IBM Library Server Print Preview

DOCNUM = SC26-8058-00
DATETIME = 02/09/95 20:11:36
BLDVERS = 1.2
TITLE = PL/I VSE Diagnosis Guide
AUTHOR =
COPYR = © Copyright IBM Corp. 1964, 1995
PATH = /home/webapps/epubs/htdocs/book

COVER Book Cover

IBM PL/I for VSE/ESA

Diagnosis Guide

Release 1

Document Number SC26-8058-00

Program Number
5686-069

File Number S370-37

ABSTRACT Abstract

This book describes how to diagnose failures in the PL/I VSE compiler. It aids you first in determining the problem, then in describing the problem, and if necessary, in reporting the problem.

NOTICES Notices

```
 ___ Note! __________________________________________________________
|                                                                    |
| Before using this information and the product it supports, be sure |
| to read the general information under "Notices" in topic FRONT_1.  |
|                                                                    |
|____________________________________________________________________|
```

EDITION Edition Notice

First Edition (April 1995)

This edition applies to Version 1 Release 1 of IBM PL/I for VSE/ESA, 5686-069, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address 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 Corporation, Department J58
P.O. Box 49023
San Jose, CA, 95161-9023
United States of America

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.

**© Copyright International Business Machines Corporation 1964, 1995.
All rights reserved.**
Note to U.S. Government Users -- Documentation related to restricted rights -- Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

CONTENTS Table of Contents

TABLES Tables

FIGURES Figures

FRONT_1 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, 500 Columbus Avenue, Thornwood, NY 10594, U.S.A.

Subtopics:

- FRONT_1.1 Trademarks

FRONT_1.1 Trademarks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

AD/Cycle	Language Environment
BookManager	OS/2
C/370	RETAIN

CICS	SAA
IBM	VSE/ESA
IBMLink	

FRONT_2 About this book

This book tells you how to diagnose failures in the IBM* PL/I for VSE/ESA* (PL/I VSE) compiler. Because PL/I VSE uses IBM Language Environment* for VSE/ESA (LE/VSE) as its run-time environment, and uses certain run-time routines during compilation, consult the *LE/VSE Diagnosis Guide* or the *LE/VSE Debugging Guide and Run-Time Messages* to diagnose product failures you encounter in the run-time environment.

The book assumes that you have already determined that the suspected failure is not a user error; that is, it was not caused by incorrect usage of PL/I VSE or by an error in the logic of the application program. In the cases where a user error is the cause of the problem, consult the *PL/I VSE Programming Guide* or the *PL/I VSE Language Reference* for more information.

This book 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, Chapter 12, "Preparing an APAR" in topic 3.1 explains how to prepare an Authorized Program Analysis Report (APAR).

Most of the information in this book is to be used across operating systems. Information that is unique to an operating system is identified as such. In this book, VSE refers to Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA).

Subtopics:

FRONT_2.1 Who this book is for

This book is for system programmers, application programmers, and IBM support personnel who are involved in PL/I VSE product diagnosis. Prerequisite knowledge for using this book is:

- A general understanding of your operating system
- Some knowledge of the PL/I VSE language and options

FRONT_2.2 Using your documentation

The publications provided with PL/I VSE are designed to help you do PL/I programming under VSE. Each publication helps you perform a different task.

Subtopics:

FRONT_2.2.1 Where to look for more information

For information about the PL/I VSE library, see Table 1.

Table 1. How to use the publications you receive with PL/I VSE

To...	**Use...**
Evaluate the product	Fact Sheet
Understand warranty information	*Licensed Program Specifications*
Install the compiler	Installation and Customization Guide
Understand product changes and adapt programs to PL/I VSE	Migration Guide
Prepare and test your programs and get details on compiler options	Programming Guide
Get details on PL/I syntax and specifications of language elements	Language Reference Reference Summary
Diagnose compiler problems and report them to IBM	*Diagnosis Guide*
Get details on compile-time messages(1)	Compile-Time Messages and Codes

Note:

1. For details on run-time messages, see the LE/VSE library.

You might also require information about IBM* Language Environment* for VSE/ESA* (LE/VSE). For information about the LE/VSE library, see Table 2.

Table 2. How to use the publications you receive with LE/VSE

To...	**Use...**
Evaluate Language Environment	*Fact Sheet* *Concepts Guide*
Install LE/VSE	*Installation and Customization Guide*
Understand the LE/VSE program models and concepts	*Concepts Guide* *Programming Guide*

Prepare your LE/VSE-conforming applications and find syntax for run-time options and callable services	*Programming Guide* *Reference Summary*
Debug your LE/VSE-conforming application and get details on run-time messages	*Debugging Guide and Run-Time Messages*
Diagnose problems that occur in your LE/VSE-conforming application	*Diagnosis Guide*
Understand warranty information	*Licensed Program Specifications*

For the complete titles and order numbers of these and other related publications, see the "Bibliography" in topic BIBLIOGRAPHY.

FRONT_2.3 What is new in PL/I VSE

This is a major new release of PL/I, containing many new features and facilities. It brings to VSE many of the functions of the MVS & VM version of PL/I (IBM SAA* AD/Cycle* PL/I MVS & VM), while retaining close source compatibility with the DOS PL/I Optimizing Compiler (DOS PL/I).

PL/I VSE enables you to integrate your PL/I applications into IBM Language Environment for VSE/ESA (LE/VSE). In addition to PL/I's already impressive features, you gain access to LE/VSE's rich set of library routines and enhanced interlanguage communication (ILC) with IBM COBOL for VSE/ESA (COBOL/VSE).

Subtopics:

- FRONT_2.3.1 IBM Language Environment for VSE/ESA support
- FRONT_2.3.2 Usability enhancements
- FRONT_2.3.3 Extended addressing enhancements

FRONT_2.3.1 IBM Language Environment for VSE/ESA support

PL/I VSE provides the following functions in the LE/VSE area:

Interlanguage communication (ILC) support:

° Object code produced by PL/I VSE Release 1 can be linked with object code produced by other LE/VSE-conforming compilers (currently only COBOL/VSE).

- PL/I VSE programs can fetch COBOL/VSE phases.

- COBOL/VSE programs can fetch PL/I VSE phases.

Note: PL/I VSE does not support ILC with:

- FORTRAN
- RPG
- DOS/VS COBOL
- C/370*

Limited ILC support is provided for VS COBOL II at Release 3.2 or later.

Common support for multiple operating environments:

- Some of the restrictions on PL/I coding in the CICS* environment have been lifted.

- Procedure OPTIONS option FETCHABLE can be used to specify the procedure that gets control within a fetched phase.

- CEETDLI is supported in addition to PLITDLI and EXEC DLI.

- LE/VSE services provide storage management and condition handling support, as well as PLIDUMP and MSGFILE support for PL/I messages and other output.

- By default, only user-generated output is written to SYSLST. All run-time generated messages are written to MSGFILE.

- ERROR conditions now get control of all system abends. The PL/I message is issued only if there is no ERROR on-unit or if the ERROR on-unit does not recover from the condition via a GOTO.

- Selected items from PL/I Package/2 (the PL/I product for OS/2*) are implemented to allow better coexistence.

  - Limited support of OPTIONS(BYVALUE and BYADDR)

  - Limited support of EXTERNAL(environment-name) allowing alternate external names

  - Limited support of OPTIONAL arguments/parameters

  - Support for %PROCESS statement

  - NOT and OR compiler options

Product packaging:

- All PL/I VSE resident library routines are now packaged with LE/VSE, and are loaded at run time rather than link-edited with the application program. Changes to the resident library no longer require PL/I programs to be re-linked.

- At link-edit time, you have the option of getting math results that are compatible with LE/VSE or with DOS PL/I.

- Installation enhancements are provided to ease product installation and migration.

For migration considerations, see the *PL/I VSE Migration Guide.*

FRONT_2.3.2 Usability enhancements

These enhancements expand the PL/I language statements and options, PL/I data types, and compiler options, to make the language easier to use.

Enhanced double-byte character set (DBCS) support: This support introduces many enhancements that facilitate processing of GRAPHIC and mixed-character data and allows the source of the PL/I program to be in DBCS and/or the single-byte character set (SBCS), rather than only in SBCS.

Hexadecimal data constants: Constants for bit and character data can now be defined in hexadecimal notation, such that each *character* (0-9 and A-F) represents 4 bits.

Interface improvements for all (sub)systems: A new compiler option, SYSTEM, lets the programmer specify the target operating environment (of the generated object code), and the format of the parameters for the MAIN procedure.

Specification of compile-time options: You can specify compile-time options on the *PROCESS statement, a new %PROCESS statement, and in the PARM option of the EXEC IEL1AA JCL statement.

Linking after errors: The COMPILE compile-time option has been enhanced to allow linking to proceed after a severe error.

Run-time options: You can specify program run-time options in the PARM option of the EXEC JCL statement. PL/I VSE and LE/VSE will use these to control the execution of PL/I programs.

Passing parameters to the MAIN procedure: VSE JCL can also be used to pass a parameter to the MAIN PL/I procedure. A slash (/) separates the run-time options from the program parameter.

OPEN statement enhancements:

- There are new parameters on the PL/I OPEN statement that allow additional file attributes to be specified at file open time. These attributes are added to those in the file declaration.
- A vendor exit on the PL/I OPEN statement can be used to change the system logical unit number of the PL/I spill file.
- Data set name sharing for VSAM files, using the DSN option of the ENVIRONMENT attribute.

Date and time enhancements: A new built-in function, DATETIME, returns consistent date and time, including the four-digit year.

PL/I statement numbering options: A new compiler option, NUMBER, specifies that PL/I statement numbers will be derived from the sequence numbers in the program source deck, instead of being allocated sequentially.

Dynamic loading of external procedures: PL/I now supports the FETCH and RELEASE statements, to load external procedures into main storage at run time instead of having them link-edited with the MAIN procedure. (If these external procedures are PL/I, they must be compiled with PL/I VSE.)

New I/O facilities: PL/I VSE provides the following new I/O facilities:

- Support for REGIONAL(2) files
- Support for V and VS formats on REGIONAL(3) files
- Support for DELETE statement on REGIONAL files
- Support for multitrack search on REGIONAL files, using the LIMCT option of the ENVIRONMENT attribute
- Support for VSAM variable-length relative-record data sets (VRDS)
- Support for V format on consecutive unbuffered files

- Support for VS and VBS formats on consecutive buffered files

System programmer functions: A number of significant new features enhance PL/I as a system programming language:

- Support of additional program execution environments.

  PL/I can now be used for some system exit routines, such as the LE/VSE initialization exit.

- Additional support for pointers.

  PL/I built-in functions are now available to perform extended operations on pointers, including pointer arithmetic.

- Additional support for entry variables.

  A new built-in function and pseudovariable, ENTRYADDR, allows programmers to manipulate entry point addresses of procedures.

FRONT_2.3.3 Extended addressing enhancements

These enhancements exploit the large amounts of storage available in the VSE/ESA environment, making programming easier.

Addressing mode: PL/I VSE programs can be link-edited with AMODE(31) and RMODE(ANY).

Location of variables: PL/I variables can now be located above the 16-megabyte line.

Fullword array subscripts: Array bounds can now be in the range -2(31) (-2,147,483,648) through +2(31)-1 (+2,147,483,647). The associated built-in functions (such as LBOUND and HBOUND) now return FIXED BINARY(31) values.

AREA and aggregate sizes: An AREA can now have a maximum size of 2,147,483,647 (2(31)-1) bytes.

An aggregate can now have a maximum size of 2,147,483,647 (2(31)-1) bytes. For unaligned BIT arrays and aggregates that contain any unaligned BIT data (arrays or non-arrays), the maximum size is 268,435,455 (2(28)-1) bytes.

These numbers include any control information bytes that might be needed.

1.0 Part 1. Determining the problem

Subtopics:

1.1 Chapter 1. Compiler overview

This chapter includes an overview of the PL/I VSE compiler and its place in the processing of a PL/I program.

Subtopics:

1.1.1 Function of the PL/I VSE compiler

The PL/I VSE compiler analyzes source programs written in the PL/I language and translates these source statements into a series of machine instructions that form an object module. The compiler operates as a problem-state program under the operating system.

1.1.2 Processing a PL/I program

Figure 1 and Figure 2 show the process through which a PL/I program passes from compilation to use.

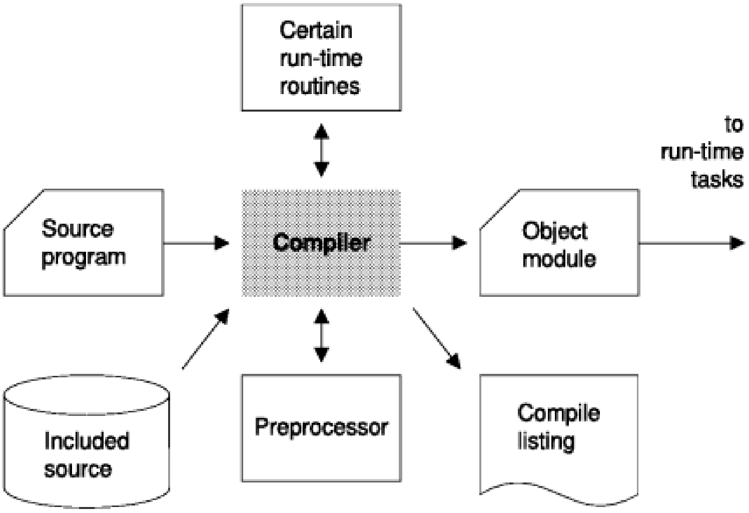

Figure 1. Compile-time tasks

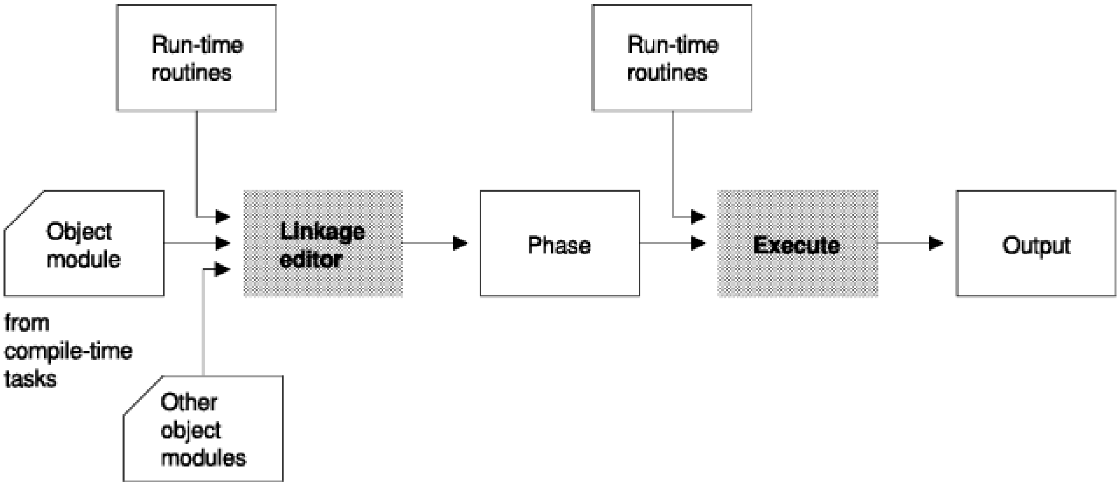

Figure 2. Run-time tasks

There are four stages in the process:

1. Writing: Coding the program and preparing it for the computer.

2. Compiling: Translating the program into machine instructions (that is, creating an object module).

3. Link-editing: Producing a phase from the object module. This includes linking the compiled code with run-time library modules, and possibly with other compiled programs. It also includes resolving the addresses within the code.

4. Running: Executing the phase.

The process is not necessarily a continuous one. The program may, for example, be kept in a compiled or link-edited form before it is run, and also be run a number of times once compiled.

1.1.3 Compiling

Compiling is the process of translating a PL/I program into machine instructions. This is done by associating PL/I statements with addresses in storage and translating executable PL/I statements into a series of machine instructions.

Subtopics:

1.1.3.1 Preprocessing stage

The compiler receives the source program either directly or through a preprocessor stage (see Figure 1 in topic 1.1.2). The preprocessor can modify source statements in the program or insert additional source statements in the program before compilation begins. You invoke the preprocessor by specifying the compile-time option MACRO. If the compiler detects an error or the possibility of an error during the preprocessor stage, it prints a message on the pages following the input listing. Thus, there are two sets of messages: one for the preprocessor and one for the compiler. Details of preprocessor and compile-time messages are given in *PL/I VSE Compile-Time Messages and Codes.*

1.1.3.2 Compiling stage

Under the control of the OPTIMIZE compile-time option, the compiler optimizes code by automatically altering the sequence of statements or operations, creating a more efficient object program. Some compile-time options useful in determining problems are shown in Table 3. For more information about these options, see the *PL/I VSE Programming Guide.*

Table 3. Compile-time options helpful in determining problems

Documentation needed	**Compile-time option to use**
Source listing	SOURCE
Cross-reference listing	XREF
Attribute table	ATTRIBUTES
Aggregate table	AGGREGATE

Storage table	STORAGE
Compile-time options	OPTIONS
Offset address	OFFSET
Object listing	LIST
Static-storage map	MAP
Statement in error	GOSTMT or GONUMBER
Diagnostic message list	FLAG(I)
Margins of source list	MARGINI
External symbol dictionary list	ESD
Long-form messages	LMESSAGE
Preprocessor information	INSOURCE MDECK MACRO
Block and do-group statement listing	NEST
Sequential statement listing	STMT
Debugging information about compiled code	TEST GOSTMT or GONUMBER

1.1.4 Link-editing

Link-editing links the compiler-generated object code with external modules requested by the compiled program. These are run-time library routines and possibly modules produced by other compilations. As well as linking the external modules, the linkage editor also resolves addresses within the object module. See the *LE/VSE Programming Guide* for a detailed discussion of link-editing.

1.1.5 Running

The PL/I compiler produces code that requires a special arrangement of control blocks and registers to run correctly. This arrangement of control blocks and registers is the *run-time environment*. Execution consequently is a three-stage process:

1. Set up the environment. (The run-time initialization routines handle this.)

2. Run the program.

3. Complete the job after the run. (This consists of closing any files left open and returning control either to the supervisor or to a calling module. The termination routines handle this.)

See the *LE/VSE Programming Guide* for a detailed discussion of how to run your PL/I program.

1.2 Chapter 2. Compile-time problem determination chart

This chapter contains the compile-time problem determination chart.

If you have already made a preliminary diagnosis of your problem and are familiar with PL/I, you can use the chart index in Table 4 to find the information you need.

Table 4. Compile-time problem determination index

Compile-time subject covered	**Block number**
Abend or program check	26
Message	28
Loop	32
Wait	34
Unusual or unexpected output	35
Performance	37

If you are just beginning your problem diagnosis, start by using the chart in Table 5. Begin with Block 1 and answer the question or perform the action specified, then go to the block indicated by the answer. Some blocks describe an action to be performed and also direct you to the next block.

Table 5. Compile-time problem determination chart

Block number	**Question**	**Answer**	**Go to block**
1	Is this a compile-time failure or a run-time failure?	Compile-time	2
	Note: The compiler requires access to LE/VSE during compilation. If the needed run-time routines are not available during compilation, compiler message IEL0995I is issued. Errors occurring in these run-time library routines are always intercepted by the compiler and reported using an associated compiler message. If your compiler message includes a run-time message number, you need to use this chart **and** refer to the *LE/VSE Debugging Guide and Run-Time Messages*.	Run-time	See the *LE/VSE Debugging and Run-Time Messages Guide*

2	Is this a U-level message?	Yes No	8 3
3	Is this a problem relating to a message?	Yes No	8 4
4	Is this a loop?	Yes No	8 5
5	Is this a wait?	Yes No	8 6
6	Does the compilation result in some type of unusual or unexpected output?	Yes No	8 7
7	Is this a performance problem?	Yes No	8 24
8	Has the program ever compiled before?	Yes No	9 11
9	Has anything in the environment changed? (Source changes, release level, maintenance fixes, compile-time options, and so on.)	Yes No	12 10
10	Is the entry from: U-level message Message other than U-level Loop Wait Unusual or unexpected output Performance **Note:** If you are here via the *environment changed* route, follow the major symptom code being experienced.	 Yes Yes Yes Yes Yes Yes	 26 28 32 34 35 37
11	Make sure PL/I coding rules were followed. Check and correct any statements causing E- or S-level messages. Check and correct any source statements causing W- or I-level messages that might relate to the problem. If you are using the OPTIMIZE(TIME) or OPTIMIZE(2) compile option, re-compile using NOOPTIMIZE. Is the problem circumvented?	Yes No	41 10
12	Has the source code of the program changed? (This includes compile-time options.)	Yes No	15 13
13	Was any maintenance applied? (PTFs, fixes)	Yes No	17 14
14	Has the release level changed?	Yes No	18 25
15	Check and correct any source statements causing E or S-level messages. Check and correct any source statements causing W- or I-level messages that might relate to the problem. Check and correct any compile-time options causing messages to be issued. Be critical of source changes. Is the problem solved?	Yes	END

	Note: If the problem is solved, but you feel the message was generated in error, follow the NO path.	No	16
16	If you are using the OPTIMIZE(TIME) or OPTIMIZE(2) compile-time option, re-compile using NOOPTIMIZE. Is the problem circumvented?	Yes No	41 10
17	Are the fixes or PTFs installed correctly? In other words, were there any system messages while installing and link-editing? Search IBMLink or INFO/ACCESS, or ask the IBM Support Center Level 1 to search RETAIN* for possible PTF errors in the form PExxxxx.	Yes No	19 20
18	Ensure that the release has been installed correctly. (Search IBMLink or INFO/ACCESS, or ask the IBM Support Center Level 1 to search RETAIN for any PTFs and errors applicable to this release. Search for PTF errors in the form: PExxxxx.) Were there any MSHP error messages or system messages while installing the release?	Yes No	19 21
19	Search IBMLink, INFO/ACCESS (Level 1), or have the IBM Support Center search RETAIN. Information about conducting a search is in Chapter 10, "Using the keyword string to search for corrections" in topic 2.8. Any hits?	Yes No	22 10
20	Reinstall the PTF or fix and test. Is the problem solved?	Yes No	END 23
21	Reinstall the release level correctly, plus any PTFs or fixes that apply, and test. Is the problem solved?	Yes No	END 23
22	Apply applicable fixes from RETAIN and test. Is the problem solved?	Yes No	END 23
23	Have the symptoms changed?	Yes No	2 10
24	Something was probably overlooked. A failure occurred, and it was one of the previously mentioned items. Review the compiler output again. If the problem does not fit any of the stated symptoms, go to Block 41.		41
25	Something was probably changed. Carefully determine what has changed with regard to this program. Is it:		
	Source code changes? This includes changes to the compile-time options, to INCLUDE files, and to different	Yes	15

	compile-time messages from the last time the program was successful.		
	Maintenance? This includes all APAR and PTF fixes to PL/I, fixes to LE/VSE that relate to PL/I, and fixes to the operating system that relate to the problem.	Yes	17
	Release-level changes? This includes changes in the release level of PL/I, LE/VSE, or the operating system.	Yes	18
	None of the above?	Yes	41
26	Compiler abends and program checks produce one of the following messages: IEL0001I U PREPROCESSOR ERROR NUMBER n DURING PHASE p IEL0230I U COMPILER ERROR NUMBER n DURING PHASE p IEL0970I U COMPILER CANNOT PROCEED. ERROR n DURING PHASE p. CORRECT SOURCE AND RE-COMPILE **Note:** Details of compiler error numbers and any recommended programmer actions are in PL/I VSE Compile-Time Messages and Codes. If the actions described do not solve the problem, go to Block 27.		27
27	Search IBMLink, INFO/ACCESS (Level 1), or have the IBM Support Center search RETAIN using the component ID 5686069, the error number from the message, and the phase ID from the message. Information about conducting a search is in Chapter 10, "Using the keyword string to search for corrections" in topic 2.8. Any hits?	Yes No	39 41
28	Is this an E- or S-level diagnostic message?	Yes No	29 30
29	Look up the message in PL/I VSE Compile-Time Messages and Codes. These messages indicate the compiler-detected error conditions in the source statements. Compilation can be complete, but the object program might not run correctly. Check and correct any statements in error. Re-run the program. Does the message still occur?	Yes No	30 END
30	If you are using the OPTIMIZE(TIME)		

	or OPTIMIZE(2) compile-time option, re-compile with NOOPTIMIZE. Is the problem circumvented?	Yes No	41 31
31	Search IBMLink, INFO/ACCESS or have the IBM Support Center search RETAIN using component ID 5686069 and MSGIELxxxxI. Information about doing your search is in Chapter 10, "Using the keyword string to search for corrections" in topic 2.8. Any hits?	Yes No	39 41
32	If a loop appears to occur, use a system-trace facility or instruction-step mode to capture all, or at least part, of the loop addresses. Then cancel the job with a dump. Find the current phase in the dump as follows: ° Register 13 points to the communications area (XCOMM). ° To check, look at the field at offset X'90' from register 13. This field contains the first source input record. If register 13 was corrupted, search for this field to locate XCOMM. ° The current phase name is at offset X'4AB' from the beginning of XCOMM. This field contains two letters. Adding IEL1 before these two letters gives you the name of the current phase. ° The phase start address is at offset X'434' from XCOMM. If you are using the OPTIMIZE compile-time option, re-compile using NOOPTIMIZE. Phase IEL1IE can appear to be in a loop if a large number of BYNAME assignments occur or if the source assigns to a large PICTURE statement. Phase IEL1IK can appear to be in a loop while sorting data names for the XREF table. Give the compiler a little more time. Other loops are caused by: ° Using a colon instead of a semicolon ° Not enough storage ° Not enough time **Note:** It is possible to cause the compiler to loop by including a preprocessor macro that contains a loop. This is a user error in the preprocessor macro. Is the problem circumvented?	Yes No	41 33

33	Search IBMLink, INFO/ACCESS, or have the IBM Support Center search RETAIN using: ° Component ID 5686069 ° LOOP and module names in which loop occurs Information about doing your search is in Chapter 10, "Using the keyword string to search for corrections" in topic 2.8. Any hits?	Yes No	39 41
34	The only wait states the compiler issues are for I/O. Investigate wait states from the system control viewpoint: ° Check to see that the partition running PL/I is not waiting for a resource owned by another partition. ° See if there are any system messages. If you still suspect the PL/I compiler is causing the wait state, go to Block 41.		41
35	Be sure all appropriate compile-time options are specified. If you are using the OPTIMIZE(TIME) or OPTIMIZE(2) compile-time option, re-compile with NOOPTIMIZE. Is the problem circumvented?	Yes No	41 36
36	Search IBMLink, INFO/ACCESS (Level 1), or have the IBM Support Center search RETAIN using component ID 5686069 and INCORROUT (a word describing what output is incorrect). Information about doing your search is in Chapter 10, "Using the keyword string to search for corrections" in topic 2.8. Any hits?	Yes No	39 41
37	Performance problems usually show up after some environment change; if not a maintenance change, then perhaps a source code or compile-time option change. Review these items. If you are using the OPTIMIZE(TIME) or OPTIMIZE(2) compile-time option, re-compile with NOOPTIMIZE. Is the problem circumvented?	Yes No	41 38
38	Search IBMLink, INFO/ACCESS (Level 1), or have the IBM Support Center search RETAIN using component ID 5686069. Information about doing your search is in Chapter 10, "Using		

	the keyword string to search for corrections" in topic 2.8. Any hits?	Yes No	39 41
39	Apply the fixes or circumvention found in RETAIN, and test. Is the problem solved?	Yes No	END 40
40	Do the symptoms change?	Yes No	2 41
41	Contact the IBM Support Center for assistance. Have available the following documentation: ° Compile listing with LIST, SOURCE, XREF, STMT, ESD, and MAP compile-time options in effect ° The JCL used to run the job ° Dump (if applicable) ° Preprocessor input (if applicable) ° List of applied fixes		

2.0 Part 2. Describing the problem

Subtopics:

2.1 Chapter 3. Introduction to diagnosis keywords

Failures in the PL/I VSE compiler can be described through the use of *diagnosis keywords*. A diagnosis keyword is a word or abbreviation used to describe one aspect of a product failure. You can use a set of keywords called a *keyword string* to describe the failure in detail. Use the procedures in this section to construct a keyword string that describes what you currently know about the compiler failure. Table 6 lists the books that discuss problems other than compiler failures.

Table 6. References for other failure information

Type of failure	**Source of information**
User compile-time problems	See the PL/I VSE Programming Guide
User run-time problems	See the *LE/VSE Debugging Guide and Run-Time Messages* or the *LE/VSE Programming Guide*
LE/VSE run-time product failures	See the *LE/VSE Diagnosis Guide*

After it is constructed, the keyword string is used as a search argument against an IBM software support database, such as the Software Support Facility (SSF). The database contains keyword and text information describing all current problems reported through Authorized Program Analysis Reports (APARs) and associated program temporary fixes (PTFs). IBM Support Center personnel have access to the software support database and are responsible for storing and retrieving the information. Using the keyword string, they search the database to retrieve records that describe similar known problems.

If you have IBMLink* or some other electronic link with IBM Service, you can do your own search for previously recorded product failures before calling the IBM Support Center.

If the keyword string search produces a match in the software support database, the search might yield a more complete description of the problem and possibly identify a correction or circumvention. Such a search might yield several matches to previously reported problems. Thus, you should review each error description carefully to determine if the problem description in the database matches the failure.

If a match is not found, use the keyword string you have constructed to describe the failure when you contact the IBM Support Center for assistance and when you submit an APAR. Keywords are intended to ensure that identical program errors are described with identical keyword strings. Spelling the keywords exactly as they are presented in this book is especially important for a successful match.

Subtopics:

2.1.1 Keyword usage

Depending upon the PL/I failure, a keyword string can contain some or all of the following items:

- Component identification
- Release level
- Type of failure
- Name of the module that failed
- Name of the system you were operating under at the time of failure
- One or more modifier keywords, depending on the type of failure

In building a keyword string, the first keyword usually identifies the failing component. The component identification for PL/I VSE should be the compiler identifier. A search of the software support database with this single keyword would locate all problems reported for the compiler. Each additional keyword added to the keyword string narrows the scope of the search argument and helps to eliminate unnecessary examination of problem descriptions that have similar, but not matching, characteristics. In some cases, a correction for a product failure might be located with less than a full string of keywords. If circumstances make it difficult to follow the instructions for selecting a particular keyword, omit that keyword to avoid incorrectly identifying the problem. In general, if you contact IBM, you are asked to identify your problem with a full set of keywords, as described here.

Follow the steps in the keyword procedures until you are directed to use the keyword string in a search argument.

Figure 3 shows the keyword string generation process for each type of failure.

```
                                          ________________
                                         |  Component ID  |
                                         |  Keyword       |
                                         |  (568606901)   |
                                         |_______ ________|

                                          ________________
                                         |  Release-Level |
                                         |  Keywords      |
                                         |  (R110)        |
                                         |_______ ________|
                                                 |
                                         Type of  Failure Keyword
        _______________ _______________ __________________ ______________ ______________

 ______________    __________    ______________    ______________    __________   ______________
| Abnormal     |  | No       |  | Performance  |  | Incorrect    |  | Message  | | Documentation|
| Termination  |  | Response |  | Problem      |  | Output       |  | Problem  | | Error        |
| (IEL0001I)   |  |          |  |              |  |              |  |          | |              |
| (IEL0230I)   |  |          |  |              |  |              |  |          | |              |
| (IEL0970I)   |  | (LOOP)   |  | (PERFM)      |  | (INCORROUT)  |  | (MSGxxxx)| | (DOC)        |
|______ _______|  |____ _____|  |______ _______|  |_______ ______|  |____ _____| |______ _______|
       |               |               |                  |              |              |
       |_______ _______|               |__________________|______________|              |
                                                         |                             |
       ________________                                   |                             |
      |     Module     |                                  |                             |
      |     Keyword    |                                  |                             |
      |                |                                  |                             |
      |________ _______|                                  |                             |
               |                                          |                             |
               |_________________________________ ________|                             |
                                                                                       |
                                          ________________                              |
                                         |  System Type   |                             |
                                         |  Keyword       |                             |
                                         |                |                             |
                                         |_______ ________|                             |
                                                 |_____________________________ ________|

                                                                        ________________
                                                                       |    Modifier    |
                                                                       |    Keyword     |
                                                                       |                |
                                                                       |________________|
```

Figure 3. PL/I VSE - Problem identification using keywords

2.1.2 Using the problem identification worksheet

You can use Chapter 11, "Problem identification worksheet" in topic 2.9 to help you construct and record a keyword string. As you identify the keywords associated with your software problem, just record them in the spaces provided.

2.1.3 Diagnosis procedure

This procedure is designed to gather the diagnostic information required for developing a keyword string to search the software support database. It describes options that you can specify to obtain all the available diagnostic information. You need this information to discuss the problem with your IBM support representative if your search against the database fails to locate a fix for your problem.

Use the following procedure only if the problem is occurring at compile time. If the problem is in the LE/VSE product, refer to the *LE/VSE Diagnosis Guide* for diagnostic information.

1. Locate the source of the problem:

   a. Determine if the program has been changed since it was last compiled successfully. If it has, examine the changes. If the error is occurring in the changed code, note the change that caused the error. If possible, retain copies of both the original and the changed programs in case you need to submit an APAR.

   b. Determine if the operating system environment has been changed. If it has, examine the changes. If the error cannot be resolved by examination of the changes, have a description of the changes available when you call the IBM Support Center.

   c. Determine if PL/I maintenance has been applied. If it has, determine if the error was due to a particular PTF and call the IBM Support Center with this information.

2. Correct all problems diagnosed by error messages. Ensure that any messages previously generated do not affect the current problem. Be sure to pay attention to warning messages (W-level messages). Message prefixes identify the system or subsystem that issued the error:

   - PL/I VSE compiler messages are prefixed by IEL.

   - For messages other than these, consult the appropriate system or subsystem messages book.

3. After you have explored and identified the failure, consider writing a small test case that re-creates the problem. This test case should help you to:

   - Pinpoint the problem
   - Distinguish between an error in the application program and an error in PL/I
   - Choose keywords that best describe the error

4. In addition to the options originally specified, specify the compile-time options described in Table 3 in topic 1.1.3.2 and re-compile the program.

   These options produce maximum diagnostic information, which helps you diagnose product errors. See the *PL/I VSE Programming Guide* for more information on how to use these options.

5. If the error symptoms change, return to step 2.

6. Record the sequence of events that led to the error condition. This information may be useful in developing a keyword string, and is needed if an APAR is required.

7. Begin developing the keyword string. Start with the procedures in Chapter 4, "Component identification keyword" in topic 2.2.

2.2 Chapter 4. Component identification keyword

This procedure shows what to specify as the component identification keyword. The component identification keyword is usually the first keyword placed in the search argument string. It comes from the PL/I compiler program number and identifies the area within the software support database that contains APARs for PL/I.

At minimum, you need to use a type-of-failure keyword as a search argument along with the component identification keyword. If you search using only the component identifier keyword, you receive a full listing of all the APARs affecting PL/I.

1. Use **568606901** as the component identification keyword. This number is the PL/I VSE compiler product identifier, 5686069, with 01 appended.

2. If service tapes have been applied to the licensed program, note the tape level of the last service tape applied. See your system programmer for the current service level of your PL/I compiler. Although the service tape level is not used in the keyword string, you might find it useful when reviewing APARs selected during the keyword search.

3. Continue the diagnostic procedure with Chapter 5, "Release-level keyword" in topic 2.3.

2.3 Chapter 5. Release-level keyword

Use the following procedure to identify the specific release level of PL/I under which you were operating when the failure occurred.

1. Locate the version, release, and modification level line at the top of the first page of your latest compiler output listing for the failing program. The heading line contains the current product identification data in the following format:

   ```
   5686-069 IBM PL/I for VSE/ESA v.r.m
   ```

   where:

   v Specifies the current version
   r Specifies the current release
   m Specifies the current modification number

   The date and time of compilation and the page number are also found on the heading line. The heading line for Version 1.1.0 of PL/I VSE would be as follows:

   ```
   5686-069 IBM PL/I for VSE/ESA Ver 1 Rel 1 Mod 0 ...
   ```

2. Specify the release-level keyword:

   ```
   R110
   ```

   The following is an example of a partial keyword string, consisting of the component identification and the release-level keyword.

   ```
   568606901 R110
   ```

3. Continue the diagnostic procedure with Chapter 6, "Type-of-failure keyword" in topic 2.4.

2.4 Chapter 6. Type-of-failure keyword

Various types of failure specific to the compiler might occur in the PL/I licensed program. Table 7 lists the books that discuss problems other than compiler failures.

Table 7. References for other failure information

Type of failure	**Source of information**
User compile-time problems	See the PL/I VSE Programming Guide
User run-time problems	See the *LE/VSE Debugging Guide and Run-Time Messages* or the *LE/VSE Programming Guide*
LE/VSE run-time product failures	See the *LE/VSE Diagnosis Guide*

Read Table 8 and select the type of failure that best describes the problem with the compiler. Then go to the associated keyword procedure listed in this table for instructions on how to complete the keyword string for that type of failure. If more than one of the keywords in the following table describes the problem you are experiencing, use the keyword that appears first in the table.

Table 8. Types of PL/I failures

Type of failure	**Symptom**	**Procedure**
Abnormal termination	The compiler has terminated abnormally, issuing one of three compiler messages: IEL0001I IEL0230I IEL0970I	See "Abnormal termination" in topic 2.4.1.
Message problems	The compiler issues an inappropriate or invalid error message.	See "Message problems" in topic 2.4.2.
No response from the compiler	Either an unexpected program suspension has occurred or the job has not completed in batch mode.	See "No response from the compiler" in topic 2.4.3.
PL/I documentation problems	Information in one of the PL/I publications or soft copy documents is incorrect or missing.	See "PL/I documentation problems" in topic 2.4.4.
Output problems	The output from the compiler is missing or invalid.	See "Output problems" in topic 2.4.5.
Performance problems	The performance of a PL/I compilation is degraded.	See "Performance problems" in topic 2.4.6.

Subtopics:

2.4.1 Abnormal termination

The compiler can terminate abnormally with either a system or user abend. When abends occur in the compiler, the compiler traps these abends and generally issues one of the following messages:

- IEL0001I
- IEL0230I
- IEL0970I

Check the error and restriction codes associated with the message received. For a list of the error and restriction codes, see *PL/I VSE Compile-Time Messages and Codes.*

Ensure that all other problems have been fixed and search the database to determine if the problem is already known. If you determine that this is a new problem, call IBM with the message information received.

2.4.2 Message problems

Before using this procedure, verify that you have received a compiler message. The PL/I compiler issues messages prefixed with IEL. Messages with other prefixes are issued by the LE/VSE run-time environment or by operating systems or subsystems and access methods, and should not be addressed as PL/I product problems. See the *PL/I VSE Compile-Time Messages and Codes* for PL/I VSE messages.

Use this procedure for any one of the following conditions:

- A message is issued under a set of conditions that should not have caused it to be issued.

- A message contains invalid data or is missing data.

Use the following procedure to construct the MSGx keyword:

1. For example, if you received message number IEL0230I, replace the x in MSGx with IEL0230I, as shown in the following example:

```
568606901 R110 MSGIEL0230I
```

2. Proceed with Chapter 8, "System-type keyword" in topic 2.6.

2.4.3 No response from the compiler

Use this keyword procedure for any of the following conditions:

- The compiler seems to be doing nothing or is doing something repetitively.

- The compile job does not reach completion.

If the problem looks like a WAIT state, it is probably a system problem. In that case, follow your local procedures for resolution. Otherwise:

1. Your set of keywords, so far, would look something like this:

   ```
   568606901 R110 LOOP
   ```

2. Continue with Chapter 7, "Module keyword" in topic 2.5.

2.4.4 PL/I documentation problems

Use this procedure when you notice a problem caused by incorrect or missing information in one of the published documents or the soft copy PL/I documents.

1. Locate the page or pages in the document (or the online panel for a soft copy document) where the problem occurs and prepare a description of the error and the problem it caused. This information is required for APAR preparation if no similar problem is found in the software support database.

2. Decide whether the documentation problem is severe enough to cause lost time for other users.

   ° If the problem is not severe, submit a Reader's Comments Form (RCF) attached to the back of the publication in question. As an option, you can fax the RCF to IBM using the fax number printed on the RCF. If the RCF is missing, send a note to the address shown on the edition notice for this book. Include the problem description you have developed, along with your name and return address, so that IBM can respond to your comments.

   ° If the problem is severe enough to cause lost time for other users, continue creating your keyword string to determine whether IBM has a record of the problem. Should this be a new problem, you will be asked to submit a severity-3 or -4 documentation (DOC) APAR.

3. Use the order number on the cover of the document along with the DOC keyword as your type-of-failure keyword, but omit the hyphens. Leave a single space between DOC and the document number. If the number following the last hyphen has only one digit, it must be preceded by a zero. For example, if the order number is SC26-8054-00 (*PL/I VSE Language Reference*), use SC26805400. Your keyword string would look something like this:

   ```
   568606901 R110 DOC SC26805400
   ```

4. Search the IBM software support database to determine if this documentation problem has already been reported. If, after searching the database, you do not find a matching description, return here to continue. To search the database, turn to Chapter 10, "Using the keyword string to search for corrections" in topic 2.8.

5. In case several levels of the document exist, you can use two

asterisks appended to the document number to search for all problems reported for the document rather than only those for a specific release of the document. Use a format similar to the following:

```
568606901 R110 DOC SC268054**
```

6. Go to Chapter 10, "Using the keyword string to search for corrections" in topic 2.8.

2.4.5 Output problems

Use this procedure when the output appears to be incorrect or missing, but the compiler terminated normally. If the data or records were repeated endlessly, follow the steps under "No response from the compiler" in topic 2.4.3 instead of this "Output problems" procedure to create your keyword string.

1. Use INCORROUT as your type-of-failure keyword.

2. Select a modifier keyword from the following table to describe the type of error in the output. For more information on the use of modifier keywords, see Chapter 9, "Modifier keyword" in topic 2.7.

Table 9. Incorrect output modifier keywords

Modifier keyword	**Type of incorrect output**
DUPLICATE	Some data or records were duplicated, but were not repeated endlessly.
INVALID	The output that appeared was not as expected; that is, the output was bad or incorrect.
MISSING	Some expected output was missing.

3. Select another modifier keyword from the following table to describe the portion of the output in which the error occurred.

Table 10. Output error location keywords

Modifier keyword	**Portion of output in error**
LIST	Assembler language expansion of source listing, global tables, literal pools, static storage map
MAP	Static storage map
MESSAGE	Diagnostic message
OFFSET	Offset table
SOURCE	Source listing
XREF	Cross-reference listing

```
|__________________|_________________________________________________|
```

For example, if you think that the compiler has given an incorrect cross-reference listing, your keyword string so far would look something like this:

```
568606901 R110 INCORROUT INVALID XREF
```

4. Continue the diagnostic procedure with Chapter 8, "System-type keyword" in topic 2.6.

2.4.6 Performance problems

Most performance problems can be related to system tuning and should be handled by system engineers and system programmers. You might want to contact your IBM system engineer who can use ASKQ to retrieve recommendations for improving product performance.

Use the following keyword procedure when the performance problem could not be corrected by system tuning and performance is significantly below explicitly stated expectations.

1. Use PERFM as your type-of-failure keyword. For example, your keyword string for performance problems might look like this:

   ```
   568606901 R110 PERFM
   ```

2. Continue the diagnostic procedure with Chapter 8, "System-type keyword" in topic 2.6.

2.5 Chapter 7. Module keyword

The following procedure demonstrates how you build a module keyword.

In messages, module names are shown as:

```
PHASE xx
```

To construct a module name from this type of message, you append IEL1 to the phase as shown in the example below.

```
IEL1EA
```

Continue the diagnostic procedure with Chapter 8, "System-type keyword" in topic 2.6.

2.6 Chapter 8. System-type keyword

System-type keywords indicate which system you were operating under when the PL/I compiler failed.

1. Use VSE as your first system-type keyword.

2. Use ESA as your next system-type keyword. For instance, your keyword search might look like this:

```
568606901 R110 MSGIEL0230I VSE ESA
```

3. Continue the diagnostic procedure with Chapter 9, "Modifier keyword" in topic 2.7.

2.7 Chapter 9. Modifier keyword

This procedure helps you determine what type of keyword (modifier or source language) you should add to your keyword string. Use this procedure to locate the point of compiler failure or the keyword you were using when the failure occurred. Table 11 lists the books that discuss problems other than compiler failures.

Table 11. References for other failure information

Type of failure	**Source of information**
User compile-time problems	See the PL/I VSE Programming Guide
User run-time problems	See the *LE/VSE Debugging Guide and Run-Time Messages* or *LE/VSE Programming Guide*
LE/VSE run-time product failures	See the *LE/VSE Diagnosis Guide*

One or more modifier keywords may be used in the same keyword string to define the compiler problem. Additional modifiers help to make the search argument more specific. Use the capitalized spelling of the modifier in the keyword string. The various types of modifier keywords include:

° Compile-time options

Select from your compiler listing those compile-time options that you consider significant to the type of failure. See Appendix A, "Compile-time options" in topic A.0 for a list of the compile-time options. The option name itself (in full or abbreviated format) is the keyword.

If the compiler failure appears to be correlated with any particular compiler option or options, such as XREF, use those options as additional modifier keywords, as in this example:

```
  568606901 R110 MSGIEL0230I XREF
```

- Message IDs

  If you receive a compile-time message, you can use the ID as an additional modifier. For example, assume that you received this message:

  ```
    IEL0230I  COMPILER ERROR 0 DURING PHASE EA
  ```

  You could then use IEL0230I as an additional modifier.

- PL/I language keywords

  If the compiler failure is peculiar to a PL/I language keyword, use the language keyword as a modifier keyword. For example, if the source of failure was the PL/I language keyword SELECT, your keyword string would look something like this:

  ```
    568606901 R110 MSGIEL0230I SELECT
  ```

Continue the diagnostic procedure with Chapter 10, "Using the keyword string to search for corrections" in topic 2.8.

2.8 Chapter 10. Using the keyword string to search for corrections

This chapter explains how to use the keyword string as a search argument against a software support database. The search can be performed by calling an IBM Support Center or using your electronic link with IBM, if available.

Your search will be more successful if you follow these rules:

- Use only the keywords given in this book.

- Spell keywords the way they are spelled in this book. Any variation in spelling may result in an unsuccessful search.

- Include all the appropriate keywords in any discussion with IBM support personnel or in an APAR.

In order to search the support database, you should perform the following steps:

1. Search the software support database, using the full set of keywords you have developed. For example, consider the following keyword string:

   ```
   568606901 R110 MSGIEL0230I IEL*EA VSE ESA
   ```

   Note: You can use a wildcard (*) when you use the module name as part of your search argument.

2. If the search produces a list of APARs, continue with step 3; otherwise, go to step 6.

3. When your search is complete, eliminate from the list of possible APAR fixes those that have already been applied to your system.

4. Compare each of the remaining closed APAR descriptions with the current failure symptoms.

5. If a match is found, apply the PTF to your system and exit this procedure.

6. If the search did **not** produce a list of APARs, or an APAR description matching the current failure is not found, broaden the search, using the following techniques:

   a. Omit the release-level keyword (for example, R110) from the search argument, thereby broadening the search to include similar failures on other release levels.

   b. Drop one keyword from the right end of the search argument string. The diagnostic procedures directed you to construct the keyword string with the most significant keywords listed first. By dropping a keyword from the right, you eliminate the least significant keyword, thereby broadening your search while maintaining the relevancy of your search argument string. Perform the search against the software support database, using your shortened search argument string. Repeat this step as necessary.

7. If a match is not found using the preceding techniques, go to Chapter 12, "Preparing an APAR" in topic 3.1.

2.9 Chapter 11. Problem identification worksheet

Record the keywords associated with your software problem as you identify them.

Component Identification: _________________________

Release Level: _________________________

Type of Failure: _________________________

Module: _________________________

System Type: _________________________

Modifiers: _________________________

____

____

Note: Some keywords may not be applicable to all problems.

3.0 Part 3. Reporting the problem

Subtopics:

3.1 Chapter 12. Preparing an APAR

Before preparing an Authorized Program Analysis Report (APAR), make sure that:

- You have followed the diagnosis procedure.
- You have eliminated user errors as a source for the problem.
- The keyword search was unsuccessful.

Open a Problem Management Record (PMR) when these steps have failed to resolve your problem.

Subtopics:

3.1.1 Opening a PMR

If you have IBMLink or some other connection to IBM databases, you may open a PMR yourself; otherwise, you can call the IBM Support Center and request that they open the PMR for you.

The PMR is used to document your problem and to record the work done on the problem by members of the IBM Support Center or the IBM Change Team. After analyzing the problem, the Support Center may recommend that an APAR be initiated.

3.1.2 Initiating an APAR

In order to initiate an APAR, you need to complete the following steps.

1. Contact the IBM Support Center for assistance. Be prepared to supply the following information:

   - Customer number and security code
   - PMR number
   - Operating system
   - Operating system release level
   - Current PL/I VSE maintenance level (PTF list and list of APAR fixes applied)
   - Current LE/VSE release level and PTF list
   - The various keyword strings used to search the software support database
   - Processor number (model and serial)

2. From the following list, you may be asked to include the applicable PL/I VSE environmental information with your APAR:

   - Job control statements
   - Compiler listings, including:

     - Source listing
     - Compiled listing
     - Storage map
     - Cross-reference listing

     Use LIST, MAP, SOURCE, XREF, and other options pertinent to the problem.

   - Machine-readable copy of the program causing the problem, including all copy members required by the program.
   - A dump on tape, if available, or if the use of the DUMP option was requested by an IBM representative.

  - The compiler on tape if requested by an IBM representative.

  - Hard copy of the job control language (JCL) for unloading the submitted machine-readable tape.

  - Any other data that may help in re-creating the problem.

Any listings supplied must be from the PL/I VSE compilation version that failed.

Subtopics:

- 3.1.2.1 Materials to submit

3.1.2.1 Materials to submit

Table 12 describes how to produce documentation required for submission with the APAR. Additional requirements are explained after the table. Many of these materials may already have been produced in their required format during the formulation of the keyword string (see "Diagnosis procedure" in topic 2.1.3).

Table 12. Summary of requirements for submitting an APAR

Materials	**When required**
Original source or failing test case	Always
JCL	Always
PL/I compiler	Only when requested
Compile listing	Always
JCL listing	Always
Applied PTFs and fixes	Always, or specify no fixes applied

Note: If you supply machine-readable material on a tape reel, describe how you created the tape.

Subtopics:

- 3.1.2.1.1 Original source information
- 3.1.2.1.2 PL/I compiler
- 3.1.2.1.3 Compile listing
- 3.1.2.1.4 JCL listing
- 3.1.2.1.5 Applied fixes

3.1.2.1.1 Original source information

You must supply source information in one of the following forms:

- Your original source in machine-readable format
- A small test case that IBM can use to re-create the problem

Note: If you do not supply one of these forms, IBM programming service might return your APAR, requesting that you supply source information.

If you send machine-readable source, submit the information on an unlabeled tape. Along with the tape, send a hard copy listing of how you created the tape. Carefully pack and clearly identify machine-readable information. Make sure the APAR number is on the tape, so that IBM can identify it if it is separated from the rest of the material you submit with the APAR.

Depending on the options and conditions you have, the source of machine-readable code differs. These sources appear in Table 13. Also, the machine-readable source should have no %NOPRINT statements, unless they relate to the problem.

Table 13. Machine-readable sources

Options and conditions	**Machine-readable source**
NOINCLUDE NOMACRO	The source is the data set assigned to SYSIN for the compile step.
INCLUDE MACRO Preprocessor failure	The source is the data set assigned to SYSIN for the compile step and the source statement library or libraries referred to by %INCLUDE statements in the program.
INCLUDE MACRO	The source is the SYSPUNCH data set the compiler produces when the MDECK compile-time option is in effect.

3.1.2.1.2 PL/I compiler

You do not need to send this unless IBM specifically asks you for it. IBM programming service needs it only if they cannot re-create your problem using programming service's own compiler.

3.1.2.1.3 Compile listing

If you think you have a compiler failure, then all listings that you supply must relate to a specific run of the compiler. Do not send information that is derived from separate compilations or runs. These can mislead the programming support personnel at IBM.

With your APAR, always send the listing which results from the compilation of the original source. Compile the program with the compile-time options listed in Table 3 in topic 1.1.3.2 unless you must use the opposite option to show the failure or unless the option masks the failure.

3.1.2.1.4 JCL listing

You must provide listings of job control statements that you use to run the program. If you have problems with a batch job, show any cataloged procedures you use.

3.1.2.1.5 Applied fixes

Also supply with your APAR a list of any program temporary fixes (PTFs) and local fixes you applied to either the compiler or the library. If you applied no fixes, indicate this specifically with your APAR.

4.0 Appendixes

A.0 Appendix A. Compile-time options

The following list contains the valid compile-time options. For more information about these options, see the *PL/I VSE Programming Guide.*

```
AGGREGATE (AG)
ATTRIBUTES (A)
CMPAT (CMP)
COMPILE (C)
CONTROL
DECK (D)
ESD
FLAG (F)
GONUMBER (GN)
GOSTMT (GS)
GRAPHIC (GR)
IMPRECISE (IMP)
INCLUDE (INC)
INSOURCE (IS)
INTERRUPT (INT)
LANGLVL
LINECOUNT (LC)
LIST
LMESSAGE (LMSG)
MACRO (M)
MAP
MARGINI (MI)
MARGINS (MAR)
```

```
MDECK (MD)
NAME (N)
NEST
NOAGGREGATE (NAG)
NOATTRIBUTES (NA)
NOCOMPILE (NC)
NODECK (ND)
NOESD
NOGONUMBER (NGN)
NOGOSTMT (NGS)
NOGRAPHIC (NGR)
NOIMPRECISE (NIMP)
NOINCLUDE (NINC)
NOINSOURCE (NIS)
NOINTERRUPT (NINT)
NOLIST
NOMACRO (NM)
NOMAP
NOMARGINI (NMI)
NOMDECK (NMD)
NONEST
NONUMBER (NNUM)
NOOBJECT (NOBJ)
NOOFFSET (NOF)
NOOPTIMIZE (NOPT)
NOOPTIONS (NOP)
NOSEQUENCE (NSEQ)
NOSOURCE (NS)
NOSTMT
NOSTORAGE (NSTG)
NOSYNTAX (NSYN)
NOT
NOTERMINAL (NTERM)
NOTEST
NOXREF (NX)
NUMBER (NUM)
OBJECT (OBJ)
OFFSET (OF)
OPTIMIZE (OPT)
OPTIONS (OP)
OR
SEQUENCE (SEQ)
SIZE (SZ)
SMESSAGE (SMSG)
SOURCE (S)
STMT
STORAGE (STG)
SYNTAX (SYN)
SYSTEM
TERMINAL (TERM)
TEST
XREF (X)
```

B.0 Appendix B. Compile-time source language keywords

The following list contains the valid compile-time source language keywords. For more information about the compile-time keywords, see the *PL/I VSE Reference Summary.*

```
A
ABS
```

```
ACOS
%ACTIVATE (%ACT)
ADD
ADDBUFF
ADDR
ALIGNED
ALL
ALLOCATE (ALLOC)
ALLOCATION (ALLOCN)
ANY
AREA
ARGi
ASCII
ASIN
ASSEMBLER (ASM)
ATAN
ATAND
ATANH
ATTENTION (ATTN)
AUTOMATIC (AUTO)
B
BACKWARDS
BASED
BEGIN
BINARY (BIN)
BINARYVALUE
BIT
BKWD
BLKSIZE
BOOL
BUFFERED (BUF)
BUFFERS
BUFND
BUFNI
BUFOFF
BUFSP
BUILTIN
BX
BY
B4
BYADDR
BYVALUE
BY NAME
C
CALL
CEIL
CHAR
CHARACTER (CHAR)
Character
CHARGRAPHIC (CHARG)
CLOSE
CMDCHN
COBOL
COLUMN (COL)
COMPAT
COMPILETIME
COMPLETION (CPLN)
COMPLEX (CPLX)
CONDITION (COND)
CONJG
CONNNECTED (CONN)
CONSECUTIVE
CONTROLLED (CTL)
CONVERSION (CONV)
COPY
COS
COSD
COSH
COUNT
COUNTER
CTLASA
```

```
CTL360
CURRENTSTORAGE
 (CSTG)
D
DATA
DATAFIELD
DATE
DATETIME
DB
%DEACTIVATE (%DEACT)
DECIMAL (DEC)
DECLARE (DCL)
%DECLARE (%DCL)
DEFAULT (DFT)
DEFINED (DEF)
DELAY
DESCRIPTORS
DIM
DIRECT
DISPLAY
DIVIDE
DO
%DO
E
EDIT
ELSE
%ELSE
EMPTY
END
%END
ENDFILE
ENDPAGE
ENTRY
ENTRYADDR
ENVIRONMENT (ENV)
ERF
ERFC
ERROR
EVENT
EXCLUSIVE (EXCL)
EXIT
EXP
EXTERNAL (EXT)
F
FB
FETCH
FETCHABLE
FILE
FILESEC
FINISH
FIXED
FIXEDOVERFLOW (FOFL)
FLOAT
FLOOR
FORMAT
FREE
FROM
G
GENERIC
GENKEY
GET
GO TO
GOTO
%GO TO
%GOTO
GRAPHIC (G)
GX
HBOUND
HIGH
IF
%IF
```

```
IGNORE
IMAG
IN
%INCLUDE
INDEX
INDEXAREA
INDEXED
INITIAL (INIT)
INPUT
INTER
INTERNAL (INT)
INTO
IRREDUCIBLE (IRRED)
iSUB
KEY
KEYED
KEYFROM
KEYLENGTH
KEYLOC
KEYTO
LABEL
LBOUND
LEAVE
LENGTH
LIKE
LIMCT
LINE
LINENO
LINESIZE
LIST
LOCATE
LOG
LOG2
LOG10
LOW
M
MAIN
MAX
MEDIUM
MIN
MOD
MPSTR
MULTIPLY
NAME
NCP
NOCHARGRAPHIC
 (NOCHARG)
NOCONVERSION
 (NOCONV)
NOEXECOPS
NOFEED
NOFIXEDOVERFLOW
 (NOFOFL)
NOLABEL
NOLOCK
NOMAP
NOMAPIN
NOMAPOUT
NOOVERFLOW (NOOFL)
%NOPRINT
NORESCAN
NOSIZE
NOSTRINGRANGE
 (NOSTRG)
NOSTRINGSIZE (NOSTRZ)
NOSUBSCRIPTRANGE
 (NOSUBRG)
%NOTE
NOTAPEMK
NOUNDERFLOW (NOUFL)
NOWRITE
```

```
NOZERODIVIDE
null
%null
NULL
OFFSET
ON
ONCHAR
ONCODE
ONCOUNT
ONFILE
ONKEY
ONLOC
ONSOURCE
OPEN
OPTIONAL
OPTIONS
ORDER
OTHERWISE (OTHER)
OUTPUT
OVERFLOW (OFL)
P
PAGE
%PAGE
PAGESIZE
PARMSET
PASSWORD
PENDING
PICTURE (PIC)
PLIRETV
POINTER (PTR)
POINTERADD
POINTERVALUE
POLY
POSITION (POS)
precision
PRECISION (PREC)
PRINT
%PRINT
PROCEDURE (PROC)
%PROCEDURE (%PROC)
*PROCESS
%PROCESS
PROD
PUT
R
RANGE
READ
REAL
RECORD
RECSIZE
RECURSIVE
REDUCIBLE (RED)
REENTRANT
REFER
REGIONAL
RELEASE
REORDER
REPEAT
REPLY
REREAD
RESCAN
RETCODE
RETURN
RETURNS
REUSE
REVERT
REWRITE
ROUND
SAMEKEY
SCALARVARYING
SELECT
```

```
SEQUENTIAL (SEQL)
SET
SIGN
SIGNAL
SIN
SIND
SINH
SIS
SIZE
SKIP
%SKIP
SNAP
SQRT
STATEMENT (STMT)
STATIC
STATUS
STOP
STORAGE
STREAM
STRING
STRINGRANGE (STRG)
STRINGSIZE (STRZ)
SUBSCRIPTRANGE
 (SUBRG)
SUBSTR
SUM
SYSIN
SYSNULL
SYSPRINT
SYSTEM
TAN
TAND
TANH
THEN
%THEN
TIME
TITLE
TO
TOTAL
TRANSLATE
TRANSMIT
TRKOFL
TRUNC
U
UNALIGNED (UNAL)
UNBUFFERED (UNBUF)
UNDEFINEDFILE (UNDF)
UNDERFLOW (UFL)
UNLOAD
UNLOCK
UNSPEC
UNTIL
UPDATE
V
VALUE
VARIABLE
VARYING (VAR)
VB
VERIFY
VOLSEQ
VSAM
WAIT
WHEN
WHILE
WRITE
WRTPROT
X
ZERODIVIDE (ZDIV)
```

BIBLIOGRAPHY Bibliography

Subtopics:

BIBLIOGRAPHY.1 IBM PL/I for VSE/ESA publications

Fact Sheet, GC26-8052

Installation and Customization Guide, SC26-8057

Licensed Program Specifications, GC26-8055

Language Reference, SC26-8054

Compile-Time Messages and Codes, SC26-8059

Diagnosis Guide, SC26-8058

Migration Guide, SC26-8056

Programming Guide, SC26-8053

Reference Summary, SX26-3836

BIBLIOGRAPHY.2 IBM Language Environment for VSE/ESA publications

Concepts Guide, GC26-8063

Fact Sheet, GC26-8062

Debugging Guide and Run-Time Messages, SC26-8066

Diagnosis Guide, SC26-8060

Installation and Customization Guide, SC26-8064

Licensed Program Specifications, GC26-8061

Programming Guide, SC26-8065

Reference Summary, SX26-3835

BIBLIOGRAPHY.3 VSE/ESA publications

VSE/ESA Version 1

Administration, SC33-6505

Messages and Codes, SC33-6507

System Control Statements, SC33-6513

System Utilities, SC33-6517

System Macros Reference, SC33-6516

Guide to System Functions, SC33-6511

VSE/VSAM Commands and Macros, SC33-6532

VSE/VSAM User's Guide, SC33-6535

VSE/ESA Version 2

Administration, SC33-6605

Messages and Codes, SC33-6607

System Control Statements, SC33-6613

System Utilities, SC33-6617

System Macros Reference, SC33-6616

Guide to System Functions, SC33-6611

VSE/VSAM Commands and Macros, SC33-6631

VSE/VSAM User's Guide, SC33-6632

BIBLIOGRAPHY.4 Related publications

CICS/VSE

Application Programming Guide, SC33-0712

Application Programming Reference, SC33-0713

System Definition and Operations Guide, SC33-0706

Resource Definition, SC33-0708

DFSORT for VSE/ESA

Application Programming Guide, SC26-7040

Sort/Merge II

DOS/VS VM/SP Sort/Merge Version 2 Application Programming Guide, SC33-4044

DL/I DOS/VS

Application Programming: High Level Programming Interface, SH24-5009

Application Programming: CALL and RQDLI Interfaces, SH12-5411

SQL/DS

SQL/Data System Application Programming Guide for VSE, SH09-8098

BIBLIOGRAPHY.5 Softcopy publications

These collections contain the LE/VSE and LE/VSE-conforming language product publications:

VSE Collection, SK2T-0060

Application Development Collection, SK2T-1237

INDEX Index

IBM Library Server Print Preview

DOCNUM = SC26-8058-00
DATETIME = 02/09/95 20:11:36
BLDVERS = 1.2
TITLE = PL/I VSE Diagnosis Guide
AUTHOR =
COPYR = © Copyright IBM Corp. 1964, 1995
PATH = /home/webapps/epubs/htdocs/book