__code` union.

2 __error
`__error` contains the return code from the system macro or utility. Refer to Table 29 on page 65 for further information.

3 __abend
This structure contains the abend code when `errno` is set to indicate a recoverable I/O abend. `__syscode` is the system cancel code and `__rc` is set

to zero. The macros `__abendcode()` and `__rsncode()` may be set to the abend code and reason code of a command when invoked with `system()`.

4 **`__feedback`**
This structure is used for VSAM only. The `__rc` field stores the VSAM register 15 and the `__fdbk` field stores the VSAM error code or reason code. See also the `__RBA` field, below.

5 **`__alloc`**
This structure contains no valid information under VSE.

6 **`__RBA`**
This is the relative byte address (RBA) value returned by VSAM after an ESDS or KSDS record is written out. For a RRDS, it is the calculated value from the record number. It may be used in subsequent calls to `flocate()`.

7 **`__last_op`**
This field contains a value that indicates the last I/O operation being performed by C at the time the error occurred. These values are shown in Table 29.

8 **`__msg`**
This structure may contain an error message from read or write operations, but is not always filled.

This structure is used by the `SIGIOERR` handler.

9 **`__error2`**
This field is not used under VSE.

10 **`__fileptr`**
This field is used by the signal `SIGIOERR` to pass back a `FILE` pointer that can then be passed to `fldata()` to get the name of the file causing the error.

__last_op Values

The `__last_op` field is the most important of the `__amrc` fields. It defines the last I/O operation C was performing at the time of the I/O error. You should note that the structure is neither cleared nor set by non-I/O operations, so querying this field outside of a `SIGIOERR` handler should only be done immediately after I/O operations. Table 29 lists `__last_op` values you may receive and where to look for further information.

Table 29. __last_op Codes and Diagnosis Information

Code	Further Information
`__IO_INIT`	Will never be seen by `SIGIOERR` exit value given at initialization.
`__BSAM_OPEN`	Sets __error with return code from VSE OPEN macro.
`__BSAM_CLOSE`	Sets __error with return code from VSE CLOSE macro.
`__BSAM_READ`	No return code (either __abend (errno == 92) or __msg (errno == 66) filled in).
`__BSAM_NOTE`	NOTE returned `0` unexpectedly, no return code.
`__BSAM_POINT`	This will not appear as an error lastop.
`__BSAM_WRITE`	No return code (either __abend (errno == 92) or __msg (errno == 65) filled in).
`__BSAM_CLOSE_T`	Not supported under VSE, but retained for compatibility.

Table 29. __last_op Codes and Diagnosis Information (continued)

Code	Further Information
`__BSAM_BLDL`	Not supported under VSE, but retained for compatibility.
`__BSAM_STOW`	Not supported under VSE, but retained for compatibility.
`__TGET_READ`	Not supported under VSE, but retained for compatibility.
`__TPUT_WRITE`	Not supported under VSE, but retained for compatibility.
`__IO_DEVTYPE`	Sets __error with return code from VSE EXTRACT macro.
`__IO_TRKCALC`	Sets __error with return code from VSE GETVCE macro.
`__IO_OBTAIN`	Not supported under VSE, but retained for compatibility.
`__IO_LOCATE`	Not supported under VSE, but retained for compatibility.
`__IO_CATALOG`	Not supported under VSE, but retained for compatibility.
`__IO_UNCATALOG`	Not supported under VSE, but retained for compatibility.
`__IO_RENAME`	Not supported under VSE, but retained for compatibility.
`__C_TRUNCATE`	Set when C truncates output data. Usually this is data written to a text file with no newline such that the record fills up to capacity and subsequent characters cannot be written. For a record I/O file this refers to an `fwrite` writing more data than the record can hold. Truncation is always rightmost data. There is no return code.
`__C_FCBCHECK`	Set when C FCB is corrupted. This is due to a pointer corruption somewhere. File cannot be used after this.
`__C_DBCS_TRUNCATE`	This occurs when writing DBCS data to a text file and there is no room left in a physical record for anymore double byte characters. A new-line is not acceptable at this point. Truncation will continue to occur until an SI is written or the file position is moved. Cannot happen if MB_CUR_MAX is 1.
`__C_DBCS_SO_TRUNCATE`	This occurs when there is not enough room in a record to start any DBCS string or else when a redundant SO is written to the file before an SI. Cannot happen if MB_CUR_MAX is 1.
`__C_DBCS_SI_TRUNCATE`	This occurs only when there was not enough room to start a DBCS string and data was written anyways, with an SI to end it. Cannot happen if MB_CUR_MAX is 1.
`__C_DBCS_UNEVEN`	This occurs when an SI is written before the last double byte character is completed, thereby forcing C to fill in the last byte of the DBCS string with a padding byte X'FE'. Cannot happen if MB_CUR_MAX is 1.
`__C_CANNOT_EXTEND`	This occurs when an attempt is made to extend a file that allows writing, but cannot be extended. Typically this is a member of a VSE Librarian sublibrary being opened for update.
`__VSAM_OPEN_FAIL`	Set when a low level VSAM OPEN fails, sets __rc and __fdbk fields in the __amrc struct.
`__VSAM_OPEN_ESDS`	Does not indicate an error; set when the low level VSAM OPEN succeeds, and the file type is ESDS.
`__VSAM_OPEN_RRDS`	Does not indicate an error; set when the low level VSAM OPEN succeeds, and the file type is ESDS.
`__VSAM_OPEN_KSDS`	Does not indicate an error; set when the low level VSAM OPEN succeeds, and the file type is ESDS.

Table 29. __last_op Codes and Diagnosis Information (continued)

Code	Further Information
`__VSAM_OPEN_ESDS_PATH`	Does not indicate an error; set when the low level VSAM OPEN succeeds, and the file type is ESDS.
`__VSAM_OPEN_KSDS_PATH`	Does not indicate an error; set when the low level VSAM OPEN succeeds, and the file type is ESDS.
`__VSAM_MODCB`	Set when a low level VSAM MODCB macro fails, sets __rc and __fdbk fields in the __amrc struct.
`__VSAM_TESTCB`	Set when a low level VSAM TESTCB macro fails, sets __rc and __fdbk fields in the __amrc struct.
`__VSAM_SHOWCB`	Set when a low level VSAM SHOWCB macro fails, sets __rc and __fdbk fields in the __amrc struct.
`__VSAM_GENCB`	Set when a low level VSAM GENCB macro fails, sets __rc and __fdbk fields in the __amrc struct.
`__VSAM_GET`	Set when the last op was a low level VSAM GET; if the GET fails, sets __rc and __fdbk in the __amrc struct.
`__VSAM_PUT`	Set when the last op was a low level VSAM PUT; if the PUT fails, sets __rc and __fdbk in the __amrc struct.
`__VSAM_POINT`	Set when the last op was a low level VSAM POINT; if the POINT fails, sets __rc and __fdbk in the __amrc struct.
`__VSAM_ERASE`	Set when the last op was a low level VSAM ERASE; if the ERASE fails, sets __rc and __fdbk in the __amrc struct.
`__VSAM_ENDREQ`	Set when the last op was a low level VSAM ENDREQ; if the ENDREQ fails, sets __rc and __fdbk in the __amrc struct.
`__VSAM_CLOSE`	Set when the last op was a low level VSAM CLOSE; if the CLOSE fails, sets __rc and __fdbk in the __amrc struct.
`__QSAM_GET`	__error is not set (if abend (errno == 92), __abend is set, otherwise if read error (errno == 66), look at __msg.
`__QSAM_PUT`	__error is not set (if abend (errno == 92), __abend is set, otherwise if write error (errno == 65), look at __msg.
`__QSAM_TRUNC`	This is an intermediate operation. You will only see this if an I/O abend occurred.
`__QSAM_CLOSE`	Sets __error to result of VSE CLOSE macro.
`__QSAM_OPEN`	Sets __error to result of VSE OPEN macro.
`__CICS_WRITEQ_TD`	Sets __error with error code from EXEC CICS WRITEQ TD.

Displaying an Error Message with the perror() Function

To find a failing routine, check the return code of all function calls. After you have found the failing routine, use the `perror()` function after the routine to display the error message. `perror()` displays the string that you pass to it and an error message corresponding to the value of `errno`. `perror()` writes to the standard error stream (`stderr`). Figure 18 on page 68 shows an example of a routine using `perror()`.

```
#include <stdio.h>
int main(void)

   FILE *fp;

   fp = fopen("myfile.dat", "w");
   if (fp == NULL)
      perror("fopen error");
```

Figure 18. Example Routine Using perror()

Using C Compiler Listings

The following sections discuss C compiler-generated listings, and explain how to use these listings to locate information, such as variable values and the timestamp, in the dump.

Generating C Listings and Maps

C listings and maps show the compile, prelink, and link-edit steps that provide many pieces of information necessary for performing problem analysis tasks.

When you are debugging, you can use one or more of the following:
- Pseudo-assembler listing
- Storage offset listing
- Structure map
- Prelinker map
- Inline report
- Linkage editor module map
- Source listing

This section gives an overview of each of these listings and specifies the compile-time option you use to obtain the listing. For a detailed description of available listings, see *LE/VSE C Run-Time Programming Guide*.

Table 30. Contents of Listing and Associated Compile-Time Options

Name	Compile-Time Option	Function
Pseudo-assembler listing	LIST	Generates a pseudo-assembler listing, which shows the source listing for the current routine
Storage Offset Listing	XREF	Produces a storage offset listing, which includes in the source listing a cross reference table of names used in the routine and the line numbers on which they were declared or referenced.
Structure Map	AGGREGATE	Causes a structure map to be included in the source listing. The structure map shows the layout of variables for the type `struct` or `union`.
Inline Report	INLINE(,REPORT,...)	Generates an inline report that summarizes all functions inlined and a provides a detailed call structure of all the functions.

Table 30. Contents of Listing and Associated Compile-Time Options (continued)

Name	Compile-Time Option	Function
Prelinker Map	MAP (prelink option)	Creates the prelinker map when invoking the Prelinker. (This is the default.) Prelinker maps can be used to determine the location of static and external C variables compiled with the RENT option.
Linkage Editor Module Map	MAP (link-time option)	Creates the linkage editor map when invoking C. (This is the default.) Linkage editor maps can expose incorrect linkage; for example, if CEESTART was not the entry point or a fetch phase was created incorrectly.
Source Listing	SOURCE	Generates the source listing, which contains the original source input statements and any compile-time diagnostic messages issued for the source.

Finding Variables

You can determine the value of a variable in the routine at the point of interrupt by using the compiled code listing as a guide to its address, then finding this address in the LE/VSE dump. The method you use depends on the storage class of variable.

This method is generally used when no symbolic variables have been dumped (symbolic variables can be dumped by using the TEST compile-time option).

It is possible for the routine to be interrupted before the value of the variable is placed in the location provided for it. This can explain unexpected values in the dump.

Finding Automatic Variables

To find automatic variables in the LE/VSE dump, use the following steps:

1. Identify the start of the stack frame. When a dump is produced, each stack frame is dumped. The stack frames can be cross-referenced to the function name in the traceback.
2. Add the offset of the variable (which is given in decimal) in the storage offset listing, shown in Figure 19, to the stack frame address.

```
aa1              85-0:85        Class = automatic,     Offset = 164(r13),    Length = 40
```

Figure 19. Example of a Storage Offset Listing in Compiled Code

In the example, variable aa1 can be found in the dump by first determining the value of the base register (in this case, GPR13) in the Saved Registers section for the function you are interested in. Add this base address to the offset of the variable. The contents of the variable can then be read in the DSA Frame section corresponding to the function the variable is contained in.

Finding the Writable Static Area

If you have C code compiled with the RENT option (hereafter called RENT code):

- You must determine the base address of the writable static area (WSA) if you want to calculate the address of a static or external variable.
- The WSA for application code is in the WSA field of the CAA (see page 87 for an explanation of WSA).
- The WSA for a fetched phase is in the WSA field of the Fetch Information section for the fetch function pointer you are interested in.

Finding the Static Storage Area

If you have C code compiled with the NORENT option (hereafter called NORENT code):

- You must determine the base address of the static storage area if you want to calculate the address of a static variable. A CSECT is generated for the static storage area for each source file. In order to determine the origin and length of the CSECT from the prelinker map, you must name the static storage area CSECT by using the `pragma csect` directive.
- External variables are stored in a corresponding CSECT with the same name. The origin and length of the external variable CSECT can be determined from the prelinker map.

Address calculation for static and external variables uses the static storage area as a base address with 1 or more offsets added to this address.

The storage associated with these CSECTs is not dumped when an exception occurs. This is dumped when you call `cdump()`, which invokes the VSE SDUMP macro. SDUMP output is written to either SYSLST or the dump sublibrary. See "Generating an LE/VSE Dump of a C Routine" on page 77 for information about `cdump()`, SDUMP, and CEE5DMP.

Finding RENT Static Variables

1. Find the address of the WSA (see "Finding the Writable Static Area"). For this example, the address of writable static is X'02D66E40'.
2. Find the offset of `@STATIC` (associated with the file where the static variable is located) in the Writable Static Map section of the prelinker map, shown in Figure 20 on page 71. In this example, the offset is X'00000058'. Add this offset to the WSA to get the base address of static variables. The result is X'02D66E98' (X'02D66E40' + X'00000058').

```
======================================================================
|                          Writable Static Map                       |
======================================================================

  OFFSET    LENGTH  FILE ID  INPUT NAME

       0         1   00001   DFHC0011
       4         1   00001   DFHC0010
       8         2   00001   DFHDUMMY
       C         2   00001   DFHB0025
      10         2   00001   DFHB0024
      14         2   00001   DFHB0023
      18         2   00001   DFHB0022
      1C         2   00001   DFHB0021
      20         2   00001   DFHB0020
      24         2   00001   DFHEIB0
      28         4   00001   DFHEIPTR
      2C         4   00001   DFHCP011
      30         4   00001   DFHCP010
      34         4   00001   DFHBP025
      38         4   00001   DFHBP024
      3C         4   00001   DFHBP023
      40         4   00001   DFHBP022
      44         4   00001   DFHBP021
      48         4   00001   DFHBP020
      4C         4   00001   DFHEICB
      50         4   00001   DFHEID0
      54         4   00001   DFHLDVER
      58       278   00001   @STATIC
     720        30   00002   @STATIC
```

Figure 20. Writable Static Map Produced by Prelinker

3. Find the offset of the static variable in the partial storage offset compiler listing shown in Figure 21. In this example, the offset of the static variable sa0 is 96 (X'00000060').

```
sa0             66-0:66       Class = static,   Location = WSA  @STATIC + 96,  Length = 4
```

Figure 21. Partial Storage Offset Listing

4. Add this offset to the base address of static variables, calculated above. In this example, the sum is X'02D66EF8' (X'02D66E98' + X'00000060'). This is the address of the value of the static variable sa0 in the LE/VSE dump.

   Figure 22 on page 72 shows the path to locate RENT C static variables by adding the address of writable static, the offset of @STATIC, and the variable offset.

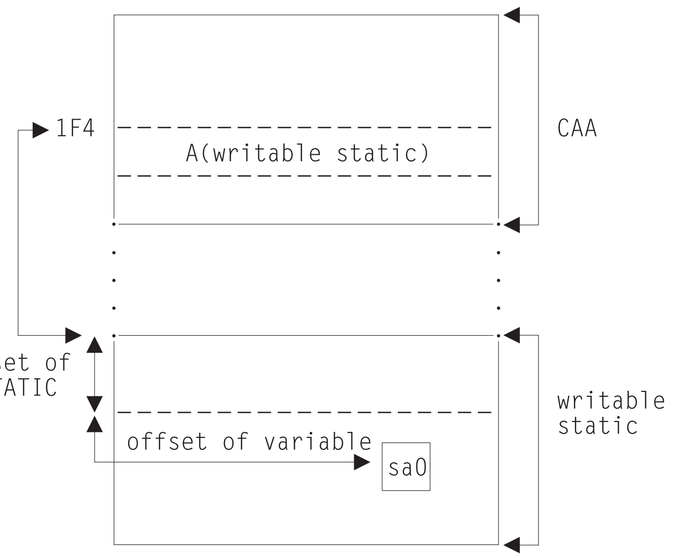


Figure 22. Location of RENT Static Variable in Storage

Finding External RENT Variables

Locating external variables in the LE/VSE dump requires several steps:

1. Find the address of the WSA (see "Finding the Writable Static Area" on page 70). In this example, the address of writable static is X'02D66E40'.
2. Find the offset of the external variable in the Prelinker Writable Static Map, shown in Figure 23 on page 73. In this example, the offset for `DFHEIPTR` is X'00000028'.

```
=======================================================================
|                          Writable Static Map                        |
=======================================================================

  OFFSET    LENGTH  FILE ID  INPUT NAME

       0         1   00001   DFHC0011
       4         1   00001   DFHC0010
       8         2   00001   DFHDUMMY
       C         2   00001   DFHB0025
      10         2   00001   DFHB0024
      14         2   00001   DFHB0023
      18         2   00001   DFHB0022
      1C         2   00001   DFHB0021
      20         2   00001   DFHB0020
      24         2   00001   DFHEIB0
      28         4   00001   DFHEIPTR
      2C         4   00001   DFHCP011
      30         4   00001   DFHCP010
      34         4   00001   DFHBP025
      38         4   00001   DFHBP024
      3C         4   00001   DFHBP023
      40         4   00001   DFHBP022
      44         4   00001   DFHBP021
      48         4   00001   DFHBP020
      4C         4   00001   DFHEICB
      50         4   00001   DFHEID0
      54         4   00001   DFHLDVER
      58       420   00001   @STATIC
```

Figure 23. Writable Static Map Produced by Prelinker

3. Add the offset of the external variable to the address of writable static. The result is X'02D66E68', (X'02D66E40' + X'00000028'). This is the address of the value of the external variable in the LE/VSE dump.

Finding NORENT Static Variables

1. Find the name and address of the static storage area (see "Finding the Static Storage Area" on page 70). For this example, the static storage area is called **STATSTOR** and has an address of X'02D66E40'.
2. Find the offset of the static variable in the partial storage offset compiler listing shown in Figure 24. The offset is 96 (X'00000060').

```
sa0             66-0:66       Class = static,       Location = STATSTOR +96,   Length = 4
```

Figure 24. Partial Storage Offset Listing

3. Add this offset to the base address of static variables, calculated above. The sum is X'02D66EA0' (X'02D66E40' + X'00000060'). This is the address of the value of the static variable in the LE/VSE dump.

   Figure 25 on page 74 shows how to locate NORENT C static variables by adding the Static Storage Area CSECT address to the variable offset.

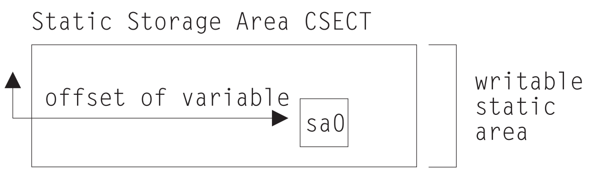


Figure 25. Location of NORENT Static Variable in Storage

Finding External NORENT Variables

1. Find the address of the external variable CSECT (see "Finding the Static Storage Area" on page 70). In this example, the address is X'02D66E40'. This is the address of the value of the external variable in the LE/VSE dump.

Finding the C Parameter List

A pointer to the parameter list is passed to the called function in register 1. This is the address of the start of the parameter list. Figure 26 shows the example code for the parameter variable.

```
func0()
   ⋮
 func1(a1,a2);
   ⋮


func1(int ppx, int pp0)
   ⋮
```

Figure 26. Example Code for Parameter Variable

Parameters *ppx* and *pp0* correspond to copies of *a1* and *a2* in the stack frame belonging to `func0`.

To locate a parameter in the LE/VSE dump:

1. Find the register and offset in the partial storage offset listing shown in Figure 27. In this example the offset is 4 (X'0000004') from register 1.

```
 pp0                62-0:62        Class = parameter,     Location = 4(r1),    Length = 4
```

Figure 27. Partial Storage Offset Listing

2. Determine the value of `GPR1` in the Saved Registers section for the function that called the function you are interested in. Add this base address to the offset of the parameter. The contents of the variable can then be read in the DSA frame section corresponding to the function the parameter was passed from.

Finding Members of Aggregates

You can define aggregates in any of the storage classes or pass them as parameters to a called function. The first step is to find the start of the aggregate. You can compute the start of the aggregate as described in previous sections, depending on the type of aggregate used.

The aggregate map provided for each declaration in a routine can further assist in finding the offset of a specific variable within a aggregate. Structure maps are generated using the AGGREGATE compile-time option. Figure 28 shows an example of a static aggregate.

```
static struct {
    short int ss01;
    char      ss02[56];
    int       sz0[6];
    int       ss03;
} ss0;
```

Figure 28. Example Code for Structure Variable

Figure 29 shows an example structure map.

```
================================================================================
| Aggregate map for:  ss0
================================================================================
|      Offset       |      Length       | Member Name
|    Bytes(Bits)    |    Bytes(Bits)    |
====================|===================|=======================================
|       0           |        2          |  ss01
|       2           |       56          |  ss02[56]
|      58           |        2          |  ***PADDING***
|      60           |       24          |  sz0[6]
|      84           |        4          |  ss03
================================================================================
```

Figure 29. Example of Structure Map

Assume the structure has been compiled as RENT. To find the value of variable *sz0[0]*:

1. Find the address of the writable static. For this example the address of writable static is X'02D66E40'.
2. Find the offset of @STATIC in the Writable Static Map. In this example, the offset is X'00000058'. Add this offset to the address of writable static. The result is X'02D66E98' (X'02D66E40' + X'00000058'). Figure 30 on page 76 shows the Writable Static Map produced by the prelinker.

```
========================================================================
|                         Writable Static Map                          |
========================================================================

  OFFSET    LENGTH  FILE ID  INPUT NAME

       0         1   00001   DFHC0011
       4         1   00001   DFHC0010
       8         2   00001   DFHDUMMY
       C         2   00001   DFHB0025
      10         2   00001   DFHB0024
      14         2   00001   DFHB0023
      18         2   00001   DFHB0022
      1C         2   00001   DFHB0021
      20         2   00001   DFHB0020
      24         2   00001   DFHEIB0
      28         4   00001   DFHEIPTR
      2C         4   00001   DFHCP011
      30         4   00001   DFHCP010
      34         4   00001   DFHBP025
      38         4   00001   DFHBP024
      3C         4   00001   DFHBP023
      40         4   00001   DFHBP022
      44         4   00001   DFHBP021
      48         4   00001   DFHBP020
      4C         4   00001   DFHEICB
      50         4   00001   DFHEID0
      54         4   00001   DFHLDVER
      58       320   00001   @STATIC
```

Figure 30. Writable Static Map Produced by Prelinker

3. Find the offset of the static variable in the storage offset listing. The offset is 96 (X'00000060').

   Figure 31 shows a partial storage offset listing.

```
ss0              66-0:66        Class = static,      Location = GPR13(96),   Length = 4
```

Figure 31. Partial Storage Offset Listing

   Add this offset to the result from step 2. The result is X'02D66EF8' (X'02D66E98' + X'00000060'). This is the address of the value of the static variable in the dump.
4. Find the offset of sz0 in the Aggregate Map, shown in Figure 29 on page 75. The offset is 60.

Add the offset from the Aggregate Map to the address of the ss0 struct. The result is X'00000060', (X'0000003C' + X'00000060'). This is the address of the values of sz0 in the dump.

Finding the Timestamp

The timestamp can be located in the compile unit block. The address for the compile unit block is located at eight bytes past the function entry point. The compile unit block is the same for all functions in the same compilation. The fourth word of the compile unit block points to the timestamp. The timestamp is 16 bytes long and has the following format:

```
YYYYMMDDHHMMSSSS
```

Generating an LE/VSE Dump of a C Routine

You can use either the CEE5DMP callable service or the `ctrace()`, `csnap()`, and `cdump()` C functions to generate an LE/VSE dump of C routines. These C functions call CEE5DMP with specific options.

To use these functions, you must add `#include <ctest.h>` to your C code.

The output, which is written to filename CEEDUMP (if present) or SYSLST (otherwise), is identified by the *dumptitle* string parameter you pass with the C function.

For more detail about the syntax of these functions, see *LE/VSE C Run-Time Library Reference*.

For more detail about CEE5DMP, see "Generating a Dump with CEE5DMP" on page 32.

cdump()

The `cdump()` function produces a main storage dump with the activation stack.

`cdump()` is equivalent to calling CEE5DMP with the option string:
TRACEBACK BLOCKS VARIABLES FILES STORAGE STACKFRAME(ALL) CONDITION ENTRY

Usage Notes:

1. If a routine calls `cdump()` more than once, use the `cdump(`*`dumptitle`*`)` parameter to help identify the beginning of each dump.
2. Output from `cdump()` is directed to the filename CEEDUMP (if defined) or SYSLST (otherwise).
3. When you call `cdump()`, the C library issues a VSE SDUMP macro to obtain a snap dump of virtual storage. The first invocation of `cdump()` results in a snap dump identifier of 0. For each successive invocation, the identifier is increased by one to a maximum of 255, after which it is reset to 0.
4. Support for SDUMP dumps using `cdump()` is provided only in the batch environment. SDUMP dumps are not produced in a CICS environment.
5. If the SDUMP does not succeed, the CEE5DMP DUMP file displays the message:
   ```
   Snap was unsuccessful
   ```
6. If the SDUMP is successful, CEE5DMP displays the message:
   ```
   Snap was successful; snap ID = nnn
   ```
   where *nnn* corresponds to the snap dump identifier described above. The identifier is incremented only if SDUMP is successful.

ctrace()

The `ctrace()` function produces a traceback and includes the offset addresses from which the calls were made.

`ctrace()` is equivalent to calling CEE5DMP with the option string:
TRACEBACK NOFILES NOBLOCKS NOVARIABLES NOSTORAGE STACKFRAME(ALL) NOCONDITION NOENTRY

csnap()

The `csnap()` function produces a condensed storage dump.

`csnap()` function is equivalent to calling CEE5DMP with the option string:
TRACEBACK FILES BLOCKS VARIABLES NOSTORAGE STACKFRAME(ALL) CONDITION ENTRY

Sample C Routine that Calls cdump()

Figure 32 shows a sample C routine. This fetches the routine in Figure 33 on page 80, which calls the `cdump()` function to generate a dump. To run this sample, you must have:

- included this JCL statement:
  ```
  // DLBL TMP0001,'MYFILE.DATA',,VSAM,CAT=VSESPUC
  ```
- created a VSAM SAM ESDS file that has a fixed block record format (RECSIZE 80, BLKSIZE 80).

"Sample LE/VSE Dump with C-Specific Information" on page 81 shows the dump output.

```
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>

void hsigfpe(int);
void hsigterm(int);
void atf1(void);
typedef int (*FuncPtr_T)(void);

int st1 = 99;
int st2 = 255;
int xcount = 0;

int main(void) {
  /*
   * 1) Open multiple files
   * 2) Register 2 signals
   * 3) Register 1 atexit function
   * 4) Fetch and execute phase
   */
  FuncPtr_T fetchPtr;
  FILE*     fp1;
  FILE*     fp2;
  int       rc;
```

Figure 32. Example C Routine Using cdump() to Generate a Dump (Part 1 of 2)

```
  fp1 = fopen("DD:TMP0001", "w");
  if (!fp1) {
    perror("Could not open myfile.data for write");
    exit(101);
  }

  fprintf(fp1, "record 1\n");
  fprintf(fp1, "record 2\n");
  fprintf(fp1, "record 3\n");

  fp2 = fopen("memory.data", "wb,type=memory");
  if (!fp2) {
    perror("Could not open memory.data for write");
    exit(102);
  }

  fprintf(fp2, "some data");
  fprintf(fp2, "some more data");
  fprintf(fp2, "even more data");

  signal(SIGFPE , hsigfpe);
  signal(SIGTERM, hsigterm);

  rc = atexit(atf1);
  if (rc) {
    fprintf(stderr,"Failed on registration of atexit function atf1\n");
    exit(103);
  }

  fetchPtr =(FuncPtr_T) fetch("PHASE1");
  if (!fetchPtr) {
    fprintf(stderr,"Failed to fetch PHASE1\n");
    exit(104);
  }
  fetchPtr();
  return(0);
  }

void hsigfpe(int sig) {
  ++st1;
  return;
}

void hsigterm(int sig) {
  ++st2;
  return;
}

void atf1() {
   ++xcount;
}
```

Figure 32. Example C Routine Using cdump() to Generate a Dump (Part 2 of 2)

Figure 33 on page 80 shows the routine fetched by the routine in Figure 32 on page 78.

```
#include <ctest.h>

#pragma linkage(func1, fetchable)
int func1(void) {
  cdump("This is a sample dump");
  return(0);
}
```

Figure 33. Fetched C Routine

Sample LE/VSE Dump with C-Specific Information

This sample dump was produced by compiling the routine in Figure 32 on page 78 with the TEST(ALL) compile-time option, then running it. Notice the sequence of calls in the traceback section: @@MNINV is the C management module that invokes `main` and @@FECBFUNC1 fetches the user-defined function `func1`, which in turn calls the library routine `__cdump`.

If source code is compiled with the `GONUMBER` or `TEST` compile option, statement numbers are shown in the trace back. If source code is compiled with the `TEST(ALL)` compile option, variables and their associated type and value are dumped out. See “Finding C Information in an LE/VSE Dump” on page 86 for more information about C-specific information contained in a dump.

```
CEE5DMP V1 R4.2: This is a sample dump                                              07/30/01 3:13:21 PM                PAGE:    1

CEE5DMP CALLED BY PROGRAM UNIT  (ENTRY POINT __cdump) AT OFFSET +039BA334.

Snap was successful; snap ID = 0

REGISTERS ON ENTRY TO CEE5DMP:

  PM....... 0100
  GPR0..... 00000000  GPR1..... 0066B5A0  GPR2..... 0066B550  GPR3..... 839BA1CE
  GPR4..... 006AB300  GPR5..... 0066B534  GPR6..... 00000000  GPR7..... 00000000
  GPR8..... 00000005  GPR9..... 00000000  GPR10.... 806AB080  GPR11.... 806AB080
  GPR12.... 00657E48  GPR13.... 0066B490  GPR14.... 8065AEEE  GPR15.... 83A01D68
  FPR0..... 4E800000  00000000            FPR2..... 487FFFFF  FF000000
  FPR4..... 41A00000  00000000            FPR6..... 40404040  40404040
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (0066B490)
        -0020 0066B470  00000000 00000000 0066B338 0066B340  8066B368 0066B4A0 006AB300 0066B49C  |............... ................|
        +0000 0066B490  1066B4A4 0066B3F8 0066B5B0 8065AEEE  83A01D68 00000000 0066B5A0 0066B550  |...u...8........c..............&|
        +0020 0066B4B0  839BA1CE 006AB300 0066B534 00000000  00000000 00000005 00000000 806AB080  |c...............................|
      STORAGE AROUND GPR 1 (0066B488)
        -0020 0066B468  00000000 00000000 00000000 00000000  0066B338 0066B340 8066B368 0066B4A0  |...................... ........|
        +0000 0066B488  006AB300 0066B49C 1066B4A4 0066B3F8  0066B5B0 8065AEEE 83A01D68 00000000  |...........u...8........c.......|
        +0020 0066B4A8  0066B5A0 0066B550 839BA1CE 006AB300  0066B534 00000000 00000000 00000005  |.......&c.......................|
:
INFORMATION FOR ENCLAVE main

  INFORMATION FOR THREAD 8000000000000000

  REGISTERS ON ENTRY TO CEE5DMP:
    PM....... 0100
    GPR0..... 00000000  GPR1..... 0066B5A0  GPR2..... 0066B550  GPR3..... 839BA1CE
    GPR4..... 006AB300  GPR5..... 0066B534  GPR6..... 00000000  GPR7..... 00000000
    GPR8..... 00000005  GPR9..... 00000000  GPR10.... 806AB080  GPR11.... 806AB080
    GPR12.... 00657E48  GPR13.... 0066B490  GPR14.... 8065AEEE  GPR15.... 83A01D68
    FPR0..... 4E800000  00000000            FPR2..... 487FFFFF  FF000000
    FPR4..... 41A00000  00000000            FPR6..... 40404040  40404040
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (00000000)
        +0000 00000000    INACCESSIBLE STORAGE.
        +0020 00000020    INACCESSIBLE STORAGE.
        +0040 00000040    INACCESSIBLE STORAGE.
      STORAGE AROUND GPR 1 (0066B5A0)
        -0020 0066B580  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  |                                |
        +0000 0066B5A0  0066B550 039BA6BB 8066B534 00000001  00000000 0066B490 0066C0F8 83A059D8  |...&..w....................8c..Q|
        +0020 0066B5C0  03A07768 0000001F 0066B8D0 00000002  00000000 00000008 0066BD50 03A06D63  |...........................&.._.|
:
```

Figure 34. Example Dump from Sample C Routine (Part 1 of 6)

```
  TRACEBACK:
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY         E ADDR    E  OFFSET   STATEMENT  STATUS
    0066B490                039BA180  +000001B4  __cdump       039BA180  +000001B4              CALL
    0066B3F8  DD:SYSIPT     006AB080  +0000006A  func1         006AB080  +0000006A           5  CALL
    0066B348                0068EC70  +00000000                0068EC70  +00000000              CALL
    0066B290                03911CE0  +0000001A  @@GETFN       03911C38  +000000C2              CALL
    0066B1E8  DD:SYSIPT     005000F8  +0000038C  main          005000F8  +0000038C          59  CALL
    0066B0D8                039BF1E6  +000000B4  @@MNINV       039BF1E6  +000000B4              CALL
    0066B018  CEEBBEXT      001FA3E0  +00000132  CEEBBEXT      001FA3E0  +00000132              CALL

  PARAMETERS, REGISTERS, AND VARIABLES FOR ACTIVE ROUTINES:
⋮
    main (DSA ADDRESS 0066B1E8):
      SAVED REGISTERS:
        GPR0..... 0068ED48  GPR1..... 83968A28  GPR2..... 0066B1E8  GPR3..... 80500144
        GPR4..... 00501538  GPR5..... 0068ED48  GPR6..... 00501A08  GPR7..... 00501664
        GPR8..... 00501660  GPR9..... 80000000  GPR10.... 839BF1DA  GPR11.... 801FA3E0
        GPR12.... 00657E48  GPR13.... 0066B1E8  GPR14.... 80500486  GPR15.... 03911C38
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (0068ED48)
        -0020 0068ED28  0068D000 00000018 00000001 0068ED38  0068EB80 00000000 0068D000 00000028  |................................|
        +0000 0068ED48  180F58FF 001007FF 0068EC70 00000000  03911C38 FFFFFFFF 006AB5E8 0068ED48  |.................j.........Y....|
        +0020 0068ED68  0068ED18 00000000 00000010 00000000  00000000 00000000 00000000 00000000  |................................|
      STORAGE AROUND GPR 1 (83968A28)
        -0020 03968A08  F018401C 50F0F01C 50F05088 47F03524  18F458D0 D00458E0 D00C982B D01C051E  |0. .&00.&0&h.0...4.........q....|
        +0000 03968A28  07070000 D200F000 10000000 FFFFFFFF  00000020 58FF0010 80000000 000000A8  |....K.0........................y|
        +0020 03968A48  00000010 0000003D 00000003 039563A8  03969460 039683D8 039540A8 03911C38  |.............n.y.om-.ocQ.n y.j..|
⋮
      LOCAL VARIABLES:
               fetchPtr         signed int (*) (void)
                                                    0x68ED48
               fp2              struct __ffile *     0x69984C
               fp1              struct __ffile *     0x69866C
               rc               signed int                   0
⋮
  CONTROL BLOCKS FOR ACTIVE ROUTINES:
    DSA FOR __cdump: 0066B490
⋮
    DSA FOR func1: 0066B3F8
      +000000  FLAGS.... 1000      member... 0000      BKC...... 0066B348  FWC...... 0066B4E0  R14...... 806AB0EC
      +000010  R15...... 039BA180  R0....... 0066B490  R1....... 0066B488  R2....... 0066B3F8  R3....... 806AB0CE
      +000024  R4....... 006AB300  R5....... 0068ED48  R6....... 00501A08  R7....... 00501664  R8....... 00501660
      +000038  R9....... 03A03D66  R10...... 03A02D67  R11...... 83A01D68  R12...... 00657E48  reserved. 0068B180
      +00004C  NAB...... 0066B490  PNAB..... 00000000  reserved. 00000000  00000000  00000000  00000000
      +000064  reserved. 00000000  reserved. 00000000  MODE..... 83A3FE6A  reserved. 00000000  00000000
      +000078  reserved. 00000000  reserved. 00000000
⋮
  STORAGE FOR ACTIVE ROUTINES:
    DSA frame: 0066B490
      +000000 0066B490  1066B4A4 0066B3F8 0066B5B0 8065AEEE  83A01D68 00000000 0066B5A0 0066B550  |...u...8........c..............&|
      +000020 0066B4B0  839BA1CE 006AB300 0066B534 00000000  00000000 00000005 00000000 806AB080  |c...............................|
      +000040 0066B4D0  806AB080 00657E48 0068B180 0066B5B0  00007FC8 0066B3F8 0066B590 8065B0BA  |......=..........."H...8........|
⋮
  CONTROL BLOCKS ASSOCIATED WITH THE THREAD:
    CAA: 00657E48
      +000000 00657E48  00000800 00000000 0066B000 0068B000  00000000 00000000 00000000 00000000  |................................|
      +000020 00657E68  00000000 00000000 0068B3C8 00000000  00000000 00000000 00000000 00000000  |...........H....................|
      +000040 00657E88  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
      +000060 00657EA8  00000000 00000000 00000000 00000000  00000000 80204A90 00000000 00000000  |................................|
      +000080 00657EC8  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
      +0000A0 00657EE8 - +00011F 00657F67             SAME AS ABOVE
⋮
```

Figure 34. Example Dump from Sample C Routine (Part 2 of 6)

```
    C CAA information :
    C Specific CTHD............ 0068E548
    C Specific CEDB............ 0068D028

    C Specific Thread block: 0068E548
      +000000 0068E548  C3E3C8C4 00000220 0068E548 00000000  03959768 00000000 00000000 00000000  |CTHD......V......np.............|
      +000020 0068E568  00000000 00000000 00000000 00000000  0068E770 00000000 00000000 00000000  |..................X.............|
      +000040 0068E588  00000000 00000000 00000000 00695838  00695920 00000000 00000000 00000000  |................................|
      +000060 0068E5A8  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
      +000080 0068E5C8 - +0000FF 0068E647            SAME AS ABOVE
⋮
    C Specific EDB block: 0068D028
      +000000 0068D028  C3C5C4C2 00000480 0068D028 00501A40  00030000 0068D610 0068D960 005000F8  |CEDB.........&. ......O...R-.&.8|
      +000020 0068D048  00000080 806A0300 00000000 00000000  00000000 0068D4B0 03963458 039622E0  |......................M..o...o..|
      +000040 0068D068  0068EC70 00000001 00000000 0068ED30  0069852C 006982AC 006983EC 0390D71A  |..................e...b...c...P.|
      +000060 0068D088  00000000 0068D5C0 40400000 00000000  00100000 00000000 00000000 00000000  |......N.  ......................|
⋮
    errno value................ 0
    memory file block chain..... 006999E8
    open FCB chain............. 00699860
    GTAB table................. 0068E8A8

    signal information :
    SIGFPE   :
    function pointer... 00500E58    WSA address...     N/A           function name... hsigfpe

    SIGTERM  :
    function pointer... 00500F90    WSA address...     N/A           function name... hsigterm

  ENCLAVE VARIABLES:
          DD:SYSIPT:>xcount
                           signed int                  0
          DD:SYSIPT:>st2   signed int                255
          DD:SYSIPT:>st1   signed int                 99

  ENCLAVE CONTROL BLOCKS:
    EDB: 00657060
      +000000 00657060  C3C5C5C5 C4C24040 C0000001 00657D08  006575F8 00000000 00000000 00000000  |CEEEDB  ......'....8............|
      +000020 00657080  006574B0 006574E0 0065A038 00656D40  00000000 80656008 00657180 00008000  |.............._ ......-.........|
      +000040 006570A0  00000000 00000000 00438000 00000000  00000000 00000000 00657120 006570B8  |................................|
      +000060 006570C0  00000000 00000000 00000000 00000000  00000000 00000000 03A892F8 00657E48  |.........................yk8..=.|
      +000080 006570E0  40000000 00000000 00000000 00000000  00000001 00000000 00000000 00000000  | ...............................|
    MEML: 00657D08
      +000000 00657D08  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
      +000020 00657D28  00000000 00000000 03A41338 00000000  0068D028 00000000 839B3B80 00000000  |.........u..............c.......|
      +000040 00657D48  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
      +000060 00657D68 - +00011F 00657E27            SAME AS ABOVE

    WSA address.................00000000

    atexit information :
    function pointer... 805010C8    WSA address...     N/A           function name... atf1

    fetch information :
    fetch pointer     : 0068EC70
    function pointer... 806AB080

  ENCLAVE STORAGE:
    Initial (User) Heap
      +000000 0068D000  C8C1D5C3 00657480 00657480 00000000  0068D000 0068ED68 00008000 00006298  |HANC............................q|
      +000020 0068D020  0068D000 00000488 C3C5C4C2 00000480  0068D028 00501A40 00030000 0068D610  |.......hCEDB.........&. ......O.|
      +000040 0068D040  0068D960 005000F8 00000080 806A0300  00000000 00000000 00000000 0068D4B0  |..R-.&.8......................M.|
⋮
```

Figure 34. Example Dump from Sample C Routine (Part 3 of 6)

```
    LE Anywhere Heap
      +000000 0069B000  C8C1D5C3 006574B0 006574B0 006574B0  0069B000 0069B908 00004000 000036F8  |HANC...................... ....8|
      +000020 0069B020  0069B000 000002D8 FB000000 FFFFFFFF  0000003C C3C5C5C4 E4D4D740 FFFFFFFF  |.......Q............CEEDUMP ....|
      +000040 0069B040  003C0015 E38889A2 4089A240 8140A281  94979385 4084A494 97000000 00000000  |....This is a sample dump.......|
      +000060 0069B060  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
      +000080 0069B080  00280013 F0F761F3 F061F0F1 40F37AF1  F37AF2F1 40D7D440 40404040 40404040  |....07/30/01 3:13:21 PM         |
      +0000A0 0069B0A0  40404040 40404040 40404040 00000014  0000000C 00000003 00000000 003C001C  |            ....................|
      +0000C0 0069B0C0  C9D5C6D6 D9D4C1E3 C9D6D540 C6D6D940  C5D5C3D3 C1E5C540 94818995 D77A0000  |INFORMATION FOR ENCLAVE mainP:..|
      +0000E0 0069B0E0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000003F2  |...............................3|
      +000100 0069B100  003C0010 C5D5C3D3 C1E5C540 E2E3D6D9  C1C7C57A C2D3D6C3 D2E27A40 E3C8C540  |....ENCLAVE STORAGE:BLOCKS: THE |
      +000120 0069B120  E3C8D9C5 C1C47AE3 C8D9C5C1 C47AC3E3  C9E5C540 D9D6E4E3 C9D5C5E2 7A000000  |THREAD:THREAD:CTIVE ROUTINES:...|
      +000140 0069B140  0000008E 003C003C D3C540C1 95A8A688  85998540 C8858197 40404040 40404040  |........LE Anywhere Heap        |
:
    LE Below Heap
      +000000 0068B000  C8C1D5C3 006574E0 006574E0 006574E0  8068B000 0068B540 00002000 00001AC0  |HANC................... ........|
      +000020 0068B020  0068B000 00000040 C8E7E2E3 001FEFF0  001FE2E8 00658800 00000000 00000000  |....... HXST...0..SY..h.........|
      +000040 0068B040  0068B068 0068B068 0001000A 0068B0F0  0068B0F0 0001000A 00000000 00000000  |...............0...0............|
      +000060 0068B060  0068B000 00000088 00000000 00000000  001FEFF0 00658800 00000009 00000000  |.......h...........0..h.........|
:
    Additional Heap, heapid = 0069B8D4
      +000000 006AC000  C8C1D5C3 0069B8D4 0069B8D4 0069B8D4  006AC000 006AC3B0 000003E8 00000038  |HANC...M...M...M......C....Y....|
      +000020 006AC020  006AC000 00000048 00000000 006AB124  006AB080 006AC088 0068EC70 40F11000  |.......................h.... 1..|
      +000040 006AC040  00000003 006AB2EC 00020000 006AC070  00000000 006AB080 006AB080 00000000  |................................|
      +000060 006AC060  006AB1DC 006AB134 006AC000 00000018  0009C4C4 7AE2E8E2 C9D7E300 00000000  |..................DD:SYSIPT.....|
:
  FILE STATUS AND ATTRIBUTES:
    File Control Block: 00699860
      +000000 00699860  00699BBD 00000000 000003DB 00699930  00699950 00000000 00699988 0000002C  |..............r...r&......rh....|
      +000020 00699880  00000030 00000000 00000000 00698680  00000000 00699860 00000000 00000000  |..............f.......q-........|
      +000040 006998A0  00000000 FFFFFFFF 00080055 006999C0  00699980 03925030 03927818 03927A68  |..............r...r..k&..k...k:.|
      +000060 006998C0  03927E98 03915BA8 00000000 00000400  00000400 00000000 00000000 00000400  |.k=q.j$y........................|
      +000080 006998E0  00699B98 00699B98 00000000 00000000  00000000 00000000 00000000 00000000  |...q...q........................|
      +0000A0 00699900  00000000 00000000 00000000 00000000  0391CE70 00000000 00000000 00000000  |.................j..............|
      +0000C0 00699920  43020008 80001080 00000000 00000000  58FF0008 07FF0000 0391C9A0 00000000  |.........................jI.....|
      +0000E0 00699940  00000000 00000000 00000000 00000000  58FF0008 07FF0000 039236B8 00000000  |.........................k......|
      +000100 00699960  00000000 00000000 00000000 00000000  01000000 00000000 00000000 00000000  |................................|
    fldata FOR FILE:   CDMPPR1.MEMORY.DATA
    __recfmF:1........ 1
    __recfmV:1........ 0
    __recfmU:1........ 0
    __recfmS:1........ 0
    __recfmBlk:1...... 0
    __recfmASA:1...... 0
    __recfmM:1........ 0
    __recfmPO:1....... 0
    __dsorgPDSmem:1... 0
    __dsorgPS:1....... 0
    __dsorgMem:1...... 1
    __dsorgTemp:1..... 0
    __dsorgVSAM:1..... 0
    __openmode:2...... 1
    __modeflag:4...... 2
    __device.......... 8
    __blksize......... 1024
    __maxreclen....... 1024
    __dsname.......... CDMPPR1.MEMORY.DATA

    FILE pointer........ 0069984C

    Buffer at current file position: 00699B98
      +000000 00699B98  A2969485 408481A3 81A29694 85409496  99854084 81A38185 A5859540 94969985  |some datasome more dataeven more|
      +000020 00699BB8  408481A3 81000000 00000000 00000000  00000000 00000000 00000000 00000000  | data...........................|
      +000040 00699BD8  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
      +000060 00699BF8 - +0003FF 00699F97             SAME AS ABOVE
    Saved Buffer........ NULL
```

Figure 34. Example Dump from Sample C Routine (Part 4 of 6)

```
first data space.... NULL

File Control Block: 00698680
  +000000 00698680  00695BE0 00000000 00000050 00698750  00698770 80000000 006987A8 0000002C  |..$........&..g&..g.......gy....|
  +000020 006986A0  00000030 00000000 00000000 00698170  00699860 00698680 00000000 E3D4D7F0  |..............a...q-..f.....TMP0|
  +000040 006986C0  F0F0F140 FFFFFFFF 0000002A 006987E0  006987A0 03B54700 03B53BD0 03B572A8  |001 ..........g...g............y|
  +000060 006986E0  03B569D0 03B57D78 00000000 00000050  00000050 00000000 00000000 00000051  |..............&...&.........169.|
  +000080 00698700  00695BE0 00695BE0 00695BE0 00000000  00000000 00000000 00000000 00000000  |..$...$...$.....................|
  +0000A0 00698720  00000000 00000000 00000000 00000000  03B2CC68 00000000 00000000 00000000  |................................|
  +0000C0 00698740  43520020 80000880 00000000 00000000  58FF0008 07FF0000 03B2C798 00000000  |..........................Gq....|
  +0000E0 00698760  00000000 00000000 00000000 00000000  58FF0008 07FF0000 03B509F0 00000000  |...........................0....|
  +000100 00698780  00000000 00000000 00000000 00000000  01000000 00000000 00000000 00000000  |................................|

fldata FOR FILE:   DD:TMP0001
__recfmF:1........ 1
__recfmV:1........ 0
__recfmU:1........ 0
__recfmS:1........ 0
__recfmBlk:1...... 0
__recfmASA:1...... 0
__recfmM:1........ 0
__recfmPO:1....... 0
__dsorgPDSmem:1... 0
__dsorgPS:1....... 1
__dsorgMem:1...... 0
__dsorgTemp:1..... 0
__dsorgVSAM:1..... 0
__openmode:2...... 0
__modeflag:4...... 2
__device.......... 0
__blksize......... 80
__maxreclen....... 80
__dsname.......... MYFILE.DATA

FILE pointer........ 0069866C
ddname.............. TMP0001

Buffer at current file position: 00695BE0
  +000000 00695BE0  99858396 998440F3 40404040 40404040  40404040 40404040 40404040 40404040  |record 3                        |
  +000020 00695C00  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  |                                |
  +000040 00695C20  40404040 40404040 40404040 40404040  40000000 00000000 000023C8 00000000  |                 ..........H....|
Saved Buffer........ NULL

DTF: 006A1818
  +000000 006A1818  00509204 0C000162 006A1300 00000000  0013C740 2064E3D4 D7F0F0F0 F10E0000  |.&k...............G ..TMP0001...|
  +000020 006A1838  00000000 0022000E 0050007E 0000007E  0000007F 000E0000 007E0000 01000050  |.........&.=...=...".....=.....&|
  +000040 006A1858  206A11C0 806A120E 076A184E 40000006  316A1850 40000005 086A1868 00000000  |...........+ ......& ...........|
  +000060 006A1878  036A1878 20000001 056A1880 20000001  316A1850 40000005 086A1888 00000000  |...................& ......h....|
  +000080 006A1898  1E6A18A8 30000008 126A18A8 00000008  00000000 01000000 9E000000 00001000  |...y.......y....................|
  +0000A0 006A18B8  00001000 00010A0A 00000000 00695B7C  00000000 00000000 00000000 00000000  |..............$@................|
  +0000C0 006A18D8  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
  +0000E0 006A18F8 - +00011F 006A1937             SAME AS ABOVE

__amrc_type structure: 0066C700
  +000000 0066C700  0013C740 00000000 00000005 00000000  00000000 00000000 00000000 00000000  |..G ............................|
  +000020 0066C720  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
  +000040 0066C740 - +0000BF 0066C7BF             SAME AS ABOVE
  +0000C0 0066C7C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000000DC  |................................|
amrc __code union fields
__error................. 1296192(13C740)
__abend.__syscode....... 19(13)
__abend.__rc............ 51008(C740)
__feedback.rc........... 19(13)
__feedback.__ftncd...... 199(C7)
__feedback.__fdbk....... 64(40)
__alloc.__svc99_info.... 19(13)
__alloc.__svc99_error... 51008(C740)
```

Figure 34. Example Dump from Sample C Routine (Part 5 of 6)

```
    __RBA.............. 0(0)
    __last_op........... 5(5)
    __msg.__str......... NULL
    __msg.__parmr0...... 0(0)
    __msg.__parmr1...... 0(0)
    __msg.__str2........ NULL

    __amrc2_type structure: 0066C600
      +000000 0066C600  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
    __error2............ 0(0)
    __fileptr........... NULL

PROCESS CONTROL BLOCKS:

  PCB: 00656D40
    +000000 00656D40  C3C5C5D7 C3C24040 04030290 00000000  00000000 00000000 00656F38 03A8A6B0  |CEEPCB  ..................?..yw.|
    +000020 00656D60  03A86838 03A89690 03A891C8 006469C8  00656118 00000000 00000000 00657060  |.y...yo..yjH...H../............-|
    +000040 00656D80  03A894E8 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |.ymY............................|

  MEML: 00656F38
    +000000 00656F38  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
    +000020 00656F58  00000000 00000000 03A41338 00000000  039B3B80 0068DE50 839B3B80 00000000  |.........u.............&c.......|
    +000040 00656F78  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
    +000060 00656F98 - +00011F 00657057            SAME AS ABOVE
```

Figure 34. Example Dump from Sample C Routine (Part 6 of 6)

Finding C Information in an LE/VSE Dump

When an LE/VSE traceback or dump is generated for a C routine, information is provided that is unique to C routines. C-specific information includes:

- Control block information for active routines
- Condition information for active routines
- Enclave level data

This information is located in the sections of the dump described below.

Storage for Active Routines

The Storage for Active Routines section of the dump shows the DSAs for the active C routines. To relate a DSA frame to a particular function name, use the address associated with the frame to find the corresponding DSA. Figure 35 on page 87 shows this section of the dump. In this example, the function `func1` DSA address is X'0066B3F8'.

```
⋮
  CONTROL BLOCKS FOR ACTIVE ROUTINES:
⋮
    DSA FOR func1: 0066B3F8
      +000000  FLAGS.... 1000      member... 0000      BKC...... 0066B348  FWC...... 0066B4E0  R14...... 806AB0EC
      +000010  R15...... 039BA180  R0....... 0066B490  R1....... 0066B488  R2....... 0066B3F8  R3....... 806AB0CE
      +000024  R4....... 006AB300  R5....... 0068ED48  R6....... 00501A08  R7....... 00501664  R8....... 00501660
      +000038  R9....... 03A03D66  R10...... 03A02D67  R11...... 83A01D68  R12...... 00657E48  reserved. 0068B180
      +00004C  NAB...... 0066B490  PNAB..... 00000000  reserved. 00000000  00000000  00000000  00000000
      +000064  reserved. 00000000  reserved. 00000000  MODE..... 83A3FE6A  reserved. 00000000  00000000
      +000078  reserved. 00000000  reserved. 00000000
⋮
  STORAGE FOR ACTIVE ROUTINES:
    DSA frame: 0066B490
      +000000 0066B490  1066B4A4 0066B3F8 0066B5B0 8065AEEE  83A01D68 00000000 0066B5A0 0066B550  |...u...8........c..............&|
      +000020 0066B4B0  839BA1CE 006AB300 0066B534 00000000  00000000 00000005 00000000 806AB080  |c...............................|
      +000040 0066B4D0  806AB080 00657E48 0068B180 0066B5B0  00007FC8 0066B3F8 0066B590 8065B0BA  |......=..........."H...8........|
⋮
```

Figure 35. Storage for Active Routines Section of a Dump

Control Blocks Associated with the Active Thread

In the Control Blocks associated with the Thread section of the dump, the following information appears:

- C fields from the CAA
- C Specific CAA
- Signal information

C Fields of the CAA

The CAA contains several fields that the C programmer can use to find information about the run-time environment. This information is dumped as shown in Figure 36. An explanation of each field follows.

```
    C CAA information :
    C Specific CTHD............. 0068E548
    C Specific CEDB............. 0068D028
⋮
    WSA address.................00000000
```

Figure 36. C Fields in the CAA

C Specific CTHD and CEDB
: For each C program, there is a C Specific Thread area and a C Specific Enclave area.

WSA address
: This is available for all C programs except those compiled with the NORENT compile-time option. It points to the base address of the writable static area.

C Specific CAA

The C-specific CAA fields that are of interest to users are shown in Figure 37.

```
    errno value................ 0
    memory file block chain..... 006999E8
    open FCB chain............. 00699860
```

Figure 37. C-Specific CAA Fields

`errno` value
: A variable used to display error information. Its value can be set to a positive number that corresponds to an error message. The functions `perror()` and `strerror()` print the error message that corresponds to the value of `errno`.

Memory file control block
: You can use the *memory file control block* (MFCB) to locate additional information about memory files. This control block resides at the C thread level. See "Memory File Control Block" on page 91 for more information about the MFCB.

Open FCB chain
: A pointer to the start of a linked list of open file control blocks (FCBs). For more information about FCBs, see "File Control Block Information" on page 89.

Signal Information

Signal information is provided in the dump to aid you in debugging. For each signal that is disabled with `SIG_IGNORE`, an entry value of 00000001 is made in the first field of the Signal Information field for the specified signal name.

For each signal that has a handler registered, the signal name and the handler name are listed. If the handler is a fetched C function, the value `@@FECB` is entered as the function name and the address of the fetched pointer is in the first field.

If you compile a C routine as NORENT, the WSA address is not available (`N/A`). See *LE/VSE C Run-Time Programming Guide* for more information about the `signal()` function.

atexit() Information

The `atexit()` information lists the functions registered with `atexit()` to be run at normal termination. The functions are listed in chronological order of registration.

If you compile a C routine as NORENT, the WSA address is not available (`N/A`). See *LE/VSE C Run-Time Library Reference* for more information about the `atexit()` function.

Figure 38 on page 89 shows `atexit()` information in the dump.

```
  ⋮
   atexit information :
   function pointer... 805010C8    WSA address...     N/A          function name... atf1
  ⋮
```

Figure 38. atexit() Information in Dump

fetch Information

The `fetch` information shows information about phases that you have dynamically loaded using `fetch()`. For each phase that was fetched, the `fetch` (stub) pointer (`ptr1` in the following example) and the function pointer addresses are included.

```
ptr1 = fetch("MOD");
```

If you compile a C routine as NORENT, the WSA address is not available (`N/A`). See *LE/VSE C Run-Time Programming Guide* for more information about the `fetch` function.

Figure 39 shows `fetch` information in the dump.

```
⋮
    fetch information :
    fetch pointer     : 0068EC70
    function pointer... 806AB080
⋮
```

Figure 39. fetch Information in Dump

File Control Block Information

This section of the dump includes the file control block (FCB) information for each C file. The FCB contains file status and attributes for files open during C active routines. You can use this information to find the data set or file name.

The FCB is a handle that points to the following file information, which is displayed when applicable, for the file:
- Access method control block (ACB) address
- Define the file (DTF) address
- RPL address
- Current buffer address
- Saved buffer address
- filename (appears in the listing as `ddname`)
- file ID (appears in the listing as `__dsname`)

Figure 40 on page 90 shows some of the fields that are dumped for a non-VSAM file.

```
 FILE STATUS AND ATTRIBUTES:
   File Control Block: 01CB92C0
     +000000 01CB92C0  00771CE9 00000000 00000085 01CB9390  01CB93B0 80000000 01CB9870 00000009  |...Z.......e..l...l.......q.....|
     +000020 01CB92E0  00000030 E2E8E2D3 E2E30000 01CB9680  00000000 01CB92C0 00000000 00000000  |....SYSLST....o.......k.........|
   ⋮

   fldata FOR FILE:   dd:SYSLST
   __recfmF:1........ 1
   __recfmV:1........ 0
   __recfmU:1........ 0
   ⋮
   __dsname..........

   FILE pointer........ 01CB92AC

   Buffer at current file position: 00771CE9
     +000000 00771CE9  E3C5E2E3 40D9C5C3 F240C6D9 D6D440C3  E3C5E2E3 F6404040 40404040 40404040  |TEST REC2 FROM CTEST6           |
     +000020 00771D09  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  |                                |
     +000040 00771D29 - +00007F 00771D68             SAME AS ABOVE
     +000080 00771D69  40404040 40000000 00229000 00000000  00000000 00000000 00000000 00000000  |     ...........................|
   Saved Buffer........ NULL

   DTF: 00777818
     +000000 00777818  00008000 0C000003 00744D90 00744DA0  011187A0 08B20909 00777851 00000000  |..........(...(...g........é....|
     +000020 00777838  07004700 0000000C 09777851 20000085  80000000 00000000 00E3C5E2 E340D9C5  |...........é...e.........TEST RE|
```

Figure 40. File Control Block Information for a non-VSAM File

Figure 41 shows some of the fields that are dumped for a VSAM data set.

```
   File Control Block: 01CB9680
     +000000 01CB9680  01CB9807 00000002 00000000 01CB9750  01CB9770 80000000 01CB97A8 0000002C  |..q...........p&..p.......py....|
     +000020 01CB96A0  00000030 00000000 00000000 01CB9170  01CB92C0 01CB9680 00000000 C9D5C6C9  |..............j...k...o.....INFI|
   ⋮

   fldata FOR FILE:   dd:INFILE
   __recfmF:1........ 0
   __recfmV:1........ 0
   __recfmU:1........ 1
   ⋮
   __dsname.......... RTURNER.CTEST6.ESDS

   FILE pointer........ 01CB966C
   ddname............. INFILE

   Buffer at current file position: 01CB97E0
     +000000 01CB97E0  E3C5E2E3 40D9C5C3 F240C6D9 D6D440C3  E3C5E2E3 F6404040 40404040 40404040  |TEST REC2 FROM CTEST6           |
     +000020 01CB9800  40404040 40404040 15000000 00000000  00000000 00000000 00000000 00000000  |        ........................|
     +000040 01CB9820  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
     +000060 01CB9840 - +00007F 01CB985F             SAME AS ABOVE
     +000080 01CB9860  00000000 00000000 80000038 01C 9000  C4C47AE2 E8E2D3E2 E3000000 00000000  |................DD:SYSLST.......|
   Saved Buffer........ NULL

   Access Method Control Block: 00771BC0
     +000000 00771BC0  A040004C 00791618 001B00D8 00000000  00005C00 28100100 00000000 C9D5C6C9  |. .<.......Q......*.........INFI|
     +000020 00771BE0  D3C54040 0074F898 00000000 00777C00  00000000 00000000 00080004 00771834  |LE  ..8q......@.................|
     +000040 00771C00  00000000 80000000 00000000 00000000  00100034 00000028 00771B6C 01CB97E0  |...........................%..p.|
   Read Request Parameter List: 00771C10
     +000000 00771C10  00100034 00000028 00771B6C 01CB97E0  00000028 00000088 00771BC0 01040000  |...........%..p........h........|
     +000020 00771C30  60100000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |-...............................|
   Write Request Parameter List: 00771C48
     +000000 00771C48  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
     +000020 00771C68  00000000 00000000 00000000 00000000  00000000 00000000 80000060 00771000  |...........................-....|
```

Figure 41. File Control Block Information for a-VSAM File

Not all FCB fields are always filled in. For example:
- RPLs are used only for VSAM data sets.
- The `ddname` (filename) field contains blanks if it is not used.
- DTFs are not used for memory files.

The save block buffer represents auxiliary buffers that are used to save the contents of the main buffers. Such saving occurs only when a reposition is performed and there is new data; for example, an incomplete text record or an incomplete fixed-block standard (FBS) block in the buffers that cannot be flushed out of the system. Since the main buffers represent the current position in the file, while the save buffers merely indicate a save has occurred, check the save buffers only if data appears to be missing from the external device and is not found in the main buffers. Also, do not infer that the presence of save buffers means that data present there belongs at the end of the file. (The buffers remain, even when the data is eventually written).

Memory File Control Block

This section of the dump, shown in Figure 42, holds the memory file control block information for each *closed* memory file the routine used.

```
  Memory File Control Block: 01CBDB58
    +000000 01CBDB58  01CBDBC8 00000000 00000000 00000000  00000000 00010000 01CBDCD0 0000002C  |...H............................|
    +000020 01CBDB78  00000030 00000000 00000000 01CBD170  01CBD680 01CBDB58 00000000 01CBD808  |..............J...O...........Q.|
    +000040 01CBDB98  01CBDB58 00000000 00000000 00000000  00000000 00000000 00000000 01CBDBC8  |...............................H|
    +000060 01CBDBB8  00010000 00000000 00000000 00000000  00000000 00000000 01CBE110 00000026  |................................|
  memory file name......... CTEST6.MEMORY.FILE.TWO
```

Figure 42. Memory File Control Block

Information for __amrc

`__amrc` is a structure defined in the `stdio.h` header file to assist in determining errors resulting from I/O operations. The contents of `__amrc` (shown in Figure 43) can be checked for system information, such as the return code for VSAM. Certain fields of the `__amrc` structure can provide useful information about what occurred previously in your routine.

```
__amrc_type structure: 000763CC
  +000000 000763CC  00000004 00000000 00000007 00000000  00000000 00000000 00000000 00000000  |................................|
  +000020 000763EC  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
  +000040 0007640C - +0000BF 0007648B             same as above
  +0000C0 0007648C  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000000DC  |................................|
amrc __code union fields
__error................. 4(4)
__abend.__syscode....... 0(0)
__abend.__rc............ 4(4)
__feedback.rc........... 0(0)
__feedback.__ftncd...... 0(0)
__feedback.__fdbk....... 4(4)
__alloc.__svc99_info.... 0(0)
__alloc.__svc99_error... 4(4)

__RBA............... 0(0)
__last_op........... 7(7)
__msg.__str......... NULL
__msg.__parmr0...... 0(0)
__msg.__parmr1...... 0(0)
__msg.__str2........ NULL

__amrc2_type structure: 000762D0
  +000000 000762D0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
__error2............ 0(0)
__fileptr........... NULL
```

Figure 43. __amrc Structure Information in Dump

For more information about `__amrc`, see “Debugging C Input/Output Programs” on page 63 and *LE/VSE C Run-Time Programming Guide*.

C Contents of the LE/VSE Trace Table

LE/VSE provides two C trace table entry types that contain character data:
- Trace entry 1 occurs when a base C library function is called.
- Trace entry 2 occurs when a base C library function returns.

The format for trace table entry 1 is:

```
—>(xxx) NameOfCallingFunction NameOfCalledFunction
```

The format for trace table entry 2 is:

```
<—(xxx) R15=value ERRNO=value
```

In both entry types, `(xxx)` is a number associated with the called library function and is used to associate a specific entry record with its corresponding return record.

For more information about the LE/VSE trace table format, see "Understanding the Trace Table Entry (TTE)" on page 55.

Debugging Example of C Routines

This section contains examples that demonstrate the debugging process for C routines. Important areas of the output are highlighted. Data unnecessary to the debugging examples has been replaced by ellipses.

Note: From LE/VSE 1.4.2 onwards, dumps taken with run-time option `TER(UADUMP...)` or `TER(DUMP...)` are capable of generating a run-time options report, *independently of* the setting for `RPTOPTS(ON|OFF)`. For an example, see Figure 51 on page 98.

Divide-by-Zero Error

Figure 44 on page 93 illustrates a C program that contains a divide-by-zero error. The code was compiled with RENT so static and external variables need to be calculated from the WSA field. The code was compiled with XREF, LIST and OFFSET to generate a listing, which is used to calculate addresses of functions and data. The code was prelinked with MAP to generate a prelinker map, which is used to calculate addresses of static and external variables. This routine was run with the option TERMTHDACT(DUMP).

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
int statint = 73;
void funcb(int *pp);

int main(void) {
  int aa, bb=1;
  aa = bb;
  funcb(&aa);
  return(99);
}

void funcb(int *pp) {
  int fa, result;
  fa = *pp;
  result = fa/(statint-73);
  return;
}
```

Figure 44. C Routine with a Divide-by-Zero Error

To debug this routine, use the following steps:

1. Locate the Current Condition message in the Condition Information for Active Routines section of the dump. In this example, the message is `CEE3209S The system detected a Fixed Point divide exception`. This message indicates the error was caused by an attempt to divide by zero. See Chapter 8, "LE/VSE Run-Time Messages," on page 163 for additional information about CEE3209S.

   The traceback section of the dump indicates that the exception occurred at offset X'00000084' within function `funcb`. This information is used along with the compiler-generated Pseudo Assembly Listing to determine where the problem occurred.

   If the TEST compile-time option is specified, variable information is in the dump. If the GONUMBER compile-time option is specified, statement number information is in the dump. Figure 45 on page 94 shows the generated traceback from the dump.

```
CEE5DMP V1 R4.2: CONDITION PROCESSING RESULTED IN THE UNHANDLED CONDITION.        07/27/01 1:22:59 PM                  PAGE:    1

INFORMATION FOR ENCLAVE main

  INFORMATION FOR THREAD 8000000000000000

  TRACEBACK:
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY         E ADDR    E  OFFSET   STATEMENT  STATUS
    00668018  CEEHDSP       03A58D38  +000029DA  CEEHDSP       03A58D38  +000029DA              CALL
    0066B288  DD:SYSIPT     00500358  +00000084  funcb         00500358  +00000084          18  EXCEPTION
    0066B1E8  DD:SYSIPT     005000F8  +00000078  main          005000F8  +00000078          11  CALL
    0066B0D8                039BF1E6  +000000B4  @@MNINV       039BF1E6  +000000B4              CALL
    0066B018  CEEBBEXT      001FA3E0  +00000132  CEEBBEXT      001FA3E0  +00000132              CALL

  CONDITION INFORMATION FOR ACTIVE ROUTINES
    CONDITION INFORMATION FOR DD:SYSIPT (DSA ADDRESS 0066B288)
      CIB ADDRESS: 00668528
      CURRENT CONDITION:
        CEE0198S THE TERMINATION OF A THREAD WAS SIGNALED DUE TO AN UNHANDLED CONDITION.
      ORIGINAL CONDITION:
        CEE3209S THE SYSTEM DETECTED A FIXED-POINT DIVIDE EXCEPTION.
      LOCATION:
        PROGRAM UNIT: DD:SYSIPT ENTRY: funcb STATEMENT: 18 OFFSET: +00000084
      MACHINE STATE:
        ILC..... 0002    INTERRUPTION CODE..... 0009
        PSW..... 071D2400 805003DE
        GPR0..... 0066B320  GPR1..... 0066B280  GPR2..... 0066B288  GPR3..... 805003A6
        GPR4..... 00000000  GPR5..... 00000001  GPR6..... 00000000  GPR7..... 00500554
        GPR8..... 00500550  GPR9..... 80000000  GPR10.... 839BF1DA  GPR11.... 801FA3E0
        GPR12.... 00657E48  GPR13.... 0066B288  GPR14.... 80500172  GPR15.... 0068B180
    STORAGE DUMP NEAR CONDITION, BEGINNING AT LOCATION: 005003CC
      +000000 005003CC  58606000 4B603066 5840D090 8E400020  1D465050 D0944400 C1AC4400 C1DC47F0  |.--..-... ... ....&&.m..A...A..0|

  PARAMETERS, REGISTERS, AND VARIABLES FOR ACTIVE ROUTINES:
    CEEHDSP (DSA ADDRESS 00668018):
      SAVED REGISTERS:
        GPR0..... 00000000  GPR1..... 006683BC  GPR2..... 00000008  GPR3..... 006535E0
        GPR4..... 00000003  GPR5..... 0065A038  GPR6..... 03A5CD34  GPR7..... 00669017
        GPR8..... 03A5BD35  GPR9..... 03A5AD36  GPR10.... 03A59D37  GPR11.... 83A58D38
        GPR12.... 00657E48  GPR13.... 00668018  GPR14.... 8065AEEE  GPR15.... 83A01D68
⋮
```

Figure 45. Dump Sections from C Example of Divide-by-Zero Error

2. Locate the instruction with the divide-by-zero error in the Pseudo Assembly Listing in Figure 46 on page 95.

   The offset (within `funcb` in Figure 46 on page 95) of the exception from the traceback (X'00000084') reveals the divide instruction: `DR     r4,r6` at that location. Instructions X'00006C' through X'000086' refer to the `result = fa/(statint-73);` line of the C routine.

```
000000                        110  funcb    DS    0F
000000  47F0  F026            111           B     38(,r15)
00001C  41E0  F03A            112           LA    r14,58(,r15)
000020  58F0  C074            113           L     r15,116(,r12)
000024  07FF                  114           BR    r15
000026  90E6  D00C            115           STM   r14,r6,12(r13)
00002A  5820  D04C            116           L     r2,76(,r13)
00002E  4100  2098            117           LA    r0,152(,r2)
000032  5500  C00C            118           CL    r0,12(,r12)
000036  4720  F01C            119           BH    28(,r15)
00003A  58F0  D048            120           L     r15,72(,r13)
00003E  90F0  2048            125           STM   r15,r0,72(r2)
000042  9210  2000            126           MVI   0(r2),16
000046  50D0  2004            127           ST    r13,4(,r2)
00004A  18D2                  128           LR    r13,r2
00004C  0530                  129           BALR  r3,r0
00004E                     End of Prolog
00004E  5010  D088            131           ST    r1,136(,r13)
                                     *
                    00015 |          *  void funcb(int *pp) {
000052  4400  C1B0            134           EX    r0,HOOK..PGM-ENTRY
                                     *    int fa, result;
                    00017 |          *    fa = *pp;
000056  4400  C1AC            137           EX    r0,HOOK..STMT
00005A  5840  D088            138           L     r4,136(,r13)
00005E  5840  4000            139           L     r4,0(,r4)
000062  D203  D090  4000      140           MVC   144(4,r13),0(r4)
                    00018 |          *    result = fa/(statint-73);
000068  4400  C1AC            142           EX    r0,HOOK..STMT
00006C  5860  ****            143           L     r6,=Q(statint)
000070  5A60  C1F4            144           A     r6,500(,r12)
000074  5860  6000            145           L     r6,0(,r6)
000078  4B60  ****            146           SH    r6,=H'73'
00007C  5840  D090            147           L     r4,144(,r13)
000080  8E40  0020            148           SRDA  r4,32
000084  1D46                  149           DR    r4,r6
000086  5050  D094            150           ST    r5,148(,r13)
                    00019 |          *    return;
00008A  4400  C1AC            152           EX    r0,HOOK..STMT
00008E  4400  C1DC            153           EX    r0,HOOK..GOTO
000092  47F0  ****            154           B     @5L2
000096  ****  ****            155           DC    A(@5L2-ep)
                    00020 |          *  }
00009A                        157 @5L2      DS    0H
00009A                     Start of Epilog
```

Figure 46. Pseudo Assembly Listing

3. Verify the value of the divisor `statint`. The procedure specified below is to be used for determining the value of static variables only. If the divisor is an automatic variable, there is a different procedure for finding the value of the variable. See “Finding Automatic Variables” on page 69 for more information about finding automatic variables in a dump.

   Because this routine was compiled with the RENT option, find the WSA address in the C CAA information section of the dump. In this example this address is X'0068EA18'. Figure 47 on page 96 shows the WSA address.

```
⋮
   C CAA information :
   C Specific CTHD............ 0068E548
   C Specific CEDB............ 0068D028
⋮
   WSA address.................0068EA18
⋮
```

Figure 47. C CAA Information in Dump

4. Routines compiled with the RENT option must also be prelinked. The prelinker produces the Writable Static Map. Find the offset of `statint` in the Writable Static Map in Figure 48. In this example, the offset is X'00000000'.

```
   ⋮
=======================================================================
|                         Writable Static Map                         |
=======================================================================

  OFFSET    LENGTH  FILE ID  INPUT NAME

       0         4   00001   statint
   ⋮
```

Figure 48. Writable Static Map

5. Add the WSA address of X'0068EA18' to the offset of `statint` (X'00000000'). The result is X'0068EA18'. This is the address of the variable `statint`, which is in the writable static area. The writable static area is storage allocated by the C run time for the C user, so it is in the user heap. The heap content is displayed in the Enclave Storage section of the dump, shown in Figure 49.
6. To find the variable `statint` in the heap, locate the closest address listed that is at or before the address of `statint`. In this case, that address is X'0068EA18'. The value at that location is X'49' (that is, `statint` is 73), and hence the fixed point divide exception.

```
⋮
  ENCLAVE STORAGE:
    Initial (User) Heap
      +000000 0068D000  C8C1D5C3 00657480 00657480 00000000  0068D000 0068ED50 00008000 000062B0  |HANC..................&........|
⋮
      +000380 0068D380  00000000 00000000 00000000 00000000  00000000 00000000 00000000 0066C37C  |.............................C@|
      +0003A0 0068D3A0  00000000 00000000 00000000 00000000  0068EA18 00657180 839B3B80 039B4348  |.......................c.......|
      +0003C0 0068D3C0  00000000 00657E48 00695948 00000008  00000000 00000000 00040000 00698170  |......=.......................a.|
⋮
```

Figure 49. Enclave Storage Section of Dump

Calling a Nonexistent Function

Figure 50 on page 97 demonstrates the error of calling a nonexistent function. This routine was compiled with the compile-time options LIST and RENT and was run with the option TERMTHDACT(DUMP).

This routine was not compiled with the TEST(ALL) compile option. As a result, arguments and variables do not appear in the dump.

The only prelinker option used was MAP.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <signal.h>
void funca(int* aa);
int (*func_ptr)(void)=0;
int main(void) {
  int aa;
  funca(&aa);
  printf("result of funca = %d\n",aa);
  return;
}
void funca(int* aa) {
  *aa = func_ptr();
  return;
}
```

Figure 50. C Example of Calling a Nonexistent Subroutine

To debug this routine, use the following steps:

1. Locate the Current Condition message in the Condition Information for Active Routines section of the dump, shown in Figure 51 on page 98. In this example, the message is `CEE3201S The system detected an operations exception.` This message suggests that the error was caused by an attempt to branch to an unknown address. See Chapter 8, “LE/VSE Run-Time Messages,” on page 163 for additional information about CEE3201S.

   The traceback section of the dump indicates that the exception occurred at offset X'00000000' within function `funca` . The zero offset indicates that the offset cannot be used to locate the instruction that caused the error. Another indication of bad data is the value of X'80000002' in the instruction address of the PSW. This address indicates that an instruction in the routine branched outside the bounds of the routine.

```
CEE5DMP V1 R4.2: CONDITION PROCESSING RESULTED IN THE UNHANDLED CONDITION.   07/27/01 1:25:18 PM                 PAGE:    1

INFORMATION FOR ENCLAVE main
  INFORMATION FOR THREAD 8000000000000000

  TRACEBACK:
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY         E ADDR    E  OFFSET   STATEMENT  STATUS
    00668018  CEEHDSP       03A58D38  +000029DA  CEEHDSP       03A58D38  +000029DA              CALL
    0066B288                005001B8  +00000000  funca         005001B8  +00000000              EXCEPTION
    0066B1E8                005000F8  +00000068  main          005000F8  +00000068              CALL
    0066B0D8                039BF1E6  +000000B4  @@MNINV       039BF1E6  +000000B4              CALL
    0066B018  CEEBBEXT      001FA3E0  +00000132  CEEBBEXT      001FA3E0  +00000132              CALL

  CONDITION INFORMATION FOR ACTIVE ROUTINES
    CONDITION INFORMATION FOR  (DSA ADDRESS 0066B288)
      CIB ADDRESS: 00668528
      CURRENT CONDITION:
        CEE0198S THE TERMINATION OF A THREAD WAS SIGNALED DUE TO AN UNHANDLED CONDITION.
      ORIGINAL CONDITION:
        CEE3201S THE SYSTEM DETECTED AN OPERATION EXCEPTION.
      LOCATION:
        PROGRAM UNIT:  ENTRY: funca STATEMENT:  OFFSET: +00000000
      MACHINE STATE:
        ILC..... 0002    INTERRUPTION CODE..... 0001
        PSW..... 071D1400 80000002
        GPR0..... 0066B318  GPR1..... 0066B280  GPR2..... 0066B288  GPR3..... 80500206
        GPR4..... 0068EA18  GPR5..... 0068EA20  GPR6..... 0066B278  GPR7..... 00000000
        GPR8..... 00500238  GPR9..... 80000000  GPR10.... 839BF1DA  GPR11.... 801FA3E0
        GPR12.... 00657E48  GPR13.... 0066B288  GPR14.... 80500220  GPR15.... 00000000
    STORAGE DUMP NEAR CONDITION, BEGINNING AT LOCATION: 00000000
      +000000 00000000  INACCESSIBLE STORAGE.

  PARAMETERS, REGISTERS, AND VARIABLES FOR ACTIVE ROUTINES:
    CEEHDSP (DSA ADDRESS 00668018):
      SAVED REGISTERS:
        GPR0..... 00000000  GPR1..... 006683BC  GPR2..... 00000008  GPR3..... 006535E0
        GPR4..... 00000003  GPR5..... 0065A038  GPR6..... 03A5CD34  GPR7..... 00669017
        GPR8..... 03A5BD35  GPR9..... 03A5AD36  GPR10.... 03A59D37  GPR11.... 83A58D38
        GPR12.... 00657E48  GPR13.... 00668018  GPR14.... 8065AEEE  GPR15.... 83A01D68
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (00000000)
        +0000 00000000    INACCESSIBLE STORAGE.
        +0020 00000020    INACCESSIBLE STORAGE.
⋮
```

Figure 51. Dump Sections from C Example of Calling a Non-Existent Function (Part 1 of 2)

```
 RUN-TIME OPTIONS REPORT ASSOCIATED WITH DUMP FOR ENCLAVE : main
  LAST WHERE SET          OPTION
  -------------------------------------------------------------------------------
  Installation default      ABPERC(NONE)
  Installation default      ABTERMENC(ABEND)
  Installation default    NOAIXBLD
  Installation default      ALL31(OFF)
  Installation default      ANYHEAP(16384,8192,ANYWHERE,FREE)
  Installation default      BELOWHEAP(8192,4096,FREE)
  Installation default      CBLOPTS(ON)
  Installation default      CBLPSHPOP(OFF)
  Installation default      CHECK(OFF)
  Non-overrideable          COUNTRY(US)
  Installation default    NODEBUG
  Installation default      DEPTHCONDLMT(10)
  Installation default      ENVAR("")
  Installation default      ERRCOUNT(20)
  Installation default      HEAP(32768,32768,ANYWHERE,KEEP,8192,4096)
  Installation default      HEAPCHK(OFF,1,0)
  Installation default      LIBSTACK(12288,4096,FREE)
  Installation default      MSGFILE(SYSLST)
  Installation default      MSGQ(15)
  Installation default      NATLANG(UEN)
  Installation default      RETZERO(OFF)
  Installation default      RPTOPTS(OFF)
  Installation default      RPTSTG(OFF)
  Installation default    NORTEREUS
  Installation default      STACK(131072,131072,BELOW,KEEP)
  Assembler user exit       STORAGE(00,NONE,NONE,32768)
  Invocation command        TERMTHDACT(DUMP,,96)
  Installation default    NOTEST(ALL,"*","PROMPT","")
  Installation default      TRACE(OFF,4096,DUMP,LE=0)
  Installation default      TRAP(ON,MAX)
  Installation default      UPSI(00000000)
  Installation default    NOUSRHDLR()
  Installation default      XUFLOW(AUTO)
```

Figure 51. Dump Sections from C Example of Calling a Non-Existent Function (Part 2 of 2)

2. Find the branch instructions for `funca` in the listing in Figure 52 on page 100. Notice the `BALR r14,r15` instruction at offset X'00000126'. This branch is part of the instruction `*aa = func_ptr();`.

```
0000C0                    91  funca    DS    0F
0000C0  47F0  F026        92           B     38(,r15)
0000DC  41E0  F03A        93           LA    r14,58(,r15)
0000E0  58F0  C074        94           L     r15,116(,r12)
0000E4  07FF              95           BR    r15
0000E6  90E7  D00C        96           STM   r14,r7,12(r13)
0000EA  5820  D04C        97           L     r2,76(,r13)
0000EE  4100  2090        98           LA    r0,144(,r2)
0000F2  5500  C00C        99           CL    r0,12(,r12)
0000F6  4720  F01C       100           BH    28(,r15)
0000FA  58F0  D048       101           L     r15,72(,r13)
0000FE  90F0  2048       102           STM   r15,r0,72(r2)
000102  9210  2000       103           MVI   0(r2),16
000106  50D0  2004       104           ST    r13,4(,r2)
00010A  18D2             105           LR    r13,r2
00010C  0530             106           BALR  r3,r0
00010E                End of Prolog
00010E  5840  C1F4       108           L     r4,500(,r12)
000112  5010  D088       109           ST    r1,136(,r13)
                                *   }
                  00013 |      *   void funca(int* aa) {
                  00014 |      *     *aa = func_ptr();
000116  5870  D088       113           L     r7,136(,r13)
00011A  5860  7000       114           L     r6,0(,r7)
00011E  5870  ****       115           L     r7,=Q(func_ptr)
000122  58F4  7000       116           L     r15,0(r4,r7)
000126  05EF             117           BALR  r14,r15
000128  50F0  6000       118           ST    r15,0(,r6)
                  00015 |      *     return;
```

Figure 52. Pseudo Assembly Listing

3. Find the offset of FUNC@PTR in the Writable Static Map, shown in Figure 53, as produced by the prelinker.

```
   ⋮
========================================================================
|                          Writable Static Map                         |
========================================================================

  OFFSET    LENGTH  FILE ID  INPUT NAME

       0         4   00001   func_ptr
       8        18   00001   @STATIC
   ⋮
```

Figure 53. Writable Static Map

4. Add the offset of FUNC@PTR (X'00000000') to the WSA address (X'0068EA18'). The result (X'0068EA18') is the address of the function pointer func_ptr in the writable static storage area within the heap. This value is 0, indicating the variable is uninitialized.

   Figure 54 on page 101 shows the sections of the CAA and heap from the dump.

```
⋮
   C CAA information :
   C Specific CTHD............. 0068E548
   C Specific CEDB............. 0068D028
⋮
   WSA address.................0068EA18

  ENCLAVE STORAGE:
   Initial (User) Heap
     +000000 0068D000  C8C1D5C3 00657480 00657480 00000000  0068D000 0068ED68 00008000 00006298  |HANC...........................q|
⋮
     +0019E0 0068E9E0  FFFFFFFF FFFFFFFF 03CB9694 03CB9694  03CB9694 03CB9694 03CB9640 E0BDADD0  |..........om..om..om..om..o ....|
     +001A00 0068EA00  C05FA15A 7B4F5B7C 79000000 00000000  0068D000 00000028 00000000 00000000  |.^.!#|$@........................|
     +001A20 0068EA20  9985A2A4 93A34096 864086A4 95838140  7E406C84 15000000 0068D000 00000148  |result of funca = %d............|
⋮
```

Figure 54. CAA and Enclave Storage Information in Dump

Chapter 5. Debugging COBOL Routines

This chapter provides information for debugging applications that contain one or more COBOL routines. The chapter includes information about:
- Determining the source of error
- Generating COBOL listings and the LE/VSE dump
- Finding COBOL information in a dump
- Debugging example for COBOL routines

Determining the Source of Error

The following sections describe tasks that you can perform to determine the source of error in your COBOL routine, and explain how the use of features such as the DISPLAY statement, *scope terminators*, declaratives, and file status keys can simplify the process of debugging COBOL routines. The following methods for determining errors are covered:
- Tracing routine logic
- Finding and handling input/output errors
- Validating data
- Assessing switch problems
- Generating information about procedures

Tracing Routine Logic

You can add DISPLAY statements to help you trace through the logic of the routine in a non-CICS environment. If, for example, you determine that the problem appears in an EVALUATE statement or in a set of nested IF statements, DISPLAY statements in each path tell you how the logic flows. You can also use DISPLAY statements to show you the value of interim results.

For example, to check logic flow, you might insert:

```
DISPLAY "ENTER CHECK PROCEDURE".
⋮
(checking procedure routine)
⋮
DISPLAY "FINISHED CHECK PROCEDURE".
```

to determine whether you started and finished a particular routine. After you are sure that the routine works correctly, put asterisks in position 7 of the DISPLAY statement lines to convert them to comment lines. See *IBM COBOL for VSE/ESA Language Reference* for a detailed description of the DISPLAY statement.

Scope terminators can also help you trace the logic of your routine because they clearly indicate the end of a statement. See *IBM COBOL for VSE/ESA Programming Guide* for a detailed description of scope terminators.

Note: Before you run the routine in production, delete all debugging aids and recompile the routine. The routine runs more efficiently and uses less storage.

Finding Input/Output Errors

VSAM file status keys can help you determine whether routine errors are due to the logic of your routine or are I/O errors occurring on the storage media.

To use file status keys as a debugging aid, include a test after each I/O statement to check for a value other than 0 in the status key. If the value is other than 0, you can expect to receive an error message. You can use a nonzero value as an indication to look at the way the I/O procedures in the routine were coded. You can also include procedures to correct the error based on the status key value.

The status key values and their associated meanings are described in *IBM COBOL for VSE/ESA Language Reference*

Handling Input/Output Errors

If you have determined that the problem lies in one of the I/O procedures in your routine, you can include the USE EXCEPTION/ERROR declarative to help debug the problem. If the file does not open, the appropriate EXCEPTION/ERROR declarative is activated. You can specify the appropriate declarative for the file or for the different open attributes—INPUT, OUTPUT, I/O, or EXTEND.

Code each USE AFTER STANDARD ERROR statement in a separate section. Code this section immediately after the Declarative Section keyword of the Procedure Division. See the rules for coding these statements in *IBM COBOL for VSE/ESA Language Reference*

Validating Data (Class Test)

If you suspect that your routine is trying to perform arithmetic on nonnumeric data or is somehow receiving the wrong type of data on an input record, you can use the class test to validate the type of data. See *IBM COBOL for VSE/ESA Programming Guide* for a detailed discussion of how to use the class test to check for incompatible data.

Assessing Switch Problems

Using INITIALIZE or SET statements to initialize a table or variable is useful when you suspect that a problem is caused by residual data left in those fields. If your problem occurs intermittently and not always with the same data, the problem could be that a switch is not initialized, but is generally set to the right value (0 or 1). By including a SET statement to ensure that the switch is initialized, you can determine whether or not the uninitialized switch is the cause of the problem. See *IBM COBOL for VSE/ESA Programming Guide* for a detailed discussion of how to use the INITIALIZE and SET statements.

Generating Information about Procedures

You can use debugging lines in your source programs to generate debugg ing information. Debugging lines are placed in your program anywhere after the OBJECT-COMPUTER paragraph, and are identified by a 'D' in position 7. Debugging lines allow you, for example, to check the value of a data-name at certain points in a procedure.

You can also use debugging statements in your source programs to generate debugging information. Debugging statements are coded in the Declaratives Section of the Procedure Division. You can use the USE FOR DEBUGGING declarative to include debugging statements in a COBOL routine and specify when they should run. Use these statements to generate information about your routine and how it is running.

For example, to check how many times a procedure is run, include a debugging procedure in the USE FOR DEBUGGING declarative, adding 1 to a counter each time control passes to that procedure.

Code each USE FOR DEBUGGING declarative in a separate section in the Declaratives Section of the Procedure Division. See the rules for coding them in *IBM COBOL for VSE/ESA Language Reference*

To use debugging lines and statements in your routine, you must include both:
- WITH DEBUGGING MODE in the SOURCE-COMPUTER paragraph in the Environment Division
- The DEBUG run-time option

With USE FOR DEBUGGING, the TEST compile-time option must have the NONE suboption specified or the NOTEST compile-time option must be specified.

Note: The TEST compile-time option and the DEBUG run-time option are mutually exclusive, with DEBUG taking precedence.

The adding-to-a-counter technique can be used as a check for:
- How many times a PERFORM ran. This shows you whether the control flow you are using is correct.
- How many times a loop routine actually runs. This tells you whether the loop is running and whether the number you have used for the loop is accurate.

Figure 55 on page 106 shows how to use the DISPLAY statement and the USE FOR DEBUGGING declarative to test a routine.

```
Environment Division
Source Computer . . . With Debugging Mode.
   ⋮
Data Division.
   ⋮
File Section.

Working-Storage Section.

  /*(among other entries you would need:)*/

01  Trace-Msg    PIC X(30)
                 Value "  Trace for Procedure-Name : ".
01  Total        PIC 99  Value Zeros.

  /*(balance of Working-Storage Section)*/

Procedure Division.
Declaratives.
Debug-Declar Section.
    Use For Debugging On 501-Some-Routine.
Debug-Declar-Paragraph.
    Display Trace-Msg, Debug-Name, Total.
Debug-Declar-End.
    Exit.

End Declaratives.

Begin-Program Section.
   ⋮
    Perform 501-Some-Routine.

      /*(within the module where you want to test, place:)*/

    Add 1 To Total

      /*(whether you put a period at the end depends on */
      /* where you put this statement.)                 */
```

Figure 55. Example of Using the WITH DEBUGGING MODE Clause

In the example in Figure 55, portions of a routine are shown to illustrate the kind of statements needed to use the USE FOR DEBUGGING declarative. The DISPLAY statement specified in the Declaratives Section issues the:

```
Trace For Procedure-Name : 501-Some-Routine nn
```

message every time the PERFORM 501-SOME-ROUTINE runs. The total shown, *nn*, is the value accumulated in the data item named TOTAL.

Another use for the DISPLAY statement technique shown above is to show the flow through your routine. You do this by changing the USE FOR DEBUGGING declarative in the Declaratives Section to:

```
USE FOR DEBUGGING ON ALL PROCEDURES.
```

and dropping the word TOTAL from the DISPLAY statement.

Using COBOL Listings

When you are debugging, you can use one or more of the following listings:
- Sorted Cross-Reference listing
- Data Map listing
- Verb Cross-Reference listing
- Procedure Division listings

This section gives an overview of each of these listings and specifies the compile-time option you use to obtain each listing. For a detailed description of available listings, sample listings, and a complete description of COBOL compile-time options, see *IBM COBOL for VSE/ESA Programming Guide*

Table 31. Contents of Listing and Associated Compile-Time Options

Name	Compile-Time Option	Function
Sorted Cross-Reference Listings	XREF	Provides sorted cross-reference listings of DATA DIVISION, PROCEDURE DIVISION, and routine names. The listings provide the location of all references to this information.
Data Map Listing	MAP	Provides information about the locations of all DATA DIVISION items and all implicitly declared variables. This option also supplies a nested routine map, which indicates where the routines are defined and provides routine attribute information.
Verb Cross-Reference Listing	VBREF	Produces an alphabetic listing of all the verbs in your routine and indicates where each is referenced.
Procedure Division Listings	LIST	Tells the COBOL compiler to generate a listing of the PROCEDURE DIVISION along with the assembler coding produced by the compiler. The list output includes the assembler source code, a map of the task global table (TGT), information about the location and size of WORKING-STORAGE and control blocks, and information about the location of literals and code for *dynamic storage* usage. Instead of the full PROCEDURE DIVISION listing with assembler expansion information, you can use the OFFSET compile-time option to get a condensed listing that provides information about the routine verb usage, global tables, WORKING-STORAGE, and literals. The OFFSET option takes precedence over the LIST option. That is, OFFSET and LIST are mutually exclusive; if you specify both, only OFFSET takes effect.

Generating an LE/VSE Dump of a COBOL Routine

Note: From LE/VSE 1.4.2 onwards, dumps taken with run-time option `TER(UADUMP...)` or `TER(DUMP...)` are capable of generating a run-time options report, *independently of* the setting for `RPTOPTS(ON|OFF)`. For an example, see Figure 69 on page 121.

Consider the two sample routines shown in Figure 56 and Figure 57 on page 109. In the examples, routine COBDUMP1 calls COBDUMP2 (sequence number COB00330 in COBDUMP1), which in turn calls the LE/VSE dump service CEE5DMP (sequence number COB00400 in COBDUMP2).

```
IDENTIFICATION DIVISION.                                            COB00010
  PROGRAM-ID.   COBDUMP1.                                           COB00020
  AUTHOR. USER NAME                                                 COB00030
                                                                    COB00040
ENVIRONMENT DIVISION.                                               COB00050
INPUT-OUTPUT SECTION.                                               COB00060
FILE-CONTROL.                                                       COB00070
    SELECT SEQ-FILE1                                                COB00080
     ASSIGN TO S-SYSFILE1                                           COB00090
     ORGANIZATION IS SEQUENTIAL.                                    COB00100
                                                                    COB00110
DATA DIVISION.                                                      COB00120
FILE SECTION.                                                       COB00130
FD SEQ-FILE1 IS GLOBAL                                              COB00140
    RECORDING MODE V.                                               COB00150
01 GLOBAL-REC1.                                                     COB00160
   05 G1-OBJ1 PIC 9.                                                COB00170
                                                                    COB00180
WORKING-STORAGE SECTION.                                            COB00190
01 SOME-WORKINGSTG.                                                 COB00200
   05 SUB-LEVEL PIC X(80).                                          COB00210
                                                                    COB00220
01   SALARY-RECORDA.                                                COB00230
 02    NAMEA   PIC X(10).                                           COB00240
 02    DEPTA   PIC 9(4).                                            COB00250
 02    SALARYA PIC 9(6).                                            COB00260
                                                                    COB00270
                                                                    COB00280
PROCEDURE DIVISION.                                                 COB00290
START-SEC.                                                          COB00300
    DISPLAY "STARTING TEST COBDUMP1".                               COB00310
    MOVE "THIS IS IN WORKING STORAGE" TO SUB-LEVEL.                 COB00320
    CALL "COBDUMP2" USING SALARY-RECORDA.                           COB00330
    DISPLAY "END OF TEST COBDUMP1"                                  COB00340
    STOP RUN.                                                       COB00350
END PROGRAM COBDUMP1.                                               COB00360
```

Figure 56. COBOL Routine COBDUMP1 Calling COBDUMP2

```
IDENTIFICATION DIVISION.                                         COB00010
  PROGRAM-ID.   COBDUMP2.                                        COB00020
  AUTHOR. USER NAME                                              COB00030
                                                                 COB00040
ENVIRONMENT DIVISION.                                            COB00050
INPUT-OUTPUT SECTION.                                            COB00060
FILE-CONTROL.                                                    COB00070
    SELECT OPTIONAL IOFSS1 ASSIGN AS-ESDS1DD                     COB00080
        ORGANIZATION SEQUENTIAL ACCESS SEQUENTIAL.               COB00090
                                                                 COB00100
DATA DIVISION.                                                   COB00110
FILE SECTION.                                                    COB00120
FD  IOFSS1 GLOBAL.                                               COB00130
  1 IOFSS1R PIC X(40).                                           COB00140
                                                                 COB00150
WORKING-STORAGE SECTION.                                         COB00160
  01  Temp4.                                                     COB00170
      05  a-1 occurs 2 times.                                    COB00180
          10  a-2 occurs 2 times.                                COB00190
              15 a-3v pic x(3).                                  COB00200
              15 a-6  Picture x(3).                              COB00210
77   DMPTITLE   PIC X(80).                                       COB00220
77   OPTIONS PIC X(255).                                         COB00230
77   FC PIC X(12).                                               COB00240
                                                                 COB00250
LINKAGE SECTION.                                                 COB00260
01   SALARY-RECORD.                                              COB00270
 02    NAME   PIC X(10).                                         COB00280
 02    DEPT   PIC 9(4).                                          COB00290
 02    SALARY PIC 9(6).                                          COB00300
                                                                 COB00310
PROCEDURE DIVISION USING SALARY-RECORD.                          COB00320
START-SEC.                                                       COB00330
    DISPLAY "STARTING TEST COBDUMP2"                             COB00340
    MOVE "COBOL/VSE DUMP" TO DMPTITLE.                           COB00350
    MOVE "XXX" to a-6(1, 1).                                     COB00360
    MOVE "YYY" to a-6(1, 2).                                     COB00370
    MOVE "ZZZ" to a-6(2, 1).                                     COB00380
    MOVE " BLOCKS STORAGE PAGE(55) FILES" TO OPTIONS.            COB00390
    CALL "CEE5DMP" USING DMPTITLE, OPTIONS, FC.                  COB00400
    DISPLAY "END OF TEST COBDUMP2"                               COB00410
    GOBACK.                                                      COB00420
END PROGRAM COBDUMP2.                                            COB00430
```

Figure 57. COBOL Routine COBDUMP2 Calling the LE/VSE Dump Service

The call in routine COBDUMP2 to CEE5DMP generates an LE/VSE dump, shown in Figure 58 on page 110. The dump includes a traceback section, which shows the names of both routines; a section on register usage at the time the dump was generated; and a variables section, which shows the stack frame and variable values for each routine. Character fields in the dump are indicated by single quotes. For an explanation of these sections of the dump, see “Finding COBOL Information in a Dump” on page 112.

```
CEE5DMP V1 R4.2: COBOL/VSE DUMP                                              07/27/01 1:50:17 PM                  PAGE:    1

CEE5DMP CALLED BY PROGRAM UNIT COBDUMP2 AT STATEMENT 40 (OFFSET +000003EE).

REGISTERS ON ENTRY TO CEE5DMP:

  PM....... 0000
  GPR0..... 006B296C  GPR1..... 006B29D0  GPR2..... 006B62E0  GPR3..... 006B61E0
  GPR4..... 005027F0  GPR5..... 006B2038  GPR6..... 0065D1AC  GPR7..... 00000000
  GPR8..... 0065DA70  GPR9..... 006B6178  GPR10.... 005028CC  GPR11.... 00502A70
  GPR12.... 00657E48  GPR13.... 006B2828  GPR14.... 8067FEEE  GPR15.... 83A01D68
  FPR0..... 40404040  40404040            FPR2..... 40404040  40404040
  FPR4..... 40404040  40404040            FPR6..... 40404040  40404040
⋮
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (006B2170)
        -0020 006B2150  00000000 00000000 00000000 00000000  006B6038 006B6070 00000000 00000000  |................,-..,-..........|
        +0000 006B2170  006B0588 00000000 005004AC 07FE07FE  00000000 00000000 00001FFF 07FE0000  |.,.h.....&......................|
        +0020 006B2190  00000000 0000C3D6 C2C4E4D4 D7F24040  40404040 40404040 40404040 40404040  |......COBDUMP2                  |
      STORAGE AROUND GPR 1 (006B21D8)
        -0020 006B21B8  00500354 00500354 006B2210 C0000000  00000000 00000000 006B2310 00000000  |.&...&...,...............,......|
        +0000 006B21D8  806B60C0 00000000 006B2038 006B0450  006B2210 006B2310 006B6038 006B6070  |.,-......,...,.&.,...,...,-..,-.|
        +0020 006B21F8  006B2000 000001C8 000001C0 00000000  006B6028 006B0440 C6C3C200 01020000  |.,.....H.........,-..,. FCB.....|
⋮
      STORAGE AROUND GPR15 (005027B8)
        -0020 00502798  0050274A 00000000 00000000 00000000  00000000 00502350 00000000 00000000  |.&...................&.&........|
        +0000 005027B8  47F0F028 00C3C5C5 00000000 00000014  47F0F001 4ACEAC00 00502864 00000000  |.00..CEE.........00......&......|
        +0020 005027D8  00000000 00000000 90ECD00C 4110F038  98EFF04C 07FF0000 005027B8 005028AC  |..............0.q.0<.....&...&..|

INFORMATION FOR ENCLAVE COBDUMP1

  INFORMATION FOR THREAD 8000000000000000

  REGISTERS ON ENTRY TO CEE5DMP:
    PM....... 0000
    GPR0..... 006B296C  GPR1..... 006B29D0  GPR2..... 006B62E0  GPR3..... 006B61E0
    GPR4..... 005027F0  GPR5..... 006B2038  GPR6..... 0065D1AC  GPR7..... 00000000
    GPR8..... 0065DA70  GPR9..... 006B6178  GPR10.... 005028CC  GPR11.... 00502A70
    GPR12.... 00657E48  GPR13.... 006B2828  GPR14.... 8067FEEE  GPR15.... 83A01D68
    FPR0..... 40404040  40404040            FPR2..... 40404040  40404040
    FPR4..... 40404040  40404040            FPR6..... 40404040  40404040
⋮
  TRACEBACK:
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY          E ADDR    E  OFFSET   STATEMENT  STATUS
    006B2828  COBDUMP2      005027B8  +000003EE  COBDUMP2       005027B8  +000003EE          40  CALL
    006B2038  COBDUMP1      00500078  +00000394  COBDUMP1       00500078  +00000394          33  CALL

  PARAMETERS, REGISTERS, AND VARIABLES FOR ACTIVE ROUTINES:
    COBDUMP2 (DSA ADDRESS 006B2828):
      SAVED REGISTERS:
        GPR0..... 006B296C  GPR1..... 006B29D0  GPR2..... 006B62E0  GPR3..... 006B61E0
        GPR4..... 005027F0  GPR5..... 006B2038  GPR6..... 0065D1AC  GPR7..... 00000000
        GPR8..... 0065DA70  GPR9..... 006B6178  GPR10.... 005028CC  GPR11.... 00502A70
        GPR12.... 00657E48  GPR13.... 006B2828  GPR14.... 8067FEEE  GPR15.... 83A01D68
⋮
```

Figure 58. Sections of the LE/VSE Dump Called from COBDUMP2 (Part 1 of 2)

```
  LOCAL VARIABLES:
          13 IOFSS1                             FILE SPECIFIED AS: OPTIONAL, ORGANIZATION=VSAM SEQUENTIAL,
                                               ACCESS MODE=SEQUENTIAL, RECFM=FIXED.  CURRENT STATUS OF
                                               FILE IS: NOT OPEN, VSAM STATUS CODE=00, VSAM FEEDBACK=000,
                                               VSAM RET CODE=000, VSAM FUNCTION CODE=000.
          14 01 IOFSS1R          X(40) DISP       '                                        '
          17 01 TEMP4            AN-GR
          18  02 A-1              AN-GR OCCURS 2
          19   03 A-2              AN-GR OCCURS 2
          20    04 A-3V             XXX
                  SUB(1,1)         DISP             '   '
                  SUB(1,2) TO SUB(2,2) ELEMENTS SAME AS ABOVE.
          21    04 A-6              XXX
                  SUB(1,1)         DISP             'XXX'
                  SUB(1,2)                          'YYY'
                  SUB(2,1)                          'ZZZ'
                  SUB(2,2)                          '   '
          22 77 DMPTITLE         X(80) DISP       'COBOL/VSE DUMP
                                                                       '
          23 77 OPTIONS          X(255) DISP      ' BLOCKS STORAGE PAGE(55) FILES

                                                                  '
          24 77 FC               X(12) DISP       '            '
          27 01 SALARY-RECORD    AN-GR
          28  02 NAME             X(10) DISP       '          '
          29  02 DEPT             9999 DISP
          30  02 SALARY           9(6) DISP
COBDUMP1 (DSA ADDRESS 006B2038):
  SAVED REGISTERS:
    GPR0..... 006B2170  GPR1..... 006B21D8  GPR2..... 006B60C0  GPR3..... 0050039A
    GPR4..... 005000B0  GPR5..... 006B0188  GPR6..... 0065D1AC  GPR7..... 00000000
    GPR8..... 0065DA70  GPR9..... 006B6070  GPR10.... 0050018C  GPR11.... 00500324
    GPR12.... 00657E48  GPR13.... 006B2038  GPR14.... 8050040E  GPR15.... 005027B8

  LOCAL VARIABLES:
          14 SEQ-FILE1                          FILE SPECIFIED AS: ORGANIZATION=SEQUENTIAL, ACCESS
                                               MODE=SEQUENTIAL, RECFM=VARIABLE.  CURRENT STATUS OF FILE
                                               IS: NOT OPEN, SAM STATUS CODE=00.
          16 01 GLOBAL-REC1      AN-GR
          17  02 G1-OBJ1          9                *** Invalid address for this item, no value displayed ***
          20 01 SOME-WORKINGSTG  AN-GR
          21  02 SUB-LEVEL        X(80) DISP       'THIS IS IN WORKING STORAGE
                                                                       '
          23 01 SALARY-RECORDA   AN-GR
          24  02 NAMEA            X(10) DISP       '          '
          25  02 DEPTA            9999 DISP
          26  02 SALARYA          9(6) DISP
⋮
```

Figure 58. Sections of the LE/VSE Dump Called from COBDUMP2 (Part 2 of 2)

Finding COBOL Information in a Dump

In addition to the standard LE/VSE dump format, dumps generated from COBOL routines contain:
- Control block information for active routines
- Storage for each active routine
- Enclave-level data
- Process-level data

Control Block Information for Active Routines

The Control Blocks for Active Routines section of the dump, shown in Figure 59, displays the following information for each active COBOL routine:

- Program Message
- COBOL compiler Version, Release, Modification, and User Level
- COBOL control blocks TGT, *CLLE*, *FCB*, *FIB*, and GMAREA

```
    PROGRAM COBDUMP2 WAS COMPILED 07/27/01 1:48:57 PM
    COBOL VERSION = 01 RELEASE = 01 MODIFICATION = 01     USER LEVEL = '    '
    TGT FOR COBDUMP2: 006B2828
      +000000 006B2828  00108001 006B2038 00690018 8067FEEE  83A01D68 006B296C 006B29D0 006B62E0  |....,..........c....,.%.,...,..|
      +000020 006B2848  006B61E0 005027F0 006B2038 0065D1AC  00000000 0065DA70 006B6178 005028CC  |.,/..&.0.,....J..........,/..&..|
      +000040 006B2868  00502A70 00657E48 F3E3C7E3 00690018  03000000 43030220 006B0188 0065D7EC  |.&....=.3TGT.............,.h..P.|
      +000060 006B2888  006B29E0 00000001 00000174 80502BA8  00000000 006B6130 00000000 00000000  |.,...........&.y.....,/.........|
:
    CLLE FOR COBDUMP2: 006B0588
      +000000 006B0588  C3D6C2C4 E4D4D7F2 00000100 00000000  84810000 005027B8 006B2828 00500078  |COBDUMP2........da...&...,...&..|
      +000020 006B05A8  00000000 006B26D4 006B27E0 006B0408  006B0000 00000088 C3E2E3D2 00000000  |.....,.M.,...,...,.....hCSTK....|
    DSA FOR COBDUMP1: 006B2038
      +000000  FLAGS.... 0010      member... 8001      BKC...... 00658608  FWC...... 00000000  R14...... 8050040E
      +000010  R15...... 005027B8  R0....... 006B2170  R1....... 006B21D8  R2....... 006B60C0  R3....... 0050039A
      +000024  R4....... 005000B0  R5....... 006B0188  R6....... 0065D1AC  R7....... 00000000  R8....... 0065DA70
      +000038  R9....... 006B6070  R10...... 0050018C  R11...... 00500324  R12...... 00657E48  reserved. F3E3C7E3
      +00004C  NAB...... 00690018  PNAB..... 03000000  reserved. 65030220  006B0188  0065D7EC  006B21E0
      +000064  reserved. 00000001  reserved. 00000064  MODE..... 00000000  reserved. 00000000  006B6028
      +000078  reserved. 00000000  reserved. 00000000

    PROGRAM COBDUMP1 WAS COMPILED 07/27/01 1:50:13 PM
    COBOL VERSION = 01 RELEASE = 01 MODIFICATION = 01     USER LEVEL = '    '
    TGT FOR COBDUMP1: 006B2038
      +000000 006B2038  00108001 00658608 00000000 8050040E  005027B8 006B2170 006B21D8 006B60C0  |......f......&...&...,...,.Q.,-.|
      +000020 006B2058  0050039A 005000B0 006B0188 0065D1AC  00000000 0065DA70 006B6070 0050018C  |.&...&...,.h..J..........,-..&..|
      +000040 006B2078  00500324 00657E48 F3E3C7E3 00690018  03000000 65030220 006B0188 0065D7EC  |.&....=.3TGT.............,.h..P.|
      +000060 006B2098  006B21E0 00000001 00000064 00000000  00000000 006B6028 00000000 00000000  |.,..................,-.........|
      +000080 006B20B8  00657E48 000001A8 00000000 00000000  00000001 00000001 E2E8E2D6 E4E34040  |..=....y................SYSOUT  |
      +0000A0 006B20D8  C9C7E9E2 D9E3C3C4 00000000 00000000  00000000 00000000 00000000 00000000  |IGZSRTCD........................|
      +0000C0 006B20F8  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
      +0000E0 006B2118  00000000 00000000 0050016C 00000001  006B21C4 006B0408 005001F0 006B2174  |.........&.%.....,.D.,...&.0.,..|
      +000100 006B2138  00500078 00500190 006B21B8 00500188  006B21C0 006B6070 00000000 00000000  |.&...&...,...&.h.,...,-.........|
      +000120 006B2158  00000000 00000000 006B6038 006B6070  00000000 00000000 006B0588 00000000  |.........,-..,-..........,.h....|
      +000140 006B2178  005004AC 07FE07FE 00000000 00000000  00001FFF 07FE0000 00000000 0000C3D6  |.&............................CO|
      +000160 006B2198  C2C4E4D4 D7F24040 40404040 40404040  40404040 40404040 40404040 00000000  |BDUMP2                      ....|
      +000180 006B21B8  00500354 00500354 006B2210 C0000000  00000000 00000000 006B2310 00000000  |.&...&...,.....................,......|
      +0001A0 006B21D8  806B60C0 00000000 006B2038 006B0450  006B2210 006B2310 006B6038 006B6070  |.,-.....,...,.&.,...,...,-..,-.|
    CLLE FOR COBDUMP1: 006B0408
      +000000 006B0408  C3D6C2C4 E4D4D7F1 00000100 00000000  94810000 00500078 006B2038 00500078  |COBDUMP1........ma...&...,...&..|
      +000020 006B0428  00000000 00000000 00000000 00000000  006B0000 00000080 00000078 00000000  |................,...............|
:
```

Figure 59. Control Block Information for Active COBOL Routines

Storage for Each Active Routine

The Storage for Active Routines section of the dump, shown in Figure 60, displays the following information for each COBOL routine:

- Routine name
- Contents of the base locators for files, WORKING-STORAGE, the linkage section, variably located areas, and EXTERNAL data.
- Working storage, including the base locator for working storage (BLW) and program class storage

```
⋮
 STORAGE FOR ACTIVE ROUTINES:
   COBDUMP2:
     CONTENTS OF BASE LOCATORS FOR FILES ARE:
          0-006B60F8

     CONTENTS OF BASE LOCATORS FOR WORKING STORAGE ARE:
          0-006B6178

     CONTENTS OF BASE LOCATORS FOR THE LINKAGE SECTION ARE:
          1-00000000        2-006B60C0

     NO VARIABLY LOCATED AREAS WERE USED IN THIS PROGRAM.

     NO EXTERNAL DATA WAS USED IN THIS PROGRAM.

     NO INDEXES WERE USED IN THIS PROGRAM.

   ESDS1DD  (BLF-0): 006B60F8
     +000000 006B60F8  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
     +000020 006B6118  00000000 00000000 00000000 00000000  006B6000 000001D0 000001C8 00000000  |................,-........H....|

   WORKING STORAGE FOR COBDUMP2
   BLW-0: 006B6178
     +000000 006B6178  000000E7 E7E70000 00E8E8E8 000000E9  E9E90000 00000000 C3D6C2D6 D361E5E2  |...XXX...YYY...ZZZ......COBOL/VS|
     +000020 006B6198  C540C4E4 D4D74040 40404040 40404040  40404040 40404040 40404040 40404040  |E DUMP                          |
     +000040 006B61B8  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  |                                |
     +000060 006B61D8  40404040 40404040 40C2D3D6 C3D2E240  E2E3D6D9 C1C7C540 D7C1C7C5 4DF5F55D  |         BLOCKS STORAGE PAGE(55)|
     +000080 006B61F8  40C6C9D3 C5E24040 40404040 40404040  40404040 40404040 40404040 40404040  | FILES                          |
     +0000A0 006B6218  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  |                                |
     +0000C0 006B6238 - +00015F 006B62D7            SAME AS ABOVE
     +000160 006B62D8  40404040 40404000 00000000 00000000  00000000 00000000 00000000 00000000  |       .........................|

   PROGRAM CLASS STORAGE: 006B6130
     +000000 006B6130  000001C8 00000000 00000000 006B2A00  00000000 00000000 00000000 00000000  |...H.........,..................|
     +000020 006B6150  00000000 00000000 E2E8E2D6 E4E34040  00000000 00000000 0E000000 00000000  |........SYSOUT  ................|
     +000040 006B6170  0F000000 00000000 000000E7 E7E70000  00E8E8E8 000000E9 E9E90000 00000000  |...........XXX...YYY...ZZZ......|
     +000060 006B6190  C3D6C2D6 D361E5E2 C540C4E4 D4D74040  40404040 40404040 40404040 40404040  |COBOL/VSE DUMP                  |
⋮
```

Figure 60. Storage for Active COBOL Routines

Enclave Level Data

The Enclave Storage section of the dump, shown in Figure 61, provides the following information:

- RUNCOM enclave control block
- Storage for all run units

```
:
  ENCLAVE CONTROL BLOCKS:
:
    RUNCOM: 006B0188
      +000000 006B0188  C3F3D9E4 D5C3D6D4 00000268 00A00040  00657060 00000000 00438000 00000000  |C3RUNCOM....... ...-............|
      +000020 006B01A8  00000000 00500078 006B0408 00000000  001F9160 00000000 00500078 00000000  |.....&...,........j-.....&......|
      +000040 006B01C8  0065DA70 0065D1AC 00000000 0065D7EC  00000000 00000000 00657E48 00000000  |......J.......P...........=.....|
      +000060 006B01E8  00000000 006B2818 006B2038 00000000  00000000 F0F0F0F0 F0F0F0F0 006B27E0  |.....,...,..........00000000.,..|
:
  ENCLAVE STORAGE:
:
    RUNUNIT CLASS STORAGE: 006B2818
      +000000 006B2818  000001E0 00000000 00000000 006B0578  00108001 006B2038 00690018 8067FEEE  |.............,......,..........|
      +000020 006B2838  83A01D68 006B296C 006B29D0 006B62E0  006B61E0 005027F0 006B2038 0065D1AC  |c....,.%.,...,...,/..&.0.,....J.|
:
    RUNUNIT CLASS STORAGE: 006B0578
      +000000 006B0578  00000040 00000000 006B2818 006B23C8  C3D6C2C4 E4D4D7F2 00000100 00000000  |... ....,...,.HCOBDUMP2........|
      +000020 006B0598  84810000 005027B8 006B2828 00500078  00000000 006B26D4 006B27E0 006B0408  |da...&...,...&.......,.M.,...,..|
    RUNUNIT CLASS STORAGE: 006B23C8
      +000000 006B23C8  00000448 00000000 006B0578 006B04C0  C8C1E340 00000100 00000000 00000000  |........,...,..HAT ...........|
      +000020 006B23E8  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
:
    RUNUNIT CLASS STORAGE: 006B04C0
      +000000 006B04C0  000000B0 00000000 006B23C8 006B2028  00000000 00000000 00000000 00000000  |.........,.H.,..................|
      +000020 006B04E0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
:
    RUNUNIT CLASS STORAGE: 006B2028
      +000000 006B2028  000001D0 00000000 006B04C0 006B03F8  00108001 00658608 00000000 8050040E  |........,...,.8......f......&..|
      +000020 006B2048  005027B8 006B2170 006B21D8 006B60C0  0050039A 005000B0 006B0188 0065D1AC  |.&...,...,.Q.,-..&...&...,.h..J.|
:
    RUNUNIT CLASS STORAGE: 006B03F8
      +000000 006B03F8  00000040 00000000 006B2028 00000000  C3D6C2C4 E4D4D7F1 00000100 00000000  |... ....,......COBDUMP1........|
      +000020 006B0418  94810000 00500078 006B2038 00500078  00000000 00000000 00000000 00000000  |ma...&...,...&.................|
:
```

Figure 61. Enclave Level Data for COBOL Routines

Process Level Data

This section of the dump, shown in Figure 62, displays COBOL process level control blocks *THDCOM*, *COBCOM*, *COBVEC*, and ITBLK [4].

```
   :
PROCESS CONTROL BLOCKS:
   :
  THDCOM: 0065DA70
    +000000 0065DA70  C3F3E3C8 C4C3D6D4 000001E8 81000000  00000100 00000000 0065D0F8 0065D1AC  |C3THDCOM...Ya..............8..J.|
    +000020 0065DA90  006AF188 00000000 C3D6C2C4 E4D4D7F1  00000000 00000000 00000000 00000000  |..1h....COBDUMP1................|
    +000040 0065DAB0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
   :
  COBCOM: 0065D0F8
    +000000 0065D0F8  C3F3C3D6 C2C3D6D4 00000978 00000000  00000000 00000000 00000000 00000000  |C3COBCOM........................|
    +000020 0065D118  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
    +000040 0065D138  00000000 00000000 00000000 00000000  00000000 00000000 908100D6 0065D1AC  |.........................a.O..J.|
   :
  COBVEC: 0065D1AC
    +000000 0065D1AC  0065D42C 0065D432 0065D438 0065D43E  0065D444 0065D44A 0065D450 0065D456  |..M...M...M...M...M...M...M&..M.|
    +000020 0065D1CC  0065D45C 0065D462 0065D468 0065D46E  0065D474 0065D47A 0065D480 0065D486  |..M*..M...M...M>..M...M:..M...Mf|
    +000040 0065D1EC  0065D48C 0065D492 0065D498 0065D49E  0065D4A4 0065D4AA 0065D4B0 0065D4B6  |..M...Mk..Mq..M...Mu..M...M...M.|
   :
```

Figure 62. Process Level Control Blocks for COBOL Routines

4. In a non-CICS environment, the ITBLK control block only appears when a VS COBOL II or DOS/VS COBOL routine is active. In a CICS environment, the ITBLK control block always appears.

Debugging Example COBOL Routines

The following examples help demonstrate techniques for debugging COBOL routines. Important areas of the dump output are highlighted. Data unnecessary to debugging has been replaced by ellipses.

Note: These examples were run using the `CHECK(ON)` run-time option. If the (*default*) `CHECK(OFF)` run-time option was used, the SSRANGE compile would not have had any effect!.

Subscript Range Error

Figure 63 illustrates the error of moving a value outside the range of an array. The COBOLX routine was compiled with LIST, TEST(STMT,SYM), and SSRANGE. The SSRANGE compile-time option causes the compiler to generate code that checks during run time for data that has been stored or referenced outside of its defined area because of incorrect indexing and subscripting. The SSRANGE option takes effect during run time, unless you specify CHECK(OFF) as a run-time option.

The routine was run with TERMTHDACT(TRACE) to generate the traceback information shown in Figure 64 on page 117.

```
CBL   LIST,SSRANGE,TEST(STMT,SYM)                                   DX100010
      ID DIVISION.                                                  DX100020
      PROGRAM-ID. COBOLX.                                           DX100030
      ENVIRONMENT DIVISION.                                         DX100040
      DATA DIVISION.                                                DX100050
      WORKING-STORAGE SECTION.                                      DX100060
      77  J     PIC 9(4) USAGE COMP.                                DX100070
      01  TABLE-X.                                                  DX100080
       02  SLOT PIC 9(4) USAGE COMP OCCURS 8 TIMES.                 DX100090
      PROCEDURE DIVISION.                                           DX100100
          MOVE 9 TO J.                                              DX100110
          MOVE 1 TO SLOT (J).                                       DX100120
          GOBACK.                                                   DX100130
```

Figure 63. COBOLX Example of Moving a Value Outside an Array Range

To understand the traceback information and debug this routine, use the following steps:

1. Locate the current error message in the Condition Information for Active Routines section of the LE/VSE traceback, shown in Figure 64 on page 117. The message is `IGZ0006S The reference to table SLOT by verb number 01 on line 000011 addressed an area outside the region of the table.` The message indicates that line 11 was the current COBOL statement when the error occurred. For more information about this message, see Chapter 12, “COBOL Run-Time Messages,” on page 307.
2. Statement 11 in the traceback section of the dump occurred in routine COBOLX.

```
CEE5DMP V1 R4.2: CONDITION PROCESSING RESULTED IN THE UNHANDLED CONDITION.        07/27/01 1:51:50 PM                  PAGE:    1
INFORMATION FOR ENCLAVE COBOLX

  INFORMATION FOR THREAD 8000000000000000

  TRACEBACK:
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY         E ADDR    E  OFFSET   STATEMENT  STATUS
    0068D018  CEEHDSP       03A58D38  +000032C0  CEEHDSP       03A58D38  +000032C0              CALL
    00690528  CEEHSGLT      03A62890  +0000005C  CEEHSGLT      03A62890  +0000005C              EXCEPTION
    00690018  IGZCMSG       0066C688  +0000037A  IGZCMSG       0066C688  +0000037A              CALL
    00501978  COBOLX        00500078  +0000021C  COBOLX        00500078  +0000021C          11  CALL

  CONDITION INFORMATION FOR ACTIVE ROUTINES
    CONDITION INFORMATION FOR CEEHSGLT (DSA ADDRESS 00690528)
      CIB ADDRESS: 0068D528
      CURRENT CONDITION:
        IGZ0006S THE REFERENCE TO TABLE SLOT BY VERB NUMBER 01 ON LINE 000011 ADDRESSED AN AREA OUTSIDE THE REGION OF THE TABLE.

      LOCATION:
        PROGRAM UNIT: CEEHSGLT ENTRY: CEEHSGLT STATEMENT:  OFFSET: +0000005C
    STORAGE DUMP NEAR CONDITION, BEGINNING AT LOCATION: 03A628DC
      +000000 03A628DC  F010D20B D0801000 58A0C2B8 58F0A01C  05EFD20B D098B118 41A0D098 50A0D08C  |0.K.......B..0....K..q.....q&...|

  PARAMETERS, REGISTERS, AND VARIABLES FOR ACTIVE ROUTINES:
    CEEHDSP (DSA ADDRESS 0068D018):
      SAVED REGISTERS:
        GPR0..... 0068D704  GPR1..... 0068D3BC  GPR2..... 0068D528  GPR3..... 0068D4D7
        GPR4..... 03A5C628  GPR5..... 0068E018  GPR6..... 03A5CD34  GPR7..... 0068E017
        GPR8..... 03A5BD35  GPR9..... 03A5AD36  GPR10.... 03A59D37  GPR11.... 83A58D38
        GPR12.... 00657E48  GPR13.... 0068D018  GPR14.... 8067FEEE  GPR15.... 83A01D68
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (0068D704)
        -0020 0068D6E4  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
        +0000 0068D704  E9D4C3C8 00D00001 00690528 006901E8  006901E8 00000000 0067F038 0067F038  |ZMCH...........Y...Y......0...0.|
        +0020 0068D724  005001DE 00000005 0065DA70 00501978  0067F038 83A62890 00657E48 00690528  |.&...........&....0.cw....=.....|
⋮
    CEEHSGLT (DSA ADDRESS 00690528):
      SAVED REGISTERS:
        GPR0..... 00690528  GPR1..... 006901E8  GPR2..... 006901E8  GPR3..... 00000000
        GPR4..... 0067F038  GPR5..... 0067F038  GPR6..... 005001DE  GPR7..... 00000005
        GPR8..... 0065DA70  GPR9..... 00501978  GPR10.... 0067F038  GPR11.... 83A62890
        GPR12.... 00657E48  GPR13.... 00690528  GPR14.... 8067FEDA  GPR15.... 801FD878
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (00690528)
        -0020 00690508  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40400000  |                              ..|
        +0000 00690528  00000000 00690018 006905E8 8067FEDA  801FD878 00690528 006901E8 006901E8  |...........Y......Q........Y...Y|
        +0020 00690548  00000000 0067F038 0067F038 005001DE  00000005 0065DA70 00501978 0067F038  |......0...0..&...........&....0.|
⋮
    IGZCMSG  (DSA ADDRESS 00690018):
      SAVED REGISTERS:
        GPR0..... 00690528  GPR1..... 006901E8  GPR2..... 006901E8  GPR3..... 00000000
        GPR4..... 0067F038  GPR5..... 0067F038  GPR6..... 005001DE  GPR7..... 00000005
        GPR8..... 0065DA70  GPR9..... 00501978  GPR10.... 006B0188  GPR11.... 8066C688
        GPR12.... 00657E48  GPR13.... 00690018  GPR14.... 8068344A  GPR15.... 83A62890
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (00690528)
        -0020 00690508  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40400000  |                              ..|
        +0000 00690528  00000000 00690018 006905E8 8067FEDA  801FD878 00690528 006901E8 006901E8  |...........Y......Q........Y...Y|
⋮
    COBOLX (DSA ADDRESS 00501978):
      SAVED REGISTERS:
        GPR0..... 00690128  GPR1..... 005001DE  GPR2..... 00000010  GPR3..... 0065D1AC
        GPR4..... 005000B0  GPR5..... 006B0188  GPR6..... 0065D1AC  GPR7..... 00000000
        GPR8..... 0065DA70  GPR9..... 00501B40  GPR10.... 00500174  GPR11.... 00500248
        GPR12.... 0050016C  GPR13.... 00501978  GPR14.... 40500296  GPR15.... 8066C688
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (00690128)
        -0020 00690108  00657180 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
        +0000 00690128  00101001 00690018 00000078 8066C688  001E0000 00000008 00000000 00000000  |..............Fh................|
⋮
      LOCAL VARIABLES:
               6 77 J              9999 COMP       00009
               7 01 TABLE-X        AN-GR
               8  02 SLOT           9999  OCCURS 8
                    SUB(1)         COMP            00000
                    SUB(2) TO SUB(8) ELEMENTS SAME AS ABOVE.
```

Figure 64. Sections of LE/VSE Dump for COBOLX

3. Find the statement on line 11 in the listing for routine COBOLX, shown in Figure 65. This statement moves the 1 value to the array SLOT (J).

```
PP 5686-068 IBM COBOL FOR VSE/ESA 1.1.1                          DATE 07/27/2001  TIME 13:51:47   PAGE     2
    NOWORD
      XREF(SHORT)
      YEARWINDOW(1900)
      ZWB
PP 5686-068 IBM COBOL FOR VSE/ESA 1.1.1                COBOLX    DATE 07/27/2001  TIME 13:51:47   PAGE     3
  LINEID  PL SL  ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7-|--+----8  MAP AND CROSS REFERENCE
/* COBOLX
  000001               ID DIVISION.
  000002               PROGRAM-ID. COBOLX.
  000003               ENVIRONMENT DIVISION.
  000004               DATA DIVISION.
  000005               WORKING-STORAGE SECTION.
  000006               77  J     PIC 9(4) USAGE COMP.                                  BLW=0000+000         2C
  000007               01  TABLE-X.                                                    BLW=0000+008         0CL16
  000008                02  SLOT PIC 9(4) USAGE COMP OCCURS 8 TIMES.                   BLW=0000+008,0000000 2C
  000009               PROCEDURE DIVISION.
  000010                   MOVE 9 TO J.                                                6
  000011                   MOVE 1 TO SLOT (J).                                         8 6
  000012                   GOBACK.
⋮
```

Figure 65. COBOLX Listing

4. Find the values of the local variables in the Arguments, Registers, and Variables for Active Routines section of the traceback, shown in Figure 64 on page 117. J, which is of type PIC 9(4) with usage COMP, has a 9 value. J is the index to the array SLOT.

   Note: The array SLOT contains eight positions. When the routine tries to move a value into the J or 9th element of the 8-element array named SLOT, the error of moving a value outside the area of the array occurs.

Calling a Nonexistent Subroutine

Figure 66 demonstrates the error of calling a nonexistent subroutine in a COBOL routine. In this example, the routine COBOLY was compiled with the compile-time options LIST and MAP, and run with the run-time option TERMTHDACT(DUMP). Figure 66 shows the routine.

COBOLY was not compiled with the SYM suboption of the TEST compile-time option. As a result, arguments and variables do not appear in the dump.

```
 CBL   LIST,MAP                                                    DX200010
       ID DIVISION.                                                DX200020
       PROGRAM-ID. COBOLY.                                         DX200030
       ENVIRONMENT DIVISION.                                       DX200040
       DATA DIVISION.                                              DX200050
       WORKING-STORAGE SECTION.                                    DX200060
       77  SUBNAME PIC X(8) USAGE DISPLAY.                         DX200070
       PROCEDURE DIVISION.                                         DX200080
           MOVE "COBSUB1" TO SUBNAME.                              DX200090
           CALL SUBNAME.                                           DX200100
           GOBACK.                                                 DX200110
```

Figure 66. COBOLY Example of Calling a Nonexistent Subroutine

To understand the traceback information and debug this routine, use the following steps:

1. Locate the error message for the current condition under the Condition Information for Active Routines section of the dump, shown in Figure 67. The message is `CEE3501S The phase` *`phase name`* `was not found.` For more information about this message, see Chapter 8, “LE/VSE Run-Time Messages,” on page 163.
2. Note the sequence of calls in the Traceback section of the dump. COBOLY called IGZCLNK; IGZCLNK (a COBOL library subroutine used for *dynamic calls*) called IGZCLDL; then IGZCLDL (a COBOL library subroutine used to load routines) attempted to load and call a nonexistent routine. This caused an exception, resulting in a call to CEEHDSP.

   This sequence indicates that the exception occurred in IGZCLDL when COBOLY was attempting to make a dynamic call.

   Note: The call statement in COBOLY is located at offset “+000002CE”.

```
CEE5DMP V1 R4.2: CONDITION PROCESSING RESULTED IN THE UNHANDLED CONDITION.        07/27/01 1:52:48 PM                  PAGE:    1

INFORMATION FOR ENCLAVE COBOLY

  INFORMATION FOR THREAD 8000000000000000

  TRACEBACK:
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY         E ADDR    E  OFFSET   STATEMENT  STATUS
    0068D018  CEEHDSP       03A58D38  +000029DA  CEEHDSP       03A58D38  +000029DA              CALL
    006901B0  IGZCLDL       0066AA20  +000000F4  IGZCLDL       0066AA20  +000000F4              EXCEPTION
    00690018  IGZCLNK       0067C780  +00000486  IGZCLNK       0067C780  +00000486              CALL
    00501980  COBOLY        00500078  +000002CE  COBOLY        00500078  +000002CE           9  CALL

  CONDITION INFORMATION FOR ACTIVE ROUTINES
    CONDITION INFORMATION FOR IGZCLDL  (DSA ADDRESS 006901B0)
      CIB ADDRESS: 0068D528
      CURRENT CONDITION:
        CEE0198S THE TERMINATION OF A THREAD WAS SIGNALED DUE TO AN UNHANDLED CONDITION.
      ORIGINAL CONDITION:
        CEE3501S THE PHASE COBSUB1  WAS NOT FOUND.
      LOCATION:
        PROGRAM UNIT: IGZCLDL  ENTRY: IGZCLDL  STATEMENT:  OFFSET: +000000F4
    STORAGE DUMP NEAR CONDITION, BEGINNING AT LOCATION: 0066AB04
      +000000 0066AB04  30045060 30085850 C2B81813 58F0501C  05EF47F0 B21A41A0 000850A0 D0D85070  |..&-...&B....&....0......&..Q&.|

  PARAMETERS, REGISTERS, AND VARIABLES FOR ACTIVE ROUTINES:
⋮
```

Figure 67. Sections of LE/VSE Dump for COBOLY

3. Use the offset of X'000002CE' from the COBOL listing, shown in Figure 68 on page 120, to locate the statement that caused the exception in the COBOLY routine. The offset X'000002CE' is an instruction for statement 9. Statement 9 is a call with the identifier SUBNAME specified.

   Note: In the Data Division Map, SUBNAME is at Base Locator for Working Storage (BLW) 0 with offset of 0. This offset points to the SUBNAME value.

```
PP 5686-068 IBM COBOL FOR VSE/ESA 1.1.1                        COBOLY    DATE 07/27/2001  TIME 13:52:45   PAGE     3
  LINEID  PL SL  ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7-|--+----8  MAP AND CROSS REFERENCE
/* COBOLY
  000001               ID DIVISION.
  000002               PROGRAM-ID. COBOLY.
  000003               ENVIRONMENT DIVISION.
  000004               DATA DIVISION.
  000005               WORKING-STORAGE SECTION.
  000006               01  SUBNAME PIC X(8) USAGE DISPLAY.                                 BLW=0000+000         8C
  000007               PROCEDURE DIVISION.
  000008                   MOVE "COBSUB1" TO SUBNAME                                       6
  000009                   CALL SUBNAME                                                    6
  000010                   GOBACK.
*/ COBOLY
PP 5686-068 IBM COBOL FOR VSE/ESA 1.1.1                        COBOLY    DATE 07/27/2001  TIME 13:52:45   PAGE     4
AN "M" PRECEDING A DATA-NAME REFERENCE INDICATES THAT THE DATA-NAME IS MODIFIED BY THIS REFERENCE.

 DEFINED   CROSS-REFERENCE OF DATA NAMES   REFERENCES

       6   SUBNAME. . . . . . . . . . . .  M8 9
PP 5686-068 IBM COBOL FOR VSE/ESA 1.1.1                        COBOLY    DATE 07/27/2001  TIME 13:52:45   PAGE     5
 DEFINED   CROSS-REFERENCE OF PROGRAMS     REFERENCES

PP 5686-068 IBM COBOL FOR VSE/ESA 1.1.1                        COBOLY    DATE 07/27/2001  TIME 13:52:45   PAGE     6
DATA DIVISION MAP
DATA DEFINITION ATTRIBUTE CODES (RIGHTMOST COLUMN) HAVE THE FOLLOWING MEANINGS:
    D = OBJECT OF OCCURS DEPENDING    G = GLOBAL                          S = SPANNED FILE
    E = EXTERNAL                      O = HAS OCCURS CLAUSE               U = UNDEFINED FORMAT FILE
    F = FIXED-LENGTH FILE             OG= GROUP HAS OWN LENGTH DEFINITION    V = VARIABLE-LENGTH FILE
    FB= FIXED-LENGTH BLOCKED FILE     R = REDEFINES                       VB= VARIABLE-LENGTH BLOCKED FILE
SOURCE   HIERARCHY AND                             BASE       HEX-DISPLACEMENT  ASMBLR DATA                 DATA DEF
LINEID   DATA NAME                                 LOCATOR    BLK   STRUCTURE   DEFINITION    DATA TYPE      ATTRIBUTES
     2  PROGRAM-ID COBOLY -------------------------------------------------------------------------------------------*
     6   01 SUBNAME . . . . . . . . . . . . . . . . . BLW=0000  000              DS 8C         DISPLAY
⋮
PP 5686-068 IBM COBOL FOR VSE/ESA 1.1.1                        COBOLY    DATE 07/27/2001  TIME 13:52:45   PAGE    11
    000280                  START    EQU  *                      COBOLY
    000280  58A0 C000                L    10,0(0,12)             CBL=1
    000284  5890 D124                L    9,292(0,13)            BLW=0
    000288  D203 D0EC A004           MVC  236(4,13),4(10)        TGTFIXD+236                      PGMLIT AT +0
    00028E  9180 D174                TM   372(13),X'80'          IPCB=1
    000292  58B0 C004                L    11,4(0,12)             PBL=1
    000296  4780 B000                BC   8,0(0,11)              GN=5(0002A8)
    00029A  5820 D05C                L    2,92(0,13)             TGTFIXD+92
    00029E  58F0 21CC                L    15,460(0,2)            V(IGZEMSG )
    0002A2  4110 A165                LA   1,357(0,10)            PGMLIT AT +353
    0002A6  05EF                     BALR 14,15
    0002A8                  GN=5     EQU  *
    0002A8  9680 D174                OI   372(13),X'80'          IPCB=1
    0002AC  9640 D174                OI   372(13),X'40'          IPCB=1
000008  *
000008  MOVE
    0002B0  D207 9000 A008           MVC  0(8,9),8(10)           SUBNAME                          PGMLIT AT +4
000009  CALL
    0002B6  D207 D188 9000           MVC  392(8,13),0(9)         TS2=0                            SUBNAME
    0002BC  DC07 D188 A016           TR   392(8,13),22(10)       TS2=0                            PGMLIT AT +18
    0002C2  5820 D05C                L    2,92(0,13)             TGTFIXD+92
    0002C6  58F0 205C                L    15,92(0,2)             V(IGZCLNK )
    0002CA  4110 A15C                LA   1,348(0,10)            PGMLIT AT +344
    0002CE  05EF                     BALR 14,15
000010  GOBACK
    0002D0  47F0 B03E                BC   15,62(0,11)            GN=1(0002E6)
    0002D4  9120 D054                TM   84(13),X'20'           TGTFIXD+84
    0002D8  47E0 B03E                BC   14,62(0,11)            GN=1(0002E6)
    0002DC  58F0 21CC                L    15,460(0,2)            V(IGZEMSG )
    0002E0  4110 A14A                LA   1,330(0,10)            PGMLIT AT +326
    0002E4  05EF                     BALR 14,15
    0002E6                  GN=1     EQU  *
    0002E6  947F D174                NI   372(13),X'7F'          IPCB=1
    0002EA  9140 D055                TM   85(13),X'40'           TGTFIXD+85
    0002EE  47E0 B054                BC   14,84(0,11)            GN=6(0002FC)
    0002F2  4110 0008                LA   1,8(0,0)
    0002F6  58F0 2020                L    15,32(0,2)             V(IGZCCTL )
    0002FA  05EF                     BALR 14,15
    0002FC                  GN=6     EQU  *
    0002FC  58F0 2224                L    15,548(0,2)            V(IGZETRM )
    000300  4110 A13A                LA   1,314(0,10)            PGMLIT AT +310
    000304  05EF                     BALR 14,15
```

Figure 68. COBOLY Listing

4. The content of the storage at offset 0 of BLW 0 is C3D6C2E2 E4C2F140 in the Storage for Active Routines section of the dump, shown in Figure 69. This indicates that the variable SUBNAME was initialized, or the VALUE clause was specified for the identifier.

```
⋮
  STORAGE FOR ACTIVE ROUTINES:
    COBOLY:
      NO FILES WERE USED IN THIS PROGRAM.

      CONTENTS OF BASE LOCATORS FOR WORKING STORAGE ARE:
           0-00501B48

      CONTENTS OF BASE LOCATORS FOR THE LINKAGE SECTION ARE:
CEE5DMP V1 R4.2: CONDITION PROCESSING RESULTED IN THE UNHANDLED CONDITION.         07/27/01 1:52:48 PM                 PAGE:    8

           1-00000000

      NO VARIABLY LOCATED AREAS WERE USED IN THIS PROGRAM.

      NO EXTERNAL DATA WAS USED IN THIS PROGRAM.

      NO INDEXES WERE USED IN THIS PROGRAM.

    WORKING STORAGE FOR COBOLY
    BLW-0: 00501B48
      +000000 00501B48  C3D6C2E2 E4C2F140 00000008 00000000  00501B70 00501B78 00000000 00501BF8  |COBSUB1 .........&...&.......&.8|
⋮
  RUN-TIME OPTIONS REPORT ASSOCIATED WITH DUMP FOR ENCLAVE : COBOLY

   LAST WHERE SET          OPTION
   -------------------------------------------------------------------------------
   Installation default      ABPERC(NONE)
   Installation default      ABTERMENC(ABEND)
   Installation default    NOAIXBLD
   Installation default      ALL31(OFF)
   Installation default      ANYHEAP(16384,8192,ANYWHERE,FREE)
   Installation default      BELOWHEAP(8192,4096,FREE)
   Installation default      CBLOPTS(ON)
   Installation default      CBLPSHPOP(OFF)
   Installation default      CHECK(OFF)
   Non-overrideable          COUNTRY(US)
   Installation default    NODEBUG
   Installation default      DEPTHCONDLMT(10)
   Installation default      ENVAR("")
   Installation default      ERRCOUNT(20)
   Installation default      HEAP(32768,32768,ANYWHERE,KEEP,8192,4096)
   Installation default      HEAPCHK(OFF,1,0)
   Installation default      LIBSTACK(12288,4096,FREE)
   Installation default      MSGFILE(SYSLST)
   Installation default      MSGQ(15)
   Installation default      NATLANG(UEN)
   Installation default      RETZERO(OFF)
   Installation default      RPTOPTS(OFF)
   Installation default      RPTSTG(OFF)
   Installation default    NORTEREUS
   Installation default      STACK(131072,131072,BELOW,KEEP)
   Assembler user exit       STORAGE(00,NONE,NONE,32768)
   Invocation command        TERMTHDACT(DUMP,,96)
   Installation default    NOTEST(ALL,"*","PROMPT","")
   Installation default      TRACE(OFF,4096,DUMP,LE=0)
   Installation default      TRAP(ON,MAX)
   Installation default      UPSI(00000000)
   Installation default    NOUSRHDLR()
   Installation default      XUFLOW(AUTO)
```

Figure 69. Storage for Active Routines Section of Dump for COBOLY

Divide-by-Zero Error

The following example demonstrates the error of calling an assembler routine that tries to divide by zero. The main routine was:

- compiled with `TEST(STMT,SYM)` compile-time option, so that at run-time, statement numbers are displayed in the traceback.
- run with the `TERMTHDACT(TRACE)` run-time option.

Figure 70 shows the main COBOL routine (COBOLZ1), which dynamically calls:
    COBOL subroutine (COBOLZ2), which dynamically calls:
        assembler routine (ASSEMZ3).

Both programs COBOLZ1 and COBOLZ2 were compiled using the DYNAM compile option (so that they are loaded dynamically at run time), and the XREF compile option (so that a detailled view of registers is generated). The use of the XREF option is demonstrated in Figure 72 on page 125.

Note: The application structure shown above, which uses dynamically-called subroutines, requires that each routine is generated as a single phase unit.

```
[Main Program]

    ID DIVISION.
    PROGRAM-ID. COBOLZ1.
    ENVIRONMENT DIVISION.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    77  D-VAL PIC 9(4) USAGE COMP VALUE 0.
    PROCEDURE DIVISION.
        CALL "COBOLZ2" USING D-VAL.

[Subroutine]


    ID DIVISION.
    PROGRAM-ID. COBOLZ2.
    ENVIRONMENT DIVISION.
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    77  DV-VAL PIC 9(4) USAGE COMP.

    LINKAGE SECTION.
    77  D-VAL PIC 9(4) USAGE COMP.
    PROCEDURE DIVISION USING D-VAL.
        MOVE D-VAL TO DV-VAL.
        CALL 'ASSEMZ3' USING DV-VAL.
        GOBACK.
```

Figure 70. Main COBOLZ1 Routine, COBOLZ2 Subroutine, and ASSEMZ3 Routine (Part 1 of 2)

```
[Assembler Routine]

    ASSEMZ3  CSECT
    ASSEMZ3  AMODE 31
    ASSEMZ3  RMODE ANY
             STM   14,12,12(13)        SAVE CALLERS REGS
             LR    12,15               SET BASE REG
             USING ASSEMZ3,12
             LA    2,SAVE              CROSS CHAIN SAVE AREAS
             ST    13,4(2)               "
             ST    2,8(,13)              "
             ST    13,0(,2)              "
             LR    13,2                NEW SAVE AREA PTR TO R13
             LA    5,2348              LOW ORDER PART OF QUOTIENT
             SR    4,4                 HI ORDER PART OF QUOTIENT
             L     6,0(1)              GET POINTER TO DIVISOR
             LA    6,0(6)              CLEAR HI BIT
             D     4,0(6)              DO DIVISION
             L     13,4(13)            CALLERS SAVE AREA PTR TO R13
             RETURN (14,12),RC=0       RETURN TO CALLER
    SAVE     DC    9D'0'
    ASSEMZ3  CSECT
             END   ASSEMZ3
```

Figure 70. Main COBOLZ1 Routine, COBOLZ2 Subroutine, and ASSEMZ3 Routine (Part 2 of 2)

To debug this application, use the following steps:

1. Locate the error message for the current condition in the Condition Information section of the traceback, shown in Figure 66 on page 118. The message is `CEE3209S The system detected a fixed-point divide exception.`

   See Chapter 8, "LE/VSE Run-Time Messages," on page 163 for additional information about this message.
2. Note the sequence of calls in the call chain in the traceback section:
   a. COBOLZ1 called IGZCLNK, which is a COBOL library subroutine used for *dynamic calls*.
   b. IGZCLNK called COBOLZ2.
   c. COBOLZ2 then called IGZCLNK.
   d. IGZCLNK attempted to call an unlisted routine. The exception occurred at this point, resulting in a call to CEEHDSP, an LE/VSE condition handling routine.

   The call to the unlisted routine occurred at statement 11 of COBOLZ2. The offset is reported as X'00000000' because the routine (in this case, ASSEMZ3) is not LE/VSE-enabled. To calculate the offset, subtract the program unit address from the program status word, then subtract the instruction length code from the result (offset = `PSW - PU Addr - ILC`). Using this calculation, you can determine that the exception occurred at offset +26 in the unknown routine:

   ```
   (X'6405AA' - X'640580' - X'000004') = X'26'
   ```

   If your system is operating in extended-addressing mode, ignore the high-order bit of the address in the PSW. If your system is operating in 370 mode, ignore the high-order byte of the address in the PSW.

```
CEE5DMP V1 R4.2: CONDITION PROCESSING RESULTED IN THE UNHANDLED CONDITION.        07/27/01 1:56:25 PM                  PAGE:    1

INFORMATION FOR ENCLAVE COBOLZ1

  INFORMATION FOR THREAD 8000000000000000

  TRACEBACK:
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY         E ADDR    E  OFFSET   STATEMENT  STATUS
    0068D018  CEEHDSP       03A58D38  +000032C0  CEEHDSP       03A58D38  +000032C0              CALL
    006405C0                00640580  +00000000                00640580  +00000000              EXCEPTION
    006901B0  IGZCLNK       0067C780  +000003D8  IGZCLNK       0067C780  +000003D8              CALL
    006B2488  COBOLZ2       006B6000  +00000258  COBOLZ2       006B6000  +00000258          11  CALL
    00690018  IGZCLNK       0067C780  +000003D8  IGZCLNK       0067C780  +000003D8              CALL
    00501918  COBOLZ1       00500078  +000001D8  COBOLZ1       00500078  +000001D8           8  CALL

  CONDITION INFORMATION FOR ACTIVE ROUTINES
    CONDITION INFORMATION FOR  (DSA ADDRESS 006405C0)
      CIB ADDRESS: 0068D528
      CURRENT CONDITION:
        CEE3209S THE SYSTEM DETECTED A FIXED-POINT DIVIDE EXCEPTION.
      LOCATION:
        PROGRAM UNIT:  ENTRY:  STATEMENT:  OFFSET: +00000000
      MACHINE STATE:
        ILC..... 0004    INTERRUPTION CODE..... 0009
        PSW..... 071D0000 806405AA
        GPR0..... 00690348  GPR1..... 006B2620  GPR2..... 006405C0  GPR3..... 006B231C
        GPR4..... 00000000  GPR5..... 0000092C  GPR6..... 006B9070  GPR7..... 806B6000
        GPR8..... 8067CB5C  GPR9..... 006B2488  GPR10.... 006B0188  GPR11.... 8067C780
        GPR12.... 80640580  GPR13.... 006405C0  GPR14.... 8067CB5A  GPR15.... 80640580
    STORAGE DUMP NEAR CONDITION, BEGINNING AT LOCATION: 00640596
      +000000 00640596  18D24150 092C1B44 58610000 41660000  5D460000 58DD0004 41F00000 58ED000C  |.K.&...../......)........0......|

  PARAMETERS, REGISTERS, AND VARIABLES FOR ACTIVE ROUTINES:
    CEEHDSP (DSA ADDRESS 0068D018):
      SAVED REGISTERS:
        GPR0..... 0068D704  GPR1..... 0068D3BC  GPR2..... 0068D528  GPR3..... 0068D4D7
        GPR4..... 03A5C628  GPR5..... 0068E018  GPR6..... 03A5CD34  GPR7..... 0068E017
        GPR8..... 03A5BD35  GPR9..... 03A5AD36  GPR10.... 03A59D37  GPR11.... 83A58D38
        GPR12.... 00657E48  GPR13.... 0068D018  GPR14.... 8067FEEE  GPR15.... 83A01D68
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (0068D704)
        -0020 0068D6E4  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
        +0000 0068D704  E9D4C3C8 00D00001 00690348 006B2620  006405C0 006B231C 00000000 0000092C  |ZMCH.........,.......,..........|
        +0020 0068D724  006B9070 806B6000 8067CB5C 006B2488  006B0188 8067C780 80640580 006405C0  |.,...,-....*.,.h.,.h..G.........|
⋮
     (DSA ADDRESS 006405C0):
      SAVED REGISTERS:
        GPR0..... 00690348  GPR1..... 006B2620  GPR2..... 006405C0  GPR3..... 006B231C
        GPR4..... 00000000  GPR5..... 0000092C  GPR6..... 006B9070  GPR7..... 806B6000
        GPR8..... 8067CB5C  GPR9..... 006B2488  GPR10.... 006B0188  GPR11.... 8067C780
        GPR12.... 80640580  GPR13.... 006405C0  GPR14.... 8067CB5A  GPR15.... 80640580
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (00690348)
        -0020 00690328  006B7FB0 00656D40 00000000 006B7C08  0069040C 00000000 006903C4 03A4619F  |.,"..._ .....,@............D.u/.|
        +0000 00690348  00001001 0068D018 00690E90 83A059D8  03A07768 0000001F 00690668 00000001  |............c..Q................|
        +0020 00690368  00000000 00000004 00690AE8 03A06D63  03A05D64 03A04D65 03A03D66 03A02D67  |...........Y.._...)...(.........|
⋮
    IGZCLNK  (DSA ADDRESS 006901B0):
      SAVED REGISTERS:
        GPR0..... 00690348  GPR1..... 006B2620  GPR2..... 00000000  GPR3..... 006B231C
        GPR4..... 006B6169  GPR5..... 006B25C0  GPR6..... 006B0520  GPR7..... 806B6000
        GPR8..... 8067CB5C  GPR9..... 006B2488  GPR10.... 006B0188  GPR11.... 8067C780
        GPR12.... 00657E48  GPR13.... 006901B0  GPR14.... 8067CB5A  GPR15.... 80640580
    GPREG STORAGE:
      STORAGE AROUND GPR 0 (00690348)
        -0020 00690328  006B7FB0 00656D40 00000000 006B7C08  0069040C 00000000 006903C4 03A4619F  |.,"..._ .....,@............D.u/.|
        +0000 00690348  00001001 0068D018 00690E90 83A059D8  03A07768 0000001F 00690668 00000001  |............c..Q................|
        +0020 00690368  00000000 00000004 00690AE8 03A06D63  03A05D64 03A04D65 03A03D66 03A02D67  |...........Y.._...)...(.........|
⋮
```

Figure 71. Sections of LE/VSE Dump for Routine COBOLZ1 (Part 1 of 2)

```
   COBOLZ2 (DSA ADDRESS 006B2488):
     SAVED REGISTERS:
       GPR0..... 006902C0  GPR1..... 006B6169  GPR2..... 0065D7EC  GPR3..... 006B6220
       GPR4..... 006B6038  GPR5..... 00690018  GPR6..... 0065D1AC  GPR7..... 00000000
       GPR8..... 0065DA70  GPR9..... 006B9070  GPR10.... 006B6100  GPR11.... 006B6198
       GPR12.... 006B60F4  GPR13.... 006B2488  GPR14.... 806B625A  GPR15.... 8067C780
   GPREG STORAGE:
     STORAGE AROUND GPR 0 (006902C0)
       -0020 006902A0  006B0188 8067C780 00657E48 00000000  00000000 006B6177 00000000 006B25C0  |.,.h..G...=..........,/......,..|
       +0000 006902C0  006B610C 006B2620 806B6000 00000000  8066F340 00010D34 49C3C5C5 00000000  |.,/..,...,-.......3 .....CEE....|
       +0020 006902E0  006B0534 006902CC 006902D4 0065D0F8  0065D1AC 0065DA70 0065D1AC C1E2E2C5  |.,.........M...8..J.......J.ASSE|
⋮
     LOCAL VARIABLES:
              6 77 DV-VAL           9999 COMP        00000
              8 77 D-VAL            9999 COMP        00000
⋮
   IGZCLNK  (DSA ADDRESS 00690018):
     SAVED REGISTERS:
       GPR0..... 006901B0  GPR1..... 00501AA8  GPR2..... 0065D119  GPR3..... 00501AA8
       GPR4..... 005001D1  GPR5..... 00501A4C  GPR6..... 006B0450  GPR7..... 00500078
       GPR8..... 8067CB5C  GPR9..... 00501918  GPR10.... 006B0188  GPR11.... 8067C780
       GPR12.... 00657E48  GPR13.... 00690018  GPR14.... 8067CB5A  GPR15.... 806B6000
   GPREG STORAGE:
     STORAGE AROUND GPR 0 (006901B0)
       -0020 00690190  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
       +0000 006901B0  00102401 006B2488 006B2620 8067CB5A  80640580 00690348 006B2620 00000000  |.....,.h.,.....!........,.......|
       +0020 006901D0  006B231C 006B6169 006B25C0 006B0520  806B6000 8067CB5C 006B2488 006B0188  |.,..../..,...,...,-....*.,.h.,.h|
⋮
   COBOLZ1 (DSA ADDRESS 00501918):
     SAVED REGISTERS:
       GPR0..... 00690128  GPR1..... 005001D1  GPR2..... 0065D1AC  GPR3..... 00000000
       GPR4..... 005000B0  GPR5..... 006B0188  GPR6..... 0065D1AC  GPR7..... 00000000
       GPR8..... 0065DA70  GPR9..... 00501AE8  GPR10.... 00500174  GPR11.... 00500228
       GPR12.... 0050016C  GPR13.... 00501918  GPR14.... 50500252  GPR15.... 8067C780
   GPREG STORAGE:
     STORAGE AROUND GPR 0 (00690128)
       -0020 00690108  006B0188 8067C780 00657E48 00000000  00000000 005001DF 00000000 00501A4C  |.,.h..G...=..........&.......&.<|
       +0000 00690128  0050017C 00501AA8 00500078 006B7C08  00000000 00000000 00000000 83A6AD68  |.&.@.&.y.&...,@............cw..|
       +0020 00690148  006B0464 00690134 0069013C 00000000  00000000 00000000 00000000 C3D6C2D6  |.,.........................COBO|
⋮
     LOCAL VARIABLES:
              6 77 D-VAL            9999 COMP        00000
```

Figure 71. Sections of LE/VSE Dump for Routine COBOLZ1 (Part 2 of 2)

3. Locate statement 11 in the COBOL listing for the COBOLZ2 routine, shown in Figure 72. This is a call to the assembler routine ASSEMZ3.

```
PP 5686-068 IBM COBOL FOR VSE/ESA 1.1.1                        COBOLZ2   DATE 07/27/2001  TIME 13:55:40   PAGE     3
  LINEID  PL SL  ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7-|--+----8  MAP AND CROSS REFERENCE
/* COBOLZ2
  000001               ID DIVISION.
  000002               PROGRAM-ID. COBOLZ2.
  000003               ENVIRONMENT DIVISION.
  000004               DATA DIVISION.
  000005               WORKING-STORAGE SECTION.
  000006               77  DV-VAL PIC 9(4) USAGE COMP VALUE 0.                                BLW=0000+000        2C
  000007               LINKAGE SECTION.
  000008               77  D-VAL PIC 9(4) USAGE COMP.                                         BLL=0001+000        2C
  000009               PROCEDURE DIVISION USING D-VAL.                                        8
  000010                   MOVE D-VAL TO DV-VAL.                                              8 6
  000011                   CALL 'ASSEMZ3' USING DV-VAL.                                       EXT 6
  000012                   GOBACK.
⋮
```

Figure 72. COBOL Listing for COBOLZ2

4. Check offset +26 in the listing for the assembler routine ASSEMZ3, shown in Figure 73 on page 126.

   This shows an instruction to divide the contents of register 4 by the variable pointed to by register 6. You can see the two instructions preceding the divide instruction load register 6 from register 1, and prepare register 6 for the divide. Because of linkage conventions, you can infer that register 1 contains a pointer to a parameter list that passed to ASSEMZ3. Register 6 points to a 0 value because that was the value passed to ASSEMZ3 when it was called by a higher level routine.

Note: To translate assembler instructions, see *System/390 Reference Summary*

```
                                              External Symbol Dictionary                                   Page    2
Symbol   Type   Id     Address  Length     LD ID  Flags Alias-of                            HLASM R4.0  2001/07/27 13.55
ASSEMZ3   SD 00000001 00000000 00000088            06
                                                                                                            Page    3
  Active Usings: None
  Loc  Object Code    Addr1 Addr2  Stmt   Source Statement                                  HLASM R4.0  2001/07/27 13.55
000000                00000 00088     1 ASSEMZ3  CSECT
                                      2 ASSEMZ3  AMODE 31
                                      3 ASSEMZ3  RMODE ANY
000000 90EC D00C            0000C     4          STM   14,12,12(13)         SAVE CALLERS REGS
000004 18CF                           5          LR    12,15                SET BASE REG
                 R:C  00000           6          USING ASSEMZ3,12
000006 4120 C040            00040     7          LA    2,SAVE               CROSS CHAIN SAVE AEREAS
00000A 50D2 0004            00004     8          ST    13,4(2)              CROSS CHAIN SAVE AEREAS
00000E 5010 D008            00008     9          ST    1,8(,13)             CROSS CHAIN SAVE AEREAS
000012 50D0 2008            00008    10          ST    13,8(,2)             CROSS CHAIN SAVE AEREAS
000016 18D2                          11          LR    13,2                 NEW SAVE AEREA PTR TO R13
000018 4150 092C            0092C    12          LA    5,2348               LOW ORDER PART OF QUOTIENT
00001C 1B44                          13          SR    4,4                  HI ORDER PART OF QUOTIENT
00001E 5861 0000            00000    14          L     6,0(1)               GET POINTER TO DIVISOR
000022 4166 0000            00000    15          LA    6,0(6)               CLEAR HI BIT
000026 5D46 0000            00000    16          D     4,0(6)               DO DIVISION
00002A 58DD 0004            00004    17          L     13,4(13)             CALLERS SAVE AEREA PTR TO R13
                                     18          RETURN (14,12),RC=0        RETURN TO CALLER
                                     19+* SUPERVISOR - RETURN - 5686-032-06
00002E 41F0 0000            00000    20+         LA    15,0                  LOAD RETURN CODE          @D51IDMZ 01-RETUR
000032 58ED 000C            0000C    21+         L     14,12+4*(14+2-(14+2)/16*16)(13)                          01-RETUR
000036 980C D014            00014    22+         LM    0,12,12+4*(0+2-(0+2)/16*16)(13)                          01-RETUR
00003A 07FE                          23+         BR    14                                                       01-RETUR
00003C 00000000
000040 0000000000000000              24 SAVE     DC    9D'0'
000088                00000 00088    25 ASSEMZ3  CSECT
000000                               26          END   ASSEMZ3
⋮
```

Figure 73. Listing for ASSEMZ3

5. Check local variables for COBOLZ2 in the Local Variables section of the traceback, shown in Figure 74. From the traceback and listings, you know that COBOLZ2 called ASSEMZ3 and passed a parameter in the variable DV-VAL. The two variables DV-VAL and D-VAL have 0 values.

```
   ⋮
LOCAL VARIABLES:
          6 77 DV-VAL           9999 COMP        00000
          8 77 D-VAL            9999 COMP        00000
   ⋮
```

Figure 74. Variables Section of LE/VSE Dump for COBOLZ2

6. In the COBOLZ2 subroutine, the variable D-VAL is moved to DV-VAL, the parameter passed to the assembler routine. D-VAL appears in the Linkage section of the COBOLZ2 listing, shown in Figure 75 on page 127, indicating that the value passed from COBOLZ1 to COBOLZ2.

```
PP 5686-068 IBM COBOL FOR VSE/ESA 1.1.1                        COBOLZ2   DATE 07/27/2001  TIME 13:55:40   PAGE     3
  LINEID  PL SL  ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7-|--+----8  MAP AND CROSS REFERENCE
/* COBOLZ2
  000001               ID DIVISION.
  000002               PROGRAM-ID. COBOLZ2.
  000003               ENVIRONMENT DIVISION.
  000004               DATA DIVISION.
  000005               WORKING-STORAGE SECTION.
  000006               77  DV-VAL PIC 9(4) USAGE COMP VALUE 0.                                 BLW=0000+000        2C
  000007               LINKAGE SECTION.
  000008               77  D-VAL PIC 9(4) USAGE COMP.                                          BLL=0001+000        2C
  000009               PROCEDURE DIVISION USING D-VAL.                                         8
  000010                   MOVE D-VAL TO DV-VAL.                                               8 6
  000011                   CALL 'ASSEMZ3' USING DV-VAL.                                        EXT 6
  000012                   GOBACK.
```

Figure 75. Listing for COBOLZ2

7. In the Local Variables section of the dump for routine COBOLZ1, shown in Figure 76, D-VAL has a 0 value. This indicates that the error causing a fixed-point divide exception in ASSEMZ3 was actually caused by the value of D-VAL in COBOLZ1.

```
   ⋮
LOCAL VARIABLES:
          6 77 D-VAL            9999 COMP        00000
   ⋮
```

Figure 76. Variables Section of LE/VSE Dump for COBOLZ1

Chapter 6. Debugging PL/I Routines

This chapter contains information that can help you debug applications that contain one or more PL/I routines. Following a discussion about potential errors in PL/I routines, the first part of this chapter discusses how to use compiler-generated listings to obtain information about PL/I routines, and how to use PLIDUMP to generate an LE/VSE dump of a PL/I routine. The last part of the chapter provides examples of PL/I routines and explains how to debug them using information contained in the traceback information provided in the dump. The topics covered are listed below.
- Determining errors in PL/I routines
- Using PL/I compiler listings
- Generating an LE/VSE dump of a PL/I routine
- Finding PL/I information in a dump
- Debugging examples for PL/I routines

Determining Errors in PL/I Routines

Most errors in PL/I routines can be identified by the information provided in PL/I run-time messages, which begin with the 'IBM' prefix. For a list of these messages, see Chapter 11, "PL/I Run-Time Messages," on page 261.

A malfunction in running a PL/I routine can be caused by:
- Errors in logic in the source routine
- Invalid use of PL/I
- Unforeseen errors
- Invalid input data
- Compiler or run-time routine malfunction
- System malfunction
- Unidentified routine malfunction
- Overlaid storage

Errors in Logic in the Source Routine

Errors of this type are often difficult to detect, and sometimes it may appear as if there has been a compiler or library malfunction.
- Not converting correctly from arithmetic data
- Incorrect arithmetic and string manipulation operations
- Not matching data lists with their format lists

Invalid Use of PL/I

A misunderstanding of the language or a failure to provide the correct environment for using PL/I can result in an apparent malfunction of a PL/I routine.

Any of the following, for example, might cause a malfunction:
- Using uninitialized variables
- Using controlled variables that have not been allocated
- Reading records into incorrect structures
- Misusing array subscripts
- Misusing pointer variables
- Incorrect conversion
- Incorrect arithmetic operations

- Incorrect string manipulation operations

Unforeseen Errors

If an error is detected during run time, and no ON-unit is provided in the routine to terminate the run or attempt recovery, the job terminates abnormally. However, the status of a routine at the point where the error occurred can be recorded by using an ERROR ON-unit that contains the statements:

```
ON ERROR
 BEGIN;
  ON ERROR SYSTEM;
  CALL PLIDUMP;         /* generates a dump */
  PUT DATA;             /* displays variables */
  GOTO ENDLABEL;        /* handle condition */
 END;
```

The statement ON ERROR SYSTEM, contained in the ON-unit, ensures that further errors do not result in a permanent loop.

Invalid Input Data

A routine should contain checks to ensure that any incorrect input data is detected before it can cause the routine to malfunction.

Use the COPY option of the GET statement to check values obtained by stream-oriented input. The values are listed on the file named in the COPY option. If no file name is given, SYSPRINT is assumed.

Compiler or Run-Time Routine Malfunction

If you are certain that the malfunction is caused by a compiler or run-time routine error, see “Diagnosing Problems with LE/VSE,” on page 347 for information about determining whether the problem has been previously documented and, if not, how to report the problem to IBM. Meanwhile, you can try an alternative way to perform the operation that is causing the trouble. A bypass is often feasible, since the PL/I language frequently provides an alternative method of performing operations.

System Malfunction

System malfunctions include machine malfunctions and operating system errors. System messages identify these malfunctions and errors to the operator.

Unidentified Routine Malfunction

In most circumstances, an unidentified routine malfunction does not occur when using the compiler. If your routine terminates abnormally without an accompanying LE/VSE run-time diagnostic message, it is probable that the error causing the termination also inhibited the production of a message. This might occur:

- If your job control statements are in error, particularly in defining files.
- If your routine overwrites main storage areas containing executable instructions. This can happen if you accidentally do any of the following:
  - Assign a value to a nonexistent array element. For example:

    ```
    DCL ARRAY(10);
       ⋮
    DO I = 1 TO 100;
    ARRAY(I) = VALUE;
    ```

To detect this type of error in a compiled *module*, enable the SUBSCRIPTRANGE condition so that each attempt to access an element outside the declared range of subscript values raises the SUBSCRIPTRANGE condition. If there is no ON-unit for this condition, a diagnostic message is printed and the ERROR condition is raised. This facility, though expensive in run time and storage space, is a valuable routine-testing aid.

– Use an incorrect *locator* value for a locator (pointer or offset) variable. This type of error can occur if a locator value is obtained by means of record-oriented transmission. Ensure that locator values created in one routine, transmitted to a data set, and subsequently retrieved for use in another routine, are valid for use in the second routine.

– Attempt to free a nonbased variable. This can happen when you free a based variable after its qualifying pointer value has been changed. For example:

```
DCL A STATIC,B BASED (P);
ALLOCATE B;
P = ADDR(A);
FREE B;
```

– Use the SUBSTR pseudovariable to assign a string to a location beyond the end of the target string. For example:

```
DCL X CHAR(3);
I=3;
SUBSTR(X,2,I) = 'ABC';
```

To detect this type of error, enable the STRINGRANGE condition during compilation.

Storage Overlay Problems

If you suspect an error in your PL/I application is a storage overlay problem, check for the following:

- The use of a subscript outside the declared bounds (SUBSCRIPTRANGE)
- An attempt to assign a string to a target with an insufficient maximum length (STRINGSIZE)
- The failure of the arguments to a SUBSTR reference to comply with the rules described for the SUBSTR built-in *function* (STRINGRANGE)
- The loss of significant last high-order (left-most) binary or decimal digits during assignment to an intermediate result or variable or during an input/output operation (SIZE)
- The reading of a variable-length file record into a variable that is too small
- The misuse of a pointer variable
- The invocation of an LE/VSE callable service with fewer arguments than are required

The first four situations are associated with the indicated PL/I conditions, all of which are disabled by default. If you suspect one of these problems exists in your routine, use the appropriate condition prefix on the suspected statement or on the BEGIN or PROCEDURE statement of the containing block.

The fifth situation occurs when you read a data record into a variable that is too small. This type of problem only happens with variable-length files. You can often isolate the problem by examining the data in the file information and buffer.

The sixth situation occurs when you misuse a pointer variable. This type of storage overlay is particularly difficult to isolate. There are a number of ways pointer variables can be misused:

- When a READ statement runs with the SET option, a value is placed in a pointer. If you then run a WRITE statement or another READ SET option with another pointer, you overlay your storage if you try to use the original pointer.
- When you try to use a pointer to address storage that has already been freed, you can also cause a storage overlay.
- When you attempt to use a pointer set with the ADDR built-in function as a base for data with different attributes, you can cause a storage overlay.

The last situation occurs when an LE/VSE callable service is passed fewer arguments than its interface requires. The following example might cause a storage overlay because LE/VSE assumes that the fourth item in the argument list is the address of a feedback code, when in reality it could be residue data pointing anywhere in storage.

```
DCL CEEDATE ENTRY OPTIONS(ASM);
CALL CEEDATE(x,y,z);      /* invalid */
```

These examples illustrate valid calls:

```
DCL CEEDATE ENTRY(*,*,*,* OPTIONAL) OPTIONS(ASM);
CALL CEEDATE(x,y,z,*);    /* valid   */
CALL CEEDATE(x,y,z,fc);   /* valid   */
```

Using PL/I Compiler Listings

The following sections explain how to use PL/I compile-time options to obtain listings that contain information about your routine, and describe the contents of those listings.

Generating PL/I Listings

PL/I listings show machine instructions, constants, and external or internal addresses that the *linkage editor* resolves. This information can help you find other information, such as variable values, in a dump of a PL/I routine.

Note: From LE/VSE 1.4.2 onwards, dumps taken with run-time option `TER(UADUMP...)` or `TER(DUMP...)` are capable of generating a run-time options report, *independently of* the setting for `RPTOPTS(ON|OFF)`.

Table 32 shows compiler-generated listings that you might find helpful when you use information in dumps to debug PL/I routines. For more information about compiler output, see *IBM PL/I for VSE/ESA Programming Guide*

Table 32. Contents of Listing and Associated Compile-Time Options

Name	Contents	Compile-Time Option
Source program	Source program statements	SOURCE
Cross reference	Cross reference of names with attributes	XREF and ATTRIBUTES
Aggregate table	Names and layouts of structures and arrays	AGGREGATE
Variable map	Offsets of automatic and static internal variables (from their defining base)	MAP
Object code	Contents of the program control section in hexadecimal notation and translated into a pseudo-assembler format	LIST

Table 32. Contents of Listing and Associated Compile-Time Options (continued)

Name	Contents	Compile-Time Option
Variable Map, Object Code, Static Storage	Same as MAP and LIST options above, plus contents of static internal and static external control sections in hexadecimal notation with comments	MAP and LIST

Finding Information in PL/I Listings

Figure 77 shows an example of a PL/I routine that was compiled with the LIST and MAP compile-time options.

```
*PROCESS SOURCE, LIST, MAP;
   ⋮
                   SOURCE LISTING


    STMT


       1  EXAMPLE: PROC OPTIONS(MAIN);
       2             DCL EXTR ENTRY EXTERNAL;
       3             DCL B(2,2)  FIXED BIN(31) STATIC EXTERNAL INIT((4)0);
       4             DCL C CHAR(20) STATIC INIT('SAMPLE CONSTANT');
       5             DCL D FIXED BIN(31) STATIC;
       6             DCL E FIXED BIN(31);
       7             FETCH EXTR;
       8             CALL EXTR(A,B,C,D,E);
       9             DISPLAY(C);
      10           END;
```

Figure 77. PL/I Routine Compiled with LIST and MAP

Figure 78 on page 134 shows the output generated from this routine, including the static storage map, variable storage map, and the object code listing. The sections following this example describe the contents of each type of listing.

```
                          STATIC INTERNAL STORAGE MAP

000000  E00000C4              PROGRAM ADCON
000004  00000008              PROGRAM ADCON
000008  00000096              PROGRAM ADCON
00000C  00000096              PROGRAM ADCON
000010  00000096              PROGRAM ADCON
000014  00000000              A..IBMSJDSA
000018  00000000              A..IBMSPFRA
00001C  00000000              A..STATIC
000020  0000000000000044      LOCATOR..B
000028  0000008800140000      LOCATOR..C
000030  91E091E0              CONSTANT
000034  0A000000C5E7E3D9      FECB..EXTR
        40404040
000040  80000034              A..FECB..EXTR
000044  0000000C00000008      DESCRIPTOR
        0000000200000001
        0000000400000002
        00000001
000060  80000034              A..FECB..EXTR
000064  00000000              A..B
000068  00000000              A..A
00006C  00000020              A..LOCATOR
000070  00000028              A..LOCATOR
000074  000000A0              A..D
000078  80000000              A..E
00007C  00000000              A..ENTRY EXTR
000080  80000028              A..LOCATOR
000084
000088  E2C1D4D7D3C540C3      INITIAL VALUE..C
        D6D5E2E3C1D5E340
        40404040



                          STATIC EXTERNAL CSECTS


000000  0000000000000000      CSECT FOR EXTERNAL VARIABLE
        0000000000000000
   ⋮
                                  VARIABLE STORAGE MAP


IDENTIFIER                        LEVEL         OFFSET       (HEX)   CLASS     BLOCK

E                                     1            184          B8   AUTO      EXAMPLE
D                                     1            160          A0   STATIC    EXAMPLE
C                                     1            136          88   STATIC    EXAMPLE
A                                     1            188          BC   AUTO      EXAMPLE
   ⋮
  OBJECT LISTING
                                                            * STATEMENT NUMBER  7
                                                            000096  58 B0 C 004               L     11,4(0,12)
                                                            00009A  58 FB 0 000               L     15,PR..EXTR
* STATEMENT NUMBER  1                                       00009E  59 F0 C 064               C     15,100(0,12)
000000                            DC    C'EXAMPLE'          0000A2  47 70 2 01E               BNE   CL.5
000007                            DC    AL1(7)              0000A6  41 10 3 040               LA    1,64(0,3)
                                                            0000AA  58 F0 3 018               L     15,A..IBMSPFRA
* PROCEDURE                             EXAMPLE             0000AE  05 EF                     BALR  14,15
                                                            0000B0  58 FB 0 000               L     15,PR..EXTR
* REAL ENTRY                                                0000B4                    CL.5    EQU   *
000008  90 EC D 00C               STM   14,12,12(13)
00000C  47 F0 F 04C               B     *+72
000010  00000000                  DC    A(STMT. NO. TABLE)  * STATEMENT NUMBER  8
```

Figure 78. Compiler-Generated Listings from Example PL/I Routine (Part 1 of 2)

```
000014  000000D8                DC    F'216'
000018  00000000                DC    A(STATIC CSECT)
00001C  00000000                DC    A(SYMTAB VECTOR)
000020  00000000                DC    A(COMPILATION INFO)
000024  A8000000                DC    X'A8000000'
000028  00010100                DC    X'00010100'
00002C  00000000                DC    X'00000000'
000030  00000000                DC    X'00000000'
000034  00000000                DC    A(ENTRY LIST VECTOR)
000038  80000000                DC    X'80000000'
00003C  01000200                DC    X'01000200'
000040  00000000                DC    A(REGION TABLE)
000044  00000002                DC    X'00000002'
000048  00000000                DC    A(PRIMARY ENTRY)
00004C  00000000                DC    X'00000000'
000050  00000000                DC    X'00000000'
000054  58 30 F 010             L     3,16(0,15)
000058  58 10 D 04C             L     1,76(0,13)
00005C  58 00 F 00C             L     0,12(0,15)
000060  1E 01                   ALR   0,1
000062  55 00 C 00C             CL    0,12(0,12)
000066  47 D0 F 068             BNH   *+10
00006A  58 F0 C 074             L     15,116(0,12)
00006E  05 EF                   BALR  14,15
000070  58 E0 D 048             L     14,72(0,13)
000074  18 F0                   LR    15,0
000076  90 E0 1 048             STM   14,0,72(1)
00007A  50 D0 1 004             ST    13,4(0,1)
00007E  92 80 1 000             MVI   0(1),X'80'
000082  92 21 1 001             MVI   1(1),X'21'
000086  92 02 1 076             MVI   118(1),X'02'
00008A  41 D1 0 000             LA    13,0(1,0)
00008E  D2 03 D 054 3 030       MVC   84(4,13),48(3)
000094  05 20                   BALR  2,0

* PROCEDURE BASE
```

```
0000B4  D2 13 D 0C0 3 068              MVC   192(20,13),104(3)
0000BA  41 70 D 0BC                    LA    7,A
0000BE  50 70 D 0C0                    ST    7,192(0,13)
0000C2  41 70 D 0B8                    LA    7,E
0000C6  50 70 D 0D0                    ST    7,208(0,13)
0000CA  96 80 D 0D0                    OI    208(13),X'80'
0000CE  58 FB 0 000                    L     15,PR..EXTR
0000D2  59 F0 C 064                    C     15,100(0,12)
0000D6  47 70 2 052                    BNE   CL.6
0000DA  41 10 3 060                    LA    1,96(0,3)
0000DE  58 F0 3 018                    L     15,A..IBMSPFRA
0000E2  05 EF                          BALR  14,15
0000E4  58 FB 0 000                    L     15,PR..EXTR
0000E8                          CL.6   EQU   *
0000E8  1B 55                          SR    5,5
0000EA  41 10 D 0C0                    LA    1,192(0,13)
0000EE  05 EF                          BALR  14,15

* STATEMENT NUMBER  9
0000F0  41 10 3 080                    LA    1,128(0,3)
0000F4  58 F0 3 014                    L     15,A..IBMSJDSA
0000F8  05 EF                          BALR  14,15

* STATEMENT NUMBER  10
0000FA  18 0D                          LR    0,13
0000FC  58 D0 D 004                    L     13,4(0,13)
000100  58 E0 D 00C                    L     14,12(0,13)
000104  98 2C D 01C                    LM    2,12,28(13)
000108  05 1E                          BALR  1,14

* END PROCEDURE
00010A  07 07                          NOPR  7

* END PROGRAM
```

Figure 78. Compiler-Generated Listings from Example PL/I Routine (Part 2 of 2)

Static Storage Map

The static storage map is a formatted listing of the contents of the static internal and static external control sections. To obtain this listing, specify the MAP and LIST options in the %PROCESS statement.

To obtain a complete variable storage map and static storage map, but not a complete LIST, specify a single statement for LIST to minimize the size of the listing; for example, LIST(1).

Each line of the static storage map contains the following information:

1. Six-digit hexadecimal offset.
2. Hexadecimal text, in 8-byte sections where possible.
3. Comment, indicating the type of item to which the text refers. The comment appears on the first line of the text for an item.

Table 33 lists typical comments found in a static storage listing.

Table 33. Comments Found in a Static Storage Listing with Explanations

Comment	**Explanation**
A..*xxx*	Address constant for *xxx*.
COMPILER LABEL CL.*n*	Compiler-generated label *n*.
CONDITION CSECT	Control section for programmer-named condition.
CONSTANT	Constant.

Table 33. Comments Found in a Static Storage Listing with Explanations (continued)

Comment	Explanation
CSECT FOR EXTERNAL VARIABLE	Control section for external variable.
D..*xxx*	*Descriptor* for *xxx*.
DED..*xxx*	Data element descriptor for *xxx*.
DESCRIPTOR	Data descriptor.
ENVB	Environment control block.
FECB..*xxx*	Fetch control block for *xxx*.
DCLCB	Declare control block.
FED..*xxx*	Format element descriptor for *xxx*.
KD..*xxx*	Key descriptor for *xxx*.
LOCATOR..*xxx*	Locator for *xxx*.
ONCB	ON statement control block.
PICTURED DED..*xxx*	Pictured data element descriptor for *xxx*.
PROGRAM ADCON	Program address constant.
RD..*xxx*	Record descriptor for *xxx*.
SYMBOL TABLE ELEMENT	Symbol table address.
SYMBOL TABLE..*xxx*	Symbol table for *xxx*.
SYMTAB DED..*xxx*	Symbol table DED for *xxx*.
USER LABEL..*xxx*	Source program label for *xxx*.
xxx	Variable with name *xxx*.. If the variable is not initialized, no text appears against the comment. There is also no static offset if the variable is an array (the static offset can be calculated from the array descriptor, if required).

Variable Storage Map

The variable storage map shows how automatic and static internal variables are mapped in storage. To obtain this listing, specify the MAP compiler option in the %PROCESS statement.

For automatic and static internal variables, the variable storage map contains the following information:
- PL/I identifier name
- Level
- Storage class
- Name of the PL/I block in which it is declared
- Offset from the start of the storage area, in both decimal and hexadecimal form

If the LIST option is also specified, a map of the static internal and external control sections, called the static storage map, is also produced.

Object Code Listing

By specifying the LIST compile-time option in the %PROCESS statement, you can obtain a listing of the compiled code, known as the object code listing. This listing consists of the machine instructions and a translation of these instructions into a form that resembles assembler and includes comments, such as source program statement numbers.

To limit the size of the listing, specify a certain statement or range of statements to be listed; for example, LIST(20) or LIST(10,30).

The machine instructions are formatted into blocks of code, headed by the statement or line number in the PL/I source program listing. Generally, only executable statements appear in the listing. DECLARE statements are not normally included. To simplify understanding of the listing, the names of PL/I variables are listed, rather than the addresses that appear in the machine code. Special mnemonics are used to refer to some items, including test hooks, descriptors, and address constants.

Statements in the object code listing are ordered by block, as they are sequentially encountered in the source program. Statements in the external procedure are given first, followed by the statements in each inner block. As a result, the order of statements frequently differs from that of the source program.

Every object code listing begins with the name of the external procedure. The actual entry point of the external procedure immediately follows the heading comment `REAL ENTRY`. The subsequent machine code is the prolog for the block, which performs block activation. The comment `PROCEDURE BASE` marks the end of the prolog. Following this is a translation of the first executable statement in the PL/I source program. Table 34 contains the comments used in the listing.

Table 34. Comments and Their Functions in the Object Code Listing

Comment	Function
BEGIN BLOCK *xxx*	Indicates the start of the begin block with label *xxx*
BEGIN BLOCK NUMBER *n*	Indicates the start of the begin block with number *n*
CALCULATION OF COMMONED EXPRESSION FOLLOWS	Indicates that an expression used more than once in the routine is calculated at this point
CODE MOVED FROM STATEMENT NUMBER *n*	Indicates object code moved by the optimization process to a different part of the routine and gives the number of the statement from which it originated
COMPILER GENERATED SUBROUTINE *xxx*	Indicates the start of compiler-generated subroutine *xxx*
CONTINUATION OF PREVIOUS REGION	Identifies the point at which addressing from the previous routine base recommences
END BLOCK	Indicates the end of a begin block
END INTERLANGUAGE PROCEDURE *xxx*	Identifies the end of an ILC procedure *xxx*
END OF COMMON CODE	Identifies the end of code used in running more than one statement
END OF COMPILER GENERATED SUBROUTINE	Indicates the end of the compiler-generated subroutine
END PROCEDURE	Identifies the end of a procedure
END PROGRAM	Indicates the end of the external procedure
INITIALIZATION CODE FOR *xxx*	Indicates the start of initialization code for variable *xxx*
INITIALIZATION CODE FOR OPTIMIZED LOOP FOLLOWS	Indicates that some of the code that follows was moved from within a loop by the optimization process

Table 34. Comments and Their Functions in the Object Code Listing (continued)

Comment	Function
INTERLANGUAGE PROCEDURE *xxx*	Identifies the start of an implicitly generated ILC procedure *xxx*
METHOD OR ORDER OF CALCULATING EXPRESSIONS CHANGED	Indicates that the order of the code following was changed to optimize the object code
ON-UNIT BLOCK NUMBER *n*	Indicates the start of an ON-unit block with number *n*
ON-UNIT BLOCK END	Indicates the end of the ON-unit block
PROCEDURE *xxx*	Identifies the start of the procedure labeled *xxx*
PROCEDURE BASE	Identifies the address loaded into the base register for the procedure
PROGRAM ADDRESSABILITY REGION BASE	Identifies the address where the routine base is updated if the routine size exceeds 4096 bytes and consequently cannot be addressed from one base
PROLOGUE BASE	Identifies the start of the prolog code common to all entry points into that procedure
REAL ENTRY	Precedes the actual executable entry point for a procedure
STATEMENT LABEL *xxx*	Identifies the position of source program statement label *xxx*
STATEMENT NUMBER *n*	Identifies the start of code generated for statement number *n* in the source listing

In certain cases the compiler uses mnemonics to identify the type of operand in an instruction and, where applicable, follows the mnemonic by the name of a PL/I variable. Table 35 lists typical mnemonics.

Table 35. Mnemonics the Compiler Uses to Identify the Type of Operand

Mnemonic	Explanation
A..*xxx*	Address constant for *xxx*
ADD..*xxx*	Aggregate descriptor for *xxx*
BASE..*xxx*	Base address of variable *xxx*
BLOCK.*n*	Identifier created for an otherwise unlabeled block
CL.*n*	Compiler-generated label number *n*
D..*xxx*	Descriptor for *xxx*
DED..*xxx*	Data element descriptor for *xxx*
HOOK...ENTRY	Debugging aid block entry hook
HOOK...BLOCK-EXIT	Debugging aid block exit hook
HOOK...PGM-EXIT	Debugging aid program exit hook
HOOK...PRE-CALL	Debugging aid pre-call hook
HOOK...INFO	Additional pre-call hook information
HOOK...POST-CALL	Debugging aid post call hook
HOOK...STMT	Debugging aid statement hook
HOOK...IF-TRUE	Debugging aid IF true hook
HOOK...IF-FALSE	Debugging aid ELSE hook
HOOK...WHEN	Debugging aid WHEN true hook

Table 35. Mnemonics the Compiler Uses to Identify the Type of Operand (continued)

Mnemonic	Explanation
HOOK...OTHERWISE	Debugging aid OTHERWISE true hook
HOOK...LABEL	Debugging aid label hook
HOOK...DO	Debugging aid iterative DO hook
HOOK...ALLOC	Debugging aid ALLOCATE controlled hook
WSP.*n*	Workspace, followed by identifying number *n*
L..*xxx*	Length of variable *xxx*
PR..*xxx*	Pseudoregister vector slot for *xxx*
LOCATOR..*xxx*	Locator for *xxx*
RKD..*xxx*	Record or key descriptor for *xxx*
VO..*xxx*	*Virtual origin* for *xxx* (the address where element 0 is held for a one-dimensional array, element 0,0 for a two-dimensional array, and so on)

Generating an LE/VSE Dump of a PL/I Routine

To obtain a dump of a PL/I routine, you can either invoke the LE/VSE callable service CEE5DMP or specify a call to PLIDUMP.

PLIDUMP Syntax and Options

A call to PLIDUMP results in calls to intermediate PL/I library routines, which convert most PLIDUMP options to CEE5DMP options. In the following list of PLIDUMP options, for those that have a corresponding CEE5DMP option, the CEE5DMP option is given as the definition. For more information about these options, see Table 21 on page 32.

Some PLIDUMP options do not have corresponding CEE5DMP options, but continue to function as PL/I default options. The list following the syntax diagram provides a description of those options.

The syntax and options for PLIDUMP are shown below.

Syntax

```
►►──PLIDUMP──(──character-string-expression 1──,──────────────────────►

►──character-string-expression 2──)─────────────────────────────────►◄
```

character-string-expression 1
A dump options character string consisting of one or more of the following:

B	BLOCKS (PL/I hexadecimal dump).
C	Continue. The routine continues after the dump.
E	Exit. The enclave terminates with a dump.
F	FILES.
H	STORAGE.

Note: When the H option is specified, the VSE SDUMP system macro is used to produce a dump of storage. Output is directed either to SYSLST or the dump sublibrary.

K	BLOCKS (when running under CICS). The Transaction Work Area is included.
NB	NOBLOCKS.
NF	NOFILES.
NH	NOSTORAGE.
NK	NOBLOCKS (when running under CICS).
NT	NOTRACEBACK.
S	Stop. The enclave terminates with a dump.
T	TRACEBACK.

T, F, and C are the default options.

character-string-expression 2
A user-identified string that is printed as the dump header (the string can be up to 80 characters long, but is truncated by CEE5DMP to 60 characters).

PLIDUMP Usage Notes

If you use PLIDUMP, the following considerations apply:

- If a routine calls PLIDUMP a number of times, use a unique user-identifier for each PLIDUMP invocation. This simplifies identifying the beginning of each dump.
- Output from PLIDUMP is directed to the file PL1DUMP, PLIDUMP, or CEEDUMP, or, if you do not define one of these files in your JCL, to SYSLST.

  If PL1DUMP is available, PLIDUMP output is directed to it. If PL1DUMP is not available, PLIDUMP output is directed to PLIDUMP. If neither PL1DUMP nor PLIDUMP is available, PLIDUMP is directed to CEEDUMP. If none of these files is available, PLIDUMP output is directed to SYSLST.
- The file (PL1DUMP, PLIDUMP, or CEEDUMP) should have a logical record length (LRECL) of at least 131 to prevent dump records from wrapping.
- When you specify the H option in a call to PLIDUMP, the PL/I library issues a VSE SDUMP macro to obtain a snap dump of virtual storage. The first invocation of PLIDUMP results in a snap dump identifier of 1. For each successive invocation, the identifier is increased by one to a maximum of 256, after which it is reset to 1.
- Support for SDUMP dumps using PLIDUMP is provided only in the batch environment. SDUMP dumps are not produced in a CICS environment.
  - If the SDUMP does not succeed, the CEE5DMP DUMP file displays the message:

    ```
    Snap was unsuccessful
    ```
  - If the SDUMP is successful, CEE5DMP displays the message:

    ```
    Snap was successful; snap ID = nnn
    ```

    where *nnn* corresponds to the snap dump identifier described above.

    The identifier is incremented only if SDUMP is successful.

To ensure portability across system platforms, use PLIDUMP to generate a dump of your PL/I routine.

Finding PL/I Information in a Dump

The following sections discuss PL/I-specific information located in the following sections of an LE/VSE dump:
- Traceback
- Control Blocks for Active Routines
- Control Block Associated with the Thread
- File Status and Attributes

Traceback

Examine the traceback section of the dump, shown in Figure 79, for condition information about your routine and information about the statement number and address where the exception occurred.

```
CEE5DMP V1 R4.2: Plidump called from error ON-unit                              07/30/01 3:21:51 PM                 PAGE:    1

DUMP WAS CALLED FROM STATEMENT NUMBER 6 AT OFFSET +000000EA FROM ERR  ON-UNIT WITH ENTRY ADDRESS 005001F8

INFORMATION FOR ENCLAVE EXAMPLE

  INFORMATION FOR THREAD 8000000000000000

  TRACEBACK:
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY         E ADDR    E  OFFSET   STATEMENT  STATUS
    006A79B0  CEEKKMRA      00227FA0  +000008C4  CEEKKMRA      00227FA0  +000008C4              CALL
    006C77C0  IBMRKDM       00675258  +000000BA  IBMRKDM       00675258  +000000BA              CALL
    006A78B0  EXAMPLE       005000F8  +000001EA  ERR  ON-UNIT  005001F8  +000000EA           6  CALL
    006A76A0  IBMRERPL      0068CF98  +0000063E  IBMRERPL      0068CF98  +0000063E              CALL
    006A75B8  CEEEV010      00658800  +00000126  CEEEV010      00658800  +00000126              CALL
    006A4018  CEEHDSP       03A58D38  +000012FE  CEEHDSP       03A58D38  +000012FE              CALL
    006A7430  IBMRERRI      0068DDD8  +0000036E  IBMRERRI      0068DDD8  +0000036E              EXCEPTION
    006A7360  EXAMPLE       005000F8  +000002E2  LABL1: BEGIN  00500308  +000000D2          11  CALL
    006A7260  EXAMPLE       005000F8  +000000E0  EXAMPLE       00500100  +000000D8           8  CALL
    006A71C0  IBMRPMIA      00692E60  +0000028E  IBMRPMIA      00692E60  +0000028E              CALL
    006A70D8  CEEEV010      00658800  +00000282  CEEEV010      00658800  +00000282              CALL
    006A7018  CEEBBEXT      001FA3E0  +00000132  CEEBBEXT      001FA3E0  +00000132              CALL

  CONDITION INFORMATION FOR ACTIVE ROUTINES
    CONDITION INFORMATION FOR IBMRERRI (DSA ADDRESS 006A7430)
      CIB ADDRESS: 006A4528
      CURRENT CONDITION:
        IBM0281I A PRIOR CONDITION WAS PROMOTED TO THE 'ERROR' CONDITION
      ORIGINAL CONDITION:
        IBM0421I 'ONCODE'=520  'SUBSCRIPTRANGE' CONDITION RAISED
      LOCATION:
        PROGRAM UNIT: IBMRERRI ENTRY: IBMRERRI STATEMENT:  OFFSET: +0000036E
    STORAGE DUMP NEAR CONDITION, BEGINNING AT LOCATION: 0068E136
      +000000 0068E136  5030D080 58A0C2B8 58F0A01C 4110D080  05EF9108 204F4710 B3849104 204F47E0  |&.....B..0........j..|...dj..|..|
...
```

Figure 79. Traceback Section of Dump

Condition Information

If the dump was called from an ON-unit, the type of ON-unit is identified in the traceback as part of the entry information. For ON-units, the values of any relevant condition built-in functions (for example, ONCHAR and ONSOURCE for conversion errors) appear. In cases where the cause of entry into the ON-unit is not stated, usually when the ERROR ON-unit is called, the cause of entry appears in the condition information.

Statement Number and Address Where Error Occurred

This information, which is the point at which the condition that caused entry to the ON-unit occurred, can be found in the traceback section of the dump.

If the condition occurs in compiled code, and you compiled your routine with either GOSTMT or GONUMBER, the statement numbers appear in the dump. To identify the assembler instruction that caused the error, use the traceback information in the dump to find the program unit (PU) offset of the statement number in which the error occurred. Then find that offset and the corresponding instruction in the object code listing.

Control Blocks for Active Routines

This section shows the stack frames for all active routines, and the *static storage*. Use this section of the dump to identify variable values, determine the contents of parameter lists, and locate the timestamp.

Figure 80 shows this section of the dump.

```
CEE5DMP V1 R4.2: Plidump called from error ON-unit                                  07/30/01 3:21:51 PM                 PAGE:    1

...
  CONTROL BLOCKS FOR ACTIVE ROUTINES:
    DSA FOR CEEKKMRA: 006A79B0
      +000000  FLAGS.... 0000      member... 40C3      BKC...... 006C77C0  FWC...... 006A7EF8  R14...... 80696EEE
      +000010  R15...... 83A01D68  R0....... 00000001  R1....... 006A7A40  R2....... 006C7338  R3....... 0000003C
      +000024  R4....... 00000001  R5....... 006A7C90  R6....... 006C7180  R7....... 00000000  R8....... 00696038
      +000038  R9....... 00000000  R10...... 00228F9F  R11...... 80227FA0  R12...... 00657E48  reserved. 006C78B0
      +00004C  NAB...... 006A7EF8  PNAB..... D6D94040  reserved. 40404040  40404040  40404040  40404040
      +000064  reserved. 40404040  reserved. 40404040  MODE..... 80228866  reserved. 40404040  40404040
      +000078  reserved. 40404040  reserved. 40404040
...
    DSA FOR ERR  ON-UNIT: 006A78B0
      +000000  FLAGS.... CC25      member... AC8F      BKC...... 006A76A0  FWC...... 00657E48  R14...... 8068A45A
      +000010  R15...... 80675258  R0....... 006A79B0  R1....... 0050049C  R2....... 80500284  R3....... 00500420
      +000024  R4....... 006A7260  R5....... 00000000  R6....... 006A7968  R7....... 006A7788  R8....... 006A7980
      +000038  R9....... 006A79A4  R10...... 00000064  R11...... 0068DF97  R12...... 03A0658C  reserved. 006C77C0
      +00004C  NAB...... 006A79B0  PNAB..... 006A79B0  reserved. 91E091E0  006A7260  005004E0  03A065BC
      +000064  reserved. 006A7950  reserved. 006A7950  MODE..... 805002E4  reserved. 006A7968  006A0224
      +000078  reserved. 006A77C4  reserved. 006A77C8
    DYNAMIC STORAGE AREA (ERR  ON-UNIT): 006A78B0
      +000000 006A78B0  CC25AC8F 006A76A0 00657E48 8068A45A  80675258 006A79B0 0050049C 80500284  |..........=...u!.........&...&.d|
      +000020 006A78D0  00500420 006A7260 00000000 006A7968  006A7788 006A7980 006A79A4 00000064  |.&.....-...........h.......u....|
      +000040 006A78F0  0068DF97 03A0658C 006C77C0 006A79B0  006A79B0 91E091E0 006A7260 005004E0  |...p.....%..........j.j....-.&..|
      +000060 006A7910  03A065BC 006A7950 006A7950 805002E4  006A7968 006A0224 006A77C4 006A77C8  |.......&...&.&.U...........D...H|
      +000080 006A7930  006A77CC 006A7B64 006A77D0 006A77D8  006A77DC 006A77E0 006A7D58 00000000  |......#........Q..........'.....|
      +0000A0 006A7950  00000000 00000000 00000000 0E654940  40404040 40404040 0C014040 40404040  |...............         ..      |
      +0000C0 006A7970  A3829586 A2404040 006A7970 00050000  D7938984 A4949740 83819393 85844086  |tbnfs   ........Plidump called f|
      +0000E0 006A7990  99969440 85999996 9940D6D5 60A49589  A340C9D5 006A7980 00210000 C1D5C4D3  |rom error ON-unit IN........ANDL|
    STATIC FOR PROCEDURE EXAMPLE    TIMESTAMP: 30 JULY 01 15:21:49
    STARTING FROM: 00500420
      +000000 00500420  E00001E4 00500100 005001A8 005001B8  005001F8 00500284 00500308 0050037E  |...U.&...&.y.&...&.8.&.d.&...&.=|
      +000020 00500440  0050037E 0050037E 0050037E 00501420  B0000002 1F800004 00000000 00500474  |.&.=.&.=.&.=.&...............&..|
      +000040 00500460  00000000 00050000 00000000 00210000  91E091E0 00000004 00000004 0000000A  |................j.j.............|
      +000060 00500480  00000001 91A091A0 00000014 00000001  0A200000 0000000A 00000002 006A7978  |....j.j.........................|
      +000080 005004A0  806A79A4 005010F0 A3829586 A2D79389  84A49497 40838193 93858440 86999694  |...u.&.0tbnfsPlidump called from|
      +0000A0 005004C0  40859999 969940D6 D560A495 89A30000  005001C4 005001B8 0C160000 005001F8  | error ON-unit...&.D.&.......&.8|
      +0000C0 005004E0  0C960000 00000000 00500620 00000000  00000000 0050051C 00500530 0050054C  |.o.......&...........&...&...&.<|
      +0000E0 00500500  00500564 00000000 005004E8 00000000  005004F4 00000000 005004F4 85000001  |.&.......&.Y.....&.4.....&.4e...|
      +000100 00500520  00500452 000000C8 00000000 0001C900  85000001 00500452 000000CC 00000000  |.&.....H.......I.e....&.........|
      +000120 00500540  0009C1D9 D9C1E86D C5D5C400 81000101  00500452 000000C0 00000000 0005C1D9  |..ARRAY_END.a....&............AR|
      +000140 00500560  D9C1E800 0D020001 00500450 005004D0  00000000 0005D3C1 C2D3F100 00000001  |RAY......&.&.&........LABL1.....|
      +000160 00500580  00500100 000000F6 005005A4 00000001  00BD0004 00C40008 00C90008 00DF000E  |.&.....6.&.u.........D...I......|
      +000180 005005A0  00F80004 005001F8 00000108 005005C4  00000004 00910005 00990006 00F10007  |.8...&.8.....&.D.....j...r...1..|
      +0001A0 005005C0  01080008 00500308 00000116 005005EC  00000008 007B0009 008B000A 00AF000B  |.....&.......&.......#..........|
      +0001C0 005005E0  00E5000C 0105000D 0104000D 0E0E0E0E  F3F040D1 E4D3E840 F0F140F1 F57AF2F1  |.V..............30 JULY 01 15:21|
      +0001E0 00500600  7AF4F940 80000011 00500078 00000000  01000001 00500100 005010C8 00000000  |:49 .....&...........&...&.H....|
```

Figure 80. Control Blocks for Active Routines Section of the Dump

Automatic Variables

To find automatic variables, use an offset from the stack frame of the block in which they are declared. This information appears in the variable storage map generated when the MAP compile-time option is in effect. If you have not used the MAP option, you can determine the offset by studying the listing of compiled code instructions.

Static Variables

If your routine is compiled with the MAP option, you can find static variables by using an offset in the variable storage map. If the MAP option is not in effect, you can determine the offset by studying the listing of compiled code.

Based Variables

To locate based variables, use the value of the defining pointer. Find this value by using one of the methods described above to find static and automatic variables. If the pointer is itself based, you must find its defining pointer and follow the chain until you find the correct value.

The following is an example of typical code for X BASED (P), with P AUTOMATIC:

```
58 60 D 0C8       L 6,P

58 E0 6 000       L 14,X
```

P is held at offset X'000000C8' from register 13. This address points to X.

Take care when examining a based variable to ensure that the pointers are still valid.

Area Variables

Area variables are located using one of the methods described above, according to their storage class.

The following is an example of typical code:

For an area variable A declared AUTOMATIC

```
41 60 D 0F8       LA 6,A
```

The area starts at offset X'000000F8' from register 13.

Variables in Areas

To find variables in areas, locate the area and use the offset to find the variable.

Contents of Parameter Lists

To find the contents of a passed parameter list, first find the register 1 value in the save area of the calling routine's stack frame. Use this value to locate the parameter list in the dump. If R1=0, no parameters were passed. For additional information about parameter lists, see *IBM PL/I for VSE/ESA Programming Guide*

Timestamp

If the TSTAMP compiler installation option is in effect, the date and time of compilation appear within the last 32 bytes of the static internal control section. The last three bytes of the first word give the offset to this information. The offset indicates the end of the timestamp. Register 3 addresses the static internal control section. If the BLOCK option is in effect, the timestamp appears in the static storage section of the dump.

Control Blocks Associated with the Thread

This section of the dump, shown in Figure 81 on page 144, includes information about PL/I fields of the CAA and other control block information.

```
CEE5DMP V1 R4.2: Plidump called from error ON-unit                                07/30/01 3:21:51 PM                PAGE:    1

:
  CONTROL BLOCKS ASSOCIATED WITH THE THREAD:
    CAA: 00657E48
      +000000 00657E48  00000000 006C77A8 006A7000 006C7000  00000000 00657E58 00000000 006C7300  |.....%.y.....%........=......%..|
      +000020 00657E68  00000000 00000000 006C7180 00000000  006C7390 00000000 006C7358 00000000  |.........%.......%.......%......|
      +000040 00657E88  006C7320 00000000 006929A0 00000000  00693220 0068CE70 00000000 00000000  |.%..............................|
      +000060 00657EA8  00000000 0068CDC8 006C77A8 00692530  00692858 80204A90 0068DDD8 10001111  |.......H.%.y...............Q....|
:
    DUMMY DSA: 00658608
      +000000  FLAGS.... 0000      member... 0000      BKC...... 00438000  FWC...... 006A7018  R14...... 80500B3C
      +000010  R15...... 801FA3E0  R0....... 00500078  R1....... 001F9160  R2....... 00000000  R3....... 00000000
      +000024  R4....... 00000000  R5....... 00000000  R6....... 00000000  R7....... 00696038  R8....... 00500610
      +000038  R9....... 00438618  R10...... 00000000  R11...... 80500A12  R12...... 00657E48  reserved. 006C73E0
      +00004C  NAB...... 006A7018  PNAB..... 006A7018  reserved. 00000000  00000000  00000000  00000000
      +000064  reserved. 00000000  reserved. 00000000  MODE..... 00000000  reserved. 00000000  00000000
      +000078  reserved. 00000000  reserved. 00000000
    TCA IMPLEMENTATION APPENDAGE (TIA): 006C7180
      +000000 006C7180  00000000 00000000 00000000 00000030  00000000 00000000 00000000 00000000  |................................|
      +000020 006C71A0  006C7338 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |.%..............................|
      +000040 006C71C0  00000000 00000000 00000000 00000000  00657E48 00000000 00000000 00000000  |.................=..............|
      +000060 006C71E0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
      +000080 006C7200 - +0000DF 006C725F            SAME AS ABOVE
      +0000E0 006C7260  00000000 00000000 00000000 00000000  00000000 00000000 00008000 00000000  |................................|
      +000100 006C7280  00000000 00000000 00000000 00000000  00000000 00000000 00000000 006C9034  |.............................%..|

  ENCLAVE CONTROL BLOCKS:
    EDB: 00657060
      +000000 00657060  C3C5C5C5 C4C24040 C0000001 00657D08  006575F8 00000000 00000000 00000000  |CEEEDB  ......'....8............|
      +000020 00657080  006574B0 006574E0 00696038 00656D40  00000000 00000000 001F9160 00008000  |..........-..._ ..........j-....|
      +000040 006570A0  00000000 00000000 00438000 00000000  00000000 00000000 00657120 006570B8  |................................|
      +000060 006570C0  00000000 00000000 00000000 00000000  00000000 00000000 03A892F8 00657E48  |.........................yk8..=.|
      +000080 006570E0  40000000 00000000 00000000 00000000  00000001 00000000 00000000 00000000  | ...............................|
    MEML: 00657D08
      +000000 00657D08  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
      +000020 00657D28 - +00009F 00657DA7            SAME AS ABOVE
      +0000A0 00657DA8  00000000 00000000 80658800 00000000  00000000 00000000 03A41338 00000000  |..........h..............u......|
      +0000C0 00657DC8  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
      +0000E0 00657DE8 - +00011F 00657E27            SAME AS ABOVE

PROCESS CONTROL BLOCKS:

  PCB: 00656D40
    +000000 00656D40  C3C5C5D7 C3C24040 04030290 00000000  00000000 00000000 00656F38 03A8A6B0  |CEEPCB  ..................?..yw.|
    +000020 00656D60  03A86838 03A89690 03A891C8 006469C8  00656118 00000000 00000000 00657060  |.y...yo..yjH...H../............-|
    +000040 00656D80  03A894E8 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |.ymY............................|

  MEML: 00656F38
    +000000 00656F38  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
    +000020 00656F58 - +00009F 00656FD7            SAME AS ABOVE
    +0000A0 00656FD8  00640838 00000000 80658800 00000000  00000000 00000000 03A41338 00000000  |..........h..............u......|
    +0000C0 00656FF8  00000000 00000000 03A41338 00000000  00000000 00000000 03A41338 00000000  |.........u...............u......|
    +0000E0 00657018 - +00011F 00657057            SAME AS ABOVE

  PL/I PROCESS CONTROL BLOCK: 00640838
    +000000 00640838  E9D7D9C2 40404040 00640800 00000000  00000000 00000000 0068CDC8 00000000  |ZPRB    ...................H....|
    +000020 00640858  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
    +000040 00640878 - +00005F 00640897            SAME AS ABOVE
```

Figure 81. Control Blocks Associated with the Thread Section of the Dump

The CAA

The address of the CAA control block appears in this section of the dump. If the BLOCK option is in effect, the complete CAA (including the PL/I implementation appendage) appears separately from the body of the dump. Register 12 addresses the CAA.

File Status and Attribute Information

This part of the dump includes the following information:
- The default and declared attributes of all open files
- Buffer contents of all file buffers

- The contents of FCBs, DTFs, DCLCBs, IOCBs, and control blocks for the process or enclave

Debugging Example PL/I Routines

This section contains examples of PL/I routines and instructions for using information in the LE/VSE dump to debug them. Important areas in the source code and in the dump for each routine are highlighted.

Subscript Range Error

Figure 82 illustrates an error caused by an array subscript value outside the declared range. In this example, the declared array value is 10.

This routine was compiled with the options `LIST`, `TEST(ALL)`, `GOSTMT`, and `MAP`. In the ERROR ON-unit, PLIDUMP is called with option T to generate a traceback for the condition.

```
5686-069 IBM PL/I FOR VSE/ESA          VER 1 REL 1 MOD 0 with MLE                30 JULY 2001  15:21:49     PAGE   1
OPTIONS SPECIFIED
*PROCESS INCLUDE,SYSTEM(VSE),GOSTMT,MAP,LIST,TEST(ALL);
*PROCESS LINECOUNT(100);
⋮
5686-069 IBM PL/I FOR VSE/ESA          EXAMPLE:  PROC  OPTIONS(MAIN);                                       PAGE   2
                    SOURCE LISTING
    STMT

       1  EXAMPLE:  PROC  OPTIONS(MAIN);
       2   DCL Array(10) Fixed bin(31);
       3   DCL (I,Array_End)   Fixed bin(31);
       4   On error
            Begin;
       5      On error system;
       6      Call plidump('tbnfs','Plidump called from error ON-unit');
       7    End;
       8  (subrg):              /* Enable subscriptrange condition */
            Labl1: Begin;
       9      Array_End = 20;
      10      Do I = 1 to Array_End;   /* Loop to initialize array     */
      11        Array(I) = 2;    /* Set array elements to 2         */
      12      End;
      13    End Labl1;
      14  End Example;
⋮
5686-069 IBM PL/I FOR VSE/ESA          EXAMPLE:  PROC  OPTIONS(MAIN);                                       PAGE   4
                                  VARIABLE STORAGE MAP
IDENTIFIER                        LEVEL        OFFSET       (HEX)   CLASS     BLOCK

I                                   1           200          C8   AUTO      EXAMPLE
ARRAY_END                           1           204          CC   AUTO      EXAMPLE
ARRAY                               1           208          D0   AUTO      EXAMPLE
⋮
```

Figure 82. Example of Moving a Value Outside an Array Range

Figure 83 shows sections of the dump generated by a call to PLIDUMP.

```
CEE5DMP V1 R4.2: Plidump called from error ON-unit                                07/30/01 3:21:51 PM              PAGE:    1

DUMP WAS CALLED FROM STATEMENT NUMBER 6 AT OFFSET +000000EA FROM ERR  ON-UNIT WITH ENTRY ADDRESS 005001F8

INFORMATION FOR ENCLAVE EXAMPLE

  INFORMATION FOR THREAD 8000000000000000

  TRACEBACK:
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY        E ADDR    E  OFFSET   STATEMENT  STATUS
    006A79B0  CEEKKMRA      00227FA0  +000008C4  CEEKKMRA     00227FA0  +000008C4              CALL
    006C77C0  IBMRKDM       00675258  +000000BA  IBMRKDM      00675258  +000000BA              CALL
    006A78B0  EXAMPLE       005000F8  +000001EA  ERR  ON-UNIT 005001F8  +000000EA           6  CALL
    006A76A0  IBMRERPL      0068CF98  +0000063E  IBMRERPL     0068CF98  +0000063E              CALL
    006A75B8  CEEEV010      00658800  +00000126  CEEEV010     00658800  +00000126              CALL
    006A4018  CEEHDSP       03A58D38  +000012FE  CEEHDSP      03A58D38  +000012FE              CALL
    006A7430  IBMRERRI      0068DDD8  +0000036E  IBMRERRI     0068DDD8  +0000036E              EXCEPTION
    006A7360  EXAMPLE       005000F8  +000002E2  LABL1: BEGIN 00500308  +000000D2          11  CALL
    006A7260  EXAMPLE       005000F8  +000000E0  EXAMPLE      00500100  +000000D8           8  CALL
    006A71C0  IBMRPMIA      00692E60  +0000028E  IBMRPMIA     00692E60  +0000028E              CALL
    006A70D8  CEEEV010      00658800  +00000282  CEEEV010     00658800  +00000282              CALL
    006A7018  CEEBBEXT      001FA3E0  +00000132  CEEBBEXT     001FA3E0  +00000132              CALL

  CONDITION INFORMATION FOR ACTIVE ROUTINES
    CONDITION INFORMATION FOR IBMRERRI (DSA ADDRESS 006A7430)
      CIB ADDRESS: 006A4528
      CURRENT CONDITION:
        IBM0281I A PRIOR CONDITION WAS PROMOTED TO THE 'ERROR' CONDITION
      ORIGINAL CONDITION:
        IBM0421I 'ONCODE'=520  'SUBSCRIPTRANGE' CONDITION RAISED
      LOCATION:
        PROGRAM UNIT: IBMRERRI ENTRY: IBMRERRI STATEMENT:  OFFSET: +0000036E
    STORAGE DUMP NEAR CONDITION, BEGINNING AT LOCATION: 0068E136
      +000000 0068E136  5030D080 58A0C2B8 58F0A01C 4110D080  05EF9108 204F4710 B3849104 204F47E0  |&.....B..0........j..|...dj..|..|

  CONTROL BLOCKS FOR ACTIVE ROUTINES:
:
    DYNAMIC STORAGE AREA (EXAMPLE): 006A7260
      +000000 006A7260  C0257060 006A71C0 00000000 805001DA  00500308 006A7360 006A7260 805001B8  |...-.........&...&.....-...-.&..|
      +000020 006A7280  00500420 00000001 006A7260 006A7330  00500614 005004D8 006A7098 00500614  |.&.........-.....&...&.Q...q.&..|
      +000040 006A72A0  00000003 00657E48 006C73E0 006A7360  006A7360 91E091A0 8800000C 005004D8  |......=..%.....-...-j.j.h....&.Q|
      +000060 006A72C0  006A74D8 8069B03E 83A58B30 00000000  006A7318 00690238 0065F868 006579B8  |...Q....cv................8.....|
      +000080 006A72E0  00657060 00640838 00657E58 006C72ED  006C72A0 00660867 00689038 00657E48  |...-......=..%...%...........=.|
      +0000A0 006A7300  00000000 006A74D8 00000000 00000080  00000000 00000000 0C010000 00000000  |.......Q........................|
      +0000C0 006A7320  006A7330 00500474 0000000B 00000014  00000002 00000002 00000002 00000002  |.....&..........................|
      +0000E0 006A7340  00000002 00000002 00000002 00000002  00000002 00000002 00000000 00000000  |................................|
```

Figure 83. Sections of the LE/VSE Dump

To debug this routine, use the following steps:

1. In the dump, PLIDUMP was called by the ERROR ON-unit in statement 6. The traceback information in the dump shows that the exception occurred following statement 11.
2. Locate the Original Condition message in the Condition Information for Active Routines section of the dump. The message is `IBM0421S ONCODE=520 SUBSCRIPTRANGE CONDITION RAISED`. This message indicates that the exception occurred when an array element value exceeded the subscript range value (in this case, 10). For more information about this message, see Chapter 11, “PL/I Run-Time Messages,” on page 261.
3. Locate statement 9 in the routine in Figure 82 on page 145. The instruction is `Array_End = 20`. This statement assigns a 20 value to the variable `Array_End`.
4. Statement 10 begins the DO-loop instruction `Do I = 1 to Array_End`. Since the previous instruction (statement 9) specified that `Array_End = 20`, the loop in statement 10 should run until I reaches a 20 value.

The instruction in statement 2, however, declared a 10 value for the array range. Therefore, when the I value reached 11, the SUBSCRIPTRANGE condition was raised.

Note: The following steps provide another method for finding the value that raised the SUBSCRIPTRANGE condition.

1. Locate the offset of variable I in the variable storage map in Figure 82 on page 145. Use this offset to find the I value at the time of the dump. In this example, the offset is X'000000C8'.
2. Now find offset X'000000C8' from the start of the stack frame in Figure 83 on page 146.

   The block located at this offset contains the value that exceeded the array range, X'B' or 11.

Calling a Nonexistent Subroutine

Figure 84 demonstrates the error of calling a nonexistent subroutine. This routine was compiled with the LIST, MAP, and GOSTMT compile-time options. In the ERROR ON-unit, PLIDUMP is called with option T to generate a traceback.

```
5686-069 IBM PL/I FOR VSE/ESA          VER 1 REL 1 MOD 0 with MLE                 2 MAR 2001    07:17:15     PAGE   1
OPTIONS SPECIFIED
*PROCESS INCLUDE,SYSTEM(VSE),GOSTMT,MAP,LIST;
   ⋮
5686-069 IBM PL/I FOR VSE/ESA          EXAMPLE1:  PROC  OPTIONS(MAIN);                                       PAGE   2
                    SOURCE LISTING
    STMT

       1  EXAMPLE1:  PROC  OPTIONS(MAIN);
       2   DCL Prog01      entry external;
       3   On error
             Begin;
       4       On error system;
       5       Call plidump('tbnfs','Plidump called from error ON-unit');
       6     End;
       7   Call Prog01;              /* Call external program PROG01   */
       8  End Example1;
```

Figure 84. Example of Calling a Nonexistent Subroutine

The routine in Figure 84 was compiled with the LIST compile-time option, which generated an object code listing, part of which is shown in Figure 85 on page 148.

```
* STATEMENT NUMBER  7
0000B8  1B 11                SR    1,1
0000BA  1B 55                SR    5,5
0000BC  58 F0 3 040          L     15,64(0,3)
0000C0  05 EF                BALR  14,15


* STATEMENT NUMBER  8
0000C2  18 0D                LR    0,13
0000C4  58 D0 D 004          L     13,4(0,13)
0000C8  58 E0 D 00C          L     14,12(0,13)
0000CC  98 2C D 01C          LM    2,12,28(13)
0000D0  05 1E                BALR  1,14

* END PROCEDURE
```

Figure 85. Object Code Listing from Example PL/I Routine

Figure 86 shows the traceback and condition information from the dump.

```
CEE5DMP V1 R4.2: Plidump called from error ON-unit                        07/30/01 3:16:49 PM               PAGE:    1

DUMP WAS CALLED FROM STATEMENT NUMBER 5 AT OFFSET +000000BE FROM ERR  ON-UNIT WITH ENTRY ADDRESS 005001CC

INFORMATION FOR ENCLAVE EXAMPLE1

  INFORMATION FOR THREAD 8000000000000000

  TRACEBACK:
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY         E ADDR    E  OFFSET   STATEMENT  STATUS
    006A7718  CEEKKMRA      00227FA0  +000008C4  CEEKKMRA      00227FA0  +000008C4              CALL
    006C78A0  IBMRKDM       00675258  +000000BA  IBMRKDM       00675258  +000000BA              CALL
    006A7618  EXAMPLE1      005000F8  +00000192  ERR  ON-UNIT  005001CC  +000000BE           5  CALL
    006A7408  IBMRERPL      0068CF98  +0000063E  IBMRERPL      0068CF98  +0000063E              CALL
    006A7320  CEEEV010      00658800  +00000126  CEEEV010      00658800  +00000126              CALL
    006A4018  CEEHDSP       03A58D38  +000012FE  CEEHDSP       03A58D38  +000012FE              CALL
    006A7260  EXAMPLE1      005000F8  -005000F6  EXAMPLE1      00500104  -00500102              EXCEPTION
    006A71C0  IBMRPMIA      00692E60  +0000028E  IBMRPMIA      00692E60  +0000028E              CALL
    006A70D8  CEEEV010      00658800  +00000282  CEEEV010      00658800  +00000282              CALL
    006A7018  CEEBBEXT      001FA3E0  +00000132  CEEBBEXT      001FA3E0  +00000132              CALL

  CONDITION INFORMATION FOR ACTIVE ROUTINES
    CONDITION INFORMATION FOR EXAMPLE1 (DSA ADDRESS 006A7260)
      CIB ADDRESS: 006A4528
      CURRENT CONDITION:
        CEE3201S THE SYSTEM DETECTED AN OPERATION EXCEPTION.
      LOCATION:
        PROGRAM UNIT: EXAMPLE1 ENTRY: EXAMPLE1 STATEMENT:  OFFSET: -005000F6
      MACHINE STATE:
        ILC..... 0002    INTERRUPTION CODE..... 0001
        PSW..... 071D0E00 80000002
        GPR0..... 006A7320  GPR1..... 00000000  GPR2..... 805001AC  GPR3..... 005002A0
        GPR4..... 00000001  GPR5..... 00000000  GPR6..... 006A7318  GPR7..... 0050038C
        GPR8..... 00500310  GPR9..... 006A7098  GPR10.... 0050038C  GPR11.... 00000003
        GPR12.... 00657E48  GPR13.... 006A7260  GPR14.... 805001BA  GPR15.... 00000000
    STORAGE DUMP NEAR CONDITION, BEGINNING AT LOCATION: 00000000
      +000000 00000000  INACCESSIBLE STORAGE.
⋮
```

Figure 86. Sections of the LE/VSE Dump

To debug this example routine, use the following steps:

1. Find the Current Condition message in the Condition Information for Active Routines section of the dump. The message is `CEE3201S The system detected an operation exception.` For more information abo ut this message, see Chapter 8, “LE/VSE Run-Time Messages,” on page 163.
2. The Condition Information for Active Routines section of the dump provides the name of the active routine and, if available, the current statement number

at the time the condition occurred. In this example, because the condition occurred as a result of a call to a nonexistent subroutine, the statement number is not available. When the statement number is unavailable, you can use the address in the PSW and the instruction length code (ILC) from the Machine State section of the dump to determine where the condition occurred.

Subtract the ILC (X'0002') from the address in the PSW (X'00000002') to get the address of the instruction that caused the condition. If your system is operating in extended-addressing mode, ignore the high-order bit of the address in the PSW. If your system is operating in 370 mode, ignore the high-order byte of the address in the PSW.

3. In this example, the condition occurred at address X'00000000'. An instruction address of X'00000000' usually indicates a call to a nonexistent subroutine. Standard linkage convention means that the address of the instruction immediately following the call to the nonexistent subroutine is in register 14. In this example, the address in register 14 is X'005001BA'.
4. Determine the PU address of the routine in control (EXAMPLE1) from the traceback listing. Subtract the PU address (X'005000F8') from the address in register 14 to obtain the offset of the instruction following the call. Use the resultant offset (X'000000C2') to locate the instruction following the call in the compiler-generated object listing. As Figure 85 on page 148 shows, the statement at offset X'000000C2' is statement 8. Therefore, you can see from the compilation listing in Figure 84 on page 147 that the statement in error is statement 7, the call to Prog01.
5. Check the linkage editor output for error messages.

Divide-by-Zero Error

Figure 87 on page 150 demonstrates a divide-by-zero error. In this example, the main PL/I routine passed bad data to a PL/I subroutine. The bad data in this example is 0, and the error occurred when the subroutine SUB1 attempted to use this data as a divisor.

```
                SOURCE LISTING
    STMT

       1   SAMPLE: PROC  OPTIONS(MAIN) ;
       2    On error
            begin;
       3      On error system;         /* prevent nested error conditioNS */
       4      Call PLIDUMP('TBC','PLIDUMP called from error ON-unit');
       5      Put Data;                /* Display variables               */
       6    End;
       7    DECLARE
              A_number    Fixed Bin(31),
              My_name     Char(13),
              An_Array(3) Fixed Bin(31) init(1,3,5);
       8    Put skip list('Sample Starting');
       9    A_number = 0;
      10    My_name = 'G. Sutherland';
      11    Call Sub1(a_number, my_name, an_array);
      12      SUB1:  PROC(divisor, name1, Array1);
      13        Declare
                  Divisor   Fixed Bin(31),
                  Name1     Char(13),
                  Array1(3) Fixed Bin(31);
      14        Put skip list('Sub1 Starting');
      15        Array1(1) = Array1(2) / Divisor;
      16        Put skip list('Sub1 Ending');
      17      End SUB1;
      18    Put skip list('Sample Ending');
      19   End;
```

Figure 87. PL/I Routine with a Divide-by-Zero Error

Since variables are not normally displayed in a PLIDUMP dump, this routine included a PUT DATA statement, which generated a listing of arguments and variables used in the routine. Figure 88 shows this output.

```
Sample Starting
Sub1 Starting           A_NUMBER=            0 MY_NAME='G. Sutherland' AN_ARRAY(1)=            1
AN_ARRAY(2)=             3                       AN_ARRAY(3)=            5;
```

Figure 88. Variables from Routine SAMPLE

The routine in Figure 87 was compiled with the LIST compile-time option, and run with the GOSTMT and MAP run-time options (otherwise, the ERR ON-UNIT statement number is not generated in the traceback).

The object code listing shown in Figure 89 on page 151 was generated.

```
* STATEMENT NUMBER  15
000372  58 B0 D 0C8           L     11,200(0,13)
000376  58 40 B 004           L     4,4(0,11)
00037A  58 90 3 0AC           L     9,172(0,3)
00037E  5C 80 4 004           M     8,4(0,4)
000382  58 70 3 0CC           L     7,204(0,3)
000386  5C 60 4 004           M     6,4(0,4)
00038A  58 80 D 0C0           L     8,192(0,13)
00038E  58 60 B 000           L     6,0(0,11)
000392  5F 60 4 000           SL    6,0(0,4)
000396  58 E7 6 000           L     14,VO..ARRAY1(7)
00039A  8E E0 0 020           SRDA  14,32
00039E  5D E0 8 000           D     14,DIVISOR
0003A2  50 F9 6 000           ST    15,VO..ARRAY1(9)
```

Figure 89. Object Code Listing from Example PL/I Routine

Figure 90 on page 152 shows the LE/VSE dump for routine SAMPLE.

```
CEE5DMP V1 R4.2: PLIDUMP called from error ON-unit                 07/30/01 3:22:01 PM                  PAGE:    1

DUMP WAS CALLED FROM STATEMENT NUMBER 4 AT OFFSET +000000BE FROM ERR  ON-UNIT WITH ENTRY ADDRESS 005002A4

INFORMATION FOR ENCLAVE SAMPLE

  INFORMATION FOR THREAD 8000000000000000

  TRACEBACK:
    DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY         E ADDR    E  OFFSET   STATEMENT  STATUS
    006A78B0  CEEKKMRA      00227FA0  +000008C4  CEEKKMRA      00227FA0  +000008C4              CALL
    006C78A0  IBMRKDM       00675258  +000000BA  IBMRKDM       00675258  +000000BA              CALL
    006A7788  SAMPLE        005000F8  +0000026A  ERR  ON-UNIT  005002A4  +000000BE           4  CALL
    006A7578  IBMRERPL      0068CF98  +0000063E  IBMRERPL      0068CF98  +0000063E              CALL
    006A7490  CEEEV010      00658800  +00000126  CEEEV010      00658800  +00000126              CALL
    006A4018  CEEHDSP       03A58D38  +000012FE  CEEHDSP       03A58D38  +000012FE              CALL
    006A7390  SAMPLE        005000F8  +0000039E  SUB1          005003C0  +000000D6          15  EXCEPTION
    006A7260  SAMPLE        005000F8  +0000015C  SAMPLE        00500100  +00000154          11  CALL
    006A71C0  IBMRPMIA      00692E60  +0000028E  IBMRPMIA      00692E60  +0000028E              CALL
    006A70D8  CEEEV010      00658800  +00000282  CEEEV010      00658800  +00000282              CALL
    006A7018  CEEBBEXT      001FA3E0  +00000132  CEEBBEXT      001FA3E0  +00000132              CALL

  CONDITION INFORMATION FOR ACTIVE ROUTINES
    CONDITION INFORMATION FOR SAMPLE (DSA ADDRESS 006A7390)
      CIB ADDRESS: 006A4528
      CURRENT CONDITION:
        IBM0281I A PRIOR CONDITION WAS PROMOTED TO THE 'ERROR' CONDITION
      ORIGINAL CONDITION:
        CEE3209S THE SYSTEM DETECTED A FIXED-POINT DIVIDE EXCEPTION.
      LOCATION:
        PROGRAM UNIT: SAMPLE ENTRY: SUB1 STATEMENT: 15 OFFSET: +0000039E
      MACHINE STATE:
        ILC..... 0004    INTERRUPTION CODE..... 0009
        PSW..... 071D2E00 8050049A
        GPR0..... 006A7490  GPR1..... 006A7468  GPR2..... 8050042C  GPR3..... 005004F0
        GPR4..... 005005AC  GPR5..... 006A7260  GPR6..... 006A7334  GPR7..... 00000008
        GPR8..... 006A7330  GPR9..... 00000004  GPR10.... 005007A4  GPR11.... 006A7328
        GPR12.... 00657E48  GPR13.... 006A7390  GPR14.... 00000000  GPR15.... 00000003
    STORAGE DUMP NEAR CONDITION, BEGINNING AT LOCATION: 00500486
      +000000 00500486  5860B000 5F604000 58E76000 8EE00020  5DE08000 50F96000 41E0D0D8 50E03124  |.-..^- ..X-.....)...&9-....Q&...|

  CONTROL BLOCKS FOR ACTIVE ROUTINES:
:
    DSA FOR ERR  ON-UNIT: 006A7788
      +000000  FLAGS.... CC25      member... AC8F      BKC...... 006A7578  FWC...... 00657E48  R14...... 8068A45A
      +000010  R15...... 80675258  R0....... 006A78B0  R1....... 005005E8  R2....... 80500318  R3....... 005004F0
      +000024  R4....... 006A7260  R5....... 00000000  R6....... 006A7840  R7....... 006A7660  R8....... 006A7878
      +000038  R9....... 006A78A8  R10...... 00000064  R11...... 0068DF97  R12...... 03A0658C  reserved. 006C78A0
      +00004C  NAB...... 006A78B0  PNAB..... 006A78B0  reserved. 91E091E0  006A7260  00500690  03A065BC
      +000064  reserved. 006A7828  reserved. 006A7828  MODE..... 80500364  reserved. 006A7840  006A02FC
      +000078  reserved. 006A769C  reserved. 006A76A0
    DYNAMIC STORAGE AREA (ERR  ON-UNIT): 006A7788
      +000000 006A7788  CC25AC8F 006A7578 00657E48 8068A45A  80675258 006A78B0 005005E8 80500318  |..........=...u!.........&.Y.&..|
      +000020 006A77A8  005004F0 006A7260 00000000 006A7840  006A7660 006A7878 006A78A8 00000064  |.&.0...-....... ...-.......y....|
      +000040 006A77C8  0068DF97 03A0658C 006C78A0 006A78B0  006A78B0 91E091E0 006A7260 00500690  |...p.....%..........j.j....-.&..|
      +000060 006A77E8  03A065BC 006A7828 006A7828 80500364  006A7840 006A02FC 006A769C 006A76A0  |.............&..... ...........|
      +000080 006A7808  006A76A4 006A7A3C 006A76A8 006A76B0  006A76B4 006A76B8 006A7C30 00000000  |...u..:....y..............@.....|
      +0000A0 006A7828  00000000 00000000 00000000 0E6C9070  40404040 40404040 0C014040 40404040  |.............%..        ..      |
      +0000C0 006A7848  40404040 40404040 40404040 40404040  00500039 C3D6D5C4 C9E3C9D6 D540D7D9  |                .&..CONDITION PR|
      +0000E0 006A7868  D6C3C5E2 E2C9D5C7 40D9C5E2 E4D3E3C5  E3C2C3D5 006A7878 00030000 D7D3C9C4  |OCESSING RESULTETBCN........PLID|
      +000100 006A7888  E4D4D740 83819393 85844086 99969440  85999996 9940D6D5 60A49589 A3404040  |UMP called from error ON-unit   |
      +000120 006A78A8  006A7884 00210000 0000C5D5 006C78A0  006A7DF8 80696EEE 83A01D68 00000001  |...d......EN.%....'8..>.c.......|
    STATIC FOR PROCEDURE SAMPLE    TIMESTAMP: 30 JULY 01 15:21:59
    STARTING FROM: 005004F0
      +000000 005004F0  E00002A4 00500100 005001A8 005001DE  005002A4 00500318 005003C0 0050042C  |...u.&...&.y.&...&.u.&...&...&..|
      +000020 00500510  0050042C 0050042C 0050042C 00501280  00501298 005012B0 00501358 00501460  |.&...&...&...&...&.q.&...&...&.-|
      +000040 00500530  005014A8 005014D8 00501538 00501580  005015E0 005015F8 00501610 00501640  |.&.y.&.Q.&...&...&...&.8.&...&. |
      +000060 00500550  20000002 1F800000 0050061C 000F0000  00000000 000D0000 00000000 005005AC  |.........&...................&..|
      +000080 00500570  00500638 000D0000 00000000 00030000  00000000 00210000 00500669 000D0000  |.&......................&.......|
      +0000A0 00500590  00500676 000B0000 91E091E0 00000001  00000003 00000005 00000000 00000004  |.&......j.j.....................|
      +0000C0 005005B0  00000004 00000003 00000001 00000002  005007B0 005007B0 006A7368 8050059C  |.................&...&.......&..|
```

Figure 90. LE/VSE Dump from Example PL/I Routine (Part 1 of 2)

```
    +0000E0 005005D0  006A7330 006A7320 806A7328 005007B0  00000000 8050059C 006A787C 806A78A8  |.............&.......&.....@...y|
    +000100 005005F0  00501490 005007B0 80000000 00000000  80500698 005007B0 006A7468 8050059C  |.&...&...........&.q.&.......&..|
    +000120 00500610  005007B0 00000000 8050059C E2819497  938540E2 A38199A3 899587C7 4B40E2A4  |.&.......&..Sample StartingG. Su|
    +000140 00500630  A3888599 93819584 E2819497 938540C5  95848995 87E3C2C3 D7D3C9C4 E4D4D740  |therlandSample EndingTBCPLIDUMP |
    +000160 00500650  83819393 85844086 99969440 85999996  9940D6D5 60A49589 A3E2A482 F140E2A3  |called from error ON-unitSub1 St|
    +000180 00500670  8199A389 9587E2A4 82F140C5 95848995  87000000 00000000 0C160000 005002A4  |artingSub1 Ending............&.u|
    +0001A0 00500690  0C960000 00000000 005006AC 005006C8  005006E4 00000000 00000000 85000001  |.o.......&...&.H.&.U........e...|
    +0001C0 005006B0  00500552 000000D0 00000000 0008C16D  D5E4D4C2 C5D90000 81000001 00500550  |.&............A_NUMBER..a....&.&|
    +0001E0 005006D0  000000C0 00000000 0007D4E8 6DD5C1D4  C5000000 81000101 00500552 000000C8  |..........MY_NAME...a....&.....H|
    +000200 005006F0  00000000 0008C1D5 6DC1D9D9 C1E80000  00000001 00500100 000001A4 00500734  |......AN_ARRAY.......&.....u.&..|
    +000220 00500710  00000001 00DE0002 00E20008 01200009  0128000A 012E000B 01560012 01940013  |.........S...................m..|
    +000240 00500730  01A40002 005002A4 00000112 00500758  00000002 00740003 00780004 00C00005  |.u...&.u.....&..................|
    +000260 00500750  01020006 0114000C 005003C0 0000012C  0050077C 0000000C 006C000E 00AA000F  |.........&.......&.@.....%......|
    +000280 00500770  00DE0010 011C0011 011C0011 0E0E0E0E  F3F040D1 E4D3E840 F0F140F1 F57AF2F1  |................30 JULY 01 15:21|
    +0002A0 00500790  7AF5F940 80000011 00500078 00000000  01000001 00500100 00501258 00000000  |:59 .....&...........&...&......|
:
   DSA FOR SUB1: 006A7390
    +000000  FLAGS.... 8025      member... 0000      BKC...... 006A7260  FWC...... 00000000  R14...... 8050049A
    +000010  R15...... 00000003  R0....... 006A7490  R1....... 006A7468  R2....... 8050042C  R3....... 005004F0
    +000024  R4....... 005005AC  R5....... 006A7260  R6....... 006A7334  R7....... 00000008  R8....... 006A7330
    +000038  R9....... 00000004  R10...... 005007A4  R11...... 006A7328  R12...... 00657E48  reserved. 006C73E0
    +00004C  NAB...... 006A7490  PNAB..... 006A7490  reserved. 91E091E0  006A7260  00000000  00000000
    +000064  reserved. 00000000  reserved. 00000000  MODE..... 8050046A  reserved. 00000000  00000200
    +000078  reserved. 00000000  reserved. 00000000
   CIB FOR SUB1: 006A4528
    +000000 006A4528  C3C9C240 00000000 00000000 010C0002  00000005 E2E30000 00030119 59C9C2D4  |CIB ................ST.......IBM|
    +000020 006A4548  00000000 006A4634 00030C89 59C3C5C5  00000001 00000005 006A7260 80658800  |...........i.CEE...........-..h.|
    +000040 006A4568  00000000 006A7390 0050049A 00653928  0000000A 00000000 00000000 00000000  |.........&......................|
    +000060 006A4588  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
    +000080 006A45A8  00000000 00000000 00000000 00000000  00540100 00008000 00000000 00000000  |................................|
    +0000A0 006A45C8  00000000 00000000 00000000 00000000  44234000 00000020 00000009 40404040  |................. .........     |
    +0000C0 006A45E8  40404040 00000000 006A7390 006A7390  00500496 00000000 00000000 00000001  |    .............&.o...........|
    +0000E0 006A4608  006A7260 0000000A 00000064 00000000  FFFFFFFC 00000000 00000000 006C77C0  |...-.........................%..|
    +000100 006A4628  00000000 00653A0C 00000000 E9D4C3C8  00D00001 006A7490 006A7468 8050042C  |............ZMCH.............&..|
CEE5DMP V1 R4.2: PLIDUMP called from error ON-unit                                  07/30/01 3:22:01 PM              PAGE:    4

   DYNAMIC STORAGE AREA (SUB1): 006A7390
    +000000 006A7390  80250000 006A7260 00000000 8050049A  00000003 006A7490 006A7468 8050042C  |.......-.....&...............&..|
    +000020 006A73B0  005004F0 005005AC 006A7260 006A7334  00000008 006A7330 00000004 005007A4  |.&.0.&.....-.................&.u|
    +000040 006A73D0  006A7328 00657E48 006C73E0 006A7490  006A7490 91E091E0 006A7260 00000000  |......=..%..........j.j....-....|
    +000060 006A73F0  00000000 00000000 00000000 8050046A  00000000 00000200 00000000 00000000  |.............&..................|
    +000080 006A7410  00000000 C9C2D4D9 D6D7C1C1 0050061C  000F0000 E6E2C1E3 006A7390 00000000  |....IBMROPAA.&......WSAT........|
    +0000A0 006A7430  88000000 006A41E0 006A7638 80234B8C  80234A48 80640480 006A44E0 00000002  |h...............................|
    +0000C0 006A7450  006A7330 006A7320 006A7328 00000001  006A7468 005007A4 00500588 00500550  |....................&.u.&.h.&.&|
    +0000E0 006A7470  80234A48 8050059C 00407300 006CD010  00000000 802042B4 00010000 006C75C0  |.....&... ...%...............%..|
   DSA FOR SAMPLE: 006A7260
    +000000  FLAGS.... C025      member... 7060      BKC...... 006A71C0  FWC...... 00000000  R14...... 80500256
    +000010  R15...... 005003C0  R0....... 006A7390  R1....... 005005D0  R2....... 805001DE  R3....... 005004F0
    +000024  R4....... 00000005  R5....... 006A7260  R6....... 006A7338  R7....... 006A7328  R8....... 00000001
    +000038  R9....... 006A7098  R10...... 005007A4  R11...... 00000003  R12...... 00657E48  reserved. 006C73E0
    +00004C  NAB...... 006A7390  PNAB..... 006A7390  reserved. 91E091E0  8800000C  00500688  006A74D8
    +000064  reserved. 8069B03E  reserved. 83A58B30  MODE..... 80500220  reserved. 006A7318  00690238
    +000078  reserved. 0065F868  reserved. 006579B8
   DYNAMIC STORAGE AREA (SAMPLE): 006A7260
    +000000 006A7260  C0257060 006A71C0 00000000 80500256  005003C0 006A7390 005005D0 805001DE  |...-.........&...&.......&...&..|
    +000020 006A7280  005004F0 00000005 006A7260 006A7338  006A7328 00000001 006A7098 005007A4  |.&.0.......-...............q.&.u|
    +000040 006A72A0  00000003 00657E48 006C73E0 006A7390  006A7390 91E091E0 8800000C 00500688  |......=..%..........j.j.h....&.h|
    +000060 006A72C0  006A74D8 8069B03E 83A58B30 80500220  006A7318 00690238 0065F868 006579B8  |...Q....cv...&............8.....|
    +000080 006A72E0  00657060 00640838 00657E58 006C72ED  006C72A0 00660867 00689038 00657E48  |...-......=..%...%............=.|
    +0000A0 006A7300  00000000 006A74D8 00000000 00000080  00000000 00000000 0C010000 00000000  |.......Q........................|
    +0000C0 006A7320  006A7344 000D0000 006A7338 005005AC  00000000 00000000 00000001 00000003  |.............&..................|
    +0000E0 006A7340  00000005 C74B40E2 A4A38885 99938195  84000000 00000000 00000000 00000000  |....G. Sutherland...............|
    +000100 006A7360  006A7368 00000000 00500558 00500550  00000000 8050059C 00400000 006CD010  |.........&...&.&.....&... ...%..|
    +000120 006A7380  10000000 00000000 00010000 006C75C0  80250000 006A7260 00000000 8050049A  |.............%.........-.....&..|
```

Figure 90. LE/VSE Dump from Example PL/I Routine (Part 2 of 2)

To understand the dump information and debug this routine, use the following steps:

1. Notice the title of the dump: `PLIDUMP called from error ON-unit`. This was the title specified when PLIDUMP was invoked, and it indicates that the ERROR condition was raised and PLIDUMP was called from within the ERROR ON-unit.
2. Locate the messages in the Condition Information section of the dump.

There are two messages. The current condition message indicates that a prior condition was promoted to the ERROR condition. The promotion of a condition occurs when the original condition is left unhandled (no PL/I ON-units are assigned to gain control). The original condition message is `CEE3209S The system detected a fixed-point divide exception`. The original condition usually indicates the actual problem. For more information about this message, see Chapter 8, "LE/VSE Run-Time Messages," on page 163.

3. In the traceback section, note the sequence of calls in the call chain. SAMPLE called SUB1 at statement 11, and SUB1 raised an exception at statement 15, PU offset X'0000039E'.
4. Find the statement in the listing for SUB1 that raised the ZERODIVIDE condition. If SUB1 was compiled with GOSTMT and SOURCE, find statement 15 in the source listing.

   Since the object listing was generated in this example, you can also locate the actual assembler instruction causing the exception at offset X'0000039E' in the object listing for this routine, shown in Figure 89 on page 151. Either method shows that *divisor* was used as the divisor in a divide operation.
5. You can see from the declaration of SUB1 that *divisor* is a parameter passed from SAMPLE. Because of linkage conventions, you can infer that register 1 in the SAMPLE save area points to a parameter list that was passed to SUB1. *divisor* is the first parameter in the list.
6. In the SAMPLE DSA, the R1 value is X'005005D0'. This is the address of the parameter list, which is located in static storage.
7. Find the parameter list in the static storage section of the dump; the first parameter address points to location X'006A7330', which is in the dynamic storage area for SAMPLE.
8. The value of the parameter at location X'006A7330' is X'00000000'. Thus, the exception occurred when SAMPLE passed a 0 value used as a divisor in subroutine SUB1.

Chapter 7. Debugging under CICS

This chapter provides information for debugging under the Customer Information Control System (CICS). The following sections explain how to access debugging information under CICS, and describe features unique to debugging under CICS.

For quick reference, Table 36 shows debugging information and where the debugging information appears.

Table 36. Location of Debugging Information under CICS

Debugging Information	Location
LE/VSE Run-Time Messages	CESE *Transient Data Queue*
LE/VSE Traceback	CESE Transient Data Queue
LE/VSE Dump Output	CESE Transient Data Queue
CICS Transaction Dump	CICS DFHDMPA or DFHDMPB data set
LE/VSE Abend and Reason Codes	System Console
LE/VSE Return Codes to CICS	System Console
CICS Shutdown Statistics	CSSL Transient Data Queue
Interactive User Interface (IUI) OLPD records, including abend information from LE/VSE	The Interactive Interface "ONLINE PROBLEM DETERMINATION" dialog. **Notes:** 1. The IUI-supplied exit IESPDATX is predefined. It is auto-enabled using the LE/VSE–CICS abnormal termination exit. Standard OLPD records that fail and that are related to LE/VSE-conforming applications, will include LE/VSE CICS 40xx-type abends and LE/VSE traceback information. 2. For detailed information about how OLPD is used, refer to the section "Online Analysis of CICS TS Transaction Abends" in the manual *z/VSE Guide for Solving Problems*, SC33-8232.

If the EXEC CICS HANDLE ABEND command is active and the application or CICS initiates an abend or application interrupt, LE/VSE does not produce any run-time messages, tracebacks, or dumps.

Accessing Debugging Information

The following sections list the debugging information available to CICS users, and describe where you can find this information.

Under CICS the LE/VSE run-time messages, LE/VSE traceback, and LE/VSE dump output are written to the CESE transient data queue. The transaction identifier, terminal identifier, date, and time precede the data in the queue. For detailed information about the format of records written to the transient data queue, see *LE/VSE Programming Guide*.

The CESE transient data queue is defined in the CICS *destination control table (DCT)*. The CICS macro DFHDCT is used to define entries in the DCT. For a detailed explanation of how to define a transient data queue in the DCT, see *CICS Transaction Server for VSE/ESA Resource Definition (Macro)* For details about specific LE/VSE definition requirements in the DCT, see *LE/VSE Customization Guide*. If you are not sure how to define the CESE transient data queue, see your system programmer.

For information about using the Interactive Interface dialog "Inspect Message Log" to access the CESE transient data queue, see *LE/VSE Customization Guide*.

Locating LE/VSE Run-Time Messages

Under CICS, LE/VSE run-time messages are written to the CESE transient data queue. Figure 91 shows an example of the LE/VSE message that appears when an application abends due to an unhandled condition from an EXEC CICS command.

```
 P039UTV9 19960916145313 CEE3250C The system or user abend AEI0 was issued.
 P039UTV9 19960916145313          From program unit UT9CVERI at entry point UT9CVERI at offset +0000011E at
 P039UTV9 19960916145313          address 0006051E.
```

Figure 91. LE/VSE Message in CESE Transient Data Queue

Locating the LE/VSE Traceback

Under CICS, the LE/VSE traceback is written to the CESE transient data queue. Because LE/VSE invokes your application routine, the LE/VSE routines that invoked your routine appear in the traceback. Figure 92 shows an example LE/VSE traceback written to the CESE transient data queue. Data unnecessary for this example has been replaced by ellipses.

```
 A0019132 19960418105907 CEE3209S THE SYSTEM DETECTED A FIXED-POINT DIVIDE EXCEPTION.
 A0019132 19960418105907          FROM COMPILE UNIT CIC9132 AT ENTRY POINT CIC9132 AT COMPILE UNIT OFFSET +00000674 AT ADDRESS 014D7
1A0019132 19960418105907 CEE5DMP V1 R4.0: CONDITION PROCESSING RESULTED IN THE UNHANDLED CONDITION.        05/14/96 10:59:07 AM
 A0019132 19960418105907
 A0019132 19960418105907 INFORMATION FOR ENCLAVE CIC9132
 A0019132 19960418105907
 A0019132 19960418105907   INFORMATION FOR THREAD 8000000000000000
 A0019132 19960418105907
 A0019132 19960418105907   TRACEBACK:
 A0019132 19960418105907     DSA ADDR  PROGRAM UNIT  PU ADDR   PU OFFSET  ENTRY         E ADDR    E  OFFSET   STATEMENT  STATUS
 A0019132 19960418105907     014C8798  CEEHDSP       005C96E8  +00001D4C  CEEHDSP       005C96E8  +00001D4C              CALL
 A0019132 19960418105907     014D08D0  CEECGEX       005C4688  +000001A6  CEECGEX       005C4688  +000001A6              CALL
 A0019132 19960418105907     014D47B0                00000000  +00000000                00000000  +00000000              CALL
 A0019132 19960418105907     014D15E0  CIC9132       014D75A0  +00000674  CIC9132       014D75A0  +00000674              EXCEPTION
 A0019132 19960418105907     014D04C0  IGZCEV5       016D5600  +00000814  IGZCEV5       016D5600  +00000814              CALL
 A0019132 19960418105907     014D0398  CEECRINV      005C7A28  +000003D0  CEECRINV      005C7A28  +000003D0              CALL
 A0019132 19960418105907     014D02C0  CEECCICS      005B1480  +0000039C  CEECCICS      005B1480  +0000039C              CALL
   ⋮
```

Figure 92. LE/VSE Traceback Written to the Transient Data Queue

Locating the LE/VSE Dump

Under CICS, the LE/VSE dump output is normally written to the transient data queue specified in the MSGFILE run-time option (which defaults to CESE). Alternatively, the run-time option TERMTHDACT allows you to specify the LSTQ sub-option for writing dump output to `VSE POWER LST QUEUE`. For active routines, the LE/VSE dump contains the traceback, condition information, variables, storage, and control block information for the thread, enclave, and process levels. Use the LE/VSE dump with the CICS transaction dump to locate problems when operating under CICS.

For an example LE/VSE dump, see “Understanding an LE/VSE Dump” on page 36.

Using CICS Transaction Dump

The CICS transaction dump is generated to the DFHDMPA or DFHDMPB data set. The offline CICS dump utility routine converts the transaction dump into formatted, understandable output.

The CICS transaction dump contains information for the storage areas and resources associated with the current transaction. This information includes the Communication Area (COMMAREA), Transaction Work Area (TWA), Exec Interface Block (EIB), and any storage obtained by the EXEC CICS commands. This information does not appear in the LE/VSE dump. It can be helpful to use the CICS transaction dump with the LE/VSE dump to locate problems when operating under CICS.

When the location of an error is uncertain, it can be helpful to insert EXEC CICS DUMP statements in and around the code suspected of causing the problem. This generates CICS transaction dumps close to the error for debugging reference.

For information about interpreting CICS dumps, see *CICS Transaction Server for VSE/ESA Problem Determination Guide*

Using CICS Register and Program Status Word Contents

When a routine interrupt occurs (code = ASRA or ASRC) and a CICS dump is generated, CICS formats the contents of the program status word (PSW) and the registers at the time of the interrupt.

The address of the interrupt can be found from the second word of the PSW, giving the address of the instruction following the point of interrupt. The address of the entry point of the function can be subtracted from this address, giving the offset of the statement that caused the interrupt.

For C routines, you can find the address of the entry point in register 3.

If register 15 is corrupted, the contents of the first phase of the active enclave appear in the program storage section of the CICS transaction dump.

Using LE/VSE Abend and Reason Codes

An application can end with an abend in two ways:

- User-specified abend (that is, an abend requested by the assembler user exit or the ABTERMENC run-time option).
- LE/VSE-detected unrecoverable error (in which case there is no LE/VSE condition handling).

When LE/VSE detects an unrecoverable error under CICS, LE/VSE terminates the transaction with an EXEC CICS ABEND command. The abend code has a number between 4000 and 4095. A write-to-operator (WTO) is performed to write a CEE1000S message to the system console. This message contains the abend code and its associated reason code.

Note: The WTO is performed only for unrecoverable errors detected by LE/VSE. No WTO occurs for user-requested abends.

Although this type of abend is performed only for unrecoverable error conditions, an abend code of 4000–4095 does not necessarily indicate an internal error within LE/VSE. For example, an application routine can write a variable outside its storage and corrupt the LE/VSE control blocks.

Possible causes of a 4000–4095 abend are corrupted LE/VSE control blocks and internal LE/VSE errors. See Chapter 13, "LE/VSE Abend Codes," on page 329 for more information about abend codes 4000–4095. Figure 93 shows an example LE/VSE abend and reason code. Abend codes and reason codes appear in hexadecimal.

```
F2 0002 CEE1000S LE/VSE internal abend. ABCODE = 00000FFD
        REASON = 00000028
```

Figure 93. LE/VSE Abend and Reason Code

Using LE/VSE Return Codes to CICS

When the LE/VSE condition handler encounters a severe condition that i s specific to CICS, the condition handler generates a CICS-specific return code. This return code is written to the system console.

Possible causes of an LE/VSE return code to CICS are:
- Incorrect partition size
- Incorrect DCT
- Incorrect CSD definitions

See Chapter 14, "Return Codes to CICS," on page 337 for a list of the reason codes written only to CICS. Figure 94 shows an example return code that was returned to CICS.

```
 +DFH1568 A CICS request to LE for VSE/ESA has failed. Reason code 0012030.
```

Figure 94. Return Code Written Only to CICS

Ensuring Transaction Rollback

If your application does not run to normal completion and there is no CICS transaction abend, take steps to ensure that transaction rollback (the backing out of any updates to recoverable resources made by the malfunctioning application) takes place.

There are two ways to ensure that a transaction rollback occurs when an unhandled condition of severity 2 or greater is detected:
- Use the ABTERMENC run-time option with the ABEND suboption (ABTERMENC(ABEND)). This is the IBM-supplied default under CICS.
- Use an assembler user exit that requests an abend for unhandled conditions of severity 2 or greater.

See "Using the Assembler User Exit" on page 16 and *LE/VSE Programming Guide* for more information about the assembler user exit.

Finding Data When LE/VSE Returns a Nonzero Reason Code

This example shows the output generated when LE/VSE returns a nonzero reason code to CICS. LE/VSE does not write any messages to the CESE transient data queue. Table 37 shows example output and the location where the output appears.

Table 37. Output When a Nonzero Reason Code is Returned to CICS

Output Message	Location	Issued By
DFH2206 14:43:54 DBDCCICS Transaction UTV2 has failed with abend AEC7. Resource backout was successful.	User's terminal	CICS
DFH1568 A CICS request to LE VSE/ESA has failed. Reason code 0012030.	System console	CICS
DFH2236 14:43:48 DBDCCICS Transaction UTV2 abend AEC7 in routine UT2CVERI term P021 backout successful.	Transient data queue CSMT	CICS

Finding Data When LE/VSE Abends Internally

This example shows the output generated when LE/VSE abends internally. LE/VSE does not write any messages to the CESE transient data queue. Table 38 shows example output and the location where the output appears.

Table 38. Output When LE/VSE Abends Internally

Output Message	Location	Issued By
DFH2206 14:35:24 LE03CC01 Transaction UTV8 has failed with abend 4095. Resource backout was successful.	User's terminal	CICS
CEE1000S LE/VSE internal abend. ABCODE = 00000FFF REASON = 00001234	System console	LE/VSE
DFH2236 14:35:24 DBDCCICS Transaction UTV8 abend 4095 in routine UT8CVERI term P021 backout successful.	Transient data queue CSMT	CICS

Finding Data When LE/VSE Abends from an EXEC CICS Command

This example shows the output generated when an application abends from an EXEC CICS command. Table 39 shows example output and the location where the output appears.

This error assumes the use of LE/VSE run-time option TERMTHDACT(MSG).

Table 39. Output When LE/VSE Abends from an EXEC CICS Command

Output Message	Location	Issued By
DFH2206 14:35:34 DBDCCICS Transaction UTV8 has failed with abend AEI0. Resource backout was successful.	User's terminal	CICS

Table 39. Output When LE/VSE Abends from an EXEC CICS Command (continued)

Output Message	Location	Issued By
DFH2236 14:35:17 DBDCCICS Transaction UTV9 abend AEI0 in routine UT9CVERI term P021 backout successful.	Transient data queue CSMT	CICS
P021UTV9 19960421143516 CEE3250C The system or user abend AEI0 was issued.	Transient data queue CESE	LE/VSE

Part 3. Run-Time Messages and Codes

This part of the book provides lists of LE/VSE and LE/VSE-component run-time messages and abend and reason codes that can appear as a result of errors in your routine.

Chapter 8. LE/VSE Run-Time Messages

The following messages relate to LE/VSE. 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.

For a description of the format of LE/VSE run-time messages, see "Interpreting Run-Time Messages" on page 25.

Within this chapter, these LE/VSE run-time messages are contained in their own sections:
- "VSE/POWER LSTQ Support Run-Time Messages" on page 210
- "Attention Routine Support Run-Time Messages" on page 212
- "CLER CICS Transaction Support Message" on page 216

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

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 raised 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: LE/VSE 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: CEEnsure 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.

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

Note: CEE088 may 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.

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 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: CEE0B4

CEE0374C CONDITION = *condition-id* **TOKEN =** *condition-token* **WHILE PROCESSING PROGRAM** *program-unit* **WHICH STARTS AT** *address* **AT THE TIME OF INTERRUPT**

Explanation: An unrecoverable condition has occurred while processing a previously-unhandled condition. The `condition-id` corresponds to the LE/VSE message number associated with the failure, and `condition-token` is the internal representation of the event (where the `condition-id` and `condition-token` translate to each other). The message is issued to the operator console when nested conditions occur. The message display includes the registers and PSW information relating to the displayed condition token. This is the condition that was being processed when the unrecoverable error occurred. If this message appears more than once, the conditions appear in order of occurrence.

PSW	*psw*			
GPR 0-3	*gpr 0*	*grp 1*	*gpr 2*	*gpr 3*
GPR 4-7	*gpr 4*	*gpr 5*	*gpr 6*	*gpr 7*
GPR 8-B	*gpr 8*	*gpr 9*	*gpr A*	*gpr B*
GPR C-F	*gpr C*	*gpr D*	*gpr E*	*gpr F*

Programmer Response: The earliest condition is the one that should be investigated and corrected. For information on condition tokens and how the CEE0374C message is used to debug errors, refer to the section "Using Condition Information" of "Chapter 1. Preparing Your Routine for Debugging" in the *LE/VSE Debugging Guide and Run-Time Messages*.

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 LE/VSE to retry the operation with new input.

Programmer Response: No programmer response is required.

System Action: LE/VSE 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 LE/VSE to retry the operation with new output.

Programmer Response: No programmer response is required.

System Action: LE/VSE 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: 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 A–Z, a–z, or 0–9. Processing will continue.

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: Facility ID parameter must be a 3-alphanumeric character field.

Programmer Response: Provide a facility ID made up of three alphanumeric characters that corresponds to a product recognized by LE/VSE. The IBM-supplied facility IDs are IBM, 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_index.

Symbolic Feedback Code: CEE0E7

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

Explanation: An error was detected trying to access the given message file *filename*. If *filename* is SYSLST, this message indicates that SYSLST is unassigned.

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 phase containing the table of message file names cannot be located. The name of each IBM-supplied phase is *xxx*MSGT, where *xxx* is the facility ID. The name of each user-defined phase is *txxx*MSGT, where *t* is 'U' for a user-assigned facility ID, and *xxx* is the facility ID. MSGT is 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 partition GETVIS size is sufficient to run the application. Verify that the storage size specified in the HEAP run-time option is reasonable, given the partition GETVIS 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 exists. 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 will have its i_s_info field wrap around 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 instance specific information 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 has been 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 an LE/VSE 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

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: LE/VSE 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 tha the original allocation. For example, allocating a 100-byte area and then writing 120 bytes to this area could cause damage to storage headers.

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) prior to 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 prior to 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 *LE/VSE Programming Reference*.

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

Symbolic Feedback Code: CEE0P6

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 cannot 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. This message indicates that storage management could not obtain sufficient storage from the operating system to create an additional heap segment or stack segment.

Programmer Response: Ensure that the partition GETVIS 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 partition GETVIS size allocated to the application. Verify that you are using storage options that get your storage from above the 16MB 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 satisfy a get storage (CEECZST) request. Request was for *nbytes* bytes from *location* storage.**

Explanation: There was not enough free storage available to satisfy a get storage call to CEEGTST, or to reallocate request call to CEECZST for *nbytes* from *location*.

Programmer Response: Ensure that the partition GETVIS size is sufficient to run the application. Ensure that the “size” parameter in the get storage request is not too large. Verify that the storage sizes specified in the HEAP run-time option are correct in relation to the partition size allocated to the application. Verify that the storage options being used obtain storage from above the 16MB line (if possible), since running out of storage below the line can happen more easily.

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

Symbolic Feedback Code: CEE0PE

CEE1000S **Language Environment internal abend. ABCODE =** ***abend-code*** **REASON =** ***reason-code***

Explanation: This message was issued to the operator’s console in CICS to indicate that LE/VSE has abended, with the abend code and reason code as specified in the message, in hexadecimal.

Programmer Response: See Chapter 13, “LE/VSE Abend Codes,” on page 329 for information about 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 identified as *identifier*.**

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

identifier	**HLL**
03	C
05	COBOL
10	PL/I

15 Assembler

Programmer Response: This is a language-specific restriction. For more information about running LE/VSE-conforming applications under CICS, see *LE/VSE Programming Guide*.

System Action: No system action is performed.

Symbolic Feedback Code: CEE0V9

CEE2001E **For an exponentiation operation (R**S) 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 (I**J) 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 (R**I) 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 input parameters 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.

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: 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

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

Explanation: An underflow has 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

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, or an invalid date was specified in the DATE job control statement. The current time cannot be determined.

Programmer Response: Notify systems support personnel that the system clock is in an invalid state, or correct the date specified in the DATE job control statement.

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 date/time, and 23:59:59.999 31 December 9999 being the last supported date/time.

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 Japanese (<JJJJ>) or Chinese (<CCCC> or <CCCCCCCC>) Era 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 a Japanese (<JJJJ>) or Chinese (<CCCC> or <CCCCCCCC>) 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 is 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 may also be returned for any non-existent date value such as June 31st, 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 Japanese or Chinese Era passed to CEEDAYS or CEESECS was not recognized.

Explanation: The value in the Japanese (<JJJJ>) or Chinese (<CCCC> or <CCCCCCCC>) Era field passed in a call to CEEDAYS or CEESECS did not contain a supported Japanese or Chinese Era name.

Programmer Response: Verify that the format of the input data matches the picture string specification and that the spelling of the Japanese or Chinese 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 does not contain a number between 0 and 23, or the “AP” (a.m./p.m.) field is present and the HH field does 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, specify year 1990, not year 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.

System Action: The output value is set to 0.

Symbolic Feedback Code: CEE2EJ

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 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 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 contained more than one era descriptor, such as both Japanese (<JJJJ>) and Chinese (<CCCC>) being specified in the same picture string.

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 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 The Japanese (<JJJJ>) or Chinese (<CCCC> or <CCCCCCCC>) Era 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 a Japanese or Chinese 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 nonzero 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 (AP, A.P., etc.) 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: A debugging tool has terminated the enclave at the user’s request. Abend code 4094, reason code X'28' is issued.

Programmer Response: No programmer response is necessary.

System Action: The enclave is terminated with abend code 4094, reason code X'28'.

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: No system action is taken.

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, or an invalid date was specified in the DATE job control statement. The current time cannot be determined.

Programmer Response: Notify systems support personnel that the system clock is in an invalid state, or correct the date specified in the DATE job control statement.

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: No system action is taken; 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 double-byte character set (DBCS) 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 by specifying enough Ms or Ws 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 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

CEE2537W **An input date outside the specified Japanese (<JJJJ>) or Chinese (<CCCC>) Era was used in a call to CEECBLDY or CEEDAYS.**

Explanation: The name of a Japanese or Chinese Era was specified in a call to CEECBLDY or CEEDAYS, but the input date does not fall in the specified Era.

Programmer Response: Modify input date or corresponding Era as required.

System Action: The output date is calculated as if the input date falls in the specified Era.

Symbolic Feedback Code: CEE2F9

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

CEE3102E Invalid CEE5DMP options or suboptions were found and ignored.

Explanation: Invalid options or suboptions were found in the options parameter to CEE5DMP.

Programmer Response: Check the options string passed to CEE5DMP. Make sure it has correct syntax and values for options and suboptions as specified in *LE/VSE 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 filename was specified with the FNAME option to CEE5DMP. The default file name is CEEDUMP, or CESE transient data queue under CICS.

Programmer Response: Make sure the file name is correct as specified in the options to CEE5DMP. 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, then 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 LE/VSE returned this condition to the common component of LE/VSE when an error had occurred in the language component's dump event handler that was not covered in the conditions returned by the low level dump CWI services.

Programmer Response: Not applicable. This is an internal condition within LE/VSE, and is never seen by the application programmer.

System Action: The common component of LE/VSE 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: A parameter to the CEEVDMP low level CWI service to format and write a variable value to the dump 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, the lengths of the strings must be greater than zero and less than the size specified in the LE/VSE CWI documentation.

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

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 if necessary.

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 LE/VSE anchor was the address of the LE/VSE main control block, the CAA. The underlying operating system must provide the celv. anchor support for LE/VSE to obtain 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 LE/VSE 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

CEE3194W An error occurred writing to the dump sublibrary. SDUMP output is on SYSLST.

Explanation: The dump sublibrary was either full, not defined, or in error.

Programmer Response: Determine the cause of the problem with the dump sublibrary.

System Action: The SDUMP output is redirected to SYSLST.

Symbolic Feedback Code: CEE33Q

CEE3195W SDUMP output could not be written to SYSLST.

Explanation: SYSLST is unassigned, assigned ignore, or assigned to an unsupported device type. The SDUMP output could not be written to SYSLST.

Programmer Response: If an SDUMP dump is desired, determine the reason the file could not be opened and correct the problem.

System Action: No SDUMP 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 SDUMP 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 An SDUMP dump was requested on an unsupported system.

Explanation: The SDUMP dump service was called to produce an SDUMP dump on a system not running an ESA-mode operating system, or was called when running under CICS.

Programmer Response: Report the error to your systems support personnel.

System Action: The SDUMP dump was not produced.

Symbolic Feedback Code: CEE33U

CEE3199S An error was returned from the SDUMP system function.

Explanation: The SDUMP dump service was called. It invoked the SDUMP system service which failed.

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

System Action: The SDUMP dump was not produced.

Symbolic Feedback Code: CEE33V

CEE3200S The System detected a Segment Translation Exception

Explanation: The segment-table entry indicated by the segment-index portion of a virtual address is outside of the allocated segment table. The Interrupt Code for this error is X'10'. See the *S/390 Principles of Operations* manual for more details.

Programmer Response: Examine the base registers of the operands of the failing instruction to determine the invalid address being used. Some possible causes are storage overlays, incorrect register save area conventions or incorrect address calculation results.

System Action: The thread is terminated.

Symbolic Feedback Code: CEE340

CEE3201S The system detected an operation exception.

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 Operations* 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.

Explanation: Attempted to execute a privileged operation code while the machine was in a problem state. See a *Principles of Operations* 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.

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 Operations* manual for a full list of execute exceptions.

Programmer Response: Check your application for errors in the EXECUTE instructions. See a *Principles of Operations* 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.

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 Operations* 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.

Explanation: Your program attempted to reference a main-storage location that was not available in the configuration. See a *Principles of Operations* 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 Operations* 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.

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 Operations* 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.

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

Programmer Response: Check the 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.

Explanation: Your program attempted to use signed binary arithmetic or signed left-shift operations and an overflow occurred. See a *Principles of Operations* 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.

Explanation: Your program attempted to perform a signed binary division and the divisor is zero. See a *Principles of Operations* 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.

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 Operations* 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.

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 Operations* 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.

Explanation: Your program attempted a floating-point operation and the result characteristics exceeded 127 and the result fraction was not zero. See a *Principles of Operations* 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.

Explanation: Your program attempted a floating-point operation and the result characteristics is less than zero and the result fraction was not zero. See a *Principles of Operations* 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.

Explanation: Your program attempted a floating-point addition or subtraction and the resulting fraction was zero. See a *Principles of Operations* 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.

Explanation: Your program attempted a do a floating-point divide and the divisor had a zero fraction. See a *Principles of Operations* 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

CEE3220S A math service attempted to execute an extended-precision floating-point-divide machine instruction while running in 370 mode.

Explanation: The math routine attempted to execute the DXR (extended-precision floating-point divide) machine instruction on a system running that does not support the DXR instruction. This has caused an operation exception to occur.

Programmer Response: Contact your systems programmer to ensure your hardware is capable of supporting the DXR instruction. If running under a software emulated S/390 environment, contact the vendor that supplied the emulation software to ensure their software can support the DXR and other 370/XA or later instructions.

System Action: The thread is terminated.

Symbolic Feedback Code: CEE34K

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

Explanation: A CICS or user abend has occurred in the application running under CICS.

Programmer Response: For CICS abend codes, see the transaction abend codes in *VSE/ESA Messages and Codes* . For user abend codes, see your application's user documentation.

System Action: The program is terminated abnormally.

Symbolic Feedback Code: CEE35I

CEE3252C The SORT service *service-name* is not supported in this environment.

Explanation: The SORT CWI service *service-name* was called in the wrong execution environment. Either CEE5SMG was called in CICS or CEE5SRT was called in batch.

Programmer Response: Ensure the appropriate SORT CWI service is used for the execution environment.

System Action: The thread is terminated.

Symbolic Feedback Code: CEE35K

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 parameter list was passed to CEE5SRT.

Explanation: Under CICS, the parameter list for the CWI CEE5SRT must be the extended-format parameter list specified by products such as DFSORT/MVS.

Programmer Response: Correct the parameter list for CEE5SRT.

System Action: The thread is terminated.

Symbolic Feedback Code: CEE35M

CEE3255C An attempt to call the SORT service *service-name* **was made from within a SORT exit routine.**

Explanation: Only one sort can be active at a time. A program called during the execution of SORT has attempted to invoke sort again. The value of *service-name* is either CEE5SMG in the batch environment or CEE5SRT in the CICS environment.

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 requested function was not performed. The condition manager has 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 LE/VSE. This service is intended to be removed in the future.

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 CEE5CIB 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 your service representative.

System Action: The requested function is not performed.

Symbolic Feedback Code: CEE35V

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

Explanation: The CEEHDHDL 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 CEEHDHDL 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. If the condition is not corrected the application may be abended.

Programmer Response: LE/VSE 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 CEE5SPM did not contain all of the settings, because the returned string was truncated.

Explanation: The QUERY option of the CEE5SPM service needed a larger character string to represent the conditions.

Programmer Response: Try a character string larger than 80.

System Action: Some items might have been filled in.

Symbolic Feedback Code: CEE36V

CEE3296E Some of the data in the condition string from CEE5SPM 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 CEE5SPM attempted to PUSH settings onto a full stack.

Explanation: There was not enough storage for the CEE5SPM PUSH service to save all of the conditions.

Programmer Response: Obtain more storage.

System Action: No settings were changed.

Symbolic Feedback Code: CEE372

CEE3299E CEE5SPM attempted to POP settings off an empty stack.

Explanation: A call to CEE5SPM 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

CEE3300C The action parameter in CEE5SPM was not one of the digits 1 to 5.

Explanation: A call to CEE5SPM was made requesting an invalid action.

Programmer Response: Use an action parameter between 1 and 5 when invoking CEE5SPM.

System Action: No settings were changed.

Symbolic Feedback Code: CEE374

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. This is an internal error. The internal routine was called with an improper argument.

Programmer Response: Contact your service representative.

System Action: No system action is performed.

Symbolic Feedback Code: CEE375

CEE3320C PSW at ABEND *psw*.

Explanation: This message is issued to the operator's console to indicate that an abend has occurred and shows the value of the PSW at the time of the abend.

Programmer Response: Use the value of the PSW to determine the cause of the abend.

System Action: The thread is terminated.

Symbolic Feedback Code: CEE37O

CEE3321C Execution abnormally terminated with VSE cancel code *cancel-code* and interruption code *interruption-code*. *cancel-code-information*

Explanation: This message is issued to the operator's console to indicate that an abend has occurred and shows the VSE cancel code and, if applicable, the interruption code, both in hexadecimal. A VSE cancel code of 20 indicates a program check has occurred. The following is a description of the interruption codes that can accompany a cancel code of 20:

01	Operation exception
02	Privileged-operation exception
03	Execute exception
04	Protection exception
05	Addressing exception
06	Specification exception
07	Data exception
08	Fixed-point overflow exception
09	Fixed-point divide exception
0A	Decimal-overflow exception
0B	Decimal-divide exception
0C	Exponent-overflow exception
0D	Exponent-underflow exception
0E	Significance exception
0F	Floating-point divide exception

For some cancel codes, VSE provides additional information in the SVUABINF field of the STXIT AB exit save area. When provided, the message displays this *cancel-code-information* in hexadecimal as follows:

```
SVUABINF: hex-digits
```

Programmer Response: See *z/VSE Messages and Codes* for a list of VSE cancel codes.

System Action: The thread is terminated.

Symbolic Feedback Code: CEE37P

CEE3322C Execution abnormally terminated with user-abend code *abend-code* and reason code *reason-code*.

Explanation: This message was issued to the operator's console to indicate that an abend has occurred and shows the user abend code and reason code, both in decimal.

Programmer Response: See Chapter 13, "LE/VSE Abend Codes," on page 329 for a list of the abend codes and reason codes issued by LE/VSE.

System Action: The thread is terminated.

Symbolic Feedback Code: CEE37Q

CEE3323S An internal error was detected by condition-management services while trying to establish a STXIT exit.

Explanation: The STXIT exit address was not provided.

Programmer Response: Contact your service representative.

System Action: No system action is performed.

Symbolic Feedback Code: CEE37R

CEE3324W The STXIT exit is currently established.

Explanation: Condition-management services determined that the STXIT exit being established is already established.

Programmer Response: No programmer response required.

System Action: No system action is taken.

Symbolic Feedback Code: CEE37S

CEE3325W A STXIT exit could not be canceled by condition-management services.

Explanation: Condition-management services was called to cancel a STXIT exit, but the STXIT exit could not be found.

Programmer Response: No programmer response required.

System Action: No system action is taken.

Symbolic Feedback Code: CEE37T

CEE3326W No STXIT exit was found to reestablish.

Explanation: Condition-management services was called to reestablish a previously established STXIT exit, but no previously established STXIT exit could be found.

Programmer Response: No programmer response is required.

System Action: No system action is taken.

Symbolic Feedback Code: CEE37U

CEE3329S Shunt already established.

Explanation: A call was made to the Establish Shunt CWI to establish a shunt, but a previously established shunt has not been reset.

Programmer Response: Ensure that the previously established shunt is reset before establishing a new shunt.

System Action: The shunt is not established.

Symbolic Feedback Code: CEE381

CEE3330S A shunt routine address was not supplied to CEEHSXT.

Explanation: A call to the CEEHSXT Establish SHUNT CWI did not pass an address for the shunt routine.

Programmer Response: Ensure an address for a shunt routine is passed when calling the CEEHSXT Establish SHUNT CWI.

System Action: A shunt routine is not established.

Symbolic Feedback Code: CEE382

CEE3331S The shunt routine type parameter was not AB or PC.

Explanation: A call to the CEEHSXT Establish SHUNT CWI was made specifying an invalid shunt routine type.

Programmer Response: Ensure the shunt routine type passed to the CEEHSXT establish SHUNT CWI is either AB or PC.

System Action: The shunt routine service was not performed.

Symbolic Feedback Code: CEE383

CEE3332S An abnormal termination exit with OPTION=EARLY is currently established.

Explanation: A call to the CEEHSXT Establish SHUNT CWI or condition-management services was made to establish an abnormal termination exit. The exit could not be established because an abnormal termination exit with OPTION=EARLY was found to be established.

Programmer Response: Contact your systems programmer to determine the subsystem that has established the abnormal termination exit with OPTION=EARLY.

System Action: No system action is performed.

Symbolic Feedback Code: CEE384

CEE3333W A shunt routine was not currently established.

Explanation: A call to the CEEHRXT Reset SHUNT CWI was made to disable a shunt routine, but no shunt routine was established.

Programmer Response: Ensure there is a shunt routine established when calling the CEEHRXT Reset SHUNT CWI.

System Action: No system action is performed.

Symbolic Feedback Code: CEE385

CEE3350S Unable to find the event handler.

Explanation: LE/VSE could not load the event handler for one of the languages in your application. The event handlers required by each language are:

C	CEEEV003
COBOL	CEEEV005
PL/I	CEEEV010

One possible cause for this message is that your application contains a call to an LE/VSE locale callable service, but the C run-time component of LE/VSE has not been installed. LE/VSE locale callable services require the C run-time component.

Programmer Response: If you are using LE/VSE locale callable services, check with your systems programmer to ensure that the C run-time component of LE/VSE has been installed. Ensure that the partition GETVIS size is sufficient to run the application. Ensure that the necessary event handlers are available in one of the sublibraries in your LIBDEF PHASE search chain, and run the application again. If this problem persists, there may be an internal error; 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 LE/VSE attempted to initialize a required member support phase.

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

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

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

Explanation: The stack frame parameter passed to the CEEHSMS 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 CEEHSMS 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 explicitly via a call to the CEEBCRE CWI, 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 call to the CEEBCRI CWI.

Symbolic Feedback Code: CEE391

CEE3362S **No main or fetchable procedure or function was present within the phase.**

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

Programmer Response: Correct the phase.

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 `system()`.

Programmer Response: Correct the phase.

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 is 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 enclave name is not set.

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 phase was not recognized by Language Environment.**

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

Programmer Response: No response is required.

System Action: 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 in a condition name that could not be translated into a corresponding LE/VSE 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 was not able 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: 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 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

CEE3420W The offset from UTC/GMT to local time was set to zero.

Explanation: The DATE job control statement was used to override the system date. When you use the DATE job control statement to override the system date, LE/VSE sets UTC/GMT to local time, and sets the offset from UTC/GMT to local time to zero.

Programmer Response: If your application requires an accurate offset from UTC/GMT to local time, remove the DATE job control statement from your JCL.

System Action: The offset from UTC/GMT is set to zero.

Symbolic Feedback Code: CEE3AS

CEE3421W The local time was calculated using a date supplied in the JCL.

Explanation: The DATE job control statement was used to override the system date. When you use the DATE job control statement to override the system date, LE/VSE calculates local time using the date specified in the DATE job control statement. However, VSE does not increment the specified date at midnight. Therefore, if your job runs past midnight, the local time returned by LE/VSE might not be what you expect.

Programmer Response: If your application requires an accurate local time, or if your job might run past midnight, remove the DATE job control statement from your JCL.

System Action: Local time is calculated using the specified date.

Symbolic Feedback Code: CEE3AT

CEE3422W GMT was set to local time.

Explanation: The DATE job control statement was used to override the system date. When you use the DATE job control statement to override the system date, LE/VSE sets GMT to local time.

Programmer Response: If your application requires an accurate GMT, remove the DATE job control statement from your JCL.

System Action: GMT is set to local time.

Symbolic Feedback Code: CEE3AU

CEE3423S Non-numeric data was detected in the year field of the date supplied in the JCL.

Explanation: A non-numeric year value was specified on the DATE job control statement. The century window could not be initialized.

Programmer Response: Correct the date supplied on the DATE job control statement.

System Action: The first year of the century window is set to zero.

Symbolic Feedback Code: CEE3AV

CEE3424S CEE5SMO was called from outside a user-written condition handler.

Explanation: CEE5SMO can only be called from within a user-written condition handler

Programmer Response: Only code calls to CEE5SMO 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 the condition they were entered to handle to new conditions. 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. (LE/VSE 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 any user-written condition handler 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 CEE5LNG. The current language was returned in the desired language parameter.

Explanation: CEE5LNG cannot POP since the resulting stack will be 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 CEE5LNG 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 *LE/VSE 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 CEE5LNG 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 CEE5LNG 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 CEE5CTY. The current country code was returned in the country code parameter.

Explanation: CEE5CTY cannot POP since the resulting stack will be 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 CEE5CTY 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 *LE/VSE 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 CEE5CTY 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

CEE3459S The *function* requested in CEE5CTY 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 CEE5MDS.

Explanation: The decimal separator parameter must be a 2-character field. The resulting decimal separator might not be valid. The decimal separator is 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 CEE5MDS. 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 is returned.

Programmer Response: Provide a valid country_code parameter. A list of the valid country codes is provided in *LE/VSE 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 CEE5MCS.

Explanation: The currency symbol parameter must be a 2-character field. The resulting currency symbol may not be valid. The currency symbol is 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 CEE5MCS. 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 is returned.

Programmer Response: Provide a valid country code parameter. A list of the valid country codes is provided in *LE/VSE 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 CEE5MTS.**

Explanation: The thousands separator parameter must be a 2-character field. The resulting thousands separator might not be valid. The thousands separator is 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 CEE5MTS. 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 is returned.

Programmer Response: Provide a valid country code parameter. A list of the valid country codes is provided in *LE/VSE 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* may not be valid. The *date-pic-string* is 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 is returned.

Programmer Response: Provide a valid country code parameter. A list of the valid country codes is provided in *LE/VSE 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* is 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 is returned.

Programmer Response: Provide a valid country code parameter. A list of the valid country codes is provided in *LE/VSE Programming Reference*.

System Action: The default time picture string is returned.

Symbolic Feedback Code: CEE3CD

CEE3470E The date and time string *datetime-string* was truncated and was not defined in CEEFMDT.

Explanation: The *datetime-string* parameter must be an 80-character field. The resulting *datetime-string* might not be valid. The *datetime-string* was left-justified and padded on the right with a blank if necessary.

Programmer Response: Provide an 80-character *datetime-string* parameter.

System Action: The *datetime-string* 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-string* was returned.

Explanation: The *country-code* parameter was not a valid 2-character country code. The default datetime string was returned.

Programmer Response: Provide a valid country code parameter. A list of the valid country codes is provided in the *LE/VSE Programming Reference*.

System Action: The default datetime 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

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 filename or destination is not valid or is 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 I/O error was detected while trying to OPEN, WRITE, or CLOSE a given filename or destination.

Programmer Response:
- If the error occurs under CICS, check that CESE is not full and is available.
- If the error occurs in BATCH with TERMTHDACT(,,MSGFL,) set, ensure SYSLST is available.
- If a CEEDUMP DLBL has been specified, ensure that the file is available and not full.
- When TERMTHDACT(,,LSTQ,) is set in BATCH, ensure that there is sufficient free space available in the VSE/POWER data files and queue files, to receive the required dump output. Also check that VSE/POWER is active and able to accept spool requests from external sources.

- For further diagnosis information when TERMTHDACT(,,LSTQ,) is set, refer to the associated CEEL011S message that is issued to the VSE operator console.

System Action: The enclave is terminated.

Symbolic Feedback Code: CEE3D4

CEE3493W An internal message services error occurred while attempting to close the message file.

Explanation: LE/VSE could not close the specified filename, because either LE/VSE 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 filename.

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 is 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 filename used for the message file is a valid name, and the file is usable.

System Action: No system action is taken.

Symbolic Feedback Code: CEE3DB

CEE3500S Not enough storage was available to load ***phase-name*****.**

Explanation: Not enough storage was available to load the requested phase into virtual memory.

Programmer Response: Ensure that the partition GETVIS size is sufficient to run the application. If necessary, delete phases not currently needed by the application or free unused storage and retry the load request.

System Action: Phase is not loaded. The application might abend.

Symbolic Feedback Code: CEE3DC

CEE3501S The phase ***phase-name*** **was not found.**

Explanation: The system could not find the phase, whose name was specified on the parameter list to the LE/VSE load service, in any of th sublibraries in the sublibrary search chain, or in the SVA.

Programmer Response: Make sure the requested program name is correct, and that the requested phase is available in one of the sublibraries in the sublibrary search chain, or in the SVA. Correct the error, and execute the job again.

System Action: Phase is not loaded. The application might abend.

Symbolic Feedback Code: CEE3DD

CEE3502S **The phase name** *phase-name* **was too long.**

Explanation: The phase name length is greater than the name length supported by the underlying operating system.

Programmer Response: Correct the phase name length and execute the job again.

System Action: Name length is truncated to the name length supported by the underlying operating system. The requested phase might or might not be loaded.

Symbolic Feedback Code: CEE3DE

CEE3503S **The load request for phase** *phase-name* **was unsuccessful.**

Explanation: The system could not load the phase.

Programmer Response: Make sure the requested program name is correct, and that the requested phase is available in one of the sublibraries in the sublibrary search chain, or in the SVA. Check that the requested phase is not a move-mode phase.

System Action: Phase was not loaded. The application might abend.

Symbolic Feedback Code: CEE3DF

CEE3504S **The delete request for phase** *phase-name* **was unsuccessful.**

Explanation: The phase might already have been deleted, was never loaded, or exceeded the maximum load count.

Programmer Response: Make sure the requested phase name is correct.

System Action: None.

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 phase** *phase-name* **could not be loaded.**

Explanation: During *LIBVEC* initialization the library vector table descriptor phase could not be found.

Programmer Response: Make sure you passed the name of the library vector table descriptor rather than the entry address. Make sure the name was not incorrectly modified and that the phase is available in one of the sublibraries in the sublibrary search chain, or in the SVA.

System Action: LIBVEC initialization is not performed.

Symbolic Feedback Code: CEE3DH

CEE3506S **The library packaged subroutine phase** *phase-name* **could not be loaded.**

Explanation: The system could not find the phase, whose name was specified in the library vector table (LIBVEC) descriptor phase, in any of the sublibraries in the sublibrary search chain, or in the SVA.

Programmer Response: Make sure the requested program name is correct, and that the requested phase is available in one the sublibraries in the sublibrary search chain, or in the SVA. Correct the error, and execute the job again.

System Action: Phase 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 library vector table (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** *phase-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 ***phase-name*** **either exceeded 1024 or was less than 1.**

Explanation: A minimum of 1 library routine entry name is required to build the library vector table. A maximum of 1024 library routines entry names is allowed in a library vector table.

Programmer Response: If library vector slots exceed the maximum, then you may have to build more than one library vector table.

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

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 library vector table.

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 is unsupported in the LE/VSE environment.

Programmer Response: LE/VSE will run under the control of, or in conjunction with, the following operating systems/subsystems: VSE/ESA, z/VSE, CICS/VSE, CICS Transaction Server for VSE/ESA.

System Action: LE/VSE terminates the application.

Symbolic Feedback Code: CEE3DQ

CEE3515I No phases were loaded.

Explanation: No application phases have been loaded via LE/VSE load service in the current application.

Programmer Response: No programmer response is necessary.

System Action: No system action is taken.

Symbolic Feedback Code: CEE3DR

CEE3520 The token passed to the CEERELES macro was invalid.

Explanation: The token passed to the CEERELES macro was not the one that was returned from the CEEFETCH macro.

Programmer Response: Ensure CEEFETCH returned successfully, and the returned token is the same as the one passed to the CEERELES macro. (For details of the CEEFETCH and CEERELES macros, refer to the *LE/VSE Programming Guide*).

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

Symbolic Feedback Code: CEE3E0

CEE3550I LE/VSE C/VSE Run-Time Initialized.

Explanation: During LE/VSE initialization under CICS, LE/VSE has successfully initialized the C for VSE/ESA Run-Time Environment.

Programmer Response: None.

System Action: Initialization Continues.

Symbolic Feedback Code: None.

CEE3551I LE/VSE COBOL Run-Time Initialized.

Explanation: During LE/VSE initialization under CICS, LE/VSE has successfully initialized the COBOL Run-Time Environment.

Programmer Response: None.

System Action: Initialization Continues.

Symbolic Feedback Code: None.

CEE3552I LE/VSE PL/1 Run-Time Initialized.

Explanation: During LE/VSE initialization under CICS, LE/VSE has successfully initialized the PL/1 for VSE/ESA Run-Time Environment.

Programmer Response: None.

System Action: Initialization Continues.

Symbolic Feedback Code: None.

CEE3553I CICS Options Newcopy Started.

Explanation: The CICS session operator has started the CICS-wide installation default runtime options *newcopy* facility.

Programmer Response: None.

System Action: Activation of the new CICS-wide options begins.

Symbolic Feedback Code: None.

CEE3554I CICS Options Newcopy Complete.

Explanation: The newcopy facility has finished activating the new CICS-wide default runtime options.

Programmer Response: None.

System Action: Activation of the new CICS-wide options has completed sucessfully.

Symbolic Feedback Code: None.

CEE3555E CICS Options Newcopy Failed.

Explanation: The newcopy facility has failed while trying to activate the CICS-wide default runtime options. These are the return codes and reason codes you can receive, and their explanations:

Return Code 08
Reason Code 3 – NOSTG returned from CICS during GETMAIN request.
All other reason codes – report to your Systems Programmer or IBM Support Center.

Return Code 12
Reason Code 2 – PGMIDERR returned from CICS during LOAD request on CEECOPT.
Reason Code 3 – NOTAUTH returned from CICS during LOAD request on CEECOPT.
All other reason codes – report to your Systems Programmer or IBM Support Center.

Return Code 16
Reason Code 2 – PGMIDERR returned from CICS during RELEASE of CEECOPT.
Reason Code 3 – NOTAUTH returned from CICS during RELEASE of CEECOPT.
All other reason codes – report to your Systems Programmer or IBM Support Center.

Return Code 20
Reason Code 2 – PGMIDERR returned from CICS during NEWCOPY request for CEECOPT.
Reason Code 3 – NOTAUTH returned from CICS during NEWCOPY request for CEECOPT.
All other reason codes – report to your Systems Programmer or IBM Support Center.

Programmer Response: Review CICS output log (MSGFILE destination) for error information:
- For reason code 2, check that CEECOPT has been defined to the CICS system by either the PPT or CSD definitions.
- For reason code 3 with RC=08, check that sufficent 31-bit GETVIS (for CICS/VSE) or EDSA (for CICS/VSE/TS) storage is available.
- For reason code 3 with RC=20,16 or 12, check that the correct authority has been granted to the NEWC transaction, and that the CICS user has sufficent authority to request the newcopy function.
- For any return or reason codes not listed above, please report the failure with details from the CICS output log, to your Systems Programmer or IBM support centre.

System Action: Activation of new CICS options is aborted. Previous CICS-wide default runtime options remain in effect.

Symbolic Feedback Code: None.

CEE3556W Use of TD Queue *queue* has failed due to a *resp*. Switching to CESE for this Enclave.

Explanation: The transient data queue specified in the active MSGFILE run time option experienced a failure when LE/VSE attempted to write messages to it.

Programmer Response: The field *resp* indicates the response code returned from CICS when LE/VSE issued the `EXEC CICS WRITE` command. Refer to the *CICS Application Programming Guide* for an explanation of the *resp* code.

Correct the MSGFILE run-time option so that it specifies a transient data queue that has been defined correctly, and that is available for LE/VSE to use.

System Action: LE/VSE automatically switches to the CESE transient data queue and continues processing the message request. All subsequent messages for this enclave will be directed to the CESE transient data queue, irrespective of the MSGFILE setting.

Symbolic Feedback Code: None.

CEE3557W Write to LSTQ has failed. Dump production aborted for this process.

Explanation: A failure has occurred while attempting to output an LE/VSE dump to the LSTQ destination. For further information, refer to message CEEL011S displayed on the VSE operators console.

Programmer Response: For error diagnosis information, refer to message CEEL011S.

System Action: The application terminates and the dump is lost.

Symbolic Feedback Code: None.

CEE3558I LE/VSE LSTQ support unavailable.

Explanation: When LE/VSE attempted to establish the resources required for the LSTQ dump output support, one or more of these resources either failed or could not be obtained.

Programmer Response: Check that:
1. Module CEELEDT is available to the CICS system, or CEEBLEDT is available in BATCH.
2. The security settings for loading the appropriate module are set correctly.
3. A CEELOPT macro has been included in the generated run-time options module.
4. The included CEELOPT macro describes valid LSTQ options that are supported by VSE/POWER.

System Action: LE/VSE execution continues but with LSTQ support disabled.

Symbolic Feedback Code: None.

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 a comma, a right parenthesis, or a 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* is an option from a previous release that is not supported or mapped by LE/VSE.

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* is an option from a previous release that is supported by LE/VSE 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.

Explanation: *option* is not an LE/VSE option or an option that is recognized by LE/VSE for previous release language compatibility.

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: 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 *LE/VSE 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) is an option from a previous release that is supported by LE/VSE for compatibility, but ignored if TRAP is specified.

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* has been defined as non-overridable at installation time.

Programmer Response: Correct the run-time options string to not specify this *option*.

System Action: The option is ignored.

Symbolic Feedback Code: CEE3GV

CEE3616I The string ' *string* ' was not a valid suboption of the run-time option *option*.

Explanation: *string* was not in the set of recognized values.

Programmer Response: Correct the run-time options string.

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.

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* is not valid from the invocation command.

Programmer Response: Correct the run-time options string.

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 run-time options string.

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* is not valid from the assembler user exit.

Programmer Response: Correct the run-time options string.

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 run-time options string.

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 sub-option is eight.

Programmer Response: Correct the run-time options string.

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 will then 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 preformed.

Symbolic Feedback Code: CEE3HB

CEE3628I The run-time option *option* was not valid from the programmer defaults.

Explanation: *option* is not valid from the programmer defaults.

Programmer Response: Correct the run-time options string.

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* is an old language option that is being supported by LE/VSE for compatibility. The user should use *le-option* instead.

Programmer Response: Change the run-time options string to use the *le-option* instead.

System Action: *old-option* is partially mapped to its LE/VSE 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 *LE/VSE 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: 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 LE/VSE for compatibility.

Programmer Response: Change the run-time options string to use the TRAP option instead.

System Action: STAE, SPIE, NOSTAE, or NOSPIE are mapped to TRAP.

Symbolic Feedback Code: CEE3HF

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

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. X'00'

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. X'00'

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* X'00'

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

CEE3649W The parameter string returned from CEE5PRM exceeded the maximum length of 80 bytes and was truncated.

Explanation: The output character string passed to CEE5PRM is not long enough to contain the text of the user parameters. It has been truncated.

Programmer Response: Execute with a shorter user parameter string.

System Action: The user parameter string is truncated to fit the character string argument passed to CEE5PRM. The truncated value is returned to the caller in the character string argument.

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: None.

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: Previous messages in the message file will contain the address, and the expected and actual data for each damaged area found.

System Action: The enclave is terminated with abend code U4042.

Symbolic Feedback Code: CEE3JM

CEE3703I In *controlblock* Control Block, the *fieldname* is damaged.

Explanation: An LE 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.

System Action: None.

Symbolic Feedback Code: CEE3JN

CEE3704I Expected data at *address* : *data*.

Explanation: Provides the address *address* and expected data *data* when heap damage is found.

Programmer Response: The message following this one in the message file will provide the actual data found at the damaged location.

System Action: None.

Symbolic Feedback Code: CEE3JO

CEE3705I Pointer at *address* should point to a valid *controlblock*.

Explanation: The pointer at location *address* should point to an LE control block with a *controlblock* eye catcher.

Programmer Response: The message following this one in the message file will provide the actual data found at the damaged location.

System Action: None.

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* contain header information and will not match the heap_free_value. They are not used in the comparison.

Programmer Response: The message following this one in the message file will provide the actual data found at the damaged location.

System Action: None.

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 message following this one in the message file will provide the actual data found at the damaged location.

System Action: None.

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 message following this one in the message file will provide the actual data found at the damaged location.

System Action: None.

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 its length does not match a Heap Storage Element.

Programmer Response: The message following this one in the message file will provide the actual data found at the damaged location.

System Action: None.

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 message following this one in the message file will provide the actual data found at the damaged location.

System Action: None.

Symbolic Feedback Code: CEE3JU

CEE3711I **Processing of the HEAPCHK runtime 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.

System Action: None.

Symbolic Feedback Code: CEE3JV

CEE3750S **Insufficient storage was available for use by *module-name*.**

Explanation: There is not enough free storage available for the DTF builder CWI *module-name* to build a DTF.

Programmer Response: Ensure that the GETVIS storage available in the partition is sufficient to run your application. Verify that the storage sizes specified in the HEAP run-time option are reasonable.

System Action: The DTF is not built.

Symbolic Feedback Code: CEE3L6

CEE3751S **A search of the LIOCS phase by *module-name* did not locate the routine *routine-name*.**

Explanation: The DTF builder CWI *module-name* was unable to find the routine *routine-name* in its associated LIOCS phase.

Programmer Response: Add the required routine to the LIOCS phase using the CEEXCDMD, CEEXDUMD or CEEXPRMD macro. For more information about these customization macros, see *LE/VSE Customization Guide*.

System Action: The DTF is not built.

Symbolic Feedback Code: CEE3L7

CEE3756S **The parameter** *parameter-name* **was invalid in a call to** *module-name***.**

Explanation: The DTF builder CWI *module-name* was passed an invalid value in parameter *parameter-name*.

Programmer Response: Correct the parameter value.

System Action: The DTF is not built.

Symbolic Feedback Code: CEE3LC

CEE3757S **The parameter** *parameter-name1* **conflicts with the parameter** *parameter-name2* **in a call to** *module-name***.**

Explanation: The DTF builder CWI *module-name* was passed the parameter *parameter-name1* which conflicts with parameter *parameter-name2*.

Programmer Response: Correct the conflicting parameters.

System Action: The DTF is not built.

Symbolic Feedback Code: CEE3LD

CEE3758S **The parameter** *parameter-name1* **conflicts with the parameters** *parameter-name2* **and** *parameter-name3* **in a call to** *module-name***.**

Explanation: The DTF builder CWI *module-name* was passed the parameter *parameter-name1* which conflicts with parameters *parameter-name2* and *parameter-name3*.

Programmer Response: Correct the conflicting parameters.

System Action: The DTF is not built.

Symbolic Feedback Code: CEE3LE

CEE3775W **A conflict was detected between the TERMTHDACT suboption LSTQ and TERMTHDACT level setting. TERMTHDACT level setting has been set to TRACE.**

Explanation: The thread termination dump output-level setting conflicts with the destination setting of LSTQ. A level of TRACE or more must be used with LSTQ.

Programmer Response: Change the value of the detail level to TRACE or more.

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 number 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 0 is used.

Symbolic Feedback Code: CEE3M5

CEE3782E **The value of the REGSTOR option of CEE5DMP was not in the valid range 0-256.**

Explanation: The value of the REGSTOR option of CEE5DMP was not a number in the range of 0-256.

Programmer Response: Change the value of the REGSTOR option to be the valid range.

System Action: The value is ignored. The default value of 0 is used.

Symbolic Feedback Code: CEE3M6

CEE3800S **The address passed to the stack segment service was not within any LE stack segment.**

Explanation: The address passed to the stack segment bounds service was not within any currently allocated LE/VSE stack segment.

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

System Action: The bounds, segment type and chain are undefined.

Symbolic Feedback Code: CEE3MO

CEE3817 **The member event handler did not return a usable function pointer.**

Explanation: The member language which compiled the input load module either does not support the fetch function, or encounted an unrecoverable error.

Programmer Response: Ensure the input load module is compiled from a language which supports the fetch function.

System Action: The called service returns an unusable function pointer.

Symbolic Feedback Code: CEE3N9

CEE3818 The member event handler encountered an error.

Explanation: The member language which compiled the input load module either does not support the release function, or encounted an unrecoverable error.

Programmer Response: Ensure the input load module is compiled from a language which supports the release function.

System Action: The function is not released.

Symbolic Feedback Code: CEE3NA

CEE3900S The function code passed to CEE5USR was not 1 or 2.

Explanation: The function_code specified in a CEE5USR call is invalid.

Programmer Response: Invoke CEE5USR 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 CEE5USR was not 1 or 2.

Explanation: The field number specified in a CEE5USR call is invalid.

Programmer Response: Invoke CEE5USR with field_number 1 or 2.

System Action: Neither SET nor QUERY is performed.

Symbolic Feedback Code: CEE3PT

CEE3907I Read-Access or Update-Access to Supplied Storage Address is Unavailable

Explanation: The CEE5TSTG service has been called with an address that is not allowed for read-access.

Programmer Response: Any attempt to use the supplied address for any processing in the current execution key will result in a program check.

System Action: Processing continues with the next sequential instruction after the call to CEE5TSTG.

Symbolic Feedback Code: CEE3PZ

CEE3908I Update-Access to Supplied Storage Address is Unavailable

Explanation: The CEE5TSTG service has been called with an address that is not allowed for update-access.

Programmer Response: Any attempt to modify storage pointed-to by the address supplied to CEE5TSTG in the current execution key, will result in a program check. Read-access is permitted in the current execution key.

System Action: Processing continues with the next sequential instruction after the call to CEE5TSTG.

Symbolic Feedback Code: CEE3Q0

CEE3909I Incompatible TRAP run-time option set.

Explanation: The CEE5TSTG service has been called when TRAP(OFF) is set.

Programmer Response: The CEE5TSTG service is only available when TRAP(ON,MIN) or TRAP(ON,MAX) are set.

System Action: Processing continues with the next sequential instruction after the call to CEE5TSTG.

Symbolic Feedback Code: CEE3Q1

CEE4001S General Failure: Service could not be completed.

Explanation: An error was encountered attempting to complete a C 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 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 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

CEE5176S There was insufficient storage to process an environment variable.

Explanation: The environment variable processing service CEEBENV was called to get, set, or clear environment variable(s). However, there was insufficient system storage available to do so.

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 internal environment variable processing service CEEBENV was called to get or set an environment variable. 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 internal environment variable processing service CEEBENV was called to get, set, or clear environment variable. However, it encountered an invalid anchor or array address.

Programmer Response: If `**environ` is used to set or clean 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

VSE/POWER LSTQ Support Run-Time Messages

The following messages relate to the LE/VSE support of VSE/POWER LSTQ.

CEEL000S **VSE/POWER is currently terminating. Connection request denied.**

Explanation: A request to output a failing dump to the VSE/POWER LSTQ was aborted as VSE/POWER was in termination mode and cannot accept any new connections.

Programmer Response: Ensure VSE/POWER is currently active and not in shutdown mode.

System Action: The request to send the dump output to a separate LSTQ member is ignored. Dump output is instead directed to the default environment-dependent location.

Symbolic Feedback Code: None

CEEL001S **VSE/POWER in abnormal termination. Connection disrupted.**

Explanation: While sending dump output to the VSE/POWER LSTQ, the request was aborted as VSE/POWER abnormally terminated.

Programmer Response: Ensure VSE/POWER is available for the complete duration of dump production. Alternatively, set TERMTHDACT to MSGFL and re-run failing transaction.

System Action: The dump request is aborted. Dump production is aborted.

Symbolic Feedback Code: None

CEEL002S **Connection to VSE/POWER could not be completed within a 10 second delay limit.**

Explanation: While requesting a connection to VSE/POWER, the time limit of 10 seconds was reached before the connection could be completed.

Programmer Response: Ensure VSE/POWER is available and sufficient system GETVIS storage is available to process the required number of concurrent connections to VSE/POWER.

System Action: The request to send the dump output to a separate LSTQ member is ignored. Dump output is instead directed to the default environment-dependent location.

Symbolic Feedback Code: None

CEEL003S **Insufficient storage to establish identification with z/VSE.**

Explanation: While sending an identification request to VSE, an “insufficient storage” response was returned.

Programmer Response: Ensure sufficient system GETVIS storage is available to process the required number of concurrent connections to VSE/POWER.

System Action: The request to send the dump output to a separate LSTQ member is ignored. Dump output is instead directed to the default environment-dependent location.

Symbolic Feedback Code: None

CEEL004S **Insufficient storage to establish XPCC connection with z/VSE.**

Explanation: While requesting an XPCC connection with VSE, an “insufficient storage” response was returned.

Programmer Response: Ensure sufficient system GETVIS storage is available to process the required number of concurrent XPCC connections.

System Action: The request to send the dump output to a separate LSTQ member is ignored. Dump output is instead directed to the default environment-dependent location.

Symbolic Feedback Code: None

CEEL005I **Removing Existing VSE/POWER Connection.**

Explanation: A previous task which was using the LSTQ support (for details, see “Generating a Dump with TERMTHDACT” on page 34) has failed. This task has left an open connection to VSE/POWER.

Programmer Response: None (information only).

System Action: The open connection to VSE/POWER is removed. A new connection to VSE/POWER is established for the currently-active task.

Symbolic Feedback Code: None

CEEL009S **Abend queue missing while trying to remove existing link. Restart CICS.**

Explanation: While attempting to remove an old connection with VSE/POWER, the queue that holds required information for the removal process could not be located.

Programmer Response: To remove the old link, a restart of CICS is required.

System Action: The link is not removed. Dump production to the LSTQ is aborted. Dump output is re-directed to the MSGFILE destination. Use of the LSTQ destination from this terminal is no longer available until CICS is restarted.

Symbolic Feedback Code: None

CEEL010W **Write to TS Queue failed. Unable to recover from connection failures.**

Explanation: The LE/VSE spool interface stores recovery information in a task-specific CICS temporary storage queue. This queue is used if a later spool connection is required but a duplication is detected. Without the information in this CICS TS queue, removal of orphaned spool connections is not possible. A CICS restart would be required to terminate these links.

Programmer Response: Ensure that as a minimum, CICS "main" temporary storage is available at all times, and that the TS queue VSAM file is not full.

System Action: Any failure experienced by this spool connection before completion is established cannot be dynamically removed. A CICS restart would be required.

Symbolic Feedback Code: None

CEEL011S **REQ=*reqid* Failed at: block VSE-RC/REAS=*??*/*??* PWR-RC/FDBK=*??*/*??***

Explanation: While attempting to output an LE/VSE dump to the VSE/POWER LSTQ, a failure occurred. The VSE-RC/REAS fields refer to XPCC return code and reason codes. The PWR-RC/FDBK fields refer to VSE/POWER Spool-access return and reason codes. Dump production for this application is prematurely terminated.

Programmer Response: Refer to the appropriate VSE/POWER manual using the displayed return and reason codes. Correct the error before attempting to use the LSTQ support again.

System Action: Dump production is terminated. The dump is lost.

Symbolic Feedback Code: None

Attention Routine Support Run-Time Messages

The following messages relate to the LE/VSE support of certain *attention routine* (AR) commands. The messages are displayed in response to the attention routine commands that are supported by LE/VSE.

CEL4000I *exit-name exit-type exit-status version*.

Explanation: Message Title. If available, the assembler user exit shown is currently active in this LE/VSE BATCH environment.

Programmer Response: None (information only).

System Action: None

Symbolic Feedback Code: None

CEL4001I Currently defined Lang. Env. exits are *list-of-exits*.

Explanation: Message Title. The exits shown (if any) are currently active in this LE/VSE BATCH environment.

Programmer Response: None (information only).

System Action: None

Symbolic Feedback Code: None

CEL4002I *exit-name exit-type dump-cntrl*.

Explanation: Message Title only.

Programmer Response: None (information only).

System Action: None

Symbolic Feedback Code: None

CEL4003E Unknown Lang. Env. for VSE/ESA command entered.

Explanation: An operator's console command was entered that contained the 'D CEE' prefix, but the completed command was not recognized by LE/VSE.

Programmer Response: Correct the command and re-enter.

System Action: No command processing is performed.

Symbolic Feedback Code: None

CEL4004E Lang. Env. EXIT module(s) not found in SVA.

Explanation: A 'D CEE,CEEEXIT' command was issued, but there were no LE/VSE abnormal termination exit modules found in the SVA to interrogate.

Programmer Response: The LE/VSE CEEBXTAN module must be resident in the SVA before the 'D CEE,CEEEXIT' command can display information about an abnormal termination exit.

System Action: No abnormal termination exit information is displayed.

Symbolic Feedback Code: None

CEL4005E Lang. Env. Assembler User Exit module not found in SVA.

Explanation: A 'D CEE,CEEEXIT' command was issued, but there were no LE/VSE assembler user exit modules found in the SVA to interrogate.

Programmer Response: The LE/VSE CEEBXITA module must be resident in the SVA before the 'D CEE,CEEEXIT' command can display information about assembler user exits.

System Action: No assembler user exit information is displayed.

Symbolic Feedback Code: None

CEL4006W Command processing incomplete.

Explanation: The LE/VSE attention routine command that was last entered, was not processed successfully.

Programmer Response: Review the last entered LE/VSE attention routine command and correct any errors reported.

System Action: No LE/VSE information is displayed.

Symbolic Feedback Code: None

CEL4007I *exit-name exit-type dump-control*.

Explanation: The displayed abnormal termination exits are active in the BATCH environment and have been defined with *dump-control* processing:

exit-name
: The name of the exit that LE/VSE will call when an abnormal termination condition occurs.

exit-type
: The exit type for this exit name. Can be an abnormal termination exit.

dump-control
: Displays the setting of the 'type' sub-option on the CEEXART macro.

Programmer Response: None (information displayed describes the active exits details).

System Action: LE/VSE exit information is displayed.

Symbolic Feedback Code: None

CEL4008S **AR-Interface Initialization has not completed successfully.**

Explanation: The execution of job CEEWARC has either not been performed, or it has failed in some way.

Programmer Response: Ensure that the CEEWARC job has been run and that it has completed successfully.

System Action: The LE/VSE command is not performed.

Symbolic Feedback Code: None

CEL4009S **AR-Interface initialization only partially completed.**

Explanation: The job CEEWARC has been run but it has not completed successfully.

Programmer Response: Re-run the CEEWARC job and ensure that it completes successfully with no error messages. Verify that there is at least 4K (approximately) of SVA-31 GETVIS available.

System Action: The issued LE/VSE command is not performed.

Symbolic Feedback Code: None

CEL4010I *exit-name exit-type exit-status version*.

Explanation: The LE/VSE assembler user exit information is displayed. *version* will only be displayed if the *exit-status* reported is 'default'.

exit-name
The catalogued name of the assembler user exit.

exit-type
Can be an assembler user exit.

exit-status
If the exit is detected to simply be the supplied default version or another OEM version.

version The *version,release* and *modification level* information available from the default or user-tailored sample versions of the assembler user exit. It might not be possible to report on OEM versions when non-standard entry code has been used.

Programmer Response: None

System Action: None

Symbolic Feedback Code: None

CEL4011I **Language Environment Version:** *version,release modification-level APAR-level*

Explanation: The LE/VSE *version,release* and *modification-level* are displayed, followed by the latest APAR that has been correctly installed.

Programmer Response: None

System Action: None

Symbolic Feedback Code: None

CEL4012I *language installed SVA-resident SVA-address APAR-level*

Explanation: Message title only.

Programmer Response: None (information only).

System Action: None

Symbolic Feedback Code: None

CEL4013I *language installed SVA-residency SVA-load-address APAR-level*

Explanation: The system displays the available information for each of the available LE/VSE-supported languages. The *language* may also be shown as a module name when the system displays information on LE/VSE core modules. The *SVA-load-address* information can only be displayed providing the specific language component's event handler is resident in the SVA.

Programmer Response: If the *APAR-level* shown is "unknown", the listed language component is either not installed, or the component's level module is unavailable. Ensure that *xxx*LVL.PHASE is resident in the Language Environment installation library, where *xxx* is the language component prefix identifier. For example:
- IGZ = COBOL
- IBM = PL/I
- EDC = C language

System Action: None

Symbolic Feedback Code: None

CEL4014I **Current Language Environment for VSE/ESA Status Information.**

Explanation: The message title displays the latest APAR level information for the current base components of Language Environment for VSE/ESA.

Programmer Response: None

System Action: None

Symbolic Feedback Code: None

CEL4015I *module installed SVA-resident SVA-address*

Explanation: Message Title only.

Programmer Response: None (information only).

System Action: None

Symbolic Feedback Code: None

CEL4016E CDLOAD failed with unexpected return code for module CEEBXTAN.

Explanation: The load for CEEBXTAN has failed.

Programmer Response: Check that CEEBXTAN is loaded in the SVA, and that the operator issuing the command has the required authority to access this module.

System Action: The command is not processed.

Symbolic Feedback Code: None

CEL4017E CDLOAD failed with unexpected return code for module CEEBXITA.

Explanation: The load for CEEBXITA has failed.

Programmer Response: Check that CEEBXITA is loaded in the SVA, and that the operator issuing the command has the required authority to access this module.

System Action: The command is not processed.

Symbolic Feedback Code: None

CEL4018I Produced by CEL4CMDR at X'????????' APAR ***APAR level*****.**

Explanation: The system displays the current attention routine interface details. This information may be requested by Level 2 support for any problem diagnosis.

Programmer Response: None

System Action: None

Symbolic Feedback Code: None

CEL4026W Load for CEELVL failed. APAR details are unavailable.

Explanation: When initializing the attention routine support, an attempt to load the CEELVL module failed. LE/VSE APAR information will not be available until this module is made available to load.

Programmer Response: Ensure that the CEELVL module is in the LE/VSE installation library, and that the executing job step has the authority required to load this module.

System Action: None. Initialization continues with LE/VSE APAR information unavailable.

Symbolic Feedback Code: None

CEL4032S Attention Interface cannot continue. Terminating.

Explanation: A severe error has occurred while attempting to initialize the attention routine interface for LE/VSE.

Programmer Response: Review the previous message(s) issued prior to this one for problem investigation.

System Action: Initialization of the attention routine interface is terminated.

Symbolic Feedback Code: None

CEL4033I Refreshing Lang. Env. Status Info. Interface already activated.

Explanation: The CEL4VNDR program (job CEEWARC) has been executed when the AR interface is already active. The Language Environment status information has therefore been updated and placed online.

Programmer Response: None (information only).

System Action: Any new status changes that may have been made since the last time CEL4VNDR was executed have now been placed online. They will be available via the D CEE,CEESTAT command.

Symbolic Feedback Code: None

CEL4034E GETVIS for Partition Storage failed.

Explanation: Insufficient partition GETVIS storage is available to execute the attention routine interface for LE/VSE.

Programmer Response: Run the attention routine initialization job in a larger partition, or make more partition GETVIS (31/ANY) available.

System Action: Initialization of the attention routine interface is terminated.

Symbolic Feedback Code: None

CEL4036E Load of module CEL4CMDR has failed.

Explanation: For the attention routine support, module CEL4CMDR must be loaded in the SVA.

Programmer Response: Ensure CEL4CMDR is loaded in the SVA. If this module is already resident in the SVA, collect the dump produced and contact your systems programmer or IBM support center.

System Action: Initialization of the attention routine interface is terminated.

Symbolic Feedback Code: None

CEL4037E Attention routine Interface module CEL4CMDR not found in the SVA.

Explanation: Module CEL4CMDR loaded successfully but was not found resident in the SVA.

Programmer Response: Ensure CEL4CMDR is loaded in the SVA, and that this version is the first to be found in the active PHASE search chain.

System Action: Initialization of the attention routine interface is terminated.

Symbolic Feedback Code: None

CEL4039E Insufficient SVA GETVIS-31 storage available.

Explanation: While attempting to acquire some SVA-31 GETVIS storage, the request failed.

Programmer Response: Ensure there is sufficient contiguous SVA-31 GETVIS storage available for the attention routine. At least 4096 bytes of contiguous storage needs to be available.

System Action: Initialization of the attention routine interface is terminated.

Symbolic Feedback Code: None

CEL4040I SVA-31 GETVIS Storage has been freed.

Explanation: The SVA-31 storage that is required by the AR interface task has been freed. This is because either this action was requested, or because of an initialization problem.

Programmer Response: None (informational only).

System Action: None

Symbolic Feedback Code: None

CEL4041E SVA-31 Storage FREEVIS request has failed.

Explanation: While attempting to free the SVA-31 GETVIS storage previously acquired, the FREEVIS request failed.

Programmer Response: Gather the dump produced and contact your systems programmer or IBM support center. An IPL will be required to free the orphaned SVA-31 storage. Subsequent executions of CEEWARC will acquire another 4 KB (approximately) of SVA-31 GETVIS storage.

System Action: Deactivation of the attention routine interface completed but the SVA-31 GETVIS storage is not freed.

Symbolic Feedback Code: None

CLER CICS Transaction Support Message

The following messages relate to the LE/VSE support of the CICS **CLER** transaction, which allows you to interactively display and modify LE/VSE CICS-wide default runtime options while your CICS system is running. For details about the CLER transaction, refer to the *LE/VSE Customization Guide*.

CEL4101E Executing on an Unsupported LE/VSE Run-Time.

Explanation: While attempting to use the CICS CLER transaction, it was found that the executing LE/VSE level is not at the level required to support the version of the CLER function being used. A possible mismatch exists between the installed LE/VSE level and the VSE version being used.

Programmer Response: Check that the correct LE/VSE level is installed for the executing VSE level, and that the LE/VSE level is executing in the CICS system where the CLER transaction is being used. The options reports generated by the NEWC or ROPC transactions can be used to determine the currently-active LE/VSE level in the CICS system. The SIR console command can be used to determine the currently-active VSE level.

System Action: The application is terminated.

Symbolic Feedback Code: None

Chapter 9. C Utility Messages

This chapter describes return codes and messages from the:

- Prelinker utility (see *page 218*)
- `localedef` utility (see *page 223*)
- `iconv` utility (see *page 233*)
- `genxlt` utility (see *page 235*)
- DSECT utility (see *page 240*)
- `uconvdef()` utility (see *page 236*)

For more information about the prelinker, see *LE/VSE Programming Guide*. For more information about the other C utilities, see *LE/VSE C Run-Time Programming Guide*.

Prelinker Messages

This section describes return codes and messages from the prelinker.

Return Codes

The prelinker can issue the following return codes, indicating the degree of prelinking success:

Table 40. Return Codes from Prelinker

Return Code	Meaning
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.

Messages

The message numbers in this section contain a suffix indicating the message severity:

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

The C prelinker can issue the following messages:

EDC4000S Unable to open ***filename*****.**

Explanation: An error was encountered during a file open.

Programmer Response: Make sure that the file has the proper attributes (that is, RECFM, BLKSIZE, LRECL).

System Action: Processing terminates.

EDC4001S Unable to read ***filename*****.**

Explanation: An error was encountered during a file read.

Programmer Response: Make sure that the file has the proper attributes (that is, RECFM, BLKSIZE, LRECL). If the file has been corrupted, recreate it.

System Action: Processing terminates.

EDC4002S Unable to write to ***filename*****.**

Explanation: An error was encountered during a file write.

Programmer Response: Make sure that the file has the proper attributes (that is, RECFM, BLKSIZE, LRECL). Also ensure that sufficient disk space is available.

System Action: Processing terminates.

EDC4004W Invalid options: *option-string*

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 statements.

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 statements).

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 an 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 resolve 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 *object-name* 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.

EDC4019S Invalid or missing XSD statements.

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 statement missing for *statement-type* control statement.

Explanation: A control statement of *statement-type* was encountered with the continuation column set, but there was no next statement or the next statement was not a valid continuation statement.

Programmer Response: Add the appropriate continuation statement or set continuation column 72 to blank if no continuation statement is required.

System Action: The statement is ignored and processing continues.

EDC4021W Invalid syntax specified on *statement-type* control statement.

Explanation: A control statement with invalid syntax was encountered during processing.

Programmer Response: If the statement is required, correct the syntax errors and repeat the step. If the statement is not required, the warning message can be removed by deleting the invalid statement.

System Action: If the statement was an INCLUDE statement, the statement is processed up to the syntax error and the remainder of the statement is ignored. If the statement was not an INCLUDE statement, the statement is ignored. In either case, processing continues.

EDC4022W More than one *statement-type* statement found in *filename*.

Explanation: More than one control statement of *statement-type* was encountered during the processing of *filename*.

Programmer Response: No recovery is necessary unless the incorrect statement was chosen or incorrect processing was performed. In this case, remove the incorrect statement and repeat the step.

System Action: The statement is ignored and processing continues.

EDC4023W Continuation statements not allowed for *statement-type* statement. Statement ignored.

Explanation: A control statement of type *statement-type* was found to be expecting a continuation statement. Information for a statement of this type must be specified on one statement.

Programmer Response: Correct the statement if necessary, set continuation column 72 to blank, and repeat the step.

System Action: The statement is ignored and processing continues.

EDC4024W RENAME statement cannot be used for short name *name*.

Explanation: A RENAME statement was encountered that attempted to rename a short name to another name. RENAME statements 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 statement.

System Action: The statement is ignored and processing continues.

EDC4025W Multiple RENAME statements found for *name*. First valid one taken.

Explanation: More than one RENAME statement was encountered for *name*.

Programmer Response: The prelinker map shows which output name was chosen. If this was not the intended name, remove the duplicate RENAME statement(s) and repeat the step.

System Action: The first RENAME statement with a valid output name is chosen.

EDC4026W May not RENAME long name *from-name* to another long name *to-name*.

Explanation: A RENAME statement 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 statement with a valid output name and repeat the step. To remove the warning message, delete the invalid RENAME statement.

System Action: The statement is ignored and processing continues.

EDC4027W May not RENAME defined long name *from-name* to defined name *to-name*.

Explanation: A RENAME statement had been encountered that attempted to rename the defined long name *from-name* to another defined name *to-name*

Programmer Response: The prelinker map shows which output name was chosen. If this was not the intended name, replace the invalid RENAME statement with a valid output name and repeat the step. To remove the warning message, delete the invalid RENAME statement.

System Action: The statement is ignored and processing continues.

EDC4028W RENAME statement of *from-name* to *to-name* ignored since *to-name* is target of another RENAME.

Explanation: Multiple RENAME statements had been encountered attempting to rename two different names the same name *to-name*.

Programmer Response: The prelinker map shows which name was renamed to *to-name*. If the output name was not the intended name, change the name and repeat the step. To remove the warning message, delete the extra RENAME statement(s).

System Action: The first valid RENAME statement for *to-name* is chosen.

EDC4029W *from-name-1* and *from-name-2* mapped to same name (*to-name* due to UPCASE option.

Explanation: A name (*from-name-1*) that was made uppercase because of the UPCASE option collided with the output name (*to-name*) of another name (*from-name-2*).

Programmer Response: If both names (*from-name-1* and *from-name-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 statement to rename one of the names to something other than *to-name*.
- 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 (*from-name-1* and *from-name-2*) are mapped to *to-name*

EDC4031W File *filename* 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.

EDC4036E Invalid command operands: *operands*.

Explanation: Invalid operands was specified on the invocation of this command.

Programmer Response: Specify the correct operands and repeat the step.

System Action: Processing terminates.

EDC4037E File *filename* had invalid format.

Explanation: The specified file, *filename*, 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.

EDC4039E Library *library-name* had invalid format.

Explanation: The specified library, *library-name*, 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.

EDC4040E *message-text*.

Explanation: General error message.

EDC4041I Internal error.

Explanation: An error had occurred while prelinking. See *IBM C for VSE/ESA Diagnosis Guide* for a description of what to do. Contact your IBM service representative.

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.

EDC4053S WRONG MESSAGE FILE

Explanation: (This message is displayed in uppercase only.) The file opened is not a valid LE/VSE Version 1 Release 4 prelinker message file.

Programmer Response: Check that the message file in the sublibrary chain is in the correct format.

System Action: Processing terminates.

EDC4054W INCLUDE statement format is not supported.

Explanation: An INCLUDE statement of the format `INCLUDE` *`module-name`*`,(`*`name-list`*`)` was encountered. This format is not supported by the prelinker.

Programmer Response: Correct the statement if necessary.

System Action: The statement is ignored by the prelinker and left in SYSLNK for processing by the linkage editor.

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, these names cannot be renamed by the prelinker. The duplicate names might also have been mapped for other reasons (such as a CSECT compiler option, or a #pragma csect statement).

Programmer Response: To avoid the conflict, modify the duplicate names listed in the message.

System Action: The duplicate objects are not renamed and prelinker processing continues. To ensure that correct execution occurs, you must define the objects consistently.

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, these names cannot be renamed by the prelinker. The duplicate names might also have 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 have been identified, modify the duplicate names to avoid the conflict.

System Action: The duplicate objects are not renamed and prelinker processing continues. To ensure that correct execution occurs, you must define the objects consistently.

`localedef` Utility Messages

This section describes return codes and messages from the `localedef` utility.

Return Codes

The `localedef` utility can issue the following return codes:

Table 41. Return Codes from the `localedef` Utility

Return Code	Meaning
0	No errors were detected and the locale(s) were generated successfully .
4	Warning messages were issued and the locale(s) 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 message numbers in this section are followed by numbers indicating message severity:

10	Warning message
30	Error message
40	Severe error

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 `localedef` utility can issue the following messages:

EDC4100 40 The symbolic name '*symbolic-name*' was not the correct type.

Explanation: This message is issued in the locale definition file when using a symbolic name that was not the expected type. The most common time this error occurs is 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 '*filename*' for read.

Explanation: This message is issued if the open for read failed for any file required by the `localedef` utility. The file name passed to `fopen()` is included in the message. The locale is not created.

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 *filename* on line *line-number*.

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 *number-expected* **arguments and received** *number-received* **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 *line-number* **in module** *module-name***.**

Explanation: The `localedef` utility was unable to allocate memory.

Programmer Response: Ensure that the `localedef` utility has sufficient storage to run. Also, ensure that the correct libraries or sublibraries are specified on the LIBDEF PHASE,SEARCH or LIBDEF *,SEARCH JCL statement. Rerun the `localedef` utility.

System Action: The locale has not been created.

EDC4106 40 Could not open file '*filename***' 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: Look in the __amrc structure for further details regarding the error. See *LE/VSE C Run-Time Programming Guide* for more information on the __amrc structure. Or, check the SYSLOG for an error message.

System Action: The locale has not been created.

EDC4107 40 The '*symbol-name***' symbol was longer than <mb_cur_max>.**

Explanation: The length of value assigned to the specified symbol in the charmap file must not be as big as the value assigned to <mb_cur_max>. <mb_cur_max> defaults to 1 and can only have the values 1 or 4. 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 '*symbolic-name***' 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 '*symbolic-name***' 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, '*start-codepoint***', must be numerically less than the end of the range, '***end-codepoint***'.**

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 *symbolic-name-1* **and** *symbolic-name-2* **was incorrectly formatted.**

Explanation: The symbolic names used 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 '*string*'.

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 '*symbolic-name*', 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 '*decimal-value*'.

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 '*octal-value*'.

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 '*hex-value*'.

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 '*string*'.

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, '*character*', 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 '*statement*' statement is missing.

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 ′*character*′ is not a POSIX Portable Character.

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 ′*character*′ is missing the closing ′>′.

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, ′*string*′.

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 ′*multibyte character*′ 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, ′*character*′, had already been assigned a weight.

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 ′*low-character*...*high-character*′ already had a collation weight.

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 is 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, '*symbolic-name*', 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 *filename*, on line *line-number*, at character *character-position*.

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 following for more information.

EDC4131 10 Warning in file *filename*, on line *line-number*, at character *character-position*.

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 following for more information.

EDC4132 30 Syntax error in file *filename*, on line *line-number*, at character *character-position*.

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 had defined it as *number*.

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, '*character*' 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* 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 sort rules 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 *number-of-arguments*.

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 *category-name* 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 *category-name* 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 *category-name* 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 *high-digit* was not one greater than the file code for *low-digit*.

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 *high-digit* is not one greater than the process code for *low-digit*.

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 *symbol* has already been defined.

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 '*keyword*' 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 ′*keyword*′ keyword. Setting to POSIX defined defaults.

Explanation: The specified keyword in the LC_CTYPE category in the locale definition file did not contain minimum characters required 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 is 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 is 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 ′*symbol-1*′ and ′*symbol-2*′ 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 BLDERR option is specified, the defined ranges will not be used.

EDC4151 10 Option *option* 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 *option* option.

Explanation: The option specified had a sub option beginning with a left parenthesis but no right parenthesis was present.

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 Symbol name *symbol-name* is undefined and was required 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 '*category-name*' 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 '*character*' 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 '*character*' 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 '*symbolic-name*' character could not be found in the charmap file. Assigned code page IBM-1047 symbol '*codepoint*'.

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 '*character-1*' and '*character-2*' 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 created.

EDC4160 10 The '*keyword*' 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 *number*. 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 string 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 '*string*' 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 '*string*' is invalid and had been truncated to '*truncated-string*'.

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 character following the −1 were ignored and the locale has been created.

EDC4166 30 The value '*number*' for '*keyword*' is invalid.

Explanation: The *number* specified is not a valid 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 assigns 0 to value *number*.

EDC4167 30 '*keyword-1*' specified with no '*keyword-2*'.

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 created.

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 created.

EDC4170 40 Write failed while writing to file *filename*.

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 Unable to fetch messages file EDCGM*nnn*.

Explanation: The `localedef` utility could not fetch the message file EDCGM*nnn*.

nnn	**NATLANG (National Language)**
UEN	Uppercase U.S. English
ENU	Mixed-case U.S. English
JPN	Japanese

Programmer Response: Ensure that the `localedef` utility has sufficient storage to run. Also, ensure that the correct libraries or sublibraries are specified on the LIBDEF PHASE,SEARCH or LIBDEF *,SEARCH JCL statement.

System Action: The locale has not been created.

`iconv` Utility Messages

This section describes return codes and messages from the `iconv` utility.

Return Codes

The `iconv` utility can issue the following return codes:

Table 42. Return Codes from the `iconv` Utility

Return Code	Meaning
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 message numbers in this section are followed by numbers indicating message severity:

10	Warning message
30	Error message

The `iconv` utility can issue the following messages:

EDC4151 10 Option *option* 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 *option* 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 *from-codeset* to *to-codeset*.

Explanation: The `iconv` utility could not locate the conversion from *from-codeset* to *to-codeset*.

Programmer Response: Check the *from-codeset* and *to-codeset* 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 EDCIM*nnn*

Explanation: The `iconv` utility could not fetch the message file EDCI*nnn*.

nnn	**NATLANG (National Language)**
UEN	Uppercase U.S. English
ENU	Mixed-case U.S. English
JPN	Japanese

Programmer Response: Ensure that the `iconv` utility has sufficient storage to run. Also, ensure that the correct libraries or sublibraries are specified on the LIBDEF PHASE,SEARCH or LIBDEF *,SEARCH JCL statement.

System Action: The file has not been converted.

genxlt Utility Messages

The message numbers in this section are followed by numbers indicating message severity:

10	Warning message
30	Error message

The genxlt utility can issue the following messages:

EDC4151 10 Option *option* 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.

EDC4152 10 No matching right parenthesis for option *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.

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 *index*.

Explanation: The character *index* 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 *line-number*.

Explanation: The line *line-number* 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 EDCGM*nnn*.

Explanation: The genxlt utility could not fetch the message file EDCGM*nnn*.

nnn	**NATLANG (National Language)**
UEN	Uppercase U.S. English
ENU	Mixed-case U.S. English
JPN	Japanese

Programmer Response: Ensure that the genxlt utility has sufficient storage to run. Also, ensure that the correct libraries or sublibraries are specified on the LIBDEF PHASE,SEARCH or LIBDEF *,SEARCH JCL statement.

System Action: The file has not been converted.

`uconvdef()` Utility Messages

This section contains messages that are used with the LE/VSE C language `uconvdef()` utility. These messages are in the range EDC4201 to EDC4237, and are followed by numbers indicating the message severity:

00	Informational message
10	Warning message
30	Error message

The `uconvdef()` utility can issue the following messages:

EDC4201 00 Start processing the source file ***filename*****.**

Explanation: Processing of the specified input file has started.

Programmer Response: None.

System Action: Processing continues.

EDC4202 00 ***Filename*** **created.**

Explanation: `uconvdef` completed processing, and the output is in the file specified.

Programmer Response: None.

System Action: None.

EDC4203 30 Output file ***filename*** **was not created.**

Explanation: Output for `uconvdef` was not created. An accompanying message provides further information about why processing did not complete.

Programmer Response: Use the information provided by the accompanying message to determine and fix the source of the error, or contact your system programmer.

System Action: `uconvdef` terminates with no output.

EDC4204 30 Not enough space for allocation.

Explanation: The `uconvdef()` function issued a system call to obtain working storage in which to build output. The call failed and storage was not obtained.

Programmer Response: Ensure that the `uconvdef()` utility has sufficient storage to run.

System Action: `uconvdef()` terminates with no output.

EDC4205 30 Cannot open input file ***filename*****:** ***error-message*****.**

Explanation: The fopen() function for the named file failed. The associated error message is included if the `VERBOSE` option has been specified.

Programmer Response: Verify that the file name is correct and that the file/library member exists.

System Action: `uconvdef()` terminates with no output.

EDC4206 30 Cannot open temporary file: ***error-message*****.**

Explanation: The fopen() function for a memory work file failed. The associated error message is included if the `VERBOSE` option has been specified.

Programmer Response: Ensure that the `uconvdef()` utility has sufficient storage to run.

System Action: `uconvdef()` terminates with no output.

EDC4207 30 Cannot close input file ***filename*****:** ***error-message*****.**

Explanation: The fclose() function for the named file failed. The associated error message is included if the `VERBOSE` option has been specified.

Programmer Response: Use the error message to determine the cause of the failure.

System Action: `uconvdef()` terminates with no output.

EDC4208 30 Cannot close temporary file: ***error-message*****.**

Explanation: The fclose() function for a memory work file failed. The associated error message is included if the `VERBOSE` option has been specified.

Programmer Response: Use the error message to determine the cause of the failure.

System Action: `uconvdef()` terminates with no output.

EDC4210 30 Error while reading input file ***filename*****:** ***error-message*****.**

Explanation: The fgets() function for the named file failed. The associated error message is included if the `VERBOSE` option has been specified.

Programmer Response: Make sure that the file has the proper attributes (that is, RECFM, BLKSIZE, and LRECL). If the file has been corrupted, recreate it.

System Action: `uconvdef()` terminates with no output.

EDC4211 30 Error while writing temporary file: *error-message*.

Explanation: The fwrite() function for a memory work file failed. The associated error message is included if the `VERBOSE` option has been specified.

Programmer Response: Ensure that the `uconvdef()` utility has sufficient storage to run.

System Action: `uconvdef()` terminates with no output.

EDC4212 30 MB_CUR_MIN is greater than MB_CUR_MAX default.

Explanation: The value of MB_CUR_MIN cannot exceed the value of MB_CUR_MAX. MB_CUR_MAX was not specified. The default value of MB_CUR_MAX is 1.

Programmer Response: Specify a value of MB_CUR_MIN that is less than MB_CUR_MAX in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4213 30 Line *line-number*: space at the beginning of a line is not allowed.

Explanation: No line in the input file may begin with a space. The message identifies the number of the line that begins with a space.

Programmer Response: Remove the space in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4214 30 Line *line-number*: invalid line format.

Explanation: The number of the line with the invalid format is identified in the message.

Programmer Response: Correct the line format in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4215 30 Line *line-number*: invalid token.

Explanation: The number of the line with the invalid token is identified in the message.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4216 30 Line *line-number*: invalid value for token *string*.

Explanation: The number of the line with the invalid token value is identified in the message.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4217 30 Line *line-number*: token *string* must have a value.

Explanation: A token must be assigned a value. The message identifies the number of the line and token without a value.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4218 30 Line *line-number*: code set name can be defined only once.

Explanation: A code set name can be defined only once. The message identifies the number of the line that specifies the name of the code set being redefined.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4219 30 Line *line-number*: UCONV_CLASS can be defined only once.

Explanation: UCONV_CLASS can be defined only once. The message identifies the number of the line that attempts to define UCONV_CLASS again.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4220 30 Line *line-number*: CHARMAP section is already started.

Explanation: An input file can contain only one CHARMAP section. The message identifies the number of the line that attempts to begin a second section.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4221 30 Line *line-number*: no token found.

Explanation: A token could not be found on the input line identified in the message.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4222 30 Line *line-number*: **conflict with UCONV_CLASS.**

Explanation: There is a conflict with the specified UCONV_CLASS. The length of the code point on the line identified in the message is greater or smaller than the length allowed by the UCONV_CLASS.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4223 30 Line *line-number*: **conflict with MB_CUR_MIN or MB_CUR_MAX.**

Explanation: The length of a code point cannot be less than MB_CUR_MIN or greater than MB_CUR_MAX. The error was detected in the line identified in the message.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4224 30 Line *line-number*: **conflicting code length.**

Explanation: An invalid code point length was detected while processing a multibyte character set. The error was detected in the line identified in the message.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4225 30 Line *line-number*: **range exceeds limit of one byte.**

Explanation: The range on the line identified in the message exceeds the allowable limit of one byte.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4226 30 Line *line-number*: **invalid code point.**

Explanation: The value of the code point on the line identified in the message is greater than allowed.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4227 30 Line *line-number*: **escape and comment characters must be distinct.**

Explanation: The escape and comment characters must be distinct. The error was detected in the line identified in the message.

Programmer Response: Correct the line in the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4228 30 No code set name is defined.

Explanation: The input file must contain a statement that identifies the code set.

Programmer Response: Add a code set name statement to the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4229 30 No UCONV_CLASS is defined.

Explanation: The input file must contain a statement that identifies the UCONV_CLASS.

Programmer Response: Add a UCONV_CLASS statement to the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4230 30 CHARMAP section must start with CHARMAP.

Explanation: The CHARMAP section of the input file must start with a CHARMAP statement.

Programmer Response: Add a CHARMAP statement to the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4231 30 CHARMAP section has no END CHARMAP.

Explanation: The input file must contain an END CHARMAP statement.

Programmer Response: Add an END CHARMAP statement to the input file and run `uconvdef()` again with the changed input.

System Action: `uconvdef()` terminates with no output.

EDC4232 30 Cannot open output file *filename*: *error-message*.

Explanation: The fopen() function for the named file failed. The associated error message is included if the `VERBOSE` option has been specified.

Programmer Response: Use the error message to determine the cause of the failure.

System Action: `uconvdef()` terminates with no output.

EDC4233 30 Cannot close output file ***filename*****:** *error-message***.**

Explanation: The fclose() function for the named file failed. The associated error message is included if the `VERBOSE` option has been specified.

Programmer Response: Use the error message to determine the cause of the failure.

System Action: `uconvdef()` terminates with incomplete output.

EDC4234 30 Cannot reopen temporary file: *error-message***.**

Explanation: The freopen() function for a memory work file failed. The associated error message is included if the `VERBOSE` option has been specified.

Programmer Response: Use the error message to determine the cause of the failure.

System Action: `uconvdef()` terminates with no output.

EDC4236 30 Error while writing output file ***filename*****:** *error-message***.**

Explanation: The fwrite() function for the named file failed. The associated error message is included if the `VERBOSE` option has been specified.

Programmer Response: Use the error message to determine the cause of the failure.

System Action: `uconvdef()` terminates with incomplete output.

EDC4237 30 Unable to fetch message file EDCUM*nnn*.

Explanation: The `uconvdef()` utility could not fetch the message file EDCUM*nnn*, where *nnn* is either UEN (uppercase U.S. English) or ENU (mixed case U.S. English), depending on the setting of `NATLANG` run-time option.

Programmer Response: Ensure that the `uconvdef()` utility has sufficient storage to run and that the correct libraries or sub-libraries are specified on the LIBDEF PHASE,SEARCH or LIBDEF *,SEARCH JCL statement.

System Action: `uconvdef()` terminates with no output.

DSECT Utility Messages

This section describes return codes and messages from the DSECT utility.

Return Codes

The DSECT utility can issue the following return codes:

Table 43. Return Codes from the DSECT utility

Return Code	Meaning
0	Successful completion.
4	Successful completion, warnings issued.
8	DSECT utility failed, error messages issued.
12	DSECT utility failed, severe error messages issued.
16	DSECT utility failed, insufficient storage to continue processing.

Messages

The message numbers in this section are followed by numbers indicating message severity:

00	Informational message
10	Warning message
30	Error message
40	Severe error

The DSECT utility can issue the following messages:

EDC5500 10 Option *option* **is not valid and is ignored.**

Explanation: The option specified in the message is not valid DSECT utility option or a valid option has been specified with an invalid value. The specified option is ignored.

User Response: Rerun the DSECT utility with the correct option.

EDC5501 30 No DSECT or CSECT names were found in the SYSADATA file.

Explanation: The SECT option was not specified or SECT(ALL) was specified. The SYSADATA was searched for all DSECTs and CSECTs but no DSECTs or CSECTs were found.

User Response: Rerun the DSECT utility with a SYSADATA file that contains the required DSECT or CSECT definition.

EDC5502 30 Sub option *sub-option* **for option** *option* **is too long**

Explanation: The sub option specified for the option was too long and is ignored.

EDC5503 30 Section name *section* **was not found in SYSADATA File.**

Explanation: The section name specified with the SECT option was not found in the External Symbol records in the SYSADATA file. The C structure is not produced.

User Response: Rerun the DSECT utility with a SYSADATA file that contains the required DSECT or CSECT definition.

EDC5504 30 Section name *section* **is not a DSECT or CSECT.**

Explanation: The section name specified with the SECT option is not a DSECT or CSECT. Only a DSECT or CSECT names may be specified. The C structure is not produced.

EDC5505 00 No fields were found for section *section***, structure is not produced.**

Explanation: No field records were found in the SYSADATA file that matched the ESDID of the specified section name. The C structure is not produced.

EDC5506 30 Record length for file *"filename"* **is too small for the SEQUENCE option, option ignored**

Explanation: The record length for the output file specified is too small to enable the SEQUENCE option to generate the sequence number in columns 73 to 80. The available record length must be greater than or equal to 80 characters. The SEQUENCE option is ignored.

EDC5507 40 Insufficient storage to continue processing.

Explanation: No further storage was available to continue processing.

User Response: Rerun the DSECT utility with a larger partition GETVIS size.

EDC5508 30 Open failed for file *"filename"***:** *error-message*

Explanation: This message is issued if the open fails for any file required by the DSECT utility. The file name passed to `fopen()` and the error message returned by `strerror(errno)` is included in the message.

User Response: The message text indicates the cause of the error. If the file name was specified incorrectly on the OUTPUT option, rerun the DSECT utility with the correct file name.

EDC5509 40 *function-name* **failed for file** *"filename"***:** *error-message*

Explanation: This message is issued if any error occurs reading, writing or positioning on any file by the DSECT utility. The name of the function that failed (Read, Write, fgetpos, fsetpos), file name and text from strerror(errno) is included in the message.

User Response: This message may be issued if an error occurs reading or writing to a file. This may be caused by an error within the file, such as an I/O error or insufficient disk space. Correct the error and rerun the DSECT utility.

EDC5510 40 Internal Logic error in function *function-name*

Explanation: The DSECT utility has detected that an error has occurred while generating the C structure. Processing is terminated and the C structure is not produced.

User Response: This may be caused by an error in the DSECT utility or by incorrect input in the SYSADATA file. Contact your systems administrator.

EDC5511 10 No matching right parenthesis for *option* **option.**

Explanation: The option specified had a sub option beginning with a left parenthesis but no right parenthesis was present.

User Response: Rerun the DSECT utility with the parenthesis for the option correctly paired.

EDC5512 10 No matching quote for *option* **option.**

Explanation: The OUTPUT option has a sub option beginning with a single quote but no matching quote was found.

User Response: Rerun the DSECT utility with the quotes for the option correctly paired.

EDC5513 10 Record length too small for file *"filename"***.**

Explanation: The record length for the Output file specified is less than 10 characters in length. The minimum available record length must be at least 10 characters.

User Response: Rerun the DSECT utility with an output file with a available record length of at least 10 characters.

EDC5514 30 Too many sub options were specified for option *option***.**

Explanation: More than the maximum number of sub options were specified for the particular option. The extra sub options are ignored.

EDC5515 00 HDRSKIP option value greater than length for section *section***, structure is not produced.**

Explanation: The value specified for the HDRSKIP option was greater than the length of the section. A structure was not produced for the specified section.

User Response: Rerun the DSECT utility with a smaller value for the HDRSKIP option.

EDC5516 10 SECT and OPTFILE options are mutually exclusive, OPTFILE option is ignored

Explanation: Both the SECT and OPTFILE options were specified, but the options are mutually exclusive.

User Response: Rerun the DSECT utility with either the SECT or OPTFILE option.

EDC5517 10 Line *line-number* from "*filename*" does not begin with SECT option

Explanation: The line from the file specified on the OPTFILE option did not begin with the SECT option. The line was ignored.

User Response: Rerun the DSECT utility without OPTFILE option, or correct the line in the input file.

EDC5518 10 setlocale() failed for locale name "*locale-name*".

Explanation: The `setlocale()` function failed with the locale name specified on the LOCALE option. The LOCALE option was ignored.

User Response: Rerun the DSECT utility without LOCALE option, or correct the locale name specified with the LOCALE option.

EDC5519 10 Long names were detected and truncated. Check output.

Explanation: The DSECT utility detected at least one name whose length exceeds the maximum allowed. It has therefore truncated the name, and appended "..." to the end of the name to signify the condition. If the input name is within limits and the UNIQUE option is specified, the mapping of national characters in the input name might have extended the name length beyond the maximum allowed.

User Response: Check the DSECT utility output. Long names are truncated, which is indicated by "..." at the end of the name. If applicable, modify the UNIQUE option field. Or, modify the input name so that it does not exceed the maximum length when expanded.

EDC5520 40 Architecture Level *i* of SYSADATA is not supported. The latest supported level is *d*.

Explanation: The SYSADATA file has probably been produced by a recent HLASM release which is not yet supported by the DSECT utility.

User Response: Contact your IBM representative.

EDC5521 40 Architecture Level *i* of SYSADATA is no longer supported. The earliest supported level is *d*.

Explanation: The SYSADATA file has probably been produced by an obsolete HLASM release.

User Response: Use a supported HLASM release to produce the SYSADATA file.

EDC5522 10 Edition *d* of record X'04*x*' is not supported. Edition *d* is assumed.

Explanation: The likely reason is that HLASM maintenance has introduced an updated layout of this SYSADATA record type. This should not cause a problem unless the offsets of fixed fields processed by the DSECT utility have changed. The message can be ignored unless the produced output is incorrect.

User Response: If the DSECT utility is producing incorrect output, then please contact your IBM representative.

EDC5599 30 Unable to fetch messages file EDDN*nnn*.

Explanation: The DSECT utility could not fetch the message file EDDN*nnn*.

nnn	**NATLANG (National Language)**
UEN	Uppercase U.S. English
ENU	Mixed-case U.S. English
JPN	Japanese

User Response: Ensure that the DSECT utility has sufficient storage to run. Also, ensure that the correct libraries or sublibraries are specified on the LIBDEF PHASE,SEARCH or LIBDEF *,SEARCH JCL statement.

System Action: The conversion table has not been built.

Chapter 10. C Run-Time Messages

The following run-time messages pertain to C. 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 message numbers in this section contain a suffix indicating the message severity:

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: This error occurred because an input argument to one of the math functions was outside the domain over which the mathematical function is defined.

Programmer Response: See *LE/VSE 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: This error occurred because the result of the math function could not be represented as a double value.

Programmer Response: See *LE/VSE C Run-Time Library Reference* for the math function that produced this error.

System Action: The math function fails.

Symbolic Feedback Code: EDC4SA

EDC5003I Truncation of a record occurred during an I/O operation.

Explanation: Truncation occurred because either (1) the specified record length on the write operation was larger than the record buffer size or (2) 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 is read/written up to the point where truncation occurred.

Symbolic Feedback Code: EDC4SB

EDC5004I The size of the specified record was not large enough.

Explanation: The record length was too small because: the specified record size for an `fwrite()` or `fupdate()` function was smaller that 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 cannot immediately follow a read operation.

Explanation: If the last operation on a non-VSAM file opened for record I/O was read, a write operation cannot 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 cannot immediately follow a write operation.

Explanation: If the last operation on a file was a write, a read operation cannot 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 with a larger partition GETVIS size.

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: 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 fails.

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 (e.g. `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 due to the fact that were already pushed back characters waiting to be read and the number of pushed back characters already equalled 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, then either move the data set or do not use the positioning functions.

System Action: The function fails.

Symbolic Feedback Code: EDC4SK

EDC5014I An attempt was made to acquire a position that is prior to 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 prior to 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 prior to the start of the file, unless you either read them or discard them via a reposition first.

System Action: `ftell()` or `fgetpos()` fail.

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. The maximum depends on the blksize. Smaller blksizes allows more records to be encoded. The maximum will be 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).

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: The `fread()`, `fwrite()`, and `fupdate()` functions are the only functions that can 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 irrecoverable error has marked the file permanently 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. Once 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: LE/VSE's attempt at obtaining memory in order to satisfy the current library request has failed.

Programmer Response: Ensure that the partition GETVIS 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 partition GETVIS size allocated to the application.

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

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 DLBL 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 DLBL so a valid combination is created. See *LE/VSE C Run-Time Programming Guide* for details on attribute merging and valid combinations.

System Action: The `fopen()/freopen()` function fails.

Symbolic Feedback Code: EDC4ST

EDC5022I An error occurred while generating a temporary name.

Explanation: The `tmpnam()` function was called in order to generate a temporary file name. However, a name could not be generated, which did not already exist, 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 which were created via the `tmpnam()` function, and retry 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 *LE/VSE C Run-Time Programming Guide* for more information on the __amrc structure.

System Action: The `ftell()/fgetpos()` function fails.

Symbolic Feedback Code: EDC4SV

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

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 previous to the start of the stream; or (3) the specified position lay 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, the file should be opened in binary mode if the user wishes 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 due to 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: 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 prior to 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 invoking the `system()` function 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 data set, 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 which is opened for write. Since 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 it appears to be the old member, now extended. For a concatenated data set, you cannot extend the concatenation.

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.

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 to the `setvbuf()` function was made 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

EDC5038I An error occurred when flushing console output, before the system retrieves console input.

Explanation: When a console read cannot get any data from the buffer and must perform a system console read, all console output data not yet flushed to the system must be output. While writing out the unflushed console output data, an error occurred.

Programmer Response: Change the code so that all console output is completed and flushed to the console before the console input operation. Check all return codes to find out if any output operation gets an error and then check the errno value for further information regarding any error encountered.

System Action: The console input operation fails.

Symbolic Feedback Code: EDC4TE

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 *LE/VSE C Run-Time Programming Guide* for more information on the __amrc structure. Or, check the SYSLOG 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 open, 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 and the name did not match an existing memory file. A name of this format is not allowed to be used to create a memory file. The name is generated internally by C when a user opens a memory file with a name of '*' for output. C 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 via `fldata()`, make sure it has not been closed and removed prior to the open using 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

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, but the file 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 prevents 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.

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, or is not a valid memory file name.

Programmer Response: Specify a valid file name according to the system or 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 unexpectedly failed.

Explanation: An internal call to an LE/VSE internal routine has failed but the failure is not anticipated and cannot be recovered from.

Programmer Response: Contact your IBM Support Center.

System Action: Current library function using an internal routine fails.

Symbolic Feedback Code: EDC4TO

EDC5049I The file name specified 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 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 file name specified exists.

System Action: The `fopen()`, `freopen()`, or `rename()` functions fails.

Symbolic Feedback Code: EDC4TP

EDC5051I An error occurred when renaming a file.

Explanation: A rename error have occured.

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. Check the `__amrc` for further details.

System Action: The `rename()` function fails.

Symbolic Feedback Code: EDC4TR

EDC5052I The application is running AMODE=24 while the run-time library was installed above the line.

Explanation: The application which is accessing the run-time library is running AMODE=24. But, the run-time library was installed above the 16M line which the application cannot address.

Programmer Response: Ensure the amode of the application matches that of the run-time library. If you must run AMODE=24, then the run-time library must be installed below the line. Otherwise, relink your application to be AMODE=31.

System Action: Application is terminated with 3000 abend.

Symbolic Feedback Code: EDC4TS

EDC5053I The language environment run-time library phase EDCZ24 could not be loaded.

Explanation: An error has occurred when LE/VSE tried to load the run-time library phase EDCZ24.

Programmer Response: Ensure that the partition GETVIS size is sufficient to run the application. Check the files in the LIBDEF PHASE search chain for the job to ensure EDCZ24 is available (for example, check PRD2.SCEEBASE, the default installation sublibrary).

System Action: The program terminates and a traceback or dump is issued, depending upon 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 will return failure, 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.

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 *LE/VSE C Run-Time Programming Guide* for more information on the __amrc structure.

System Action: The `flocate()` function fails. The `rewind()` function will 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 invoking `fsetpos()`, 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 *LE/VSE C Run-Time 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 for VSE, VSAM, etc.

Programmer Response: Check the __amrc structure for more information. See *LE/VSE C Run-Time Programming Guide* for 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 for VSE, VSAM, etc.

Programmer Response: Check the __amrc structure for more information. See *LE/VSE C Run-Time 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.

Programmer Response: Ensure that the file to be opened for read exists.

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 fails.

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()`. Only TMP_MAX calls are guaranteed to 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: Specifiy 'type=record' on the `fopen()/freopen()` statement.

System Action: The `fopen()/freopen()` function fails.

Symbolic Feedback Code: EDC4UI

EDC5076I An fread() was not performed prior to calling the fdelrec() or fupdate() functions.

Explanation: The `fdelrec()` and `fupdate()` functions cannot be invoked without previously 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 was not able to 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 *VSE/VSAM Commands and Macros* SC33-6532. __rc corresponds to the register 15 value, __fdbk corresponds to the Reason Code. See *LE/VSE C Run-Time Programming Guide* for more information on the __amrc structure.

System Action: The `fdelrec()` function fails.

Symbolic Feedback Code: EDC4UL

EDC5078I The requested operation is only valid for VSAM data sets.

Explanation: The `fdelrec()`, `flocate()` and `fupdate()` functions can only be invoked with VSAM data sets.

Programmer Response: Use the the `fseek()/ftell()` or `fgetpos()/fsetpos()` functions for positioning within a non-VSAM file. For updating, use `fread()/fwrite()` or the byte I/O functions instead of `fupdate()`.

System Action: The `fdelrec()`, `flocate()` and `fupdate()` functions fails.

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 fail.

Symbolic Feedback Code: EDC4UN

EDC5080I An invalid option was passed to the flocate() function.

Explanation: The parameter that specifies the position options 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`.

System Action: The `flocate()` function fails.

Symbolic Feedback Code: EDC4UO

EDC5083I An error occurred attempting to load a phase into storage.

Explanation: The library has attempted to dynamically load a phase 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 partition size. Check the job log for messages which will help to pinpoint the name of the phase.

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

EDC5087I The file characteristics specified 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) reading a display or printer; (2) writing to a character reader; (3) updating a magnetic tape device; (4) opening SYSIPT or SYSLST for 'append' or 'update'; (5) opening SYSIPT for anything besides 'read'; or (6) opening SYSLST '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 re-directed SYSLST or SYSPCH file.

Explanation: A variable or undefined record format was specified.

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 see the *LE/VSE C Run-Time 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 *LE/VSE C Run-Time 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 will fail.

Symbolic Feedback Code: EDC4V4

EDC5094I An attempt was made to push back the EOF character using ungetc().

Explanation: The `ungetc()` function can 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

EDC5099I The function specified is not supported under CICS.

Explanation: The function specified is not supported under CICS.

Programmer Response: See *LE/VSE C Run-Time Programming Guide* for more information on running under C 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: Under CICS, 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 the first write (and all subsequent writes) to stdout/stderr will fail.

Symbolic Feedback Code: EDC4VC

EDC5101I The transient data queue was not enabled for the standard streams.

Explanation: When running under CICS and an attempt has been made to write to stdout or stderr, 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 (via 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

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.

Programmer Response: See *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC4VP

EDC5117I File exists.

Explanation: An existing file was specified in an inappropriate manner.

Programmer Response: See *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC4VT

EDC5118I Bad address.

Explanation: The system detected an invalid address in attempting to use an argument of a call. Note that not all functions will detect this error.

Programmer Response: See *LE/VSE 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. Application execution continues.

Symbolic Feedback Code: EDC4VU

EDC5119I File too large.

Explanation: The size of a file would exceed the maximum allowed.

Programmer Response: Ensure that enough space is available in the file.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC4VV

EDC5121I Invalid argument.

Explanation: An argument supplied was invalid.

Programmer Response: See *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC501

EDC5122I Input/output error.

Explanation: An input or output error occurred.

Programmer Response: See *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC502

EDC5127I Too many open files in system.

Explanation: The system reached its predefined limit for simultaneously open files.

Programmer Response: The request may succeed at a later time if fewer files are in use.

System Action: The request has failed. Application execution continues.

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 write-only device.

Programmer Response: See *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC508

EDC5131I No locks available.

Explanation: No file and/or record locks are available, the system limit has been reached.

Programmer Response: See *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC50B

EDC5132I Not enough memory.

Explanation: There is not enough memory space available to create the new object.

Programmer Response: See *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC50C

EDC5134I Function not implemented.

Explanation: The function to be executed has not been implemented.

Programmer Response: The function cannot be used by the application program.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC50E

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.

Programmer Response: See *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

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, when a tape drive is not online.

Programmer Response: Refer to the *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC50I

EDC5145I Output buffer too small to contain all converted characters.

Explanation: The outbut buffer specified for the function is too small to contain all of the converted characters produced. Refer to the *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC50P

EDC5147I Illegal byte sequence.

Explanation: The string contains an illegal sequence of bytes. For example, an unmatched shift out / shift in condition exists.

Programmer Response: See *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC50R

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.

Programmer Response: See *LE/VSE C Run-Time Library Reference* for the function being attempted, for the specific reason for failure.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC518

EDC5165I System TOD clock not set.

Explanation: The system time of day (TOD) clock is in error, stopped, or in a nonoperational state.

Programmer Response: Report this problem to your system programmer.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC51D

EDC5168I Password has expired.

Explanation: The verification request has failed because the password has expired.

Programmer Response: Change the password.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC51G

EDC5169I Password is invalid.

Explanation: The verification or change password request has failed because the supplied password is invalid.

Programmer Response: Correct the supplied password, and retry the request.

System Action: The request has failed. Application execution continues.

Symbolic Feedback Code: EDC51H

EDC5180E Permanent I/O error on file *filename*.

Explanation: May be caused by a program attempting to process a file for which the program was not designed.

Programmer Response: Check the system log at the time of error for messages. Ensure the file being used matches the program specifications.

System Action: The current I/O operation fails.

Symbolic Feedback Code: EDC51S

EDC5181E Wrong length record on file *filename*.

Explanation: Specified record length does not match actual record length.

Programmer Response: Ensure the file being used matches the program specifications.

System Action: The current I/O operation fails.

Symbolic Feedback Code: EDC51T

EDC5182E No more matching/available extents for file *filename*.

Explanation: Possible explanations:
- There may not be sufficient VSAM data space for a VSAM-managed SAM output file.
- The EXTENT information for a SAM file may not have sufficient space for the program data being written.
- The program may be in a loop outputting data to a file.

Programmer Response: Check the EXTENT option of the file's DLBL statement.

For VSAM-managed SAM files, use the LISTCAT command to check VSAM data space availability. For information on using LISTCAT, see *IBM VSE/VSAM Commands and Macros* , SC33-6532.

System Action: The I/O write operation fails.

Symbolic Feedback Code: EDC51U

EDC5183E Function not supported under VSE.

Explanation: A C function has been called that is not supported by the LE/VSE C run-time.

Programmer Response: Check the program for function calls that are not listed in *LE/VSE C Run-Time Library Reference*

System Action: The function call fails.

Symbolic Feedback Code: EDC51V

EDC5184E Specified DLBL name not found.

Explanation: No matching DLBL can be found for the file-ID, DLBL name, or logical unit specified in the *filename* parameter of the `fopen()` or `freopen()` function. If the program is attempting to open SYSLNK, there might not be a DLBL for IJSYSLN.

Programmer Response: Check that the item specified by the *filename* parameter has a matching DLBL. For details on specifying `fopen()` and `freopen()` parameters, see *LE/VSE C Run-Time Programming Guide*.

System Action: The `fopen()/freopen()` function fails.

Symbolic Feedback Code: EDC520

EDC5185E Specified System Logical Unit not assigned.

Explanation: Possible explanations:
- The `remove()` function may be attempting to remove a SAM file where the DLBL/EXTENT data does not supply a validly assigned logical unit.
- The `fopen()` function may be attempting to open a file using the format:
  - `DD:SYS`*`xxx`* where the logical unit is not validly assigned.
  - `DD:`*`DLBL`* for a SAM file where the DLBL/EXTENT data does not supply a validly-assigned logical unit.
  - *`file-ID`* where the file-ID has been matched to a SAM DLBL/EXTENT which does not supply a validly-assigned logical unit.
  - *`file-ID`* where the file-ID has been matched to a TLBL JCL statement, however labeled tape files may only be opened using the `DD:SYS`*`xxx-TLBLname`* format.

Programmer Response: Check that the item specified by the *filename* parameter has a matching DLBL. Check the logical unit or file-ID specified by the DLBL or TLBL statement. For details on specifying `fopen()`, `freopen()`, and `remove()` parameters, see *LE/VSE C Run-Time Programming Guide*.

System Action: The `fopen()/freopen()` or `remove()` function fails.

Symbolic Feedback Code: EDC521

EDC5186E Specified System Logical Unit is assigned to an unsupported device.

Explanation: The assigned device type is not supported for C run-time files.

Programmer Response: Check that the logical unit specified by `fopen()` or `freopen()` is assigned to a supported device type. For a list of supported device types, see *LE/VSE C Run-Time Programming Guide*.

System Action: The `fopen()/freopen()` function fails.

Symbolic Feedback Code: EDC522

EDC5187E The System Logical Unit associated with the DLBL/TLBL for the file is not assigned to a disk/tape device.

Explanation: Attempting to open a file using DLBL/EXTENT where the logical unit is not assigned to a disk, or using a TLBL where the logical unit is not assigned to a tape device.

Programmer Response: Check the logical unit specified by `fopen()` or `freopen()`. Check the ASSIGN statement for the logical unit.

System Action: The `fopen()/freopen()` function fails.

Symbolic Feedback Code: EDC523

EDC5188E Error trying to allocate a Define The File (DTF).

Explanation: An error occurred when attempting to allocate a DTF to open a file. The console log should contain an associated message further defining the problem.

Programmer Response: Check the LE/VSE message file (default is SYSLOG) for a related message.

System Action: The `fopen()/freopen()` function fails.

Symbolic Feedback Code: EDC524

EDC5191E No blksize determined for DISK file *filename*.

Explanation: A disk file has not been opened because there was no `BLKSIZE=` provided or no `RECFM=` provided. If this information cannot be obtained from a VSAM catalog, it must be provided in the `fopen()` mode paramaters.

Programmer Response: Check the `fopen()` or `freopen()` BLKSIZE and RECFM mode parameters for this VSAM-managed SAM file.

System Action: The `fopen()/freopen()` function fails.

Symbolic Feedback Code: EDC527

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 an obsolete interface provided in stream.h for compatibility with old code.

Programmer Response: Split the call to `form()` into two or more calls.

System Action: Execution is aborted.

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 is aborted.

Symbolic Feedback Code: EDC538

EDC5500 - EDC5599

Explanation: These messages are produced by the DSECT utility. For details, see “DSECT Utility Messages” on page 240.

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 terminates and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000 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 terminates and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000 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 terminates and a traceback or dump is issued depending on the TERMTHDACT run-time option. A return code of 3000 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 terminates and a traceback or dump is issued depending on the TERMTHDACT run-time option. A return code of 3000 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 terminates and a traceback or dump is issued depending on the TERMTHDACT run-time option. A return code of 3000 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 terminates and a traceback or dump is issued depending on the TERMTHDACT run-time option. A return code of 3000 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 terminates and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 2000 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 terminates and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000 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 terminates and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000 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 terminates and a traceback or dump is issued, depending on the TERMTHDACT run-time option. A return code of 3000 is returned.

Symbolic Feedback Code: EDC5RP

TCP/IP-Related Messages

The EDCTCPIP module loads the TCP/IP vendor-provided phase $EDCTCPV in order to locate the TCP/IP callable functions. If the phase cannot be loaded or the called TCP/IP function cannot be located, an error message will be issued to `stderr` and the application abends with CEE3322C U4039.

In case the IBM provided dummy TCP/IP function body is active, a message is written to `stderr`. This may be an additional indicator if the application does not react to the return code or does not check for `errno` respective `h_errno`.

EDCT001S Unable to load phase $EDCTCPV.

Explanation: IBM provided dummy phase, or TCP/IP provided vendor phase holding the TCP/IP function bodies could not be loaded.

Programmer Response: Check if phase is available in a library contained in current LIBDEF chain, and check if sufficient space is available.

System Action: The application will be terminated.

EDCT002S *xxxxxxxxx* **implementation not found.**

Explanation: Phase $EDCTCPV does not contain the body of TCP/IP function *xxxxxxxxx*. This is a build error of the TCP/IP provider.

Programmer Response: Check with your TCP/IP provider. The phase was not properly built.

System Action: The application will be terminated.

EDCT003S Unsupported C Run-Time function called.

Explanation: An application program that has NOT been compiled and linked under VSE is running under LE/VSE. The application contains calls to C Run-Time functions that are not supported in the VSE environment.

Programmer Response: The application must be changed so that the unsupported C Run-Time functions are no longer called. If you recompile the application using the C for VSE/ESA compiler and check the compiler error messages, you might be able to determine the function(s) that are not supported.

System Action: The application will be terminated.

EDCV001I TCP/IP function *xxxxxxxxx* **not implemented.**

Explanation: Application has called a TCP/IP function, that is not implemented by the vendor-provided programming interface. The IBM-provided dummy function is still active, which just issues this message and returns to the caller. Application is getting an appropriate return code and/or global error variable `errno` respective `h_errno` is set.

Programmer Response:

1. Check that your LIBDEF statement refers to the vendor-provided version of the phase $EDCTCPV.
2. Avoid using this function with the TCP/IP version supplied by this vendor.

Operator Response: None.

EDCV002I Unexpected TCP/IP error code ***nnnn*****.**

Explanation: The TCP/IP product has returned the unexpected error code *nnnn*. The global error variable is set to EOPNOTSUPP.

Programmer Response: Please contact your service representative.

Operator Response: None.

EDCV099I Unable to issue EDCV001I to STDERR.

Explanation: Phase $EDCTCPV tried to write message EDCV001 to `stderr`. This is not successful, therefore message EDCV099I is written to SYSLOG. Application is getting an appropriate return code and/or global error variable `errno` respective `h_errno` is set.

Programmer Response: Check the standard stream `stderr` assignment. In case of further problems, check for the active phase $EDCTCPV, and contact the provider.

Operator Response: None.

Chapter 11. PL/I Run-Time Messages

The following messages 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.

For a description of the format of PL/I run-time messages, see "Interpreting Run-Time Messages" on page 25.

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 application is terminated.

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 application is terminated.

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 phase 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 phase, follow the instructions on link-editing fetchable phases in *LE/VSE Programming Guide*.

System Action: The application is terminated.

Symbolic Feedback Code: IBM006

IBM0018S Error during implicit close of file *file-name***.**

Explanation: An error occurred attempting to implicitly close a record I/O file during program termination. The file may be corrupted or unusable. The possible error conditions are:

- TRANSMIT (if last operation was a LOCATE, or if the file has the attributes REGIONAL, SEQUENTIAL, and OUTPUT).
- I/O Sequence error.
- KEY condition for VSAM.
- Out of Space error (there was insufficient space remaining in the data set for CLOSE to either flush unwritten data or write an EOF marker).

Programmer Response: Explicitly close the file (by coding a CLOSE statement) to obtain normal condition handling and run the program again. If an error still occurs during explicit close, take the recommended action for that error.

System Action: Program termination will continue.

Symbolic Feedback Code: IBM00I

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 application is terminated.

Symbolic Feedback Code: IBM00K

IBM0021S ONCODE= *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM00L

IBM0022S ONCODE= *oncode-value* **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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM00O

IBM0025S ONCODE= *oncode-value* **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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM00R

IBM0028S ONCODE= *oncode-value* **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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM00T

IBM0031S **ONCODE=** *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM00V

IBM0032S **ONCODE=** *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM010

IBM0033S **ONCODE=** *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM011

IBM0034S **ONCODE=** *oncode-value* **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 that 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 application is terminated.

Symbolic Feedback Code: IBM012

IBM0035S ONCODE= *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM013

IBM0036S ONCODE= *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM014

IBM0037S ONCODE= *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM015

IBM0038S ONCODE= *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM016

IBM0039S ONCODE= *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM017

IBM0040S ONCODE= *oncode-value* **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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM01A

IBM0043S ONCODE= *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM01B

IBM0045S ONCODE= *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM01D

IBM0046S ONCODE= *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM01E

IBM0047S ONCODE= *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM01F

IBM0048S **ONCODE=** *oncode-value* **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 application is terminated.

Symbolic Feedback Code: IBM01G

IBM0049S **ONCODE=** *oncode-value* **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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM01N

IBM0056S **ONCODE=0636 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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM01R

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM03R

IBM0124S **ONCODE=** *oncode-value* **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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM03T

IBM0140S **ONCODE=** *oncode-value* **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 application is terminated.

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, PUT, 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. Processing of an UPDATE file can continue. The ONCODE associated with this message 41.

Programmer Response: If the error recurs, obtain a dump of the output buffer areas by using PLIDUMP in a TRANSMIT ON-unit. See *IBM PL/I for VSE/ESA 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 application is terminated.

Symbolic Feedback Code: IBM04D

IBM0142S **ONCODE=** *oncode-value* **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. 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. See *IBM PL/I for VSE/ESA 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 application is terminated.

Symbolic Feedback Code: IBM04E

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM04J

IBM0148S **ONCODE=** *oncode-value* **The TRANSMIT condition was raised because a wrong-length record error occurred during input (***FILE= or ONFILE= file-name***).**

Explanation: Data management routines have detected that the length of the transmitted block was not the same as the specified block size for the file. If the length of the block was greater than the specified block size, the data read into the file buffer will have been truncated. Note that the wrong-length record condition will not be raised in the following circumstances:

- For undefined format records (except for long blocks read from tape)
- For variable format records where a short block was read
- For blocked fixed format records where a short block was read and the length is a multiple of the record size.

The ONCODE associated with this message is 42.

Programmer Response: Ensure that the blocksize and record size specified are correct for the data set. For variable format files the blocksize must be at least as large as the largest block in the data set (including record and block descriptor words). If the program is not in error then examine the data set for unexpected blocks of the wrong length. This may be the case, for example, if the data set was created with a file format different from that with which it is being read.

System Action: The application is terminated.

Symbolic Feedback Code: IBM04K

IBM0149S **ONCODE=** *oncode-value* **The TRANSMIT condition was raised because of an error assembling record segments during input (***FILE= or ONFILE= file-name***).**

Explanation: PL/I routines have detected an error while assembling the segments of a spanned data record. One of the following errors was detected:

- An unexpected segment sequence was encountered. The segments in a spanned record file must follow a logical sequence based on the segment type. For example, a “begin” segment can only be followed by a “middle” segment or an “end” segment, and never by a “complete” segment or another “begin” segment. For all sequence errors PL/I will skip segments until a “complete” segment or a “begin” segment is found (which may be the segment at which the sequence error was detected). The skipped segments are discarded.
- The block descriptor word length did not match the length of the physical block. PL/I completed processing for the error record by using the physical block length in place of the block descriptor word block length. If the program handles the error and attempts to continue processing further errors may occur if segments within the block are not internally consistent.
- The length indicated in a segment descriptor word extended past the end of the current block. PL/I continues processing the error record by truncating the segment at the end of the physical block.
- The length indicated in a block or segment descriptor word was too small (less than four for a SDW or less than eight for a BDW). If the program handles the error and attempts to continue, processing will recommence at the start of the next block. In this case the data in the record variable is unchanged.

The ONCODE associated with this message is 42.

Programmer Response: Examine the data set for incorrect spanned record data. It is possible to read the data set by coding a TRANSMIT on-unit to handle the error and continuing to read the data set. However, data included in invalid blocks or segments may not be available to the program.

System Action: The application is terminated.

Symbolic Feedback Code: IBM04L

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 application is terminated.

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. 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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 region. 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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 DLBL job control statement. 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 application is terminated.

Symbolic Feedback Code: IBM06B

IBM0204S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because a DLBL statement was not used for (***FILE= or ONFILE= file-name***).**

Explanation: The job stream for a file did not contain a DLBL statement. The job stream must contain a DLBL statement. with a file ID 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 DLBL statement to associate the file with a physical data set.

System Action: The application is terminated.

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 application is terminated.

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 application is terminated.

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 the ENVIRONMENT attribute.

Programmer Response: Specify the key length and rerun the program.

System Action: The application is terminated.

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, 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.

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.

System Action: The application is terminated.

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. The following is a list of possible conflicts:

Organization	Attributes
CONSECUTIVE	DIRECT, EXCLUSIVE, KEYED, TRANSIENT
INDEXED	STREAM, TRANSIENT, DIRECT OUTPUT, OUTPUT without KEYED
REGIONAL	STREAM, TRANSIENT, OUTPUT without KEYED
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 application is terminated.

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

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM06L

IBM0214S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because REGIONAL OUTPUT or DIRECT files must be KEYED (***FILE= or ONFILE= file-name***).**

Explanation: A REGIONAL file opened for DIRECT or OUTPUT processing must have the KEYED file attribute specified. The ONCODE associated with this message is 82.

Programmer Response: Check the file processing requirements and add KEYED to the file declaration.

System Action: The application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM06N

IBM0221S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because the device type is unsupported (** *FILE= or ONFILE= file-name* **).**

Explanation: LE/VSE does not support the device type of the system logical unit specified for the file. The ONCODE associated with this message is 88.

Programmer Response: Ensure that the MEDIUM specification and the JCL ASSGN statement for the file are correct.

System Action: The application is terminated.

Symbolic Feedback Code: IBM06T

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 application is terminated.

Symbolic Feedback Code: IBM071

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 application is terminated.

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. 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 application is terminated.

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 defined in a DLBL 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 application is terminated.

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 application is terminated.

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. 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 documentation for study by IBM.

System Action: The application is terminated.

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 was already being used by another program and could not be shared.

Programmer Response: See *IBM PL/I for VSE/ESA Programming Guide* for more information on sharing data sets.

System Action: The application is terminated.

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. See *VSE/VSAM Commands* for details.

System Action: The application is terminated.

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 application is terminated.

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***).**

Explanation: The VSAM routines detected an error during the open process. The cause of the error could not be determined explicitly. The ONCODE associated with this message is 93.

Programmer Response: Resubmit the program. If the error recurs after resubmitting the job, use PLIDUMP to obtain a storage dump and retain all the relevant documentation for study by IBM.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07B

IBM0236S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because the operating system was unable to OPEN the file (***FILE= or ONFILE= file-name***).**

Explanation: Possible reasons the file could not be opened are:

1. The DLBL statement was in error.
2. A difference existed between the parameters of the DLBL statement and the file's attributes.
3. There was no FORMAT-1 label in the VTOC with the name given for an input file.
4. An EXTENT statement for an input file did not match the FORMAT-1 label in the VTOC.
5. The logical unit specified on an EXTENT statement for the file was assigned to a different volume.
6. Vendor Exit management received a return code that was undefined.

Programmer Response: The numbered responses below apply to the correspondingly numbered explanations above.

1. Fix the DLBL statement.
2. Ensure the parameters specified on the DLBL statement are compatible with the file's attributes.
3. Check the file ID on the DLBL statement for a misspelling.
4. Check the start track and number of tracks specified on the EXTENT statement.
5. Change the logical unit name or assign the logical unit to the correct volume.
6. If a tape or disk management system is in use, ensure that it is operating correctly.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07C

IBM0237S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because a symbolic device name is not assigned or is invalid (***FILE= or ONFILE= file-name***).**

Explanation: The address in the ASSGN statement for the logical unit was UA or IGN or the logical unit was invalid. The ONCODE associated with this message is 84.

Programmer Response: Correct the JCL.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07D

IBM0238S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because an EXTENT statement is in error (***FILE= or ONFILE= file-name***).**

Explanation:

- The symbolic unit specified in the EXTENT statement was invalid.
- The volume serial numbers in the EXTENT statement do not match those in the catalog entry.
- More than 16 extents were specified.

The ONCODE associated with this message is 84.

Programmer Response: Correct the JCL.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07E

IBM0240S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because the required volume could not be mounted (***FILE= or ONFILE= file-name***).**

Explanation: Either an attempt was made to mount two volumes on the same unit or the operator was unable to mount the required volume. The ONCODE associated with this message is 92.

Programmer Response: Correct the JCL.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07G

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 application is terminated.

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 application is terminated.

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***).**

Explanation: When the ENVIRONMENT option BKWD is used on file opening, the file must be positioned at the last record. If an attempt to position the last record fails, the file is closed and the UNDEFINEDFILE condition is raised with this message.

Programmer Response: Check with the system operator or the system programmer.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07J

IBM0244S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because CONSECUTIVE files cannot be DIRECT or KEYED (***FILE= or ONFILE= file-name***).**

Explanation: CONSECUTIVE files must be SEQUENTIAL and records do not have a key. The ONCODE associated with this message is 82.

Programmer Response: Change the file declaration or the OPEN statement attributes.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07K

IBM0245S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because DIRECT files cannot be CONSECUTIVE or BUFFERED (***FILE= or ONFILE= file-name***).**

Explanation: DIRECT files must be VSAM or REGIONAL and cannot be BUFFERED. The ONCODE associated with this message is 82.

Programmer Response: Change the file declaration or the OPEN statement attributes.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07L

IBM0246S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because STREAM files must be CONSECUTIVE (***FILE= or ONFILE= file-name***).**

Explanation: VSAM or REGIONAL files cannot be accessed in STREAM mode. The ONCODE associated with this message is 82.

Programmer Response: Change the file declaration or the OPEN statement attributes.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07M

IBM0247S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because STREAM files cannot be spanned (***FILE= or ONFILE= file-name***).**

Explanation: Spanned files cannot be accessed in STREAM mode. The ONCODE associated with this message is 82.

Programmer Response: Change the file declaration or the OPEN statement attributes.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07N

IBM0248S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because PRINT files must be STREAM OUTPUT (***FILE= or ONFILE= file-name***).**

Explanation: The PRINT attribute is only valid for a STREAM OUTPUT file. The ONCODE associated with this message is 82.

Programmer Response: Change the file declaration or the OPEN statement attributes.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07O

IBM0249S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because VSAM files cannot be STREAM, BACKWARDS or PRINT (***FILE= or ONFILE= file-name***).**

Explanation: The STREAM, BACKWARDS and PRINT attributes are invalid for VSAM files. The ONCODE associated with this message is 82.

Programmer Response: Change the file declaration or the OPEN statement attributes.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07P

IBM0250S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because the REUSE option is valid only with OUTPUT files (***FILE= or ONFILE= file-name***).**

Explanation: The REUSE option is used to erase and rewrite a VSAM file. It is not permitted for INPUT or UPDATE mode. The ONCODE associated with this message is 82.

Programmer Response: Remove the REUSE option from the ENVIRONMENT attribute.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07Q

IBM0251S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because the BKWD option is valid only with SEQUENTIAL INPUT/UPDATE files (***FILE= or ONFILE= file-name***).**

Explanation: The VSAM BKWD ENVIRONMENT option cannot be used with OUTPUT or DIRECT files. The ONCODE associated with this message is 82.

Programmer Response: Remove the BKWD option from the ENVIRONMENT attribute.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07R

IBM0252S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because a VSAM file was not declared ENV(VSAM) (***FILE= or ONFILE= file-name***).**

Explanation: The file you are attempting to access is a VSAM file, but the VSAM ENVIRONMENT option was not specified in the file declaration. The ONCODE associated with this message is 82.

Programmer Response: If you wish to use this file, add the VSAM ENVIRONMENT option to the file declaration.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07S

IBM0253S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because ESDS and RRDS files cannot have a KEYLEN, KEYLOC or GENKEY option (***FILE= or ONFILE= file-name***).**

Explanation: The ENVIRONMENT options KEYLEN, KEYLOC and GENKEY are not meaningful for VSAM ESDS or RRDS files. The ONCODE associated with this message is 82.

Programmer Response: Remove the errant option from the file declaration.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07T

IBM0254S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because ESDS files must be SEQUENTIAL and ENV(VSAM) or ENV(CONSECUTIVE) (***FILE= or ONFILE= file-name***).**

Explanation: ESDS files should be declared SEQUENTIAL ENV(VSAM). ENV(CONSECUTIVE) is allowed for data portability between ESDS files and SAM files. The ONCODE associated with this message is 82.

Programmer Response: Change the file declaration.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07U

IBM0255S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because an AIX PATH cannot be SEQUENTIAL OUTPUT (***FILE= or ONFILE= file-name***).**

Explanation: Records can only be added sequentially to a VSAM base cluster (ESDS or KSDS). New records must be written to AIX PATHs using DIRECT mode only. The AIX must be unique. The ONCODE associated with this message is 82.

Programmer Response: Change the file declaration to DIRECT OUTPUT.

System Action: The application is terminated.

Symbolic Feedback Code: IBM07V

IBM0256S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because a non-unique AIX PATH cannot be DIRECT (***FILE= or ONFILE= file-name***).**

Explanation: A non-unique AIX PATH can only be read or updated sequentially. The ONCODE associated with this message is 82.

Programmer Response: Change the file declaration to SEQUENTIAL INPUT or UPDATE.

System Action: The application is terminated.

Symbolic Feedback Code: IBM080

IBM0257S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because a SEQUENTIAL OUTPUT KSDS must be KEYED (***FILE= or ONFILE= file-name***).**

Explanation: You must specify the KEYED attribute to open a KSDS for SEQUENTIAL OUTPUT. A key must be specified for each record written, and the keys must be in sequence. The ONCODE associated with this message is 82.

Programmer Response: Add the KEYED attribute to the file declaration.

System Action: The application is terminated.

Symbolic Feedback Code: IBM081

IBM0258S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because the RECSIZE option was not specified (***FILE= or ONFILE= file-name***).**

Explanation: PL/I cannot determine the record size for the indicated file. The RECSIZE ENVIRONMENT option is required. The ONCODE associated with this message is 83.

Programmer Response: Add the RECSIZE ENVIRONMENT option to the file declaration, specifying the required record length.

System Action: The application is terminated.

Symbolic Feedback Code: IBM082

IBM0259S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because the record or block size is invalid (***FILE= or ONFILE= file-name***).**

Explanation: Negative values for the RECSIZE or BLKSIZE ENVIRONMENT options are not permitted. The ONCODE associated with this message is 86.

Programmer Response: Change the RECSIZE or BLKSIZE value.

System Action: The application is terminated.

Symbolic Feedback Code: IBM083

IBM0260S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because REGIONAL(1) and (2) files must be fixed format (***FILE= or ONFILE= file-name***).**

Explanation: The Variable and Undefined record formats are not permitted for REGIONAL(1) and (2) files. The ONCODE associated with this message is 87.

Programmer Response: Change the record format specified by the ENVIRONMENT attribute.

System Action: The application is terminated.

Symbolic Feedback Code: IBM084

IBM0261S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because CONSECUTIVE UNBUFFERED files cannot be blocked or spanned (***FILE= or ONFILE= file-name***).**

Explanation: CONSECUTIVE files must have the BUFFERED attribute to access blocked or spanned records. The ONCODE associated with this message is 87.

Programmer Response: Specify the BUFFERED attribute or change the record format.

System Action: The application is terminated.

Symbolic Feedback Code: IBM085

IBM0262S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because REGIONAL or UNBUFFERED files cannot be blocked (***FILE= or ONFILE= file-name***).**

Explanation: REGIONAL data sets do not use blocking. Blocked records can only be processed via a CONSECUTIVE BUFFERED file. The ONCODE associated with this message is 87.

Programmer Response: Change the file declaration.

System Action: The application is terminated.

Symbolic Feedback Code: IBM086

IBM0263S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because REGIONAL SEQUENTIAL UNBUFFERED files cannot be spanned (***FILE= or ONFILE= file-name***).**

Explanation: A REGIONAL(3) SEQUENTIAL VS-format file must have the BUFFERED attribute. The ONCODE associated with this message is 87.

Programmer Response: Change the file declaration to specify BUFFERED.

System Action: The application is terminated.

Symbolic Feedback Code: IBM087

IBM0264S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because CONSECUTIVE UNBUFFERED files can be tape or disk only (***FILE= or ONFILE= file-name***).**

Explanation: VSE restrictions limit CONSECUTIVE UNBUFFERED files to tape and disk devices only. The ONCODE associated with this message is 88.

Programmer Response: Use a BUFFERED file to access other devices.

System Action: The application is terminated.

Symbolic Feedback Code: IBM088

IBM0265S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because REGIONAL is only permitted for disk files (***FILE= or ONFILE= file-name***).**

Explanation: The REGIONAL option is permitted only for Direct Access Storage Devices (disk devices). The ONCODE associated with this message is 88.

Programmer Response: Change the JCL to use a disk file or remove the REGIONAL option.

System Action: The application is terminated.

Symbolic Feedback Code: IBM089

IBM0266S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because BACKWARDS is only permitted for tape F and U formats (***FILE= or ONFILE= file-name***).**

Explanation: Variable format tape files can be read forwards only. The ONCODE associated with this message is 88.

Programmer Response: Remove the BACKWARDS attribute.

System Action: The application is terminated.

Symbolic Feedback Code: IBM08A

IBM0267S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because UPDATE is only permitted for disk files (***FILE= or ONFILE= file-name***).**

Explanation: The UPDATE attribute is invalid for non-disk devices. The ONCODE associated with this message is 88.

Programmer Response: Remove the UPDATE attribute. The file may be updated by reading the data, changing the data and writing the new data to another file.

System Action: The application is terminated.

Symbolic Feedback Code: IBM08B

IBM0268S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because card or printer devices cannot be blocked or spanned (***FILE= or ONFILE= file-name***).**

Explanation: Only disk and tape files can blocked or spanned. The ONCODE associated with this message is 88.

Programmer Response: Change the record format or use a different device.

System Action: The application is terminated.

Symbolic Feedback Code: IBM08C

IBM0269S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because diskette files cannot be V, U or blocked (***FILE= or ONFILE= file-name***).**

Explanation: Diskette devices use fixed length unblocked records. The ONCODE associated with this message is 88.

Programmer Response: Change the record format.

System Action: The application is terminated.

Symbolic Feedback Code: IBM08D

IBM0270S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because printer devices must be OUTPUT (***FILE= or ONFILE= file-name***).**

Explanation: The INPUT or UPDATE attribute is not meaningful for printer devices. The ONCODE associated with this message is 88.

Programmer Response: Specify the OUTPUT attribute or use a different device.

System Action: The application is terminated.

Symbolic Feedback Code: IBM08E

IBM0271S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because tape devices must be CONSECUTIVE (***FILE= or ONFILE= file-name***).**

Explanation: The REGIONAL or VSAM option was specified for a tape device. The ONCODE associated with this message is 88.

Programmer Response: Change the file declaration to specify CONSECUTIVE.

System Action: The application is terminated.

Symbolic Feedback Code: IBM08F

IBM0272S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because tape devices cannot be UPDATE or DIRECT (***FILE= or ONFILE= file-name***).**

Explanation: The UPDATE or DIRECT attribute is invalid for tape devices. The ONCODE associated with this message is 88.

Programmer Response: Specify SEQUENTIAL INPUT or OUTPUT in the file declaration.

System Action: The application is terminated.

Symbolic Feedback Code: IBM08G

IBM0273S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because the RECSIZE, KEYLENGTH or KEYLOC ENVIRONMENT option conflicts with the data set (***FILE= or ONFILE= file-name***).**

Explanation: The RECSIZE, KEYLENGTH or KEYLOC ENVIRONMENT option does not match the actual data set value. The ONCODE associated with this message is 91.

Programmer Response: Change the file declaration to specify the correct values.

System Action: The application is terminated.

Symbolic Feedback Code: IBM08H

IBM0274S **ONCODE=** *oncode-value* **The UNDEFINEDFILE condition was raised because the LE/VSE DTF builder encountered an error (***FILE= or ONFILE= file-name***).**

Explanation: An unknown error occurred attempting to generate the DTF for this file. The ONCODE associated with this message is 93.

Programmer Response: Contact your IBM representative. Try changing the file declaration as an interim solution.

System Action: The application is terminated.

Symbolic Feedback Code: IBM08I

IBM0275S ONCODE= *oncode-value* **The UNDEFINEDFILE condition was raised because there was insufficient GETVIS storage (***FILE= or ONFILE= file-name***).**

Explanation: Insufficient Partition GETVIS storage was available to allocate the necessary buffers and control blocks for this file. The ONCODE associated with this message is 93.

Programmer Response: Change the JCL to decrease the value on the EXEC statement SIZE parameter, or increase the size of the partition.

System Action:

Symbolic Feedback Code: IBM08J

IBM0276W Additional CLOSE attempt ignored for file *file-name***.**

Explanation: The request to close the file is ignored because a previous attempt to close the file has failed. The previous close attempt encountered a condition and could not complete for one of the following reasons:

- The condition was unhandled and promoted to the ERROR condition.
- The ON-UNIT for the condition performed a GOTO out of ON-UNIT.

Programmer Response: Fix the original error causing the first close attempt to fail.

System Action: The CLOSE statement is ignored.

Symbolic Feedback Code: IBM08K

IBM0277W Implicit CLOSE detected a previous CLOSE failure for file *file-name***.**

Explanation: A previous close failure was detected by implicit CLOSE during program termination. The integrity of the file may be compromised. See message IBM0276 for more information.

Programmer Response: Fix the original error causing the close attempt to fail.

System Action: The close is ignored and termination continues.

Symbolic Feedback Code: IBM08L

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 GOTO statement.

System Action: The application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM08P

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 340.

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 application is terminated.

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

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

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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.

System Action: The application is terminated.

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.

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

IBM0460S ONCODE= *oncode-value* **The OVERFLOW condition was raised by a SIGNAL statement.**

Explanation: 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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0ED

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 application is terminated.

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 (15 for decimal values and 31 for binary values).

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 application is terminated.

Symbolic Feedback Code: IBM0F2

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 string specification.**

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: IBM0FQ

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: IBM0FR

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: IBM0FS

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

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: IBM0G4

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: IBM0G6

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 date 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

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0GP

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 application is terminated.

Symbolic Feedback Code: IBM0HG

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0HK

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 application is terminated.

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 bif
- 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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0HR

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 application is terminated.

Symbolic Feedback Code: IBM0HT

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: The specified option requires a multitasking environment, but this is not supported by the current release of LE/VSE. The ONCODE associated with this message is 3915.

Programmer Response: Remove the TASK, EVENT, or PRIORITY option from the CALL statement.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0I0

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 matching that used in the FETCH statement. The ONCODE associated with this message is 9250.

Programmer Response: Ensure that the phase that is to be fetched is accessible at run-time, and that it is stored with the same name as that used in the FETCH statement.

System Action: The application is terminated.

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 phase named in the FETCH statement. The ONCODE associated with this message is 9251.

Programmer Response: Ensure that the required phase has been cataloged into the appropriate sublibrary with proper member type, and then rerun the job. If the problem recurs, inform your installation system programmer, who will take the appropriate action.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0IF

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. A debugging tool 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 application is terminated.

Symbolic Feedback Code: IBM0II

IBM0595S **ONCODE=9255 An attempt was made to release a phase containing non-PL/I high level language programs.**

Explanation: A phase containing non-PL/I high level language programs, such as C or COBOL, could not be released by a PL/I RELEASE statement. The phase will be released automatically during the enclave termination. A phase 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 phase will be released automatically during the enclave termination.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0IJ

IBM0596S **A back level PL/I module has been link-edited to a main phase.**

Explanation: The main phase has been linked with an object compiled by DOS PL/I V1R6 (or earlier). The job is canceled.

Programmer Response: Recompile any DOS PL/I programs with PL/I.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0IK

IBM0598S **ONCODE=** *oncode-value* **PL/I module** *module-name* **cannot be fetched because it was not compiled by PL/I VSE.**

Explanation: An attempt was made to FETCH a phase compiled with DOS PL/I V1R6 (or earlier). The only PL/I phases that may be fetched are those compiled with PL/I VSE. The ONCODE associated with this message is 9258.

Programmer Response: Recompile the offending phase with PL/I VSE.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0IM

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0IS

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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:

Table 44.

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, or TAND)
Binary 53<p≤109 Decimal 16<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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0L4

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0LT

IBM0722S **ONCODE=** *oncode-value* **X was assigned a value of 0 and Y was not assigned the value of a positive real number in X**Y.**

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0NE

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

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. Other possible conflicts include:

Statement and Option	Conflicting File Attribute or Organization
Any record I/O statement	STREAM
Any stream I/O statement	RECORD
READ	OUTPUT
READ SET	UNBUFFERED
READ EVENT	BUFFERED
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 application is terminated.

Symbolic Feedback Code: IBM0P9

IBM0810S **ONCODE=** *oncode-value* **A spanned record has been read that was larger than the specified record size, compromising program integrity (FILE= or ONFILE=** *file-name***).**

Explanation: While reading a file that was opened for update, data management has returned a spanned record to PL/I that is larger than the record size specified for the file. As PL/I can only detect this condition after the record has been read, it is possible that program storage will have been corrupted by the oversized record. The ONCODE associated with this message is 1011.

Programmer Response: Examine the data set to determine the size of the largest spanned record expected. Because this problem is unique to update mode, program integrity may be ensured by opening the file for input. Otherwise examine the program that created the data set to determine the size of the largest spanned record that it should have created. Note that although the size of a spanned record is theoretically unlimited, processing in update mode is limited to a maximum record size of 32756.

Important

If this error is handled and the program allowed to continue the integrity of the program cannot be guaranteed.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0PA

IBM0811S **ONCODE=** *oncode-value* **An I/O error occurred. The cause could not be determined due to insufficient data.**

Explanation: The data management routines detected an error during an input/output operation. The cause of the error could not be determined. The ONCODE associated with this message is 1011.

Programmer Response: If the error recurs after resubmitting the job, use PLIDUMP (or CEE5DMP) to obtain a storage dump and retain all the relevant documentation for study by IBM.

System Action: The application is terminated.

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 application is terminated.

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 application is terminated.

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 allowed by the operating system. For VSE, this limit is one. 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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0PI

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 application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0PN

IBM0825S ONCODE= *oncode-value* **The EVENT variable was already in use with file** *file-name***.**

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0PR

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 partition size for the VSAM application.

System Action: The application is terminated.

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 partition GETVIS size. The access method services VERIFY command can be used to obtain more information pertaining to the error. See *VSE/VSAM Commands* for details.

System Action: The application is terminated.

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 application is terminated.

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. 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. See *VSE/VSAM Commands* for details.

System Action: The application is terminated.

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 DLBL 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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0Q2

IBM0835S ONCODE= *oncode-value* **An attempt to position the file for a sequential READ failed.**

Explanation: An attempt to reposition the next highest key for subsequent sequential retrieval on a VSAM KSDS, after the 'key not found' condition, failed. If file processing continued, the next I/O statement should have a positioning KEY option. (See message IBM0831). The ONCODE associated with this message is 1029.

Programmer Response: Use the KEYTO option of the READ statement to obtain the keys of the records read. Use this information to reposition the file for a subsequent retrieval.

System Action: The application is terminated.

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 DLBL statement (that is, using the same title). The STRNO value that specifies 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 DSN ENVIRONMENT option (STRNO) 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, 'DSN(3)' should be specified in the ENVIRONMENT attribute.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0Q4

IBM0837S ONCODE= *oncode-value* **An error occurred during an index upgrade.**

Explanation: A change to a base cluster could not be reflected in one of the indexes of the cluster's upgrade set. The ONCODE associated with this message is 1030.

Programmer Response: Run the job with a larger partition GETVIS size. The problem might be related to an insufficient amount of virtual storage available to VSAM.

System Action: The application is terminated.

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 the appropriate VSAM manual for more information regarding the use of alternate index paths.

System Action: The application is terminated.

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: See *IBM PL/I for VSE/ESA Programming Guide* regarding a general description on the use of alternate index. For more information, refer to the appropriate VSAM manual.

System Action: The application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0Q8

IBM0841S **ONCODE=** *oncode-value* **Data set open for output has used all available space (***FILE= or ONFILE= file-name***).**

Explanation: All extents of the data set have been filled during sequential output. The ONCODE associated with this message is 1040.

Programmer Response: Allocate more extents for the file or modify the program to write less data to the file.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0Q9

IBM0850S **ONCODE=** *oncode-value* **The aggregate length exceeded the limit of 2**24 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 application is terminated.

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 application is terminated.

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 routine 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 routines are within the limits described above.

System Action: The application is terminated.

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 application is terminated.

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 application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0QO

IBM0870S ONCODE= *oncode-value* **The DOS/VS COBOL program is not supported for interlanguage communication in IBM Language Environment for VSE/ESA.**

Explanation: The DOS/VS COBOL program is not supported for interlanguage communication in IBM Language Environment for VSE/ESA.

Programmer Response: Compile the DOS/VS COBOL program with IBM COBOL for VSE/ESA or don't run the application with IBM Language Environment for VSE/ESA.

System Action: The application is terminated.

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 application is terminated.

Symbolic Feedback Code: IBM0RG

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 application is terminated.

Symbolic Feedback Code: IBM0S4

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 ONCODE= *oncode-value* **The CHECKPOINT/RESTART facility is not supported in a dynamic partition.**

Explanation: An attempt was made to call the CHECKPOINT/RESTART facility from PL/I. CHECKPOINT/RESTART is not supported in a dynamic partition. The ONCODE associated with this message is 9300.

Programmer Response: Remove the call to the CHECKPOINT/RESTART facility. If this facility needs to be used, run the application in a static partition which is below the 16-megabyte line.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0SU

IBM0927S ONCODE= *oncode-value* **The CHECKPOINT/RESTART facility is not supported in a partition that extends above the 16-megabyte line.**

Explanation: An attempt was made to call the CHECKPOINT/RESTART facility from PL/I. CHECKPOINT/RESTART is not supported in a partition that extends above the 16-megabyte line. The ONCODE associated with this message is 9301.

Programmer Response: Remove the call to the CHECKPOINT/RESTART facility. If this facility needs to be used, run the application in a static partition that does not extend beyond the 16-megabyte line.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0SV

IBM0928S The PLICANC facility is not supported by PL/I VSE.

Explanation: An attempt was made to call the built-in subroutine PLICANC. This facility is not available under VSE.

Programmer Response: Remove the call to PLICANC.

System Action: Processing continues with the next PL/I statement.

Symbolic Feedback Code: IBM0T0

IBM0929S The job has been canceled by PLIREST.

Explanation: Automatic restart is not available under VSE. The job is canceled to allow manual restart.

Programmer Response: Restart the job manually if necessary.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0T1

IBM0930S ONCODE= *oncode-value* **An attempt was made to call a Checkout-compiled program in the LE/VSE environment.**

Explanation: Checkout-compiled programs are not supported in the LE/VSE environment.

Programmer Response: Remove the call to the Checkout-compiled program.

System Action: The application is terminated.

Symbolic Feedback Code: IBM0T2

Chapter 12. COBOL Run-Time 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.

For a description of the format of COBOL run-time messages, see “Interpreting Run-Time Messages” on page 25.

IGZ0002S *debugging-information*

Explanation: A permanent I/O error has occurred on a SAM file. The format of the information in *debugging-information* is:

```
Permanent I/O error file name = filename
CCB = X'ccb'
```

where:

filename is the name associated with the file.

ccb is the hexadecimal-format CCB associated with the file. The CCB contains codes describing the error that occurred.

Programmer Response: For more information regarding the format of the CCB, see *z/VSE System Macros Reference*

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

IGZ0004S **There was an attempt to cancel program** *program-name* **that was statically linked with a non-COBOL high level language program.**

Explanation: A COBOL program that is linked with programs of other high level languages cannot be canceled.

Programmer Response: Remove the failing CANCEL statement.

System Action: The application was terminated.

Symbolic Feedback Code: IGZ004

IGZ0005S **DOS/VS COBOL programs in the application were found in multiple enclaves.**

Explanation: DOS/VS COBOL programs are restricted to one enclave within an application.

Programmer Response: Modify the application so that the DOS/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 the variable length group** *group-name* **exceeded the maximum defined length of the group at the time of reference by** *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 phase *phase-name* was unsuccessful.

Explanation: An attempt to delete a phase 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 sublibraries 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 *CICS Transaction Server for VSE/ESA Application Programming Reference*

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 LE/VSE. 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: COBOL does not allow recursive entries to programs which have started, but have not yet terminated. For example, if program A calls program B, program B cannot call program A.

Programmer Response: Remove the recursive call to *program-name*.

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: Make sure the system logical unit associated with the file is assigned to a valid device.

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, TESTCB, SHOWCAT, or LABEL 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

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.

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 the sort/merge operation.

Programmer Response: Determine why the sort/merge operation was unsuccessful and fix the problem. Possible reasons why the sort/merge operation was unsuccessful include:
- There was an error detected by the Sort/Merge program being used. See the Sort/Merge program 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:
- The name of the sort control file specified in the SORT-CONTROL special register was SYSIPT, but SYSIPT was not assigned to a valid device.
- The name of the sort control file specified in the SORT-CONTROL special register was the name of a VSE Librarian member, but the member was not found in any of the sublibraries defined in the LIBDEF search chain.

When the sort control file cannot be opened, user-supplied sort control cards will not be passed to the Sort/Merge program being used.

Programmer Response: If you want to pass in sort control cards from the sort control file, verify that the name is specified and the file or VSE Librarian member 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 the Sort/Merge program being used.

Programmer Response: For more information, look at any previous system messages 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-1 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.

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.

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.

Programmer Response: Check the file extents and, if necessary, reallocate the file.

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 the indicated file is correctly defined in your JCL. For a description of the file status code, see *IBM COBOL for VSE/ESA Language Reference*

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 *IBM COBOL for VSE/ESA 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

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/VSE 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/VSE or VS COBOL II 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

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 *IBM COBOL for VSE/ESA 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

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* **prior to 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 ILBDSET0, 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 ILBDSET0 can occur only at the beginning of the application. For example, this error can occur when:

- PL/I program calls ASSEMBLE program which calls ILBDSET0
- LE/VSE-enabled ASSEMBLE program calls ILBDSET0

Programmer Response: Only invoke ILBDSET0 prior to calling any program within the application that brings up LE/VSE.

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 *IBM COBOL for VSE/ESA 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 cancelation 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

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 for VSE/ESA 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 DOS/VS COBOL program** *program-name* **.**

Explanation: On CICS, a COBOL/VSE or a VS COBOL II program attempted to call an DOS/VS

COBOL program with the CALL statement. Using the CALL statement to perform calls between the following are not supported on CICS:

- COBOL/VSE programs and DOS/VS COBOL programs
- VS COBOL II programs and DOS/VS COBOL programs

Programmer Response: If you need to invoke a DOS/VS COBOL program from a COBOL/VSE or VS COBOL II program, use EXEC CICS LINK.

System Action: The application was terminated.

Symbolic Feedback Code: IGZ02F

IGZ0080I End of file or end of volume on *filename*. Reply "EOF" or "EOV".

Explanation: The program has read to the end of an unlabeled input tape. If the tape is a labeled tape, this message might occur if a TLBL JCL statement has not been provided, or it has been specified with a different filename than *filename* specified in assignment-name-1 of the ASSIGN clause for the file.

Programmer Response: Reply "EOF" if this is the last or only input volume. Reply "EOV" if there are additional input volumes for the file being processed.

System Action: The system waits for a response to the message.

Symbolic Feedback Code: IGZ02G

IGZ0081W The CHECKPOINT file *filename* was not available. Checkpointing will be by-passed.

Explanation: The file *filename* specified on the RERUN clause was not available. The checkpointing specified by the RERUN clause was by-passed.

Programmer Response: Check your job control to make sure that the checkpoint file is defined correctly.

System Action: No system action was taken.

Symbolic Feedback Code: IGZ02H

IGZ0082W The open of file with system-name *system-name* was unsuccessful. The system logical unit number was invalid or could not be determined.

Explanation: COBOL attempted to OPEN the file *system-name*, but was unable to determine a valid system logical unit number for the file.

If the file resides on a direct access device, and is not a SAM ESDS file, one of the following may have occurred:

- There was a DLBL job control statement for the file, but the system logical unit number was not specified in either an EXTENT statement associated with the DLBL, or in the COBOL ASSIGN clause.
- There was a DLBL job control statement and an EXTENT job control statement for the file, but the system logical unit number specified in the EXTENT statement was not a programmer logical unit number

If the file resides on a device other than a direct access device, the system logical unit number was not specified in the COBOL ASSIGN clause, or the system logical unit number specified in the ASSIGN clause was not a programmer logical unit in the range SYS000 - SYS254.

Programmer Response: Check your job control to make sure that the specified file is defined correctly.

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: IGZ02I

IGZ0083W The open of file with system-name *system-name* (on SYS*nnn*) was unsuccessful. The system logical unit was assigned to an unsupported device.

Explanation: The system logical unit number was assigned to a device that is not supported by COBOL, or the logical unit number was assigned to UA, and COBOL attempted to create the file.

Programmer Response: Check your job control to make sure that the specified file is defined correctly.

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: IGZ02J

IGZ0084W The open of file with system-name *system-name* (on SYS*nnn*) was unsuccessful. The multi-function device codes were invalid for the assigned device.

Explanation: The device-mode code supplied in the ASSIGN clause is not valid or the device mode is not supported by the device assigned to the system logical unit. A card device is assigned to the system logical unit, but the device does not support the optical-mark-read (OMR) or read-column-eliminate (RCE) mode specified.

Programmer Response: Check the device-mode code to ensure that it is valid for the type of device being used. Make sure that SYS*nnn* is assigned to a supported device.

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: IGZ084K

IGZ0087I The open of file with system-name *system-name* (on SYS*nnn*) was unsuccessful. The record length specified was zero.

Explanation: RECORD CONTAINS 0 CHARACTERS was specified for a file, but it was not possible to determine the record length for the input file. RECORD CONTAINS 0 CHARACTERS is not supported for output files.

Programmer Response: Modify your program to specify the record length.

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: IGZ02N

IGZ0088I The open of file with system-name *system-name* (on SYS*nnn*) was unsuccessful. The block length supplied was zero.

Explanation: BLOCK CONTAINS 0 CHARACTERS was specified for a file but, it was not possible to determine the block length for the input or output file.

Programmer Response: Modify your program to specify the block length.

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: IGZ02O

IGZ0089I The open of file with system-name *system-name* (on SYS*nnn*) was unsuccessful. An abend occurred in VSE LIOCS during open processing.

Explanation: An abend occurred during VSE LIOCS open processing that was trapped by COBOL.

Programmer Response: Check the message issued by VSE to determine the cause of the error.

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: IGZ02P

IGZ0090S Get storage request was unsuccessful.

Explanation: A request for GETVIS storage was issued, but failed. This message is usually issued when there is insufficient partition GETVIS storage available to satisfy the get storage request.

Programmer Response: Increase the amount of GETVIS storage in the partition by decreasing the value specified in the SIZE parameter of the EXEC JCL statement, or increase the partition size.

System Action: The application was terminated.

Symbolic Feedback Code: IGZ02Q

IGZ0091S Free storage request was unsuccessful.

Explanation: A request to free GETVIS storage was issued, but failed.

Programmer Response: This is an internal error. Contact your system representative.

System Action: The application was terminated.

Symbolic Feedback Code: IGZ02R

IGZ0092W The close of file with system-name *system-name* (on SYS*nnn*) was unsuccessful. An abend occurred in VSE LIOCS during close processing.

Explanation: An abend occurred during VSE LIOCS close processing that was trapped by COBOL.

Programmer Response: Check the message issued by VSE to determine the cause of the error.

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: IGZ02S

IGZ0093W The open of file with system-name *system-name* (on SYS*nnn*) was unsuccessful. The logical unit was assigned to a device different than that indicated by the JCL labels in effect.

Explanation: The DLBL or TLBL statement does not match the corresponding logical unit. This will occur if a TLBL statement exists for *system-name*, but SYS*nnn* is not assigned to tape, or if a DLBL statement exists for *system-name*, but the EXTENT statement does not specify a logical unit that is assigned to disk.

Programmer Response: Correct your job control.

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: IGZ02T

IGZ0094W **The open of file with system-name *system-name* (on SYS*nnn*) was unsuccessful. The label information for the file indicated VSAM, and errors occurred when processing the catalog information.**

Explanation: The DLBL statement for *system-name* is for a VSAM managed SAM file. The catalog entry for the File-id on the DLBL statement indicates it is not a cluster name or a VSAM managed SAM file.

Programmer Response: Correct your job control.

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: IGZ02U

IGZ0096C **A load of phase *phase-name* was unsuccessful.**

Explanation: An attempt to load a phase failed. The phase 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-number* was detected in module *module-name* .**

Explanation: An unrecoverable error was detected in run-time module *module-name*.

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

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* 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 run unit 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

IGZ0171S An attempt was made to run a COBOL/VSE program which was compiled with the compiler options NORENT and RMODE(ANY). The program name is *program-name***.**

Explanation: LE/VSE does not support COBOL/VSE programs compiled with the compiler options NORENT and RMODE(ANY).

Programmer Response: Compile the program with the RMODE(AUTO) compiler option.

System Action: The application was terminated.

Symbolic Feedback Code: IGZ05B

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 LE/VSE.

System Action: No system action is taken.

Symbolic Feedback Code: IGZ05D

IGZ0198S Corrupt External Data File Found While Running Program *program-name***.**

Explanation: While adding-to or checking the external file definition chain, LE/COBOL detected a corrupt external file definition in the chain.

Programmer Response: Check that all external file definitions are consistent between caller and called programs. Ensure that all calls to other programs with parameter fields are consistant with the called program's PROCEDURE division. Set the LE/VSE run-time option HEAPCHK(ON,1,0), to check all HEAP storage for the duration of the application's execution.The most common cause of this error message is when a storage overlay destroys the header information on an LE/COBOL external file definition table.

System Action: The application is terminated.

Symbolic Feedback Code: IGZ064

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 does not support the SEPARATE suboption.
- Running with a level of Debug Tool that could support the SEPARATE suboption but does not have current maintenance applied.

Programmer Response: Either:
- Run the program under levels of Language Environment and Debug Tool that support programs compiled with TEST(,,SEPARATE).
- 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 file.

Explanation: The program file description specified that the file was a physical sequential file and the file 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 DLBL statement associated with the ASSIGN clause are for the correct file.

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 file 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 file. For Format-F files, the record length specified in your program must exactly match the length attribute of the file. For Format-U files, the maximum record length specified in your program must exactly match the length attribute of the file. 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 file length 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: 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 file associated with the ASSIGN clause did not contain ASCII data. The OPEN statement failed.

Programmer Response: Check that the file associated with the ASSIGN clause is the correct one, and if it is, check the file 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 file 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 file associated with the ASSIGN clause is the correct one, and if it is, check the file 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 file associated with the ASSIGN clause. The OPEN statement failed.

Programmer Response: Check the data control block fields of the actual file to verify that the RECORDING MODE matches. The most common cause of this error is conflicting fixed and variable record length file 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 **A file attribute conflict was detected. File** *file-name* **in program** *program-name* **specified a block size of** *block-size* **that is not valid for the device assigned.**

Explanation: The block size specified in the file description for the file is not valid for the device associated with the ASSIGN clause. The OPEN statement failed. This may be caused by one of the following:

- The RECORDING MODE is F and the block size is not a multiple of the record size.
- The block size is less than the minimum for a tape device. The minimum block size is 12 bytes for an input file, and 18 for an output file.
- A diskette device is assigned, and the block size is not a multiple of 1, 2, 13, or 26 of the record length.

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: IGZ06D

IGZ0206W **A file attribute conflict was detected. File** *file-name* **in program** *program-name* **specified a record length of** *record-length* **that is not valid for the device assigned.**

Explanation: The record length specified in the file description for the file is not valid for the device associated with the ASSIGN clause. The OPEN statement failed. This may be caused by one of the following:

- A card device is assigned and the record length is greater than the maximum for the device type assigned.
- A printer device is assigned and the record length is greater than the maximum for the device.
- A diskette device is assigned, and the record length is greater than the maximum for the device.

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: IGZ06E

IGZ0207W **A file attribute conflict was detected. File** *file-name* **in program** *program-name* **specified RECORDING MODE** *recording-mode* **that is not valid for the device assigned.**

Explanation: The RECORDING MODE specified in the file description for the file is not valid for the device associated with the ASSIGN clause. The OPEN statement failed. This may be caused by one of the following:

- A card device is assigned and the RECORDING MODE is S.
- A card device is assigned for a file open for INPUT and the RECORDING is not F.
- The DLBL specifies a SAM ESDS file and the RECORDING MODE is S.
- A diskette device is assigned and the RECORDING is not F.
- A printer device is assigned and the RECORDING MODE is S.
- A tape device is assigned and the RECORDING MODE is S and CODESET was specified.

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.

IGZ0215S *argument-1* **for function** *function* **in program** *program* **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 was terminated.

IGZ0216S *argument-1* **for function** *function* **in program** *program* **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 was terminated.

IGZ0217S *argument-1* **for function** *function* **in program** *program* **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 was terminated.

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* **in program** *program* **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 was terminated.

IGZ0219S The base year for program *program-name* **was outside the valid range of 1900 through 1999. The sliding window value** *window-value* **resulted in a base year of** *base-year***.**

Explanation: When the 100-year window was computed using the current year and the sliding window value specified with the YEARWINDOW compiler option, the base year of the 100-year window was outside the valid range of 1900 through 1999. For example, if a COBOL program was compiled with YEARWINDOW(-99) and the COBOL program was run in the year 1998 this message would occur because the base year of the 100-year window would be 1899 (1998 - 99).

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.

IGZ0220S The current year was outside the 100-year window, *year-start* **through** *year-end***, for program** *program-name***.**

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.

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 the 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.

IGZ0235W The side file *side_file* **for program** *program-name* **was not available. A formatted variable dump cannot be produced. Librarian error occured during librarian** *libr-command* **command. Librarian return code is:** *libr-retcode***. Librarian reason code is:** *libr-rsncode***.**

Explanation: If error and reason codes indicate “file not found”, this is because the side file was not found in the “library.sublibrary” that was specified at the time of compilation. This side file was also not found in the currently-active temporary and permanent libdef PHASE search chain. By default, the member type of the side file is “SYSDEBUG”. For other error and reason codes refer to the *z/VSE System Macros Reference* for an explanation of return and reason codes for the above LIBRM *libr-command*.

Programmer Response: If error indicates “file not found”, ensure that the side file exists in either the
- *library.sublibrary* that was specified at the time of compilation.
- Active temporary or permanent libdef PHASE search chains.

The side file can be recreated by re-compiling the affected program with the SD compiler option.

For all other error and reason codes:
- Respond to the error as explained in the *z/VSE System Macros Reference*.
- See your IBM representative.

If there is a side-file-name exit being used (IGZIUXB for BATCH and IGZIUXC for CICS), the exit program might have been produced or altered the side file dataset name that is displayed.

System Action: None

Symbolic Feedback Code: IGZ07A

IGZ0236I **Unable to process the side file** *side_file* **for program** *program-name***. A formatted variable dump cannot be produced. Librarian error occured during librarian** *libr-command* **command. Librarian return code is:** *libr-retcode***. Librarian reason code is:** *libr-rsncode***.**

Programmer Response: For an explanation of the return and reason codes for the above LIBRM *libr-command*, refer to the *z/VSE System Macros Reference*.

System Action: None

Symbolic Feedback Code: IGZ07B

IGZ0237I **Unable to process the side file** *side_file* **for program** *program-name***. A formatted variable dump cannot be produced. Verification error occured.**

Explanation: This message indicates either:
- A side file with an invalid format was encountered.
- An internal logic error.

Programmer Response: See your IBM representative.

System Action: None

Symbolic Feedback Code: IGZ07C

IGZ0238I **Unable to process the side file** *side_file* **for program** *program-name***. A formatted variable dump cannot be produced. The following error occured:** *error-details*

Explanation: The reported error occurred while processing a side file.

Programmer Response: See your IBM representative.

System Action: None

Symbolic Feedback Code: IGZ07D

IGZ0239I **The side file** *side_file* **for program** *program-name* **was not found.**

Explanation: The side file was not found in either the:
- Specified “library.sublibrary”.
- Currently-active temporary or permanent libdef PHASE search chains.

By default, the member type of the side file is “SYSDEBUG”.

Programmer Response: Ensure that the side file exists in the specified library, or recompile the program.

Ensure that the side file exists in either the:
- “library.sublibrary” that was specified at the time of compilation.
- Currently-active temporary or permanent libdef PHASE search chains.

System Action: None

Symbolic Feedback Code: IGZ07E

Chapter 13. LE/VSE Abend Codes

This chapter lists the LE/VSE abend codes with descriptions and programmer responses. The hexadecimal equivalent of the abend code is shown in parentheses. Reason codes are shown in decimal with the hexadecimal equivalent in parentheses.

The abend codes described in this appendix are referred to by LE/VSE CEE-prefixed messages (for example CEE3322C). To obtain a listing on your VSE console of *all* these abend codes, type CEEABENDC on the command line and then press PF9.

Abend codes not listed here

Application programmers can use the CEE5ABD LE/VSE callable service to generate user-defined abend codes that do not appear in this chapter. For information about abend codes not listed here, see your application's documentation.

U1xxx

Explanation: An abend code in the form U1xxx might mean the enclave terminated with a COBOL-specific condition. If the abend is caused by a COBOL-specific condition, the abend is accompanied by a COBOL run-time message with a message number in the form IGZ0xxx, where xxx corresponds to the last three digits of the abend code. For example, if you receive a user abend code 1009, the corresponding COBOL run-time message is IGZ0009.

Programmer Response: Check the corresponding COBOL run-time message in Chapter 12, "COBOL Run-Time Messages," on page 307 for the recommended action.

System Action: Task terminated.

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 LE/VSE message file for message output.

System Action: Task terminated.

U4000 (X'FA0')

Explanation: LE/VSE dump services encountered an error.

Reason codes:

256 (X'100')
An invalid request was passed to the Dump File Manager.

257 (X'101')
There was an attempt to generate a dump with more than the maximum number of nested sections (5).

258 (X'102')
The Dump File Manager received a request to end a dump section when a dump section was not active.

Programmer Response: Ensure the parameters passed to the dump service routines are correct.

System Action: Enclave terminated.

U4034 (X'FC2')

Explanation: LE/VSE condition handling was bypassed. This could result from an error condition being raised while LE/VSE was dormant.

Programmer Response: Follow appropriate problem determination procedures.

System Action: Task terminated.

U4036 (X'FC4')

Explanation: A program check or abend occurred and LE/VSE determined that it could not turn the program check or abend into a condition.

Reason codes:

2 (X'02')
An abend was detected and the LE/VSE run-time option for the enclave is TRAP(OFF) or TRAP(ON,MIN). See previous abend message for a description of the detected abend.

3 (X'03')
A program check was detected and the LE/VSE run-time option for the enclave is TRAP(OFF). The address of the STXIT save area is loaded into register 10 prior to the abend being issued.

Programmer Response: Follow appropriate problem determination procedures.

System Action: Task terminated.

U4038 (X'FC6')

Explanation: The enclave ended with an unhandled LE/VSE condition of severity 2 or greater, and the run-time option ABTERMENC(ABEND) was specified.

Reason codes:

1 (X'01')
The unhandled condition was a software-raised or user-raised condition. See output on SYSLST for more information.

2 (X'02')
The unhandled condition was an error when attempting to load a phase at the enclave level.

Programmer Response: Check the LE/VSE message file for message output.

System Action: Enclave terminated.

U4039 (X'FC8')

Explanation: LE/VSE is requesting a system abend dump due to an unhandled severity 2, 3, or 4 condition, or in response to the runtime option TERMTHDACT(UADUMP) being set. This does not necessarily indicate an error condition.

Programmer Response: Refer to the original unhandled condition.

System Action: LE/VSE continues with termination.

U4040 (X'FC8')

Explanation: An unrecoverable error occurred while processing run-time options for a compiler.

Reason codes:

1 (X'01')
The fixed-sized stack overflowed.

Programmer Response: Reduce the number of run-time options provided to the compiler.

System Action: Enclave terminated.

U4041 (X'FC9')

Explanation: LE/VSE message services encountered an error.

Reason codes:

8 (X'08')
The chain of message destination control blocks was corrupted.

Programmer Response: Follow appropriate problem determination procedures.

System Action: Enclave terminated.

U4042 (X'FCA')

Explanation: The application heap data has been corrupted.

Reason codes:

0 (X'00')
The LE/VSE heap checker has detected that heap storage has been corrupted.

1 (X'01')
An unrecoverable error occurred while the heap checker was attempting to check for, or report on, heap damage.

Programmer Response: Follow appropriate problem determination procedures. The message file should contain messages indicating the address and contents of the damaged storage area.

System Action: Enclave terminated with a dump.

U4045 (X'FCD')

Explanation: An error was encountered during LE/VSE abend processing.

Reason codes:

1 (X'01')
There was insufficent GETVIS storage available to process the original condition. The original abend and reason code could not be issued.

Programmer Response: Increase the available amount of GETVIS storage for the application.

System Action: Enclave terminated.

U4082 (X'FF2')

Explanation: A second malfunction occurred while handling a condition.

Reason codes:

1 (X'01')
A second malfunction occurred while trying to initialize a second math save area.

2 (X'02')
A condition was raised prior to the point where a second condition could be recorded.

3 (X'03')
A condition was raised while LE/VSE 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 issued with this abend indicates what is invalid about the save-area that caused the abend.

Reason codes:

1 (X'01')
A save area loop exists. The save area points to itself or another save area incorrectly points to a higher save area.

2 (X'02')
Traversal of the back chain resulted in a program check.

3 (X'03')
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.

4 (X'04')
Under normal conditions, all save areas are presumed to be word aligned. In this case, a save area was improperly aligned.

5 (X'05')
A condition was raised prior to the allocation of the main stack frame, or after the main routine terminated.

15 (X'0F')
The save area chain is not intact.

Programmer Response: For applications involving assembler routines that generate their own save areas, ensure the save areas are correctly back-chained

together and conform to LE/VSE. For other applications, either:
- A storage overlay problem might have occurred.
- An unhandled condition might have occurred which caused an invalid save-area to be found on the active STACK save-area chain.

System Action: When possible, LE/VSE produces a traceback of all valid DSAs prior to the detected invalid save-area, and a dump of an active condition information block (CIB). In this situation, +X'CC' (204) in the CIB (CIB_SV1) will contain the address of the invalid save-area that was detected. Status flag 5 (CIB_FLG5) indicates the original failure that caused the save-area chain to be scanned (for example, a program-check or an abend). This should help you to determine the cause of the abend. For further information on how to use the CIB to determine the cause of an abend, see the section "Debugging with the Condition Information Block" in the manual *LE/VSE Debugging Guide and Run-Time Messages*.

Save-areas in the active chain should conform to the layout and description shown in the section "Debugging with the Stack Frame" in the manual *LE/VSE Debugging Guide and Run-Time Messages*, and should be resident on the active LE/VSE Stack. The reason code issued with this abend will indicate the part of the save-area that caused the abend.

The enclave is then terminated.

U4085 (X'FF5')

Explanation: The GOTO routine encountered an error.

Reason code:

1 (X'01')
GOTO is already active.

2 (X'02')
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:

1 (X'01')
Not enough storage to load phase.

2 (X'02')
Phase not found.

3 (X'03')
Phase not loaded.

Programmer Response: For reason code 1, increase the amount of GETVIS storage in the partition by decreasing the value specified in the SIZE parameter of the EXEC JCL statement, or increase the partition size. For reason codes 2 and 3, make sure the requested phase is in the sublibrary search chain, or in the SVA.

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:

0 (X'00')
LE/VSE condition manager was in control at the time of the condition.

2 (X'02')
A user handler routine (CEEHDLR) was processing a condition when a subsequent condition was raised.

3 (X'03')
A language-specific condition handler was processing a condition when a subsequent condition was raised.

4 (X'04')
During the LE/VSE condition manager's processing of the stack frame that precedes the stack frame for the first routine, a subsequent condition was raised.

5 (X'05')
While a language-specific event handling was being processed, a subsequent condition was raised.

6 (X'06')
A malfunction occurred while Debug Tool for VSE/ESA was in control.

7 (X'07')
While LE/VSE was trying to output a message, a subsequent condition was raised.

8 (X'08')
While attempting to output a dump, a subsequent condition was raised.

10 (X'0A')
An abnormal termination exit was in control and LE/VSE detected one of the following:
- A program check
- An abend
- A call to CEESGL to signal a condition

Programmer Response: In the case of CEEHDLR routine, recursion can occur when you use the DEPTHCONDLMT run-time option.

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:

91 (X'5B')
Stack pointer corrupted at location 1.

92 (X'5C')
Stack pointer corrupted at location 2.

93 (X'5D')
Stack pointer corrupted at location 3.

94 (X'5E')
Stack pointer corrupted at location 4.

99 (X'63')
Stack segment owning the next-available-byte (NAB) could not be found, or a zero backchain pointer was encountered.

1xx (xx = 01–16) (X'65'–X'74')
First free storage request terminated with return code xx.

2xx (xx = 01–16) (X'C9'–X'D8')
Second free storage request terminated with return code xx.

10xx (xx = 01–16) (X'3E9'–X'3F8')
First get storage terminated with return code xx, and reserve stack segment already in use. This indicates a storage condition was raised while handling the storage condition.

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 partition GETVIS size or check for infinite recursion. Using the STORAGE run-time option to increase the size of the reserve stack segment can also help. For all other reason codes, see your system programmer.

System Action: Enclave terminated.

U4091 (X'FFB')

Explanation: An unexpected condition occurred during the running of LE/VSE condition management.

Reason codes:

1 (X'01')
A GOTO was made by an enablement routine.

2 (X'02')
Invalid return code from a language-specific event handler was received during enablement processing.

3 (X'03')
LE/VSE 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.

4 (X'04')
Invalid return code from a language-specific event handler was received.

5 (X'05')
A program check was detected while LE/VSE condition manager was in control.

6 (X'06')
A request to resume the application was not accepted. The LE/VSE condition manager does not accept resumption requests with conditions, such as abends.

7 (X'07')
Invalid return code from a language-specific event handler was received during stack frame processing.

8 (X'08')
CEESGL callable service was attempting to signal a new condition. A control information block could not be allocated for that condition.

9 (X'09')
The CEEHDLR routine returned with an invalid feedback code.

10 (X'0A')
The LE/VSE library was unable to find a free control information block for a new condition. This is a critical error.

11 (X'0B')
The error count specified in the ERRCOUNT run-time option has been exceeded.

12 (X'0C')
LE/VSE signaled a condition that could not be resumed. After a resume took place, LE/VSE again attempted to terminate by signalling an imminent termination. Another resume was attempted and caused an abend.

13 (X'0D')
An invalid return code was received from Debug Tool for VSE/ESA.

14 (X'0E')
An invalid attempt to populate an ISI with qualifying data was detected.

15 (X'0F')
A condition token other than CEE000 was

returned from a member event handler. The feedback token resulted from a new condition being raised.

16 (X'10')
A request to extend stack storage could not be honored. A fixed-size stack might currently be in use.

17 (X'11')
A request for library stack storage could not be completed.

18 (X'12')
Stack storage was requested when storage management services were not available.

19 (X'13')
A request to extend stack storage could not be honored.

20 (X'14')
After being informed of a new condition, the condition handler indicated an unrecoverable error.

21 (X'15')
The maximum depth of condition nesting specified in the DEPTHCONDLMT run-time option was exceeded.

39 (X'27')
Condition management services could not find an LE/VSE STXIT exit to be established, although the application was running with TRAP(ON).

Programmer Response: If this abend was caused by a user-written condition handler, check the return codes provided to LE/VSE 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: STXIT AB or PC issued this abend because an unrecoverable error was detected. LE/VSE condition manager could not proceed.

Reason codes:

0 (X'00')
STXIT PC routine was detected.

1 (X'01')
STXIT AB routine was detected.

2 (X'02')
A CICS interface routine was detected.

Programmer Response: Follow appropriate problem determination procedures.

System Action: Enclave terminated.

U4093 (X'FFD')

Explanation: Abend issued during initialization when errors were detected.

Reason codes:

4 (X'04')
Storage management could not properly allocate the initial storage area.

8 (X'08')
LE/VSE control blocks could not be set up properly.

12 (X'0C')
System not supported.

16 (X'10')
The application's parameter list could not be processed correctly. The parameter list might be invalid.

20 (X'14')
Hardware not supported.

24 (X'18')
An error occurred when attempting to process the options specified in the application.

28 (X'1C')
Stack management could not allocate stack and/or heap storage.

32 (X'20')
Program management could not find a phase that was to be loaded.

36 (X'24')
When trying to load a phase, program management encountered a storage condition.

40 (X'28')
Program management could not be initialized properly.

42 (X'2A')
During initialization, a load request for CEEDOPT failed. Also see section "Programmer Response".

44 (X'2C')
The LE/VSE math library could not be initialized properly.

48 (X'30')
Condition management could not be initialized properly.

52 (X'34')
A language-specific event handler returned to initialization with a feedback code, causing immediate termination.

54 (X'36')
During initialization, a load request for the

abnormal termination exit table failed. If the abend occurs on a CICS system, the PHASE name is CEECXTAN. If the abend occurs in a BATCH program, the PHASE name is CEEBXTAN. Also see section “Programmer Response”.

56 (X'38')
Unable to load LE/VSE initialization phase, CEEBINIT. Also see section “Programmer Response”.

60 (X'3C')
The initial fixed-size stack overflowed.

64 (X'40')
Process level ran out of storage.

68 (X'44')
Enclave level ran out of storage.

72 (X'48')
Thread level ran out of storage.

76 (X'4C')
CAA pointer became corrupted.

80 (X'50')
PCB pointer became corrupted.

82 (X'52')
EDB or EICB pointer has become corrupted or has been overlaid causing the pointer to become unusable or invalid. Also see section “Programmer Response”.

84 (X'54')
Assembler user exit malfunctioned.

88 (X'58')
Get heap malfunctioned during initialization.

92 (X'5C')
Anchor setup malfunctioned.

96 (X'60')
The PLIST run-time option conflicts with the operating system type.

100 (X'64')
The LE/VSE anchor support was unavailable.

108 (X'6C')
The routine was compiled with an unsupported version of a compiler.

112 (X'70')
A phase did not contain a main procedure/function.

116 (X'74')
The primary entry point routine of the root phase was found with LE/VSE CEESTART, but the rest of the routines in the phase were not linked with LE/VSE library.

124 (X'7C')
An unsupported parameter style was detected.

128 (X'80')
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.

132 (X'84')
Library routines required for CICS support are not defined in the CICS CSD. See *LE/VSE Customization Guide* for the library routines required for CICS support.

136 (X'88')
Reinitialization feature is not supported in PL/I-defined pre-initialization service.

138 (X'8A')
A module was compiled with DOS PL/I V1R6 (or earlier) has been link-edited to the main phase.

156 (X'9C')
An error occurred during initialization of the COBOL run-time environment. Also see section “Programmer Response”.

200 (X'C8')
During initialisation processing, LE/VSE was unable to establish any STXITs that are required for condition handling purposes. Also see section “Programmer Response”.

208 (X'D0')
During initialisation processing, LE/VSE was unable to establish the required STXIT PC exit for the level of condition handling requested via the TRAP runtime option. A STXIT PC exit was already in progress.

212 (X'D4')
During initialisation processing, LE/VSE was unable to establish the required STXIT AB exit for the level of condition handling requested via the TRAP runtime option. An OPTION=EARLY STXIT AB had already been established.

216 (X'D8')
During initialisation processing, LE/VSE was unable to establish the required STXIT AB exit for the level of condition handling requested via the TRAP runtime option. An ’EXIT AB’ was already in progress for the Enclave.

1000–1255 (X'3E8'–X'4E7')
Unable to load event handler for a *high-level language*. The name of the event handler phase that could not be loaded is CEEEV*nnn*, where *nnn* is the last 3 digits of the reason code.

identifier	**high-level language**
003	C
005	COBOL
010	PL/I

Programmer Response: *For reason codes 4, 28, 40, 56, and 1000-1255*, increase the amount of GETVIS storage in the partition by decreasing the value specified in the SIZE parameter of the EXEC JCL statement, or increase the partition size.

For reason code 42, ensure that you have created the LE/VSE installation wide default options module CEEDOPT and it exists in an available sub-library.

For reason code 54, ensure you have at least the supplied CEEBXTAN or CEECXTAN PHASEs in a sub-library available to the user application. Further information on the creation of CEEDOPT, CEEBXTAN and CEECXTAN is available in the *LE/VSE Customization Guide*.

For reason code 56, also ensure that the LE/VSE sublibrary is available in the LIBDEF search chain at run-time.

For reason code 82 check for any CICS storage violations or application programs modifying storage outside of their allocated area(s) such as CICS commareas or table arrays. Compile COBOL programs with SSRANGE and set LE/VSE runtime option CHECK(ON) to validate all allocated table arrays for any boundary violations. For PL/1 VSE programs you can use the SUBSCRIPTRANGE procedure setting. In both the PL/1 and COBOL cases, the use of TEST(NONE,SYM) as a compile time option is recommended to ensure maximum debugging information is available at abend time.

For reason code 156, see the accompanying COBOL run-time message for a description of the error.

For reason code 2xx, ensure that there are no other STXITs active, or entered, when LE/VSE is being initialized (first call to an Language Environment-enabled HLL program). Check for any Assembler programs or software that may be establishing STXIT exits prior to the LE/VSE environment being initialised. If the failure persists, report the problem to your systems programmer or IBM Technical Support Centre.

For other reason codes, see your system programmer.

System Action: Enclave terminated.

U4094 (X'FFE')

Explanation: An abend was issued during termination, when errors were detected.

Reason codes:

4 (X'04')
: An invalid parameter to termination services was discovered.

8 (X'08')
: A language-specific event handler returned an invalid return code.

12 (X'0C')
: A language-specific event handler returned to termination with a return code, causing immediate termination. This might be due to writing beyond storage.

16 (X'10')
: Condition management could not properly terminate.

20 (X'14')
: Program management could not properly terminate.

24 (X'18')
: Storage management could not properly free stack and/or heap storage.

28 (X'1C')
: Storage management could not properly free the initial storage allocation.

32 (X'20')
: The user stack was unable to be collapsed using GOTO.

36 (X'24')
: The fixed-size termination stack overflowed.

44 (X'2C')
: Termination requested during termination.

80 (X'50')
: An error occurred during termination of the COBOL run-time environment.

Programmer Response: For reason code 80, see the accompanying COBOL run-time message for a description of the error.

For other reason codes, see your system programmer.

System Action: Enclave terminated.

U4095 (X'FFF')

Explanation: An abend was issued as a response to the fatal return code of an LE/VSE-conforming language.

Reason codes:

<300 (<X'12C')
: The reason code is set to the LE/VSE-conforming language ID.

301 (X'12D')
: The condition was provoked from a user handler routine.

Programmer Response: See your system programmer.

System Action: Enclave terminated.

Chapter 14. Return Codes to CICS

When LE/VSE detects an error and LE/VSE is not fully initialized or unable to generate a message, the component of LE/VSE in charge generates a return code. The return code passes from the LE/VSE 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.

LE/VSE Return Codes

10000 - 10255

Explanation: There is no CEEEVnnn module. nnn is the member language ID for the modules, and is found by subtracting 10000 from the return code.

Programmer Response: Ensure the CSD definitions are correct for LE/VSE. Especially ensure that for every CEECInnn defined module there is a corresponding CEEEVnnn defined module.

System Action: CICS continues system initialization with LE/VSE inactive.

11000

Explanation: Invalid parameters passed from CICS to LE/VSE for the partition initialization call.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

System Action: CICS continues system initialization with LE/VSE inactive.

11010

Explanation: Storage could not be acquired by LE/VSE to initialize in the CICS partition.

Programmer Response: Increase the size of the CICS dynamic storage area (DSA) by increasing the value on the SIZE parameter of the EXEC DFHSIP JCL statement.

System Action: CICS continues system initialization with LE/VSE inactive.

11020

Explanation: Unable to load LE/VSE modules in order to initialize LE/VSE for the CICS partition.

Programmer Response: Make sure the CSD definitions (as supplied in the member CEECCSD.Z) are correct for LE/VSE. Also, make sure that the CICS partition size is large enough to run LE/VSE.

System Action: CICS continues system initialization with LE/VSE inactive.

11030

Explanation: LE/VSE partition initialization did not succeed in a language support module.

Programmer Response: LE/VSE can write other messages to the operator's console explaining the cause of the malfunction. If there are none, this is more than likely an internal error in LE/VSE.

System Action: CICS continues system initialization with LE/VSE inactive.

11040

Explanation: An internal abend has occurred during LE/VSE initialization for the CICS partition.

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 13, "LE/VSE Abend Codes," on page 329 for more information.

System Action: CICS continues system initialization with LE/VSE inactive.

11060

Explanation: During initialization, LE/VSE requested that CICS load module CEECOPT. CICS returned a 'load failed' response.

Programmer Response: Ensure you have a valid CEECOPT.PHASE in a sub-library available to the CICS system. Also ensure you have CEECOPT defined in the active GRPLIST, if using RDO, or defined in the active CICS PPT. This is supplied in the CEECCCSD.Z member which is resident in the LE/VSE Installation sub-library (PRD2.SCEEBASE).

System Action: CICS continues system initialization with LE/VSE inactive.

11100

Explanation: Invalid parameters passed from CICS to LE/VSE for the partition termination call.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

System Action: CICS continues system termination.

11110

Explanation: Unable to release LE/VSE modules during partition termination.

Programmer Response: This is most likely an internal error in LE/VSE.

System Action: CICS continues system termination.

11120

Explanation: Unable to free storage acquired at partition initialization during partition termination.

Programmer Response: This is most likely an internal error in LE/VSE.

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 LE/VSE.

System Action: CICS continues system termination.

11150

Explanation: An internal abend occurred during LE/VSE termination for the CICS partition.

Programmer Response: There should be a CEE1000S message in the operators console describing the abend code and reason code for the abend. See Chapter 13, "LE/VSE Abend Codes," on page 329 for more information.

System Action: CICS continues system termination.

12000

Explanation: Invalid parameters passed from CICS to LE/VSE for the thread initialization call.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

System Action: CICS abnormally terminates the transaction with abend code AEC7.

12020

Explanation: Preallocated storage was expected by LE/VSE from CICS for the thread work area, but was not supplied.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

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 LE/VSE.

System Action: CICS abnormally terminates the transaction with abend code AEC7.

12100

Explanation: Invalid parameters passed from CICS to LE/VSE for the thread termination call.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

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 LE/VSE.

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 LE/VSE.

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 LE/VSE.

System Action: CICS continues with termination of the transaction.

13000

Explanation: Invalid parameters passed from CICS to LE/VSE for the run unit initialization call.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

System Action: CICS abnormally terminates the transaction with abend code AEC8.

13010

Explanation: There was not enough preallocated storage by CICS to LE/VSE to complete initialization for all languages in the application routine.

Programmer Response: This is most likely an internal error in LE/VSE.

System Action: CICS abnormally terminates the transaction with abend code AEC8.

13020

Explanation: The mix of languages in the application phase is not supported by this release of LE/VSE.

Programmer Response: See *LE/VSE Writing Interlanguage Communication Applications* 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 LE/VSE.

System Action: CICS abnormally terminates the transaction with abend code AEC8.

13040

Explanation: An invalid application routine argument list passed by CICS to LE/VSE during run unit initialization.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

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 LE/VSE 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 dynamic storage area (DSA) by increasing the value on the SIZE parameter of the EXEC DFHSIP JCL statement.

System Action: CICS abnormally terminates the transaction with abend code AEC8.

13100

Explanation: Invalid parameters passed from CICS to LE/VSE for the run unit termination call.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

System Action: CICS continues with termination of the application.

13110

Explanation: The thread token passed by CICS to LE/VSE for run unit termination is invalid.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

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 LE/VSE.

System Action: CICS continues with termination of the application.

13140

Explanation: Unable to free storage for LE/VSE control blocks.

Programmer Response: This is most likely an internal error in LE/VSE.

System Action: CICS continues with termination of the application.

13200

Explanation: Invalid parameters passed from CICS to LE/VSE for the run unit invocation call.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

System Action: CICS abnormally terminates the transaction with abend code AEC9.

13210

Explanation: Preallocated storage was expected by LE/VSE from CICS for the run unit work area, but was not supplied.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

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 LE/VSE.

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 LE/VSE.

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 LE/VSE.

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 LE/VSE 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 LE/VSE for the run unit end invocation call.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

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 LE/VSE.

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 LE/VSE.

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 LE/VSE.

System Action: CICS continues with termination of the application.

15000

Explanation: Invalid parameters passed from CICS to LE/VSE for the establish ownership call.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

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 LE/VSE were not available for the language.

Programmer Response: Make sure the CEEEVnnn language support modules of LE/VSE 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: LE/VSE establish ownership did not succeed in a language support module.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

System Action: CICS abnormally terminates the transaction with abend code APCS.

15040

Explanation: The application phase does not contain a main routine.

Programmer Response: Make sure there is a main routine in the application phase.

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.

16000

Explanation: Invalid parameters passed from CICS to LE/VSE for the determine working storage call.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

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 LE/VSE.

System Action: CICS continues running the transaction under Execution Diagnostic Facility (EDF).

16040

Explanation: LE/VSE could not determine the working storage address and length for a routine.

Programmer Response: This is most likely an internal error in LE/VSE.

System Action: CICS continues running the transaction under Execution Diagnostic Facility (EDF).

17000

Explanation: Invalid parameters passed from CICS to LE/VSE for the perform goto call.

Programmer Response: This is most likely an internal error in CICS or LE/VSE.

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 LE/VSE.

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 LE/VSE.

System Action: CICS abnormally terminates the transaction with abend code APC2.

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 non-LE/VSE-conforming C applications.

Programmer Response: Recompile using LE/VSE-conforming 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: CEECI005 called INPL when application routine is LE/VSE-conforming.

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

51001

Explanation: A COBOL library subroutine that is system sensitive (a program that is part of the environment specific library) is inappropriate for the CICS system environment. This is likely an improper concatenation sequence of sublibraries that contain the subroutine library, either during the execution or the link-edit of the COBPACK. The message number IGZ0011C is written to the system console.

Programmer Response: Check the order of the concatenation of the sublibraries in the LIBDEF statement in the CICS startup job. Ensure that the PRD2.SCEECICS sublibrary is concatenated before the PRD2.SCEEBASE sublibrary.

System Action: CICS continues system initialization with LE/VSE inactive.

51002

Explanation: An attempt to load a run-time routine for COBOL support failed. The module was not available or a system load failure occurred. The message number IGZ0096C is written to the system console.

Programmer Response: Verify that the COBOL-specific run-time routines are defined in the PPT. Also verify that the sublibraries containing the COBOL-specific run-time routines are defined in the LIBDEF statement in the CICS startup job.

System Action: CICS continues system initialization with LE/VSE inactive.

51003

Explanation: CICS GETMAIN command failed. This is an internal error. The message number IGZ0099C is written to the system console.

Programmer Response: Contact IBM service personnel.

System Action: CICS continues system initialization with LE/VSE inactive.

51101

Explanation: CICS RELEASE command failed. This is an internal error. The message number IGZ0099C is written to the system console.

Programmer Response: Contact IBM service personnel.

System Action: CICS continues system termination.

51102

Explanation: CICS FREEMAIN command failed. This is an internal error. The message number IGZ0099C is written to the system console.

Programmer Response: Contact IBM service personnel.

System Action: CICS continues system termination.

52001

Explanation: CICS GETMAIN command failed. This is an internal error. The message number IGZ0099C is written to the system console.

Programmer Response: Contact IBM service personnel.

System Action: CICS abnormally terminates the transaction with abend code AEC7.

52101

Explanation: CICS FREEMAIN command failed. This is an internal error. The message number IGZ0099C is written to the system console.

Programmer Response: Contact IBM service personnel.

System Action: CICS continues with termination of the transaction.

53101

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.

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 LE/VSE 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 LE/VSE 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 LE/VSE. 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. Diagnosing Problems with LE/VSE

This appendix provides information for diagnosing problems in the LE/VSE product. It helps you determine if a correction for a product failure similar to yours has been previously documented. If the problem has not been previously reported, it tells you how to open a Problem Management Record (PMR) to report the problem to IBM, and if the problem is with an IBM product, what documentation you need for an Authorized Program Analysis Report (APAR).

Diagnosis Checklist

Step through each of the items in the diagnosis checklist below to see if they apply to your problem. The checklist is designed to either solve your problem or help you gather the diagnostic information required for determining the source of the error. It can also help you confirm that the suspected failure is not a user error; that is, it was not caused by incorrect usage of the LE/VSE product or by an error in the logic of the routine.

1. If your failing application contains programs that were changed since they last ran successfully, review the output of the compile or assembly (listings) for any unresolved errors.
2. If there have not been any changes in your applications, check the output (job or console logs, CICS transient (CESE) queues) for any messages from the failing run.
3. Check the message prefix to identify the system or subsystem that issued the message. This can help you determine the cause of the problem. Following are some of the prefixes and their respective origins.

   EDC The prefix for C, C utility, and prelinker messages. EDC5000-6999 are from the C run-time component of LE/VSE (except for EDC5500-5599, which are from the DSECT utility). EDC4000-4999 are from the prelinker and C utilities.

   IGZ The prefix for messages from the COBOL run-time component of LE/VSE.

   IBM The prefix for messages from the PL/I run-time component of LE/VSE.

   CEE The prefix for messages from the common run-time component of LE/VSE.
4. For any messages received, check for recommendations in the "Programmer Response" sections of the messages in this manual.
5. Verify that abends are caused by product failures and not by program errors. See the appropriate chapters in this manual for a list of LE/VSE-related abend codes.
6. Your installation may have received an IBM Program Temporary Fix (PTF) for the problem. Verify that you have received all issued PTFs and have installed them, so that your installation is at the most current maintenance level.
7. The preventive service planning (PSP) bucket, an online database available to IBM customers through IBM service channels, gives information about product installation problems and other problems. Check to see whether it contains information related to your problem.
8. Narrow the source of the error. If an LE/VSE dump is available, check the traceback in the LE/VSE dump for the source of the problem. Refer to

Chapter 3, "Using LE/VSE Debugging Facilities," on page 29 for information on how to generate and use an LE/VSE dump or a system dump to isolate the cause of the error.

If a system dump is taken, follow the save area chain to find out the name of the failing module and whether IBM owns it. See "Locating the Name of the Failing Routine in a System Dump" for information on finding the routine name.

9. After you identify the failure, consider writing a small test case that re-creates the problem. The test case could help you determine whether the error is in a user routine or in the LE/VSE product. Do not make the test case larger than 75 lines of code. The test case is not required, but it could expedite the process of finding the problem.

   If the error is not an LE/VSE failure, refer to the diagnosis procedures for the product that failed.
10. Record the conditions and options in effect at the time the problem occurred. Compile your program with the appropriate options to obtain an assembler listing and data map. If possible, obtain the linkage editor output listing. Note any changes from the previous successful compilation or run. For an explanation of compile-time options, refer to the compiler-specific programming guide.
11. If you are experiencing a no-response problem, try to force a dump: CANCEL the program with the dump option.
12. Record the sequence of events that led to the error condition and any related programs or files. It is also helpful to record the service level of the compiler associated with the failing program.

Locating the Name of the Failing Routine in a System Dump

If a system dump is taken, follow the save area chain to find out the name of the failing routine and whether IBM owns it. Following are the procedures for locating the name of the failing routine, which is the **primary entry point name**.

1. Find the entry point associated with the current save area. The entry point address (EPA), located in the previous save area at offset 16 (X'00000010'), points to it.
2. Determine the entry point type, of which there are three:

Entry point type is...	If...
LE/VSE conforming	The entry point plus 4 is X'00C3C5C5'.
C	The entry point plus 5 is X'000000CE'.
Nonconforming	The entry point is neither of the above. Nonconforming entry points are for routines that follow the linking convention in which the name is at the beginning of the routine. X'47F0Fxxx' is the instruction to branch around the routine name.

   For routines with LE/VSE-conforming and C entry points, LE/VSE provides program prolog areas (PPAs). PPA1 contains the entry point name and the address of the PPA2; PPA2 contains pointers to the timestamp, where release level keyword information is found, and to the PPA1 associated with the primary entry point of the routine.

   If the entry point type of the failing routine is LE/VSE-conforming, go to step 3 on page 349.

   If the entry point type is C, go to step 5 on page 350.

If the entry point type is nonconforming, go to step 6 on page 350.

3. If the entry point type is LE/VSE-conforming, find the entry point name for the LE/VSE or COBOL program.
   a. Use an offset of X'0000000C' from the entry point to locate the offset of the PPA1 relative from the entry-point address.
   b. In the PPA1, locate the offset to the length of the name.
   c. Add this offset to the PPA1 address to find the halfword containing the length of the name, followed by the entry point name.

      The entry point name appears in EBCDIC, with the translated version in the right-hand margin of the system dump.
4. Find the LE/VSE or COBOL program name.
   a. Find the address of the PPA2 at X'00000004' from the start o the PPA1.
   b. Find the address of the compilation unit's primary entry point at X'00000010' in the PPA2.
   c. Find the entry point name associated with the primary entry point as described above. The primary entry point name is the routine name.

   Figure 95 illustrates the LE/VSE-conforming PPA1.

hex	PPA1: Entry Point Block			
00	Offset to the length of name	X'CE' Language Environment signature	Language Environment flags	Member flags
04	A(PPA2)			
08	Offset to Block Debugging Information (BDI) from entry point or zero			
0C	Reserved			
10	Reserved			
	: : :			
	Length of name		Untruncated entry/label name	

Figure 95. LE/VSE PPA1

Figure 96 on page 350 illustrates the LE/VSE PPA2.

hex

PPA2: Compile Unit Block

hex				
00	Member Identifier	Member Subid	Member Defined	Control Level (=1)
04	V(CEESTART) for phase			
08	Offset from PPA2 to Compile Debugging Information (CDI) or zero			
0C	Offset from PPA2 to timestamp/version information or zero			
10	A(PEP) - address to the compilation unit's Primary Entry Point			
	: : :			

Figure 96. Compile Unit Block

5. If the entry point type is C, find the C routine name.
   a. Use the entry point plus 4 to locate the offset to the entry point name in the PPA1.
   b. Use this offset to find the length-of-name byte followed by the routine name.

      The routine name appears in EBCDIC, with the translated version in the right-hand margin.

   Figure 97 illustrates the C PPA1.

hex

C Routine Layout Entry and PPA1

hex				
00	B xxx(0,15) Branch around prolog data			
04	X'14' Offset to the name	X'CE' Language Environment signature	Language Environment flags	Member flags
08	A(PPA2)			
0C	A (Block Debugging Information (BDI) or zero			
10	Stack frame size			
	: : :			
yy	Length of name		Untruncated entry/label name	

Figure 97. C PPA1

6. If the entry point type is nonconforming, find the PL/I routine name.
   a. Find the one byte length immediately preceding the entry point. This is the length of the routine name.
   b. Go back the number of bytes specified in the name length. This is the beginning of the routine name.
7. If the entry point type is nonconforming, find the name of the routine other than PL/I.
   a. Use the entry point plus 4 as the location of the entry point name.

b. Use the next byte as the length of the name. The name directly follows the length of name byte. The entry point name appears in EBCDIC with the translated version in the right-hand margin.

Figure 98 illustrates the dump output of this type of DSA. In this example, LISTIT is the name of the module associated with this save area.

Nonconforming entry points that can appear do not necessarily follow this linking convention. The location of data in these save areas can be unpredictable.

```
020000  =  47F0F00C  06D3C9E2  E3C9E300  90ECD00C  E0B |.00..LISTIT.....|
020010  =  18CF41B0  C29850BD  000850DB  000418DB      |....Bq&...&.....|
020020  =  4510C052  E3E8D7D3  C9D54040  01020034      |....TYPLIN  ....|
020030  =  C200001E  C5D5E3C5  D940D5E4  D4C2C5D9      |B...ENTER NUMBER|
020040  =  40D6C640  D9C5C3D6  D9C4E240  D6D940C1      | OF RECORDS OR A|
020050  =  D3D30ACA  00020058  4510C06C  E6C1C9E3      |LL.........%WAIT|
020060  =  D9C44040  010202F0  E4000000  0ACA0002      |RD  ...0U.......|
```

Figure 98. Nonconforming Save Area with Sample Dump

Searching the IBM Software Support Database

Failures in the LE/VSE product can be described through the use of keywords. A keyword is a descriptive word or abbreviation assigned to describe one aspect of a product failure. A set of keywords, called a keyword string, describes the failure in detail. You can use a keyword or keyword string as a search argument against an IBM software support database, such as the Service Information Search (SIS). The database contains keyword and text information describing all current problems reported through APARs and associated PTFs. IBM Support Center personnel have access to the software support database and are responsible for storing and retrieving the information. Using keywords or a keyword string, they will search the database to retrieve records that describe similar known problems.

If you have IBMLink or some other connection to the IBM databases, you can do your own search for previously recorded product failures before calling the IBM Support Center.

If your keyword or keyword string matches an entry in the software support database, the search may yield a more complete description of the problem and possibly identify a correction or circumvention. Such a search may yield several matches to previously reported problems. Review each error description carefully to determine if the problem description in the database matches the failure.

If a match is not found, go to "Preparing Documentation for an Authorized Program Analysis Report (APAR)."

Preparing Documentation for an Authorized Program Analysis Report (APAR)

Prepare documentation for an APAR only after you have done the following:
- Eliminated user errors as a possible cause of the problem.
- Followed the diagnostic procedures.
- You or your local IBM Support Center has been unsuccessful with the keyword search.

Having met these criteria, follow the instructions below.

1. Report the problem to IBM.

   If you have not already done so, report the problem to IBM by opening a Program Management Record (PMR).

   If you have IBMLink or some other connection to IBM databases, you can open a PMR yourself. Or, the IBM Software Support Center can open the PMR after consulting with you on the phone. The PMR is used to document your problem and to record the work that the Support Center does on the problem. Be prepared to supply the following information:

   - Customer number
   - PMR number
   - Operating system
   - Operating system release level
   - Your current LE/VSE maintenance level (PTF list and list of APAR fixes applied)
   - Keyword strings you used to search the IBM software support database
   - Processor number (model and serial)
   - A description of how reproducible the error is:
     Can it be reproduced each time?
     Can it be reproduced only sometimes?
     Have you been unable to reproduce it?
     Supply source files, test cases, macros, subroutines, and input files required to re-create the problem. Test cases are not required, but are helpful.

   If the IBM Support Center concludes that the problem described in the PMR is a problem with the LE/VSE product, they will work with you to open an APAR, so the problem can be fixed.
2. Provide APAR documentation. When you submit an APAR, you will need to supply information that describes the failure. Table 45 describes how to produce documentation required for submission with the APAR.

Table 45. Problem Resolution Documentation Requirements

Item	Materials Required	How to Obtain Materials
1	Machine-readable source program, including macros, subroutines, input files, and any other data that might help to reproduce the problem.	IBM-supplied system utility program
2	Compiler listings: Source listing Object listing Storage map Traceback Cross-reference listing JCL listing and linkage editor listing Assembler-language expansion	Use appropriate compile-time options.
3	Dumps LE/VSE dump System dump	See instructions in Chapter 3, "Using LE/VSE Debugging Facilities," on page 29 (as directed by IBM support personnel).
4	Partition/region size/virtual storage size	
5	List of applied PTFs	System programmer
6	Operating instructions or console log	Application programmer

Table 45. Problem Resolution Documentation Requirements (continued)

Item	Materials Required	How to Obtain Materials
7	JCL statements used to invoke and run the routine, including all run-time options, in machine-readable form	Application programmer
8	System output associated with the MSGFILE run-time option.	Specify MSGFILE(SYSLST)
9	Contents of the applicable catalog	
10	A hardcopy log of the events leading up to the failure.	Print the LISTLOG.

3. Submit the APAR documentation.

   When submitting material for an APAR to IBM, carefully pack and clearly identify any media containing source programs, job stream data, interactive environment information, data sets, or libraries.

   All magnetic media submitted must have the following information attached and visible:

   - The APAR number assigned by IBM.
   - A list of data sets on the tape (such as source program, JCL, data).
   - A description of how the tape was made, including:
     - The exact JCL listing or the list of commands used to produce the machine-readable source. Include the block size, LRECL, and format of each file.
     - Labeling information used for the volume and its data sets.
     - The recording mode and density.
     - The name of the utility program that created each data set.
     - The record format and block size used for each data set

   Any printed materials must show the corresponding APAR number.

   The IBM service personnel will inform you of the mailing address of the service center nearest you.

   If an electronic link with IBM Service is available, use this link to send diagnostic information to IBM Service.

After the APAR is opened and the fix is produced, the description of the problem and the fix will be in the software support database in SIS, accessible through ServiceLink.

Language Environment Glossary

A

abend. Abnormal end of application.

active routine. The currently executing routine.

additional heap. An LE/VSE heap created and controlled by a call to CEECRHP. See also *below heap*, *anywhere heap*, and *initial heap*.

addressing mode. An attribute that refers to the address length that a routine is prepared to handle upon entry. Addresses may be 24 or 31 bits long.

AMODE. Addressing mode.

anywhere heap. The LE/VSE heap controlled by the ANYHEAP run-time option. It contains library data, such as LE/VSE control blocks and data structures not normally accessible from user code. The anywhere heap may reside above 16MB. See also *below heap*, *additional heap*, and *initial heap*.

Application writer interface. Synonym for *callable service*.

argument. An expression used at the point of a call to specify a data item or aggregate to be passed to the called routine.

array. An aggregate that consists of data objects, each of which may be uniquely referenced by subscripting.

assembler user exit. A routine to tailor the characteristics of an enclave prior to its establishment.

AWI. Application writer interface. Synonym for *callable service*.

B

below heap. The LE/VSE heap controlled by the BELOWHEAP run-time option, which contains library data, such as LE/VSE control block and data structures not normally accessible from user code. Below heap always resides below 16MB. See also *anywhere heap*, *initial heap*, and *additional heap*.

byte. The basic unit of storage addressability, usually with a length of 8 bits.

C

C language. A high-level language used to develop software applications in compact, efficient code that can be run on different types of computers with minimal change.

CAA. Common anchor area.

callable services. A set of services that can be invoked by an LE/VSE-conforming high level language using the conventional LE/VSE-defined call interface, and usable by all programs sharing the LE/VSE conventions.

Use of these services helps to decrease an application's dependence on the specific form and content of the services delivered by any single operating system.

call chain. A trace of all active routines and subroutines that can be constructed by the user from information included in a system dump, such as the locations of save areas and the names of routines.

caller. A routine that calls another routine.

CIB. Condition information block.

CICS. Customer Information Control System.

CLLE. COBOL load list entry.

COBCOM. Control block containing information about a COBOL/VSE partition.

COBOL. COmmon Business-Oriented Language. A high level language, based on English, that is primarily used for business applications.

COBOL load list entry (CLLE). Entry in the load list containing the name of the program and the load address.

COBVEC. COBOL vector table containing the address of the library routines.

common anchor area (CAA). Dynamically acquired storage that represents an LE/VSE thread. Thread-related storage/resources are anchored off the CAA. This area acts as a central communications area for the program, holding addresses of various storage and error-handling routines, and control blocks. The CAA is anchored by an address in register 12.

compilation unit. An independently compilable sequence of HLL statements. Each HLL product has different rules for what makes up a compilation unit. Synonym for *program unit*.

Compiler writer interfaces. A set of low-level interfaces used by LE/VSE-conforming HLL compilers to invoke LE/VSE services.

condition. An exception that has been enabled, or recognized, by LE/VSE and thus is eligible to activate user and language condition handlers. Any alteration to the normal programmed flow of an application. Conditions can be detected by the hardware/operating system and result in an interrupt. They can also be detected by language-specific generated code or language library code.

condition handler. A user-written condition handler or language-specific condition handler (such as a PL/I ON-unit) invoked by the LE/VSE *condition manager* to respond to conditions.

condition handling. In LE/VSE, the diagnosis, reporting, and/or tolerating of errors that occur in the run-time environment.

condition manager. Manages conditions in the common execution environment by invoking various user-written and language-specific *condition handlers*.

condition information block (CIB). The platform-specific data block used by the LE/VSE condition manager as a repository for data about conditions raised in the LE/VSE run-time environment.

condition token. In LE/VSE, a data type consisting of 96 bits (12 bytes). The condition token contains structured fields that indicate various aspects of a condition including the severity, the associated message number, and information that is specific to a given instance of the condition.

Customer Information Control System (CICS). CICS is an OnLine Transaction Processing (OLTP) system that provides specialized interfaces to databases, files and terminals in support of business and commercial applications.

CWI. Compiler writer interface.

D

data aggregate. A logical collection of data elements that can be referred to either collectively or individually; in PL/I, an array or a structure.

data division. In COBOL, the part of a routine that describes the files to be used in the program and the records contained within the files. It also describes any internal working storage records that are needed.

data type. The properties and internal representation that characterize data.

DCLCB. Declare control block.

DCT. Destination control table.

declare control block (DCLCB). Control block containing file information.

default. A value that is used when no alternative is specified.

descriptor. PL/I control block that holds information about a variable, such as area size, array bounds, or string length.

Destination Control Table (DCT). A CICS table that contains an entry for each extrapartition, intrapartition, and indirect destination. Extrapartition entries address data sets external to the CICS partition. Intrapartition destination entries contain the information required to locate the queue in the intrapartition data set. Indirect destination entries contain the information required to locate the queue in the intrapartition data set.

DSA. Dynamic storage area.

dynamic call. A call that results in the resolution of the called routine at run time. Contrast with *static call*.

dynamic storage. Storage acquired as needed at run time. Contrast with *static storage*.

dynamic storage area (DSA). An area of storage obtained during the running of an application that consists of a register save area and an area for automatic data, such as program variables. DSAs are generally allocated within LE/VSE-managed stack segments. DSAs are added to the stack when a routine is entered and removed upon exit in a last in, first out (LIFO) manner. In LE/VSE, a DSA is known as a *stack frame*.

E

EBCDIC. Extended binary-coded decimal interchange code.

enclave. In LE/VSE, an independent collection of routines, one of which is designated as the main routine. An enclave is roughly analogous to a program or run unit.

entry point. In assembler language, the address or label of the first instruction that is executed when a routine is entered for execution.

environment. A set of services and data available to a program during execution. In LE/VSE, environment is normally a reference to the run-time environment of HLLs at the enclave level.

exception. The original event such as a hardware signal, software detected event, or user-signaled event which is a potential condition. This action may or may not include an alteration in a program's normal flow. See also *condition*.

execution time. Synonym for *run time*.

execution environment. Synonym for *run-time environment*.

external data. Data that persists over the lifetime of an enclave and maintains last-used values whenever a routine within the enclave is reentered. Within an enclave consisting of a single phase, it is equivalent to COBOL external data.

F

facility ID. A string of three characters identifying an LE/VSE-conforming component. The facility IDs assigned by IBM are:

CEE	LE/VSE common library
EDC	C language-specific library
IGZ	COBOL language-specific library
IBM	PL/I language-specific library

FCB. File control block.

feedback code (fc). A condition token value. If you specify *fc* in a call to a callable service, a condition token indicating whether the service completed successfully is returned to the calling routine.

FIB. File information block.

file. A named collection of related data records that is stored and retrieved by an assigned name.

file control block (FCB). Block containing the addresses of I/O routines, information about how they were opened and closed, and a pointer to the file information block.

file information block (FIB). A read-only block describing the characteristics of an I/O file.

filename. A 1- to 7-character name used within an application and in JCL to identify a file. The filename provides the means for the logical file to be connected to the physical file.

function. A routine that is invoked by coding its name in an expression. The routine passes a result back to the invoker through the routine name.

H

handled condition. A condition that has proceeded. Either the user condition handler or the HLL condition handler has dealt with the condition and deemed the routine to continue.

heap. An area of storage used for allocation of storage whose lifetime is not related to the execution of the current routine. The heap consists of the initial heap segment and zero or more increments. See also *additional heap*, *anywhere heap*, *below heap*, *heap element*, and *initial heap*.

heap element. A contiguous area of storage allocated by a call to the CEEGTST service. Heap elements are always allocated within a single heap segment.

heap increment. See *increment*.

heap segment. A contiguous area of storage obtained directly from the operating system. The LE/VSE storage management scheme subdivides heap segments into individual heap elements. If the initial heap segment becomes full, LE/VSE obtains a second segment, or increment, from the operating system.

heap storage. See *heap*.

high level language (HLL). A programming language above the level of assembler language and below that of program generators and query languages.

HLL. High level language.

hook. The location in a compiled program where the compiler inserts an instruction that allows the user to later interrupt the program (by setting breakpoints) for debugging purposes.

I

ILC. Interlanguage communication.

increment. The second and subsequent segments of storage allocated to the stack or heap.

initial heap. The LE/VSE heap controlled by the HEAP run-time option and designated by a *heap_id* of 0. The initial heap contains dynamically allocated user data. See also *additional heap*.

instance specific information (ISI). Located within the LE/VSE condition token, the ISI contains information used by the condition manager to identify and react to a specific occurrence of a condition.

interlanguage communication (ILC). The ability of routines written in different programming languages to communicate. ILC support allows the application writer to readily build applications from component routines written in a variety of languages.

interrupt. A suspension of a process, such as the execution of a computer program, caused by an event external to that process, and performed in such a way that the process can be resumed.

ISI. Instance specific information.

J

JCL. Job control language.

job control language (JCL). A sequence of commands used to identify a job to an operating system and to describe a job's requirements.

L

Language Environment. A set of architectural constructs and interfaces that provides a common run-time environment and run-time services to applications compiled by Language Environment-conforming compilers.

Language Environment for VSE/ESA. An IBM software product that is the implementation of Language Environment on the VSE platform.

LE/VSE. Short form of Language Environment for VSE/ESA.

library. A collection of functions, subroutines, or other data.

library vector table (LIBVEC). A vector table used to support access to library routines (LE/VSE and HLLs) from compiler-generated code, user-written assembly language code, and other subroutines.

LIBVEC. Library vector table

linkage editor. A program that resolves cross-references between separately assembled object modules and then assigns final addresses to create a single relocatable phase. The linkage editor then stores the phase in a program library in main storage.

locator. PL/I control block that holds the address of data such as structures or arrays and the address of the *descriptor*.

LWS. Library workspace.

M

module. A language construct that consists of procedures or data declarations and can interact with other such constructs. In PL/I, an external procedure.

multithreading. Mode of operation that provides for the concurrent, or interleaved, execution of two or more tasks, or threads.

N

NAB. Next available byte.

next available byte (NAB). The address of the next available byte of storage on a doubleword boundary. This address is a segment of stack storage.

non-reentrant. A type of program that cannot be shared by multiple users.

O

object code. Output from a compiler or assembler which is itself executable machine code or is suitable for processing to produce executable machine code.

offset. The number of measuring units from an arbitrary starting point in a record, area, or control block, to some other point.

operating system. Software that controls the running of programs; in addition, an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

P

parameter. Data items that are received by a routine.

partition. A fixed-size division of storage.

percolate. The action taken by the condition manager when the returned value from a condition handler indicates that the handler could not handle the condition, and the condition will be transferred to the next handler.

picture string. Character strings used to specify date and time formats.

PPA1 entry point block. Program Prolog Area. This block contains information about the compiled module.

PPA2 entry point block. An extension of the *PPA1 entry point block*.

PPT. Processing Program Table

process. The highest level of the LE/VSE program management model. A process is a collection of resources, both program code and data, and consists of at least one enclave.

Processing Program Table (PPT). A CICS table that contains information about CICS phases (whether the phase is in storage or not, its language, use count and entry point address, etc.) needed to complete a transaction.

program mask. A structure that describes the manner in which S/370 hardware-detected conditions are to be handled.

program status word (PSW). An area in storage used to indicate the order in which instructions are executed and to hold and indicate the status of the operating system. The program mask (bits 20 to 23) of the PSW can be manipulated to enable or disable the detection of some hardware conditions under LE/VSE.

program unit. Synonym for *compilation unit*.

PSW. Program status word.

R

reason code. A value returned to the invoker of an enclave that indicates how the enclave terminated. The value reflects whether the enclave terminated successfully, or unsuccessfully, to an unhandled condition. Under CICS, synonymous with *return code*.

reentrant. The attribute of a routine or application that allows more than one user to share a single copy of a phase.

register. To specify formally. In LE/VSE, to register a condition handler means to add a user-written condition handler onto a routine's stack frame.

register save area (RSA). Area of main storage in which contents of registers are saved.

return code. A code produced by a routine to indicate its success. It may be used to influence the execution of succeeding instructions.

routine. In LE/VSE, refers to a procedure, function, or subroutine.

RMODE. Residence mode. The attribute of a phase that specifies whether the phase, when loaded, must reside below the 16MB virtual storage line or may reside anywhere in virtual storage.

RSA. Register save area.

RUNCOM. COBOL block containing the ID and address of the main program.

run time. Any instant at which a program is being executed. Synonym for *execution time*.

run-time environment. A set of resources that are used to support the execution of a program. Synonym for *execution environment*.

run unit. One or more object programs that are executed together. In LE/VSE, a run unit is the equivalent of an *enclave*.

S

save area. Area of main storage in which contents of registers are saved.

scope terminator. Variable at the end of a statement.

severity code. A part of run-time messages that indicates the severity of the error condition (1, 2, 3, or 4).

stack. An area of storage used for suballocation of stack frames. Such suballocations are allocated and freed on a LIFO (last in, first out) basis. A stack is a collection of one or more stack segments consisting of an initial stack segment and zero or more increments.

stack frame. The physical representation of the activation of a routine. The stack frame is allocated on a LIFO stack and contains various pieces of information including a save area, condition handling routines, fields to assist the acquisition of a stack frame from the stack, and the local, automatic variables for the routine. In LE/VSE, a stack frame is synonymous with *DSA*.

static call. A call that results in the resolution of the called program statically at link-edit time. Contrast with *dynamic call*.

static storage. Storage that persists and retains its value across calls. Contrast with *dynamic storage*.

suboption. An option that can be used with compile-time and run-time options to further specify the action of the option.

symbolic feedback code. The symbolic representation of the 12-byte condition token returned by LE/VSE callable services. Symbolic feedback codes are provided so that you do not have to code the entire 12-byte condition token in a condition handling routine.

subsystem. A secondary or subordinate system, or programming support, usually capable of operating independently of or asynchronously with a controlling system. Example: CICS.

system abend. An abend caused by the operating system's inability to process a routine; may be caused by errors in the logic of the source routine.

T

task global table (TGT). Table with information about addresses and length of working storage and the program start address.

TGT. Task global table

THDCOM. Control block with COBOL/VSE thread information.

thread. The basic run-time path within the LE/VSE program management model. It is dispatched by the system with its own instruction counter and registers. The thread is where actual code resides.

token. See *condition token*.

trace. (1). A record of the execution of a computer program. It exhibits the sequence in which the instructions were executed. (2). To record a series of events as they occur.

traceback. A section of the dump that provides information about the stack frame (DSA), the program unit address, the entry point of the routine, the statement number, and status of the routines on the call-chain at the time the traceback was produced.

transient data queue. A file to which run-time messages are written under CICS. Under LE/VSE, the name of this file is CESE.

U

unhandled condition. A condition that is percolated beyond the first routine's stack frame. Compare with *handled condition*.

user abend. A request made by user code to the operating system to abnormally terminate a routine. Contrast with *system abend*.

user-written condition handler. A routine established by the CEEHDLR callable service to handle a condition or conditions when they occur in the common run-time environment. A queue of user-written condition handlers established by CEEHDLR may be associated with each stack frame in which they are established.

user exit. A routine that takes control at a specific point in an application. Two assembler user exits and one HLL user exit are provided by LE/VSE. They are invoked to perform initialization functions and both normal and abnormal termination functions.

V

virtual origin. The address of an element in an array whose subscripts are all zero.

VO. Virtual origin.

VSE (Virtual Storage Extended). A system that consists of a basic operating system (VSE/Advanced Functions) and IBM-supplied programs required to meet the data processing needs of a user.

W

word. A contiguous series of 32 bits (4 bytes) in storage, addressable as a unit. The address of the first byte of a word is evenly divisible by four.

Index

Readers' Comments — We'd Like to Hear from You

IBM Language Environment for VSE/ESA
Debugging Guide and Run-Time Messages
Version 1 Release 4 Modification Level 4

Publication No. SC33-6681-06

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SC33-6681-06

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

Fold and Tape **Please do not staple** Fold and Tape

SC33-6681-06

Cut or Fold
Along Line

Program Number: 5686-CF7

Printed in USA

SC33-6681-06

Spine information:

LE/VSE

Debugging Guide and Run-Time Messages

Version 1
Release 4 Modification
Level 4

SC33-6681-06