SC26-8056-00

IBM PL/I for VSE/ESA

Migration Guide

Release 1

This keyline indicates size
and position of graphic
only, keyline does not print.

IBM PL/I for VSE/ESA

SC26-8056-00

Migration Guide

Release 1

Note!

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

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.

Contents

Notices . . . v
Programming interface . . . v
Trademarks . . . vi

About this book . . . vii
Using your documentation . . . vii
  Where to look for more information . . . vii

Chapter 1. Considerations before migrating to PL/I VSE . . . 1
Product configuration . . . 1
  PL/I VSE and LE/VSE library names . . . 1
Differences in output . . . 2
Differences in PL/I I/O support . . . 3
  Indexed sequential data sets . . . 3
  Card devices . . . 3
  Unlabeled tapes . . . 4
  Extents for regional data sets . . . 4
  The MEDIUM ENVIRONMENT option . . . 4
  Format of REGIONAL(3) data sets . . . 5
Differences in PLIDUMP . . . 6
  Differences in using PLIDUMP . . . 6
  Differences in the output produced by PLIDUMP . . . 6
Differences in condition handling . . . 7
  Other condition handling . . . 8
Differences in interlanguage communication . . . 8
  COBOL . . . 8
  Assembler . . . 8
  General ILC considerations . . . 9
  Identifying programs with ILC . . . 9
Differences in preinitialized programs . . . 10
Retuning for better performance and storage use . . . 10
  Performance considerations for PL/I applications . . . 10
Differences in run-time options . . . 11
Features of DOS PL/I that are not supported by PL/I VSE . . . 13
  Debugging . . . 13
  Some types of interlanguage communication (ILC) . . . 13
  DOS PL/I Optimizing Compiler modules . . . 13
  Using PLICALLA and PLICALLB . . . 14
  Input/output support . . . 14
Miscellaneous compatibility considerations . . . 15

Chapter 2. Installation considerations . . . 16
Installation requirements . . . 16
Renamed product parts . . . 16
  Changing invocation names . . . 16
User-replaceable modules . . . 16

Chapter 3. Compile-time considerations . . . 17
Dependency on LE/VSE . . . 17
Differences in compile-time options . . . 17

Compile and link JCL . . . 18
Large arrays and aggregates . . . 18
Compatibility considerations for DOS PL/I source code . . . 20
Differences in user return codes . . . 20
Differences in processing for FILE OPEN errors . . . 21
Selecting math routines . . . 21
Storage report changes . . . 21
Compiler message changes . . . 21
  New compiler messages . . . 22
  Changed compiler messages . . . 22
  Compiler messages that are no longer valid . . . 22
Messages that PL/I issues for errors in the PLIXOPT string . . . 23

Chapter 4. Subsystem considerations . . . 24
CICS considerations . . . 24
  The System Initialization Table . . . 24
  Changing the Processing Program Table . . . 24
  Error handling . . . 24
  Macro-level interface . . . 24
  Link-edit changes . . . 25
  SYSTEM(CICS) compile-time option . . . 25
  Run-time output . . . 25
  Abend codes used by PL/I under CICS . . . 25
  Transaction abend handling . . . 25
  Shared Library support . . . 25
DL/I considerations . . . 26
  Interfaces to DL/I . . . 26
  Compile and link considerations . . . 26
  Storage usage considerations . . . 26
  Condition handling . . . 27
SQL/DS considerations . . . 28
  Compiling and linking . . . 28
  Condition handling . . . 28

Bibliography . . . 29
IBM PL/I for VSE/ESA publications . . . 29
IBM Language Environment for VSE/ESA publications . . . 29
VSE/ESA publications . . . 29
Related publications . . . 29
Softcopy publications . . . 29

Index . . . 30

Notices

Programming interface

This book is intended to help the customer migrate from previous releases of the DOS PL/I Optimizing Compiler to IBM PL/I for VSE/ESA (PL/I VSE). This book primarily documents General-use Programming Interface and Associated Guidance Information provided by PL/I VSE.

General-use programming interfaces allow the customer to write programs that obtain the services of PL/I VSE.

However, this book also documents Product-sensitive Programming Interface and Associated Guidance Information provided by PL/I VSE.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of PL/I VSE. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Product-sensitive programming interface

Product-sensitive Programming Interface and Associated Guidance Information...

End of Product-sensitive programming interface

Trademarks

About this book

This book contains information to help you migrate applications from the DOS PL/I Optimizing Compiler (DOS PL/I) to IBM* PL/I for VSE/ESA* (PL/I VSE). It suggests solutions to problems that arise because of differences in support between DOS PL/I and PL/I VSE.

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

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

This book also refers to facilities and services provided by IBM Language Environment* for VSE/ESA (LE/VSE). A general understanding of LE/VSE concepts is also required for readers of this book.

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.

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 on page viii.

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” on page 29.

Chapter 1. Considerations before migrating to PL/I VSE

This chapter discusses issues you should consider before you install PL/I VSE and the run-time product LE/VSE. Items discussed in this chapter can affect your decision to migrate applications to PL/I VSE and LE/VSE. This chapter includes the following sections:

- Product configuration
- Differences in output
- Differences in PL/I I/O support
- Differences in PLIDUMP
- Differences in condition handling
- Differences in interlanguage communication
- Differences in preinitialized programs
- Retuning for better performance and storage use
- Differences in run-time options
- Features of DOS PL/I that are not supported by PL/I VSE
- Miscellaneous compatibility considerations

Product configuration

LE/VSE must be available before you can compile, link-edit, or run a PL/I VSE application. The PL/I VSE compiler must have access to LE/VSE at compile time.

PL/I VSE and LE/VSE library names

As shown in Table 3, PL/I VSE is shipped as a single library and LE/VSE is shipped as two libraries.

Table 3. VSE libraries shipped with PL/I VSE and LE/VSE

Library	**Description**	**Shipped with**
PRD2.PROD	PL/I VSE	PL/I VSE
PRD2.SCEEBASE	LE/VSE modules needed at compile, link-edit, and run time (including CICS/VSE)	LE/VSE

The LE/VSE library PRD2.SCEEBASE must always be available to compile, link, and run PL/I VSE applications. If you use existing JCL, you will need to change the LIBDEF statements to include PRD2.SCEEBASE, unless your partitions have permanent LIBDEFs defined.

Differences in output

Output from PL/I applications is different in the following ways:

- User return codes are now supported, and returned to VSE. This might affect the behaviour of batch job streams. See “Differences in user return codes” on page 20 for details.
- By default, run-time messages now go to the LE/VSE MSGFILE destination rather than to the SYSPRINT STREAM PRINT file. Run-time user output still goes to SYSPRINT. If you want your run-time messages to go to SYSPRINT, specify the MSGFILE(SYSPRINT) run-time option.

  For more information, see the *LE/VSE Programming Guide.*
- PL/I VSE now directs output for run-time messages and messages from ON condition SNAP to MSGFILE by way of LE/VSE. These messages are no longer directed to the PL/I SYSPRINT stream.
- The format and content of run-time messages are different. If you have applications that analyze run-time output, you must change the applications to allow for the changes. The changes include:
  - The message number in the message prefix is now four digits instead of three digits.
  - The message severity in the message prefix can now be I, W, E, S, or C.
  - The message text of some of the messages has been enhanced.
  - Some new messages have been introduced.

  For more information, see the *LE/VSE Programming Guide* and the *LE/VSE Debugging Guide and Run-Time Messages.*
- The output produced by PLIDUMP has changed. For detailed information, see “Differences in PLIDUMP” on page 6.
- The output of the run-time storage report has changed. For more information, see the *LE/VSE Programming Guide.*
- The PLIXHD declaration is no longer used to provide the heading for the run-time storage report. You can use the CEE5RPH callable service of LE/VSE to specify the heading. If you do not use CEE5RPH, the heading includes the enclave name, date, and time of execution.
- Run-time error messages now give offset values that are relative to the start of the external procedure, rather than relative to the start of the block that contains the statement.

Differences in PL/I I/O support

Some devices and data set types that were supported by DOS PL/I are no longer supported, and the implementation of some types of data set support has changed.

Indexed sequential data sets

PL/I VSE does not support indexed sequential data sets. If INDEXED appears in the ENVIRONMENT option of a PL/I file, PL/I VSE will treat it as a VSAM KSDS. The following ENVIRONMENT options are related to indexed sequential data sets, and will be ignored if specified:

```
ADDBUF(n)
EXTENTNUMBER(n)
HIGHINDEX(device-type)
INDEXAREA[(index-area-size)]
INDEXED
INDEXMULTIPLE
KEYLOC(n)
NOWRITE
OFLTRACKS(n)
```

Card devices

PL/I VSE does not support the following ENVIRONMENT options related to direct control of card devices:

```
ASSOCIATE(filename)
COLBIN
FUNCTION(function)
OMR
RCE
STACKER(n)
```

This affects the following devices:

- IBM 2560 Card Read Punch
- IBM 3505 Card Reader
- IBM 3525 Card Punch with optional features
- IBM 5425 Card Read Punch

It is still possible to use these devices in PL/I VSE, provided only a single function of the device is used, and the cards use standard 80-column EBCDIC coding. For example, a PL/I file declared as

```
FILE RECORD SEQUENTIAL ENV(F RECSIZE(80) MEDIUM(SYS014))
```

can be used to read cards from an IBM 2560 Card Read Punch, but the 2560 cannot be used at the same time (via the ASSOCIATE option) for punching or printing on cards.

If you need to use any of these unsupported functions, you can implement them with an assembler subroutine that can be called from a PL/I program. PL/I VSE provides a sample program and subroutine that can be used as a base for implementing these facilities. The sample routines are:

IELCARDR
: This is a PL/I program that reads cards and copies the data to a disk file. It does not contain a file declaration for the card file—it reads the cards by calling an assembler subroutine.

IELCARDG
: This is an assembler subroutine that processes the cards. It contains a DTFCD macro for the card file. The DTF specifies MODE=E for standard EBCDIC card coding, but this can be changed to allow for special features such as optical mark reading or column binary mode. Comments in the assembler code indicate which DTF parameters relate to the DOS PL/I ENVIRONMENT options.

Note: These programs are provided as samples only. They should not be used in a production environment without modification and thorough testing.

Unlabeled tapes

PL/I VSE does not support the NOLABEL ENVIRONMENT option. To determine whether a tape file is labeled or not, PL/I checks if label information was provided in the run-time JCL. If there is a TLBL statement for the file, and the device assignment that matches the MEDIUM ENVIRONMENT option is a tape device, then PL/I treats the tape as labeled. If there is no TLBL statement and the device assignment that matches the MEDIUM ENVIRONMENT option is a tape, then PL/I treats it as an unlabeled data set.

The NOTAPEMK option still applies, but it relates to an unlabeled tape as described above, rather than to the NOLABEL option.

Extents for regional data sets

PL/I VSE ignores the EXTENTNUMBER ENVIRONMENT option. The number of extents for a regional data set is determined at run time from the JCL.

The MEDIUM ENVIRONMENT option

PL/I VSE ignores the *device-type* specification in the MEDIUM ENVIRONMENT option. PL/I will attempt to resolve all device specifications at run time rather than compile time.

The entire MEDIUM option can be omitted if the data set resides on DASD. PL/I relates the file name with the DLBL statement at run time, and resolves the logical unit number from the EXTENT statement.

Format of REGIONAL(3) data sets

When a REGIONAL(3) data set with F-format records is created, PL/I VSE formats the data set by writing dummy records in all the unused record locations. This ensures that all tracks in the data set are filled with fixed-length records, either real or dummy. When the data set is accessed, PL/I VSE expects the data set to be formatted in this way. Records are added by converting dummy records to real, and deleted by converting real records to dummy.

DOS PL/I differs from this in that there are no dummy records—the formatting process consists of clearing all the tracks of the data set by writing a capacity record at the beginning of each track. Records are added by writing into the first available space on the relevant track, and deletion is not supported.

Data set conversion

PL/I VSE programs can *read* REGIONAL(3) data sets created by DOS PL/I, but any attempt to update the data set will fail. REGIONAL(3) F-format data sets created with DOS PL/I must be reformatted.

Note: The COMPAT ENVIRONMENT option allows PL/I VSE to create and use REGIONAL(3) F-format data sets in the same way as DOS PL/I. However, if you recompile all your DOS PL/I applications with PL/I VSE, then we recommend that you also migrate your DOS PL/I-created REGIONAL(3) F-format data sets to the PL/I VSE format.

PL/I VSE provides a sample program to reformat REGIONAL(3) data sets. The program is called IELREG3C, and it contains:

- A REGIONAL(3) INPUT file (REG3IN). This is the data set created by DOS PL/I that is to be converted.
- A REGIONAL(3) OUTPUT file (REG3OUT). The program will create and format this data set, and copy the input data to it. This data set will then be suitable for accessing by PL/I VSE programs.
- Preprocessor variables %R3RECLEN and %R3KEYLEN. These contain the data set record length and key length. You must change these and recompile the program for each data set to be converted.
- Space to insert code to calculate the region number. Before each record is written to the output data set, the program needs to calculate the region number for the record, and include it in the key. The output file has the DIRECT attribute, so the region numbers do not need to be presented in ascending sequence.

The program should be run for each data set when you convert from DOS PL/I to PL/I VSE.

The program source code contains comments to assist you to make the relevant changes.

Differences in PLIDUMP

PLIDUMP now produces an LE/VSE-style dump. The way you use PLIDUMP is slightly different, and the dump output is significantly different.

In the following description, the term *compile unit* refers to the primary entry point of the external procedure. *Compile unit name* refers to the name of the external procedure.

Differences in using PLIDUMP

The way you use PLIDUMP has changed:

- The dump output filename can be CEEDUMP, PLIDUMP, or PL1DUMP. If you do not define one of these files, LE/VSE sends the dump output to SYSLST. The record length for the data set associated with the dump output file must be at least 133 bytes to prevent dump records from wrapping, not the 121 bytes required by DOS PL/I.
- If you use ILC, the dump file can also contain output related to COBOL/VSE.
- The identifier character string is limited to 60 bytes, and will be truncated if necessary. DOS PL/I supported 90 bytes for the identifier character string.

Differences in the output produced by PLIDUMP

The format of the output produced by PLIDUMP has changed in the following ways:

- The traceback section lists the compile unit name associated with each entry point name. When the entry point is a secondary entry point, the primary entry point name associated with the actual entry point is not listed.
- The traceback section contains offsets relative to the address of the compile unit, and offsets relative to the address of the real entry point.
- Diagnostic messages are in a separate section; they are no longer part of the traceback section.
- Condition handler save areas no longer appear in the traceback section of the dump. If you specify the BLOCKS (`'B'`) option of PLIDUMP, the condition handler save areas appear in the Block section of the dump. If you do not specify the BLOCKS option, the condition handler save areas do not appear in the dump.
- When you specify the STORAGE (`'H'`) option, the hex dump output is directed to a dump sublibrary if OPTION SYSDUMP is in effect and a dump sublibrary is defined for the partition (by a LIBDEF DUMP command). This is separate from the output for the other PLIDUMP options.
- Run-time messages now contain offsets relative to the compile unit. You can use these offsets to help you find the statement that is in error. To do this, match the compile unit offset from the message with the offset given in the pseudo-assembler listing that the compiler produces when you specify the LIST compile-time option.

  Messages in SNAP dumps and in the traceback section of PL/I dumps contain offsets relative to the entry point of the procedure. The offset table gives the offset of each statement in the program. Thus, you can readily identify the corresponding statement by looking up a given offset in the table.

- If the program was compiled with the TEST compile-time option, and a BEGIN block has a label, the BEGIN block is identified as “Label:BEGIN block.” Otherwise, the BEGIN block is identified as “%BLOCK*nn*,” where *nn* is the block count for the BEGIN block.
- Compiler-generated ILC subroutines now appear in the traceback section. They are identified as the compile unit name concatenated with the suffix ILC.
- PL/I library routines that have LE/VSE-defined Program Prologue Areas (PPAs) are identified by name in the dump. If the library routines do not have LE/VSE PPAs, they are identified as “Library(PL/I).”
- PLIDUMP now conforms to National Language Support standards.

Differences in condition handling

In general, PL/I VSE condition handling continues to operate as it did for DOS PL/I. However, its behavior might be slightly different in the following ways:

- The diagnostic message for an ERROR condition is issued only if there is no ERROR ON-unit established, or if the ERROR ON-unit does not recover from the condition by using a GOTO out of block. Therefore, you can use a GOTO out of the ERROR ON-unit to avoid a message for a PL/I ERROR condition.

  However, for PL/I conditions whose implicit action includes printing a message and raising the ERROR condition, the message is issued before control is given to an established ERROR ON-unit.

  Table 4 summarizes the differences in condition handling between DOS PL/I and PL/I VSE.

Table 4. ERROR ON-unit messages—DOS PL/I and PL/I VSE

Condition		**No ON-units**	**ERROR ON-unit no GOTO**	**ERROR ON-unit GOTO**
ERROR condition raised[1]	DOS PL/I	message	message prior to ON-unit	message prior to ON-unit
	PL/I VSE	message	message after ON-unit	no message
ZERODIVIDE condition raised[2]	DOS PL/I	message	message prior to ON-unit	message prior to ON-unit
	PL/I VSE	message	message prior to ON-unit	message prior to ON-unit

Notes:

1. Taking the square root of a negative number, data exception, etc.
2. With no ZERODIVIDE ON-unit; thus, implicit action is taken. Message is printed, ERROR condition is raised.

ON ERROR SNAP provides a SNAP traceback message that is issued before the ERROR ON-unit gains control. The SNAP traceback message is not identical to a regular ERROR message.

- The ERRCOUNT run-time option controls the maximum number of severe errors that are processed before the program terminates. For example, ERRCOUNT(3) specifies that the program terminates if more than three severe errors are encountered. To maintain compatibility, specify ERRCOUNT(0), which specifies that there is no limit. For more information, see the *LE/VSE Programming Guide.*
- PL/I run-time message numbers are now in the form IBM*nnnnx*, where *nnnn* represents the message number and *x* represents the severity of the message.
- The condition-handling behavior of the LINK from assembler is now clearly defined. For detailed information, see the *LE/VSE Programming Guide.*

  Note: With EXEC CICS LINK under CICS, the ON-unit for FINISH in the parent enclave might not receive control.

Other condition handling

PL/I VSE handles other conditions according to LE/VSE semantics. For more information, see the *LE/VSE Programming Guide.*

Differences in interlanguage communication

In the LE/VSE environment, interlanguage communication (ILC) with PL/I VSE is supported only for COBOL and Assembler.

COBOL

ILC between PL/I VSE and COBOL is supported only if the COBOL routines were compiled with:

- VS COBOL II Release 3.2 or later

or

- IBM COBOL for VSE/ESA

If you have any programs that contain both COBOL and PL/I routines, and the COBOL routines were compiled with an earlier release of COBOL, then when you recompile any of the PL/I routines with PL/I VSE, you must also recompile **all** of the COBOL routines with one of the supported COBOL compilers. (This is in addition to recompiling all of the PL/I routines, because DOS PL/I routines are also not supported by LE/VSE.)

PL/I ILC with COBOL is now reentrant. To write reentrant ILC applications, you must specify OPTIONS(REENTRANT) for all of your external procedures and ensure that you do not modify static variables. You must recompile all procedures that communicate with COBOL, using PL/I VSE, and you must re-link the application with LE/VSE.

Assembler

The format of PL/I locator/descriptor blocks has changed. If any PL/I programs call assembler subroutines without specifying OPTIONS(ASSEMBLER), you will need to change the assembler routines to handle the new format locator/descriptors. For a description of the format of locator/descriptor blocks, see *PL/I VSE Programming Guide.*

Assembler routines are no longer required to preserve the contents of register 12 for PL/I error handling.

For assembler programs calling PL/I subroutines, there are differences in the entry points used, and in how the PL/I environment is initialized. Entry points PLICALLA and PLICALLB are no longer supported, and entry points PLISTART and PLIMAIN are now called CEESTART and CEEMAIN. For details on LE/VSE environment initialization, see the *LE/VSE Programming Guide.*

General ILC considerations

Some PL/I applications that use ILC might behave differently under LE/VSE:

- Condition handling might behave differently. The major causes of differences in condition handling are that the INTER option is now ignored, and that PL/I condition handling facilities can deal with conditions occurring in non-PL/I routines whether you specify INTER or not.
- With DOS PL/I applications that used ILC, the environment initialization and termination of the involved languages, including PL/I, could occur multiple times. With LE/VSE, there is only one run-time environment, and language-specific initialization and termination occurs only once. Changes in behavior that you might see include opening and closing of files, releasing of allocated storage, and establishing ON-units.

For a complete description of how ILC works in the LE/VSE run-time environment, see the *LE/VSE Programming Guide.*

Identifying programs with ILC

Product-sensitive programming interface

You can use the CSECT names in your existing phases to help identify PL/I programs that have ILC. By searching for certain CSECT names, you can identify some phases that contain features that need special consideration when recompiling with PL/I VSE. Table 5 shows the names of CSECTs that contain ILC support code.

Note: Some ILC features cannot be identified by searching for CSECT names.

Table 5. Using CSECTs to identify problem load modules

CSECT name	**Module type**
IBMBILC1	ILC load module
IBMBIEC1	ILC module in which PL/I calls COBOL
IBMBIEF1	ILC module in which PL/I calls FORTRAN
IBMBIEP1	ILC module in which COBOL, FORTRAN, or RPG calls PL/I

End of Product-sensitive programming interface

Differences in preinitialized programs

The PL/I preinitialized program interface using the PLICALLA and PLICALLB entry points is no longer supported. The PL/I environment can only be preinitialized with LE/VSE services. See the *LE/VSE Programming Guide* for details.

PL/I VSE does **not** support the following techniques for PL/I environment initialization:

- An assembler program calls a PL/I MAIN program using entry point PLICALLA or PLICALLB. This initializes the PL/I environment, and then the PL/I program reinvokes the assembler routine. The assembler routine can then call the same or other PL/I routines multiple times without re-establishing the PL/I environment.
- An assembler routine mimics PL/I by including a PLIMAIN entry point which contains the address of a PL/I procedure. This allows the assembler routine to call any number of PL/I subroutines, without the need for a MAIN procedure.

Note: The PLICALLB entry point is still available for use by DL/I programs, but its parameters have changed. Any user-written assembler programs that call PLICALLB will not work under PL/I VSE.

Retuning for better performance and storage use

After migrating your applications to PL/I VSE, you should retune them to maximize performance and storage efficiency. This section provides an overview of performance tuning considerations. For more information on tools you can use to tune your programs, see the *PL/I VSE Programming Guide.*

Performance considerations for PL/I applications

When an existing DOS PL/I application is recompiled with PL/I VSE, run-time performance is generally comparable to the run-time performance of the same application running under DOS PL/I. However, storage usage increases. The amount of increase depends on the specification of the storage options.

Though there are specific areas where performance with LE/VSE improves, there are a few areas where performance with LE/VSE is degraded. Performance considerations vary depending on the characteristics of an application. This section summarizes some of the performance considerations for compatibility and migration. This summary is not exhaustive.

You can experience better run-time performance compared to DOS PL/I in the following areas:

- ILC between COBOL and PL/I
- many of the LE/VSE math routines

You might experience slower run-time performance compared to DOS PL/I in the following areas:

- Condition handling
- LE/VSE math routines that use degrees
- CICS support, especially EXEC CICS LINK

When applications are re-linked with LE/VSE, the size of the phases usually decreases. Resident routines are no longer link-edited into your executable phases; for each resident routine that formerly would have been present, a small stub is link-edited into the phase. The stub accesses the formerly resident routine.

If you use the IBM-supplied default storage options for LE/VSE, PL/I VSE applications use approximately one megabyte more storage during run-time than DOS PL/I. You can decrease the required storage by reducing the default storage values. You can find out how much storage your application actually needs by using the RPTSTG(ON) run-time option.

Table 6 shows recommended storage values for batch applications under LE/VSE. These values apply to both PL/I VSE and COBOL/VSE applications.

Table 6. Recommended storage values for batch applications under LE/VSE

STACK initial size	128K
STACK increment size	64K
LIBSTACK initial size	16K
LIBSTACK increment size	8K
HEAP initial size	32K
HEAP increment size	32K
ANYHEAP initial size	16K
ANYHEAP increment size	8K
BELOWHEAP initial size	8K
BELOWHEAP increment size	4K

The RPTSTG(ON) option tells you how much storage your application uses in the various LE/VSE storage classes (STACK, HEAP, LIBSTACK, and so on). It also tells you your application's minimum storage requirements and the number of GETVIS and FREEVIS requests done to obtain storage. You might want to tune your application to minimize the number of GETVIS and FREEVIS requests performed. Also, before putting the application into production, be sure to specify the RPTSTG(OFF) option so that no storage report is generated.

Condition handling can impact performance. For example, if truncation or overflow occurs repeatedly, the work required to handle this condition can increase execution time. For optimal performance, ensure that truncation does not occur.

Differences in run-time options

LE/VSE run-time options replace PL/I run-time options. Most PL/I run-time options have an equivalent LE/VSE run-time option that provides the same function. This section describes differences in the use of run-time options.

You should adapt your applications to allow for the following differences:

- LE/VSE recognizes run-time options specified in the PLIXOPT string. However, for easier future migration, you should delete all PLIXOPT strings from your programs and specify run-time options in your JCL. See the *PL/I VSE Programming Guide* for details.

- The COUNT option is ignored.
- The FLOW option is ignored.
- The ERRCOUNT option limits the number of errors that are handled at run time. ERRCOUNT(0) specifies that there is no limit, which is equivalent to the way DOS PL/I worked.
- The DEPTHCONDLMT run-time option limits the extent to which conditions can be nested. To maintain compatibility, specify DEPTHCONDLMT(0), which means there is an unlimited depth.
- The RPTSTG option replaces the REPORT option.
- The TRAP option replaces both SPIE and STAE. Either SPIE or STAE (or both) will map to the TRAP(ON) option; NOSPIE and NOSTAE, specified together, map to the TRAP(OFF) option.
- The STACK and HEAP options together replace ISASIZE. STACK controls the initial allocation for LIFO storage, and HEAP controls non-LIFO storage.
- The HEAP option is always in effect. This means that when you allocate storage for BASED and CONTROLLED variables, the storage always comes from HEAP storage. The storage does not come from a PL/I Initial Storage Area (ISA). HEAP(,,BELOW) is required for AMODE 24 applications.
- The XUFLOW option determines whether the UNDERFLOW condition is raised when underflow occurs. XUFLOW(AUTO) preserves PL/I semantics with regard to raising the UNDERFLOW condition.

Table 7 maps the DOS PL/I run-time options to the equivalent LE/VSE options.

Table 7. PL/I and LE/VSE run-time options

DOS PL/I	**LE/VSE**
COUNT|NOCOUNT	None
FLOW|NOFLOW	None
ISASIZE(init_size)	STACK(init_size) for LIFO storage HEAP(init_size) for non-LIFO storage
REPORT	RPTSTG(ON)
NOREPORT	RPTSTG(OFF)
SPIE STAE	TRAP(ON)
NOSPIE NOSTAE	TRAP(OFF)

The following run-time options provide compatibility with the DOS PL/I defaults:

```
ERRCOUNT(0)
DEPTHCONDLMT(0)
TRAP(ON)
XUFLOW(AUTO | ON)
```

For more information about run-time options, see the *LE/VSE Programming Guide.*

Features of DOS PL/I that are not supported by PL/I VSE

This section summarizes the major features of the DOS PL/I Optimizing Compiler that are not supported by PL/I VSE:

- Debugging
- Some types of interlanguage communication
- DOS PL/I Optimizing Compiler modules
- Using PLICALLA and PLICALLB
- Input/output support

Debugging

PL/I VSE does not support these DOS PL/I debugging features:

- The CHECK condition
- The COUNT compile-time option
- The COUNT run-time option
- The FLOW compile-time option
- The FLOW run-time option

Note: Conceptually, the PL/I NOCOUNT and NOFLOW options are always in effect during compilation and at run time.

Some types of interlanguage communication (ILC)

PL/I VSE does not support these DOS PL/I ILC features:

- Communication with FORTRAN
- Communication with DOS/VS COBOL
- Communication with RPG

These restrictions are not diagnosed at run time.

If you do not know whether your existing phases contain ILC, you can check them for the CSECT names that identify ILC support. For more information, see “Identifying programs with ILC” on page 9.

DOS PL/I Optimizing Compiler modules

Executable modules

LE/VSE does not support DOS PL/I phases. However, your DOS PL/I phases can continue to co-exist with PL/I VSE on your VSE/ESA system, provided you retain the DOS PL/I Transient Library. Phases shipped with LE/VSE and PL/I VSE have different names than those shipped with the DOS PL/I Transient Library, so your old programs will continue to run.

However, LE/VSE does not support phases that were compiled and linked with the DOS PL/I compiler, so your DOS PL/I programs must run as separate job steps, not in an LE/VSE environment. Programs compiled and linked under LE/VSE can not FETCH modules that were compiled with DOS PL/I.

Object modules

Object modules compiled with DOS PL/I can not be link-edited with modules compiled with PL/I VSE. When you recompile a module with PL/I VSE, you must ensure that **all** modules linked with that module have been recompiled, so that no DOS PL/I object modules are included in the executable program.

Using PLICALLA and PLICALLB

PL/I VSE does not support the entry points PLICALLA or PLICALLB (except that PLICALLB is still supported for DL/I as a special case—see “DL/I considerations” on page 26 for details). If you wish to call preinitialized PL/I programs from assembler language routines you should use LE/VSE-defined preinitialization services.

Under LE/VSE, if you attempt to enter PL/I via PLICALLA or PLICALLB, your program will abend.

For more information about PL/I-defined preinitialized programs, see “Differences in preinitialized programs” on page 10 and the *LE/VSE Programming Guide.*

Input/output support

PL/I VSE does not support the following devices and data set types:

Indexed sequential data sets
PL/I VSE does not support indexed sequential data sets in native mode. If a file declaration has the INDEXED attribute, PL/I VSE will treat the file as a VSAM KSDS.

Card devices
PL/I VSE does not support the following ENVIRONMENT options related to card devices:

```
ASSOCIATE(filename)
COLBIN
FUNCTION(function)
OMR
RCE
STACKER(n)
```

Direct control of card devices such as the IBM 2560 Card Read Punch is not possible in PL/I VSE. However, PL/I VSE provides a sample assembler subroutine that can be called from a PL/I program to provide these functions. For details, see “Card devices” on page 3.

See “Differences in PL/I I/O support” on page 3 for a description of the changes in PL/I I/O support.

Miscellaneous compatibility considerations

The following interfaces are not supported by PL/I VSE. In general, LE/VSE provides similar functions, and you should convert your programs to use the LE/VSE services.

Entry points PLICALLA and PLICALLB

Entry points PLICALLA and PLICALLB are no longer supported. To call a PL/I program from Assembler, you should use the functions provided by LE/VSE. See the *LE/VSE Programming Guide* for details.

Entry points PLISTART and PLIMAIN

The compiler no longer generates the entry points PLISTART and PLIMAIN. The entry point names are now CEESTART and CEEMAIN.

Under LE/VSE, assembler programs can no longer mimic PL/I main procedures by including an entry point called PLIMAIN. If you have assembler main programs calling PL/I subroutines, you must use standard LE/VSE services.

Linkage conventions

With LE/VSE, assembler programs that call a PL/I routine must follow the calling conventions defined by LE/VSE. That is, register 13 must point to a save area, save areas should be properly back-chained, and the first word of the save area should be zero. For detailed information, see the *LE/VSE Programming Guide.*

When a CICS program of any language issues an EXEC CICS LINK or EXEC CICS XCTL to a PL/I main procedure, an LE/VSE enclave is created. The main procedure operates in the newly created enclave. If the calling program is LE/VSE-enabled, it is already running in its own enclave, so the new enclave is said to be nested. For more information about the behavior of enclaves, see the *LE/VSE Programming Guide.*

In a batch environment, enclave nesting is not supported, so it is not possible to call a PL/I main procedure unless the calling program is not already LE/VSE-enabled. In this case, the LE/VSE environment must be initialized before the program calls the PL/I main procedure. When the PL/I program is called, a new LE/VSE enclave is created and the PL/I program runs in the new enclave. The *LE/VSE Programming Guide* give details on initializing the LE/VSE environment.

Chapter 2. Installation considerations

This chapter contains detailed information about installation. It outlines issues you must consider when you install PL/I VSE. This chapter includes the following sections:

- Installation requirements
- Renamed product parts
- User-replaceable modules

Installation requirements

You must install LE/VSE before you install PL/I VSE. LE/VSE must be available whenever you compile, link-edit, or run a PL/I VSE program.

For more information about installation requirements, see the *LE/VSE Programming Guide* and *PL/I VSE Installation and Customization Guide.*

Renamed product parts

The prefix for the PL/I VSE compiler modules is IEL1.

In addition, the following name changes have been made:

- For each DOS PL/I resident library module IBMB*xxxx*, there are two corresponding modules:
  - A library stub IBMS*xxxx* that is link-edited with your program
  - A run-time module IBMR*xxxx* that is loaded dynamically if needed
- All DOS PL/I IBMF*xxxx* modules for CICS specifics are now shipped with LE/VSE, and are called IBMY*xxxx*.

Changing invocation names

You should change PL/I invocation names when you migrate to PL/I VSE.

In your compile JCL, you should change your EXEC statement to use the program name IEL1AA instead of PLIOPT.

User-replaceable modules

The replacement of module IBMBEER with a user-written module to handle CICS transaction abends is no longer supported. See “Transaction abend handling” on page 25 for details.

Chapter 3. Compile-time considerations

This chapter describes the following compile-time considerations:

- Dependency on LE/VSE
- Differences in compile-time options
- Compile and link JCL
- Large arrays and aggregates
- Compatibility considerations for DOS PL/I source
- Differences in user return codes
- Selecting math routines
- Storage report changes
- Compiler message changes
- Messages that PL/I issues for errors in the PLIXOPT string

Dependency on LE/VSE

LE/VSE must be available whenever you compile a PL/I VSE application.

Differences in compile-time options

The following DOS PL/I compile-time options are no longer supported. If you specify them, the compiler will ignore them.

```
CATALOG('name')
CHARSET([48|60][EBCDIC|BCD])
COUNT|NOCOUNT
DECK|NODECK
DUMP|NODUMP
DYNBUF|NODYNBUF
FLOW[(n,m)]|NOFLOW
LIMSCONV|NOLIMSCONV
LINK|NOLINK[(W|E|S])
WORKFILE(device)
```

In some cases there is no equivalent option in PL/I VSE, and in some cases there is an option that provides a similar function. Each option is listed here with a note about its replacement. For full details of the new compile-time options, see the *PL/I VSE Programming Guide.*

CATALOG('name')
The name of the object module is now supplied by the NAME compiler option.

CHARSET([48|60][EBCDIC|BCD])
There is no equivalent option.

COUNT|NOCOUNT
There is no equivalent option.

DECK|NODECK
The JCL OPTION DECK statement tells the compiler to produce an object deck on SYSPCH.

DYNBUF|NODYNBUF
There is no equivalent option.

FLOW[(n,m)]|NOFLOW
There is no equivalent option.

LIMSCONV|NOLIMSCONV
There is no equivalent option.

LINK The JCL OPTION statement is now used instead of the compiler LINK option. Use OPTION LINK to link-edit a temporary phase, and OPTION CATAL for a permanent phase. With OPTION CATAL, you can also use the NAME compiler option to generate a PHASE statement for the linkage editor.

NOLINK[(W|E|S)]
PL/I VSE does not contain an equivalent option. If you want to bypass execution of the linkage editor when there are errors in the compile, you can use conditional JCL statements. See "Compile and link JCL" for details.

WORKFILE(device)
There is no equivalent option.

Compile and link JCL

In DOS PL/I, the NOLINK compile-time option specified the highest severity level of compiler errors that could occur before the link-edit would be disallowed. The compiler disallowed the link-edit by resetting the relevant JCL option (OPTION LINK or CATAL), and this caused the link to fail with a STATEMENT OUT OF SEQUENCE message.

PL/I VSE will unconditionally reset the LINK option if it encounters an error of severity S (severe) or higher (return code 12 or higher). This is equivalent to NOLINK(S) in DOS PL/I.

However, PL/I VSE will also set the system return code to indicate the highest severity error that occurred in the compile. If you want to suppress the link-edit step of a compile job for errors of lower severity, you can test the return code in your JCL. For example:

```
// EXEC IEL1AA
   (program source)
/*
// IF $RC LT 8 THEN
// EXEC LNKEDT
```

This will bypass the linkedit step if the compiler detects any errors of severity E or higher. This is equivalent to NOLINK(E) in DOS PL/I.

The *PL/I VSE Programming Guide* contains a list of the compiler return codes and error severity levels.

Large arrays and aggregates

To allow for larger arrays and aggregates, PL/I now uses FIXED BIN(31) subscripts instead of FIXED BIN(15). This should have minimal effect on your existing programs because PL/I will convert your FIXED BIN(15) subscripts internally to FIXED BIN(31) before using them to resolve array references.

However, some of the array-handling and storage control built-in functions now return FIXED BIN(31) values instead of FIXED BIN(15), and you should check your programs for the use of these functions. The built-in functions that are changed are:

```
ALLOCATION
DIM
HBOUND
LBOUND
```

Programs that rely on the format of the data returned by these built-in functions will need to be changed. Figure 1 shows the differences in data format for these built-in functions between DOS PL/I and PL/I VSE.

```
/* Built-in functions DIM, HBOUND, LBOUND and ALLOCATION return    */
/* fullword values when compiled with PL/I VSE and return halfword */
/* values when compiled with DOS PL/I.                             */


DC510: PROC OPTIONS(MAIN);

DCL BA219(25:28) BIT(4) AUTOMATIC VARYING ALIGNED;
DCL BU114(2,2) BIT(80) CONTROLLED UNALIGNED;

ALLOCATE BU114;

CALL INT;

INT: PROC;
  DCL BA221(4) BIT(15) ALIGNED INIT(HBOUND(BA219,1),
                                    LBOUND(BA219,1),
                                    ALLOCATION(BU114));
BA221(4) = ALLOCATION(BU114);
PUT DATA ((BA221(I) DO I = 1 TO 4));

DCL BA222(4) BIT(31) ALIGNED INIT(HBOUND(BA219,1),
                                  LBOUND(BA219,1),
                                  ALLOCATION(BU114));
BA222(4) = ALLOCATION(BU114);
PUT DATA ((BA222(I) DO I = 1 TO 4));
END INT;

END DC510;


Run-time output if compiled with DOS PL/I:

 BA221(1)='000000000011100'B                 BA221(2)='000000000011001'B
 BA221(3)='000000000000001'B                 BA221(4)='000000000000001'B;
 BA222(1)='0000000000111000000000000000000'B     BA222(2)='0000000000110010000000000000000'B
 BA222(3)='0000000000000010000000000000000'B     BA222(4)='0000000000000010000000000000000'B;


Run-time output if compiled with PL/I VSE:

 BA221(1)='000000000000000'B                 BA221(2)='000000000000000'B
 BA221(3)='000000000000000'B                 BA221(4)='000000000000000'B;
 BA222(1)='0000000000000000000000000011100'B     BA222(2)='0000000000000000000000000011001'B
 BA222(3)='0000000000000000000000000000001'B     BA222(4)='0000000000000000000000000000001'B;
```

Figure 1. Differences in built-in functions between DOS PL/I and PL/I VSE

Compatibility considerations for DOS PL/I source code

Source code compatibility with DOS PL/I is supported with the following exceptions:

- CHARSET(48 and BCD) is no longer supported.
- Graphic DBCS varies slightly from old EGCS in that the shift-in and shift-out code points are fixed.
- Processing of %INCLUDE statements now delimits text inclusions with “begin” and “end” comments.
- The preprocessor now treats character codes outside the range of '40'X through 'FF'X as delimiters if they are not part of a string constant.
- Suffixes that follow string constants are not replaced by the preprocessor—whether these are legal PL/I suffixes or not—unless there is a delimiter between the ending quotation mark of the string and the first letter of the suffix. For example:

  ```
  %DCL (GX, XX) CHAR;
  %GX='||FX';
  %XX='||ZZ';
  DATA = 'STRING'GX;
  DATA = 'STRING'XX;
  DATA = 'STRING' GX;
  DATA = 'STRING' XX;
  ```

  under DOS PL/I produces the source:

  ```
  DATA = 'STRING'||FX;
  DATA = 'STRING'||ZZ;
  DATA = 'STRING' ||FX;
  DATA = 'STRING' ||ZZ;
  ```

  whereas, under PL/I VSE it produces:

  ```
  DATA = 'STRING'GX;
  DATA = 'STRING'XX;
  DATA = 'STRING' ||FX;
  DATA = 'STRING' ||ZZ;
  ```

Differences in user return codes

PL/I will now pass the latest return code back to the operating system at the end of the job step. The return code used will be the last one set by a CALL PLIRETC statement. This will affect the running of your jobs if your JCL contains any conditional job steps that depend on the $RC value of prior steps.

PLIRETC is also recognized if it is called from the main procedure. DOS PL/I always ignored calls to PLIRETC unless they came from a subordinate procedure.

The PL/I built-in functions that support user return codes now have a precision of FIXED BIN(31) under PL/I. This change affects the following:

- You can use the PLIRETC built-in subroutine to set return codes with a value greater than 999 under PL/I VSE.

  In DOS PL/I, PLIRETC accepted FIXED BIN(31) values, but, if the value was greater than 999, PLIRETC reset the value to 999 and issued an informational message.

- The PLIRETV built-in function returns a value of precision FIXED BIN(31) instead of FIXED BIN(15).

If your program depends on the precision of the above built-in functions, you might need to make changes to your source application.

Differences in processing for FILE OPEN errors

In DOS PL/I, a failure in file open processing such as `'NO FORMAT 1 LABEL'` causes the program to abend. However, PL/I VSE intercepts operating system open failures and drives the PL/I UNDEFINEDFILE condition. If the application has not been coded to recover from open failures, the job step terminates with a return code 3000. This may result in differences in JCL processing. For example, JCL set up to handle file allocation failures through abend processing may need to be changed to operate on return code processing.

Selecting math routines

You can choose to use math routines that produce results consistent with either DOS PL/I or with LE/VSE. This choice also affects PL/I language elements, such as exponentiation.

You can obtain results consistent with DOS PL/I by specifying an INCLUDE statement in your link-edit JCL as follows:

```
INCLUDE IBMSDOSM
```

For more information, see the *PL/I VSE Programming Guide.*

Storage report changes

The PLIXHD variable is no longer used as the heading in storage reports. The identifier PLIXHD is no longer special; you can declare it and use it as you would declare and use any other variable.

To supply a heading for the run-time storage reports, use the CEE5RPH callable service of LE/VSE. See the *LE/VSE Programming Guide* for details.

Compiler message changes

This section lists the numbers of PL/I compiler messages that have changed. For detailed descriptions of messages, see *PL/I VSE Compile-Time Messages and Codes.*

Notes:

1. Compiler messages can be presented in a long form or a short form depending on the setting of the compile-time option LMESSAGE or SMESSAGE. Except for the changes listed below, the long form is the same as the corresponding DOS PL/I message.
2. The messages produced for run-time options specified in the PLIXOPT string are different. In most cases, these messages now refer you to the description of a similar LE/VSE run-time message. For more information about messages

produced for run-time options specified in the PLIXOPT string, see “Messages that PL/I issues for errors in the PLIXOPT string” on page 23.

3. The severity of the preprocessor messages IEL0115I, IEL0163I, and IEL0201I has been reduced to W.

New compiler messages

The following compiler messages have been added:

IEL0025I	IEL0148I	IEL0560I	IEL0669I
IEL0026I	IEL0149I	IEL0583I	IEL0670I
IEL0030I	IEL0205I	IEL0584I	IEL0983I
IEL0031I	IEL0227I	IEL0585I	IEL0985I
IEL0032I	IEL0369I	IEL0586I	IEL0989I
IEL0033I	IEL0376I	IEL0587I	IEL0990I
IEL0035I	IEL0378I	IEL0588I	IEL0995I
IEL0036I	IEL0379I	IEL0589I	IEL2232I
IEL0037I	IEL0383I	IEL0590I	IEL2261I
IEL0038I	IEL0384I	IEL0591I	IEL2262I
IEL0039I	IEL0478I	IEL0592I	IEL2263I
IEL0040I	IEL0537I	IEL0593I	IEL2264I
IEL0041I	IEL0540I	IEL0594I	IEL2270I
IEL0042I	IEL0548I	IEL0595I	IEL2271I
IEL0048I	IEL0557I	IEL0596I	IEL2272I
IEL0059I	IEL0558I	IEL0597I	IEL2273I
			IEL2274I

Changed compiler messages

The following compiler messages have been changed:

IEL0024I	IEL0047I

Compiler messages that are no longer valid

The following compiler messages are no longer valid:

IEL0007I	IEL0701I	IEL0822I	IEL0845I
IEL0008I	IEL0728I	IEL0823I	IEL0846I
IEL0010I	IEL0729I	IEL0824I	IEL0847I
IEL0017I	IEL0732I	IEL0825I	IEL0965I
IEL0020I	IEL0733I	IEL0826I	IEL0972I
IEL0021I	IEL0734I	IEL0829I	IEL0973I
IEL0022I	IEL0736I	IEL0831I	IEL0974I
IEL0029I	IEL0737I	IEL0832I	IEL0975I
IEL0044I	IEL0738I	IEL0833I	IEL0976I
IEL0062I	IEL0739I	IEL0839I	IEL0977I
IEL0063I	IEL0813I	IEL0841I	IEL0978I
IEL0064I	IEL0814I	IEL0842I	IEL0979I
IEL0167I	IEL0815I	IEL0843I	IEL0980I
IEL0547I	IEL0821I	IEL0844I	IEL0981I

Messages that PL/I issues for errors in the PLIXOPT string

The PLIXOPT variable is a varying-length character string that contains run-time options you can specify at compile time. The messages that the compiler produces to diagnose errors in these options have changed. In most cases, the PL/I messages now list an associated LE/VSE message that you should read for more information about the error.

The following messages describe different types of problems that can occur with the run-time options specified in a PLIXOPT string.

Message IEL0950I (Warning)

A severe error occurred in the PLIXOPT string. Generally, this message indicates that the error is severe enough to cause the parsing of the string to fail.

You must correct the errors that cause this message.

Message IEL0951I (Warning)

An error occurred in a run-time option in the PLIXOPT string. This message indicates that the string contains an item that is not a valid run-time option. This message is issued for items that are not recognized as valid run-time options, including run-time options that are no longer supported.

You must correct the errors that cause this message.

Message IEL0952I (Informational)

A possible problem exists with a run-time option in the PLIXOPT string. This message is issued for PL/I run-time options that have been replaced with similar LE/VSE run-time options. The PL/I options are automatically converted to the appropriate LE/VSE options, but some of the LE/VSE options might not function exactly as the PL/I options did.

You should convert these PL/I options to the appropriate LE/VSE option, and check to see if the LE/VSE option is different from the PL/I option.

Message IEL0953I (Informational)

A possible incompatibility exists in the support for a run-time option in the PLIXOPT string. This message is issued for PL/I run-time options that have been replaced by similar LE/VSE run-time options, but might not be supported exactly as DOS PL/I supported them.

For example, this message is issued for the PL/I NOSPIE and NOSTAE options because both options map to the LE/VSE TRAP option. TRAP(ON) implies both SPIE and STAE, and TRAP(OFF) implies both NOSPIE and NOSTAE; under LE/VSE, there is no support that is equivalent to the support that DOS PL/I provided for the combinations SPIE and NOSTAE, or NOSPIE and STAE.

Chapter 4. Subsystem considerations

This chapter discusses considerations for the following subsystems:

- CICS
- DL/I
- SQL/DS*

CICS considerations

LE/VSE provides the CICS support required for PL/I VSE, with CICS/VSE Version 3 Release 2 or later. You can also run DOS PL/I Release 6 programs in the same CICS partition, subject to some system and subsystem support differences. This section summarizes the system and subsystem migration considerations for applications running under CICS.

The System Initialization Table

PL/I VSE does not need the specification PLI=YES in the System Initialization Table (SIT). PL/I VSE programs are supported in CICS by LE/VSE functions, not by a CICS-PL/I interface. However, you should keep the PLI=YES specification to continue support for your DOS PL/I programs. When all of these have been recompiled with PL/I VSE, you can remove PLI=YES from the SIT.

Changing the Processing Program Table

If your CICS Processing Program Table (PPT) already contains entries for the DOS PL/I transient library, you should keep these entries so that your DOS PL/I CICS programs will continue to run. You can delete these entries when all of your DOS PL/I programs have been recompiled with PL/I VSE.

In addition to these entries, you should add the PPT entries supplied by IBM Language Environment for VSE/ESA to support your PL/I VSE programs. For more information about adding these entries, see the *LE/VSE Installation and Customization Guide.*

The PPT entries for your application programs need to specify PL/I as their programming language (PGMLANG=PLI or LANGUAGE(PLI) in the DEFINE PROGRAM command).

Error handling

A diagnostic message is issued only if there is no ERROR ON-unit established in the program, or the ERROR ON-unit does not recover from the condition by using a GOTO out of block. For more information, see the *LE/VSE Programming Guide.*

Macro-level interface

The CICS macro-level interface is no longer supported.

Link-edit changes

The PL/I-CICS interface modules DFHPL1I and DFHEPI have been replaced by an LE/VSE-CICS module DFHELII. You must change your link-edit JCL to include DFHELII instead of DFHPL1I and DFHEPI.

The CICS PL/I shared library is no longer supported, so you must remove all INCLUDE PLISHRE statements from your link-edit JCL.

SYSTEM(CICS) compile-time option

The default SYSTEM compile-time option is SYSTEM(VSE). However, because in CICS programs the parameter to the main PL/I procedure is not a varying-length character string, the compiler will generate a diagnostic message if SYSTEM(VSE) is specified or defaulted. To prevent this, you should compile all CICS programs with SYSTEM(CICS).

Run-time output

All PL/I run-time output, including messages, output to SYSPRINT, and output resulting from PLIDUMP and ON condition SNAP invocations is now transmitted to a CICS transient data queue CESE. LE/VSE ignores the MSGFILE option under CICS. Figure 2 shows the format of the output data queue.

ASA	Terminal id	Transaction id	B	Date/Time YYYYMMDDHHMMSS	B	Data
1	4	4	1	14	1	132

Figure 2. PL/I output data queue

In addition, PL/I transient queues CPLI and CPLD are no longer used. You only need to keep your Destination Control Table (DCT) entries for CPLI and CPLD for co-existence with DOS PL/I.

Abend codes used by PL/I under CICS

The APLx abend codes that were issued under DOS PL/I Release 6 are no longer issued. Instead, LE/VSE-defined abend codes are issued. For more information about LE/VSE abend codes, see the *LE/VSE Debugging Guide and Run-Time Messages.*

Transaction abend handling

PL/I no longer calls the IBMBEER abend facility when a program is terminated as a result of the ERROR condition being raised. If you want to control the abend handling and allocation of abend codes, you will need to use standard CICS and LE/VSE services. See the *LE/VSE Programming Guide* for details.

Shared Library support

The DOS PL/I Shared Library is no longer supported under PL/I VSE and LE/VSE.

When recompiling your CICS programs with PL/I VSE, you must remove the INCLUDE PLISHRE statements from your link-edit JCL.

DL/I considerations

LE/VSE provides DL/I support for PL/I VSE, with DL/I DOS/VS Version 1 Release 10 or higher. This section summarizes the migration considerations for DL/I applications.

Interfaces to DL/I

The following DL/I interfaces called from PL/I remain supported:

- PLITDLI
- EXEC DLI

Compile and link considerations

All DL/I **batch** application programs should be compiled with the SYSTEM(DLI) or SYSTEM(DL1) compile-time option. DL/I CICS programs should be compiled with SYSTEM(CICS). All other considerations for DL/I CICS programs are covered in the preceding CICS discussion. The remainder of this section is concerned only with DL/I batch programs.

CALL-level programs

The JCL used for compiling and linking DL/I batch programs remains the same as before, except that the PL/I compiler name is IEL1AA, not PLIOPT.

Note that the PLICALLB entry point is still used, but its parameters have changed. DL/I should be the only user of PLICALLB—if you have any assembler programs that use PLICALLB, you should change them to use standard LE/VSE initialization procedures. See the *LE/VSE Programming Guide* for details.

Command-level programs

The JCL used for compiling and linking EXEC DLI batch programs remains the same as before, with two differences:

- The PL/I compiler name is now IEL1AA, not PLIOPT.
- The interface module IBMBPJRA is a DOS PL/I module. The equivalent module in PL/I VSE is IBMRPJRA. You will need to change the INCLUDE IBMBPJRA statement in your link-edit JCL to INCLUDE IBMRPJRA.

Storage usage considerations

DL/I Version 1 Release 10 is the minimum release required for LE/VSE applications. There is also a PTF for DL/I V1R10 to allow 31-bit addressing. If the DL/I software at your installation does not have this enabling PTF, then all DL/I application programs need to ensure that the parameters and data areas passed to DL/I reside below the 16-megabyte line.

PL/I programs compiled with PL/I VSE will normally be link-edited with AMODE(31) and RMODE(ANY), which will not guarantee that the program's DL/I parameters are below the line.

The technique used to place the parameters below the line differs depending on the PL/I storage class used for the parameters. You should check the variables used as DL/I parameters and data areas in your PL/I programs, and use the following techniques to ensure that they are placed below the line. (Run-time options mentioned here are described in full in the *LE/VSE Programming Guide.*)

Static storage

PL/I static storage is held as part of the link-edited phase. If your program has DL/I parameters in static storage, you should link-edit the program with RMODE(24).

Automatic storage

The allocation of automatic storage is controlled by the LE/VSE STACK run-time option. To ensure that automatic storage is allocated below the line, specify STACK(,,BELOW).

Controlled storage

The allocation of controlled storage is governed by the LE/VSE HEAP run-time option. To ensure that this storage is allocated below the line, specify HEAP(,,BELOW).

Based storage

The allocation of based storage depends on how the program manages the based variable:

- If the based variable is allocated in an AREA variable, its storage class is inherited from the area that contains it.
- If the variable is allocated, but not in an AREA variable, its allocation is governed by the LE/VSE HEAP run-time option in the same manner as controlled storage.
- If the variable is never allocated, its storage class will be that of the variable that its locator is pointing to.

Condition handling

The method of handling abnormal conditions in the DL/I environment has changed. LE/VSE now gets control whenever an abnormal condition occurs, and processes the condition according to the run-time options that are specified. Then, if abnormal termination is to occur, LE/VSE will pass control to DL/I's abend processing routine, and DL/I will ensure that all relevant data base backout and cleanup processing is completed.

To ensure that this happens correctly, the following LE/VSE run-time options should be specified:

TRAP(ON)
This is the default. It fully enables the LE/VSE condition handler.

ABTERMENC(ABEND)
This is **not** the default. It specifies that LE/VSE terminates the enclave with an abend, ensuring that DL/I will get control to do its cleanup processing. If ABTERMENC(RETCODE) is specified or defaulted, the installation must have an assembler exit routine that sets a flag specifying that abend processing is to occur. (The exit routine is called CEEBXITA, and is described in the *LE/VSE Programming Guide.*)

Note: DL/I still requires that UPSI bit 7 be set to 0 to enable it to perform its cleanup processing. For details, see the *DL/I DOS/VS Application Programming* publications.

For more information on LE/VSE run-time options, see the *LE/VSE Programming Guide.*

SQL/DS considerations

PL/I VSE and LE/VSE support applications that access SQL/DS data bases, both in CICS and as VSE batch programs.

Compiling and linking

There is no change to the way that SQL/DS programs are compiled and link-edited, except that the compile JCL should execute program IEL1AA instead of PLIOPT.

Condition handling

SQL/DS must be allowed to handle any conditions that can result in abnormal termination of the PL/I program, so that it can perform any data base backout and cleanup procedures. How this is specified will depend on the environment under which the program is run.

In a VSE batch environment, in *single-user* mode, you must specify the LE/VSE TRAP(OFF) run-time option. This ensures that SQL/DS will be given control in the event of an error, so that it can perform any necessary data base cleanup.

In VSE batch *multiple-user* mode, all the data base processing is handled by the SQL/DS partition, so the PL/I program should run with TRAP(ON) to allow LE/VSE to perform its own error handling.

In a CICS environment, the LE/VSE TRAP(ON) option is always in effect. Any abnormal conditions are handled by LE/VSE and CICS, and the CICS transaction backout facility is used to clean up SQL/DS data bases.

Bibliography

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

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

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

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

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

A
abend codes
  CICS considerations 25
ABTERMENC run-time option, DL/I programs 27
ADDBUF environment option 3
ALLOCATION built-in function 19
assembler considerations, ILC 8
assembler invocation of PL/I 14
ASSOCIATE environment option 3, 14
automatic storage, DL/I programs 27

B
based storage, DL/I programs 27
built-in functions
  array-handling 19
  PLIRETV 20
  storage control 19
built-in subroutines
  PLIRETC 20

C
card devices, support for 3
CATALOG compile-time option 17
CEE5RPH callable service 2, 21
CEEBXITA exit routine
  for DL/I abend handling 27
CEEDUMP 6
CHARSET compile-time option 17
CHECK condition 13
CICS considerations
  abend codes used by PL/I 25
  Destination Control Table 25
  discussion of 24
  error handling 24
  link-edit changes 25
  macro-level interface 24
  Processing Program Table 24
  run-time output 25
  Shared Library support 25
  SYSTEM compile-time option 25
  System Initialization Table 24
  transient queues 25
  VSE library 1
COBOL considerations, ILC 8
COLBIN environment option 3, 14
COMPAT environment option 5
compatibility considerations
  DOS PL/I source code 20
  miscellaneous 15
  preprocessor 20
compile and link JCL 18
compile unit
  definition of 6
  name, definition of 6
compile-time considerations
  compile-time options 17
  compiler messages
    changed 22
    discussion of changes 21
    new 22
    no longer valid 22
  installing LE/VSE 17
  large arrays and aggregates 18
  storage reports 21
  user return codes 20
compiler invocation name
  for VSE JCL, IEL1AA 16
compiler messages
  changed 22
  compile-time considerations 21
  discussion of changes 21
  new 22
  no longer valid 22
condition handling
  discussion of 7
  DL/I programs 27
  performance impact 11
  SQL/DS programs 28
conditional link-edit 18
controlled storage, DL/I programs 27
converting REGIONAL(3) data sets 5
COUNT compile-time option 17
COUNT run-time option 11, 13
CSECTs
  discussion of 9
  using 9

D
debugging 13
DECK compile-time option 17
DEPTHCONDLMT run-time option 12
Destination Control Table (DCT) 25
device type, MEDIUM option 4
DFHELII
  for EXEC CICS programs 25
DFHEPI
  for EXEC CICS programs 25
DFHPL1I
  for EXEC CICS programs 25
DIM built-in function 19

DL/I considerations
  discussion of 26
  EXEC DLI programs 26
  interfaces 26
  interfaces to 26
  storage usage 26
  SYSTEM compile-time option 26
DOS PL/I
  phases 13
  source code compatibility 20
  transient library
    in the CICS PPT 24
  unsupported features
    CHARSET compile-time option 17
    CHECK condition 13
    COUNT compile-time option 13, 17
    COUNT run-time option 13
    DOS PL/I phases 13
    DUMP compile-time option 17
    DYNBUF compile-time option 17
    FLOW compile-time option 13, 17
    FLOW run-time option 13
    interlanguage communication with DOS/VS COBOL 13
    interlanguage communication with FORTRAN 13
    interlanguage communication with RPG 13
    LIMSCONV compile-time option 17
    overview 13
    PLICALLA 14
    PLICALLB 14
    WORKFILE compile-time option 17
DUMP compile-time option 17
DYNBUF compile-time option 17

E
entry point
  PLIMAIN 15
  PLISTART 15
ERRCOUNT run-time option 12
error handling, CICS considerations 24
EXEC DLI interface 26
EXEC DLI programs 26
EXTENTNUMBER environment option
  indexed sequential data sets 3
  regional data sets 4
extents for regional data sets 4

F
FETCHed modules
  support for 13
FILE OPEN errors, differences in processing 21
FLOW compile-time option 17
FLOW run-time option 11, 13
format of REGIONAL(3) data sets 5
FUNCTION environment option 3, 14

H
HBOUND built-in function 19
HEAP run-time option 12
  DL/I programs 27
HIGHINDEX environment option 3

I
I/O support, differences in 3
IBMBEER 16, 25
IBMBPJRA DL/I interface module 26
IEL1 prefix 16
IEL1AA
  compiler invocation name
    changing 16, 26, 28
    for PL/I VSE 16
IELCARDG sample assembler subroutine 4
IELCARDR sample PL/I program 4
IELREG3C sample program 5
ILC (interlanguage communication)
  CSECTs
    discussion of 9
    using 9
  differences in 8
  enabled languages 8
  identifying programs with 9
  support for 13
INDEXAREA environment option 3
INDEXED environment option 3
indexed sequential data sets 3, 14
INDEXMULTIPLE environment option 3
installation considerations
  differences in output 2
  discussion of 1
  product configuration, LE/VSE 1
  product configuration, SCEEBASE 1
  renamed product parts 16
  requirements 16
  user-replaceable modules 16
installing LE/VSE, compile-time considerations 17
INTER option 9
interlanguage communication (ILC)
  CSECTs
    discussion of 9
    using 9
  differences in 8
  enabled languages 8
  identifying programs with 9
  support for 13
invocation names, changing 16
ISASIZE run-time option 12

J
JCL
  compile and link 18
  differences in processing FILE OPEN errors 21

K
KEYLOC environment option 3

L
large arrays and aggregates, compile-time considerations 18
LBOUND built-in function 19
LIMSCONV compile-time option 17
LINK compile-time option 17
link-edit considerations
  CICS 25
  math routines, selecting 21

M
macro-level interface, CICS considerations 24
math routines, selecting 21
MEDIUM environment option 4
messages
  compiler
    changed 22
    discussion of changes 21
    new 22
    no longer valid 22
  PLIXOPT string errors
    discussion of 23
    IEL0950I 23
    IEL0951I 23
    IEL0952I 23
    IEL0953I 23
miscellaneous compatibility considerations 15
MSGFILE option, under CICS 25

N
NOLABEL environment option 4
NOLINK compile-time option 17
NOTAPEMK environment option 4
NOWRITE environment option 3

O
OFLTRACKS environment option 3
OMR environment option 3, 14
open failures, differences in JCL processing 21
output
  differences in 2
  from PLIDUMP, differences in 6

P
performance
  considerations for PL/I applications 10
  recommended storage values 11
  retuning for 10
phases
  DOS PL/I support 13
PLICALLA entry point
  support for 9, 14, 15
PLICALLB entry point
  DL/I considerations 26
  support for 9, 14, 15
PLIDUMP
  CEEDUMP 6
  changes to 6
  output produced by 6
  output under CICS 25
  using 6
PLIMAIN
  assembler ILC considerations 9
  description of 15
PLIOPT
  compiler invocation name
    changing 16, 26, 28
PLIRETC 20
PLISHRE 25
PLISTART
  assembler ILC considerations 9
  description of 15
PLITDLI DL/I interface 26
PLIXHD 2, 21
PLIXOPT string
  messages issued
    discussion of 23
    IEL0950I 23
    IEL0951I 23
    IEL0952I 23
    IEL0953I 23
preinitialized programs 10
preprocessor compatibility 20
Processing Program Table 24
product configuration 1
product parts, renamed 16
programs, preinitialized 10

R
RCE environment option 3, 14
REGIONAL(3) data sets, reformatting 5
REPORT run-time option 12
retuning
  discussion of 10
  performance considerations 10
RPSTG run-time option 12

run-time options
  compatibility with DOS PL/I 12
  differences in 11
run-time output, CICS considerations 25
run-time performance, tuning 10

S
sample program for converting REGIONAL(3) data sets 5
sample programs for card device support 4
SCEEBASE, installation considerations 1
Shared Library support
  CICS considerations 25
source code compatibility 20
SPIE run-time option 12
SQL/DS considerations 28
STACK run-time option 12
  DL/I programs 27
STACKER environment option 3, 14
STAE run-time option 12
static storage, DL/I programs 27
storage
  reports, compile-time considerations 21
  usage
    DL/I programs 26
    retuning for 10
  values
    recommended 11
    RPTSTG option 11
subsystem considerations
  CICS 24
  DL/I 26
  SQL/DS 28
SYSTEM compile-time option
  CICS considerations 25
  DL/I considerations 26
System Initialization Table 24

T
tape data sets, unlabeled 4
transient data destinations 25
TRAP run-time option 12
  DL/I programs 27
  SQL/DS programs 28
tuning
  run-time performance 10

U
UNDEFINEDFILE condition 21
UNDERFLOW condition 12
unlabeled tape data sets 4
unsupported DOS PL/I features
  Assembler invocation of PL/I 14

unsupported DOS PL/I features *(continued)*
  debugging
    CHECK condition 13
    COUNT compile-time option 13
    COUNT run-time option 13
    FLOW compile-time option 13
    FLOW run-time option 13
  DOS PL/I phases 13
  interlanguage communication
    with DOS/VS COBOL 13
    with FORTRAN 13
    with RPG 13
  overview 13
user return codes, compile-time considerations 20
user-replaceable modules
  installation considerations 16

W
WORKFILE compile-time option 17

X
XUFLOW run-time option 12

We'd Like to Hear from You

IBM PL/I for VSE/ESA
Migration Guide
Release 1

Publication No. SC26-8056-00

Please use one of the following ways to send us your comments about this book:

- Mail—Use the Readers' Comments form on the next page. If you are sending the form from a country other than the United States, give it to your local IBM branch office or IBM representative for mailing.
- Fax—Use the Readers' Comments form on the next page and fax it to this U.S. number: 800-426-7773.
- Electronic mail—Use one of the following network IDs:
  - Internet: COMMENTS@VNET.IBM.COM

  Be sure to include the following with your comments:

  - Title and publication number of this book
  - Your name, address, and telephone number if you would like a reply

Your comments should pertain only to the information in this book and the way the information is presented. To request additional publications, or to comment on other IBM information or the function of IBM products, please give your comments to your IBM representative or to your IBM authorized remarketer.

IBM may use or distribute your comments without obligation.

Readers' Comments

IBM PL/I for VSE/ESA
Migration Guide
Release 1

Publication No. SC26-8056-00

How satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Technically accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐
Grammatically correct and consistent	☐	☐	☐	☐	☐
Graphically well designed	☐	☐	☐	☐	☐
Overall satisfaction	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

May we contact you to discuss your comments? ☐ Yes ☐ No

Name	Address
Company or Organization	
Phone No.	

Readers' Comments
SC26-8056-00

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

Department J58
International Business Machines Corporation
PO BOX 49023
SAN JOSE CA 95161-9945

Fold and Tape **Please do not staple** Fold and Tape

SC26-8056-00

Cut or Fold
Along Line

File Number: S370-34
Program Number: 5686-069

Printed in U.S.A.

IBM PL/I for VSE/ESA Publications

GC26-8052	Fact Sheet
GC26-8055	Licensed Program Specifications
SC26-8056	Migration Guide
SC26-8057	Installation and Customization Guide
SC26-8053	Programming Guide
SC26-8054	Language Reference
SX26-3836	Reference Summary
SC26-8058	Diagnosis Guide
SC26-8059	Compile-Time Messages and Codes

SC26-8056-00