IBM Language Environment for VSE/ESA

Customization Guide

Version 1 Release 4

SC33-6682-03

IBM Language Environment for VSE/ESA

Customization Guide

Version 1 Release 4

SC33-6682-03

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

Fourth Edition (December 2001)

This edition applies to Version 1 Release 4 Modification Level 2 of IBM Language Environment for VSE/ESA, 5686-094, and to any subsequent releases and modifications until otherwise indicated in new editions.

Note!
This manual previously had the title *Installation and Customization Guide*. Since the contents of the installation chapters are now contained in the *VSE/ESA Planning*, the title is now *Customization Guide* only.

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

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

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

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

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

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

Contents

Figures

Tables

Notices

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

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

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

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

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

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

Trademarks and Service Marks

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

AT	DFSORT	OS/400
C/370	IBM	SAA
CICS	Integrated Language Environment	System/370
CICS/VSE	Language Environment	VisualAge
DB2	OS/390	VSE/ESA

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names, may be trademarks or service marks of others.

About This Book

This book is intended for system programmers and system administrators who plan for, install, customize, and maintain IBM Language Environment for VSE/ESA (LE/VSE).

To use this book, you need to be familiar with the VSE operating system, the publications that describe your system, and job control language (JCL) processing.

Linking To PDF Copies of LE/VSE and VSE/ESA Manuals!
When you open a PDF copy of an LE/VSE manual on the *VSE/ESA* Bookshelf, if you proceed to "Where to Find More Information", you can link to the PDF of an LE/VSE manual or VSE/ESA manual.

What Is LE/VSE?

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

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

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

LE/VSE consists of:

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

LE/VSE is the implementation of Language Environment on the VSE platform. Language Environment is offered on two other platforms: on OS/390 and VM as IBM Language Environment for OS/390 and VM, and on OS/400 as Integrated Language Environment.

LE/VSE-Conforming Languages

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

Table 1. LE/VSE-Conforming Languages

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

Any HLL not listed in Table 1 is known as a *non-LE/VSE-conforming* or, alternatively, a *pre-LE/VSE-conforming* language. Some examples of non-LE/VSE-conforming languages are:

- C/370
- DOS/VS COBOL
- VS COBOL II
- DOS PL/I
- DOS/VS RPG II

.

Only the following products can generate applications that run with LE/VSE:

- LE/VSE-conforming languages
- HLASM using LE/VSE-provided macros (for details, see *LE/VSE Programming Guide*)
- DOS/VS COBOL and VS COBOL II, with some restrictions (see LE/VSE Compatibility with Previous Versions of COBOL below).

LE/VSE Compatibility with Previous Versions of COBOL

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

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

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

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

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

How to Read the Syntax Diagrams

The following rules apply to the notation used in the syntax diagrams contained in this book:

- Read the syntax diagrams from left to right, top to bottom following the path of the line.
- Each syntax diagram begins with a double arrowhead (►►).
- An arrow (→►) at the end of a line indicates that the option, service, or macro syntax continues on the next line. A continuation line begins with an arrow (►—).
- If a syntax diagram contains too many items or groups to fit in the diagram, the syntax is shown by a main syntax diagram and one or more syntax fragments. A syntax fragment is referred to in the main diagram by its fragment name between two vertical bars (|).

  Each syntax fragment appears below the main syntax diagram, and begins and ends with a vertical bar (|). A heading above the fragment indicates the name of the fragment.

  Read each syntax fragment as though it were imbedded in the main syntax diagram.
- IBM-supplied default keywords appear **above** the main path or options path (see the sample on page xvi). In the parameter list, IBM-supplied default choices are underlined.
- Keywords appear in nonitalic capital letters and should be entered exactly as shown. However, some keywords may be abbreviated by truncation from the right as long as the result is unambiguous. In this case, the unambiguous truncation is shown in capital letters in the keyword, for example:

  ```
  ANyheap
  ```
- Words in lowercase letters represent user-defined parameters or suboptions.
- Enter parentheses, arithmetic symbols, colons, semicolons, commas, and greater-than signs where shown.
- Required parameters appear on the same horizontal line (the main path) as the option, service, or macro:

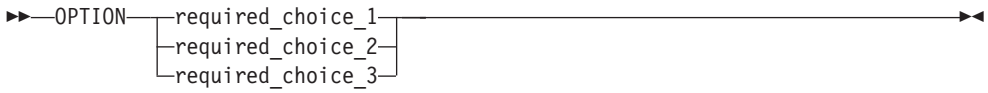
  ```
  ►►—OPTION—required_parameter———————————————————————————————►◄
  ```

- If you can choose from two or more parameters, the choices are stacked one above the other.

  If choosing one of the items is optional, the entire stack appears below the main line.

  ```
  ►►—OPTION———————————————————————————————————————————————————►◄
              ├—optional_parameter_1—┤
              ├—optional_parameter_2—┤
              └—optional_parameter_3—┘
  ```

  If you *must* choose one of the items, one item of the stack appears on the main path:

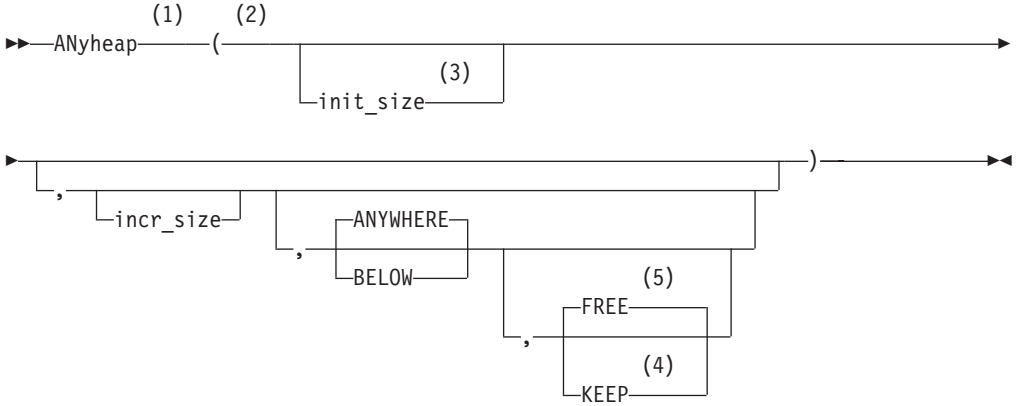


- An arrow returning to the left above a line indicates that an item can be repeated:

OR

- A comma or semicolon included in the repeat symbol indicates a separator that you must include between repeated parameters. These separators must be coded where shown.
- When entering commands, parameters and keywords must be separated by at least one blank if there is no intervening punctuation.
- A double arrow (→◄) at the end of a line indicates the end of the syntax diagram.

The following example demonstrates how to read the syntax notation. Numbers in the example correspond to explanations supplied below the example.



Notes:

1 Keyword with minimum unambiguous truncation shown in capital letters

2 Opening parenthesis (must be specified if any parameters are specified)

3 Optional parameter

4 Optional keyword

5 Optional keyword (IBM-supplied default)

Where to Find More Information

These are the manuals that describe LE/VSE:

Table 2. LE/VSE Publications

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

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

Table 3. VSE/ESA Publications

Publication	Form Number
VSE/ESA Administration	SC33-6705
VSE/ESA Messages and Codes, Volume 1	SC33-6796
VSE/ESA Messages and Codes, Volume 2	SC33-6798
VSE/ESA Messages and Codes, Volume 3	SC33-6799
VSE/ESA Planning	SC33-6703
VSE/ESA System Control Statements	SC33-6713
VSE/ESA System Macros Reference	SC33-6716
VSE/ESA System Macros User's Guide	SC33-6715
VSE/ESA System Upgrade and Service	SC33-6702
VSE/VSAM User's Guide and Application Programming	SC33-6732
VSE/VSAM Commands	SC33-6731
TCP/IP for VSE/ESA IBM Program Setup and Supplementary Information	SC33-6601

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

Table 4. IBM C for VSE/ESA Publications

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

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

Publication	Form Number
Migration Guide	SC09-2423
User's Guide	SC09-2424
Language Reference	SC09-2425
Diagnosis Guide	GC09-2426

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

Table 5. IBM COBOL for VSE/ESA Publications

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

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

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

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

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

Table 7. Debug Tool for VSE/ESA Publications

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

You might also refer to the ...

LE/VSE Home Page

You can find the LE/VSE home page at:

`http://www.ibm.com/servers/eserver/zseries/os/vse/le/`

In addition, you might refer to the ...

VSE/ESA Home Page

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

You can find the VSE/ESA home page at:

`http://www.ibm.com/servers/eserver/zseries/os/vse/`

Softcopy Publications

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

VSE Collection, SK2T-0060

Summary Of Changes

Changes Introduced with LE/VSE 1.4.2

This section describes the changes introduced with LE/VSE 1.4.2:

- Details were provided of the pre-installed LE/VSE structure (the Component Ids, CLCs, and descriptions). For details, see *page 1.*
- The IBM-supplied default run-time settings for ANYHEAP, BELOWHEAP, HEAP, LIBSTACK, and STACK, have been changed for the CICS environment. For details, see "LE/VSE Run-Time Options" on page 69.
- Run-time option TERMTHDACT has new sub-options MSGFL, LSTQ, and *reg_stor_amount*. For details, see "TERMTHDACT" on page 106.

For a summary of *all* the changes introduced with LE/VSE 1.4.2 (that is, changes contained in all re-published LE/VSE manuals), refer to the *VSE/ESA Release Guide*, SC33-6718.

Note!
This manual includes all the relevant information that was previously contained in the *LE/VSE Release Guide*, SC33-6779-00 (September 2000).

Changes Introduced with LE/VSE 1.4.1

This section describes the changes introduced with LE/VSE 1.4.1:

- Because LE/VSE is automatically installed during the installation of VSE/ESA, the original chapters 1 and 3 of this manual were no longer relevant. Instead the *VSE/ESA Planning* contains the installation details of planning to install, and installing LE/VSE.
- New defaults were introduced for the run-time options. For details, see *page 15.*
- The abnormal termination exit for LE/VSE CICS was changed. LE/VSE 1.4.1 was *pre-customized* with a null-exit capability. This null-exit capability allows you to determine the actual cause of a CICS abend, when it occurs for a LE/VSE program. For details, see *page 35.*
- A section was introduced describing how you can de-activate the *LE COBOL* and/or *LE PL/I* components, and how you should then integrate these changes into your CICS sub-systems. For details, see *page 44.*
- COBOL/CICS and COBOL/BATCH run-time no longer used the previous convention of having "duplicate" module names to control environment-specific processing. For details, see *page 45.*
- Where possible, LE/VSE used a safer method for replacing phases. User-customized modules were thereby separated from IBM-shipped code. This new method resulted in minor changes to several shipped customization jobs. For details of this new method, see *page 55*.
- Condition handling was improved by the introduction of new TRAP sub-options. For details, see *page 112.*
- A dynamic alteration facility was introduced to simplify the customization of LE/VSE CICS-wide run-time options without the need to restart CICS. For details, see *page 119.*

- A new function was provided that allows you to print LE/VSE CICS-wide run-time options to your VSE/ESA console. For details, see *page 120.*
- A description was included of what you must do to maintain a separate CICS/VSE system when performing a Fast Service Upgrade (version upgrade) from VSE/ESA 2.4.x to VSE/ESA 2.5. For details, see *page 52.*
- The appendix containing the list of APARs included in this LE/VSE release level, was updated. For details, see *page 175.*

Chapter 1. Planning to Customize LE/VSE

This chapter provides information for planning the customization of LE/VSE. It includes:
- A description of the structure of the pre-installed LE/VSE.
- Deciding whether or not to customize
- Planning to customize the IBM-supplied default run-time option values
- Planning to customize the IBM-supplied LIOCS phases
- Planning to customize the IBM-supplied default assembler user exit
- Planning to customize the IBM-supplied default abnormal termination exit
- Planning to install LE/VSE into the shared virtual area (SVA)
- Planning to customize programming language-specific features

This chapter helps you plan for these customization tasks. See "Chapter 2. Customizing LE/VSE" on page 13 for the actual customization procedure.

How the Pre-Installed LE/VSE Is Structured

The LE/VSE components are shipped as part of the VSE/ESA base system (except for the DBCS locale component). Table 8 lists these new component identifiers (COMP IDs) and component level codes (CLCs):

Table 8. LE/VSE Component IDs and CLCs

Component Id	CLC	Description
5686-066-32	65K	LE Common base, containing information written in: • uppercase and mixed-case U.S. English • Japanese NLF
5686-066-33	65L	LE C-specific base, containing information written in: • uppercase and mixed-case U.S. English • Japanese NLF
5686-066-34	65M	Optional LE DBCS Locale component (see note below)
5686-094-03	6EW	LE COBOL-specific base and CICS, containing information written in: • uppercase U.S. English • Japanese NLF
5686-094-06	6EX	LE PL/I-specific base, containing information written in: • uppercase and mixed-case U.S. English • Japanese NLF

Note: The optional LE/VSE DBCS locale component is shipped on the *VSE/ESA extended base tape.*

Deciding Whether and What to Customize

You need to choose which
- run-time options (see page 2)
- LIOCS phases (see page 5)
- assembler user exits (see page 7)

you will customize. You also need to decide
- whether to install some routines in the shared virtual area (see page 8), and

- whether to customize C locale time information (see page 12).

You should consider whether the IBM-supplied values for the run-time options that come with LE/VSE suit the needs of your site. These values control such features as:
- The national language in which messages appear
- When condition handling is invoked
- How storage is allocated to the heap and stack
- How much storage is allocated above and below the 16MB line
- The format of the program invocation parameters
- Generation of a storage and/or run-time options report
- Shared storage allocations

You should also consider whether the IBM-supplied LIOCS (logical input/output control system) phases suit the needs of your site. These LIOCS phases contain logic routines used by the VSE input/output control system when processing files assigned to card, diskette, or printer devices. The IBM-supplied phases contain logic routines for the most commonly used of these device types. You should ensure that the LIOCS phases contain all the necessary logic routines for the types of devices processed by your LE/VSE-conforming applications.

If you don't want to customize LE/VSE now, you can install it, test it, and put it into production using the IBM-supplied defaults. You can use the instructions in this book to customize LE/VSE later, if you choose. For many of the run-time options, application programmers can override the installation defaults in their code.

Application programmers at your site will be the primary users of LE/VSE. Ask them what defaults they prefer for run-time options, LIOCS phases, and user exits, which affect their work directly. Doing so will ensure that the modifications you make will best support the application programs being developed at your site.

Planning to Customize LE/VSE Run-Time Options

Why Do It

The run-time option values supplied with LE/VSE may not suit the application programmers' needs at your site. Resetting the defaults will save the programmers' time because they will not need to override the run-time option defaults as often.

You should change some of the defaults shipped with LE/VSE if you plan to use PL/I. See "An Example of Customizing LE/VSE Run-Time Options" on page 5.

Choices to Make Now

You should plan which run-time options you want to change. Refer to "Appendix A. LE/VSE Run-Time Options" on page 59 for detailed information about the run-time options, default values, and syntax. **You might not need to change every default.** You can fill in the blanks in Table 9 with the changes you plan to make in the defaults for both batch and CICS processing.

Table 9. Worksheet: Planning to Customize LE/VSE Run-Time Options

Run-time Option	Batch Default Value	Your New Batch Default	CICS Default Value	Your New CICS Default	Page
ABPERC	((NONE),OVR)	_____	N/A[1]	N/A[1]	69

Table 9. Worksheet: Planning to Customize LE/VSE Run-Time Options (continued)

Run-time Option	Batch Default Value	Your New Batch Default	CICS Default Value	Your New CICS Default	Page
ABTERMENC	((ABEND),OVR)	_____	((ABEND),OVR)	_____	71
AIXBLD	((OFF),OVR)	_____	N/A[1]	N/A[1]	73
ALL31	((OFF),OVR)	_____	((ON),OVR)	_____	74
ANYHEAP	((16K,8K,ANYWHERE, FREE), OVR)	_____	((4K,4080,ANYWHERE, FREE), OVR)	_____	75
BELOWHEAP	((8K,4K,FREE),OVR)	_____	((4K,4080,FREE),OVR)	_____	77
CBLOPTS	((ON),OVR)	_____	((ON),OVR)	_____	78
CBLPSHPOP	((OFF),OVR)	_____	((ON),OVR)	_____	79
CHECK	((OFF),OVR)	_____	((OFF),OVR)	_____	80
COUNTRY	((US),OVR)	_____	((US),OVR)	_____	81
DEBUG	((OFF),OVR)	_____	((OFF),OVR)	_____	82
DEPTHCONDLMT	((10),OVR)	_____	((10),OVR)	_____	83
ENVAR	(('),OVR)	_____	(('),OVR)	_____	84
ERRCOUNT	((20),OVR)	_____	((20),OVR)	_____	85
HEAP	((32K,32K,ANYWHERE, KEEP,8K,4K),OVR)	_____	((4K,4080,ANYWHERE, KEEP,4K,4080),OVR)	_____	86
HEAPCHK	((OFF,1,0)OVR)	_____	((OFF,1,0)OVR)	_____	88
LIBSTACK	((8K,4K,FREE),OVR)	_____	((4K,4080,FREE),OVR)	_____	89
MSGFILE	((SYSLST),OVR)	_____	((CESE),OVR)	_____	91
MSGQ	((15),OVR)	_____	((15),OVR)	_____	92
NATLANG	((UEN),OVR)	_____	((UEN),OVR)	_____	93
NOTEST	(ALL,*,PROMPT,'),OVR)	_____	(ALL,*,PROMPT,'),OVR)	_____	109
NOUSRHDLR	((),OVR)	_____	((),OVR)	_____	116
RETZERO	((OFF),OVR)	_____	((OFF),OVR)	_____	94
RPTOPTS	((OFF),OVR)	_____	((OFF),OVR)	_____	95
RPTSTG	((OFF),OVR)	_____	((OFF),OVR)	_____	97
RTEREUS	((OFF),OVR)	_____	N/A[1]	N/A[1]	100
STACK	((128K,128K,BELOW, KEEP),OVR)	_____	((4K,4080,ANYWHERE, KEEP),OVR)	_____	101
STORAGE	((00,NONE, NONE,32K), OVR)	_____	((00,NONE, NONE,0K), OVR)	_____	103
TERMTHDACT	((TRACE,,0),OVR)	_____	((TRACE,MSGFL,0),OVR)	_____	106
TRACE	((OFF,4K,DUMP,LE=0), OVR)	_____	((OFF,4K,DUMP,LE=0), OVR)	_____	111
TRAP	((ON,MAX),OVR)	_____	((ON,MAX),OVR)	_____	112
UPSI	((00000000),OVR)	_____	((00000000),OVR)	_____	115
XUFLOW	((AUTO),OVR)	_____	((AUTO),OVR)	_____	117

Note:

1. The abbreviation N/A is used for not applicable. These options are ignored under CICS.

You also need to choose which sample customization jobs you need to modify and run. Table 10 on page 5 lists the sample jobs provided on the distribution tape to help you customize LE/VSE run-time options. These jobs are included in the PRD2.SCEEBASE sublibrary. See “Changing Run-Time Options Defaults” on page 15 for instructions on how to use these jobs to customize LE/VSE run-time options.

Table 10. Customizing Run-Time Options with Sample Customization Jobs

To	Use this Sample Job
Change installation-wide defaults for run-time options.	CEEWDOPT.Z
Change installation-wide CICS defaults for run-time options.	CEEWCOPT.Z
Create an application-specific run-time options module.	CEEWUOPT.Z

Note: These jobs are available in ICCF Library 62.

An Example of Customizing LE/VSE Run-Time Options

The following are recommended run-time option settings for PL/I VSE applications:

ABTERMENC(ABEND) for compatibility with PL/I and DOS/VS COBOL.

ERRCOUNT(0) required for PL/I.

DEPTHCONDLMT(0) for compatibility with DOS PL/I.

STORAGE(00,NONE,00,32K) for Batch.

STORAGE(00,NONE,00,0K) for CICS.

XUFLOW(AUTO) recommended for PL/I.

Planning to Customize Run-Time LIOCS Phases

Why Do It

When you install LE/VSE, you are provided with default phases for three sets of run-time LIOCS phases that you can customize. The run-time LIOCS phases contain logic routines used by the VSE input/output control system when processing files assigned to card, diskette, or printer devices. When an LE/VSE-conforming HLL application opens a file assigned to one of these device types, LE/VSE loads the appropriate run-time LIOCS phase. LE/VSE then searches the loaded LIOCS phase for the appropriate logic routine. The appropriate logic routine is the routine that contains the code needed to support the attributes (such as device type, record format, number of I/O areas, and whether the file is being opened for input or output) of the file being opened. LE/VSE then stores the address of the routine in the VSE DTF (define the file) control block for the file.

LE/VSE supplies the run-time LIOCS phases because the VSE-supplied LIOCS routines for these devices are shipped with the operating system in object format only, and LE/VSE-conforming HLL applications require the LIOCS routines in loadable-phase format. The supplied phases that you can customize are:

- CEEYCD0 - LIOCS routines for card reader/punch devices
- CEEYDU0 - LIOCS routines for diskette devices
- CEEYPR0 - LIOCS routines for printer devices

The run-time LIOCS phases supplied with LE/VSE contain only the LIOCS routines for the most commonly used combinations of file attributes. The LIOCS phases do not contain LIOCS routines that support all possible combinations of file attributes as this would make the phases unnecessarily large.

If your LE/VSE-conforming HLL application opens a file with a certain set of file attributes, but the necessary LIOCS routine for that set of file attributes is not

included in the supplied LIOCS phase, then you must modify the appropriate run-time LIOCS phase. If LE/VSE is unable to find the appropriate logic routine, it issues message CEE3751S.

Choices to Make Now

You should plan which devices you want to be able to access with your LE/VSE-conforming HLL applications, and what attributes files assigned to those devices will have. Refer to "Appendix B. LE/VSE Run-Time LIOCS Phases" on page 123 for detailed information about the LIOCS phases, default values, and syntax.

Note!:

1. During customization, you are recommended not to delete or change any shipped LE/VSE LIOCS definitions. If LIOCS definitions for dummy card punch, card reader, and printer devices are missing, the startup of the CICS Transaction Server might be severely affected.
2. The Interactive Interface's *Hardware Configuration* dialog also contains definitions for dummy card punch, card reader, and print devices. These definitions reflect the currently-shipped LE/VSE LIOCS. You are recommended not to delete or change any of these definitions.
3. If, however, you do decide to change any of the dummy device defnitions, *ensure you have customized the appropriate LIOCS definitions for these devices!.*

Choose which sample customization jobs you need to modify and run. Table 11 lists the sample jobs provided on the distribution tape to help you customize LE/VSE LIOCS phases. These jobs are included in the PRD2.SCEEBASE sublibrary. See "Changing Run-Time LIOCS Phases Defaults" on page 24 for instructions on how to use these jobs to customize LE/VSE run-time LIOCS phases.

Table 11. Customizing LIOCS Phases with Sample Customization Jobs

To	Use this Sample Job
Change the card-device run-time LIOCS phase.	CEEWDCD0.Z
Change the diskette-device run-time LIOCS phase.	CEEWDDU0.Z
Change the printer-device run-time LIOCS phase.	CEEWDPR0.Z

Note: These jobs are available in ICCF Library 62.

VS COBOL II Users: If you plan to run VS COBOL II programs compiled with the NORES option, and without relink-editing with LE/VSE, you may need to customize a VS COBOL II-only LIOCS phase. Three sample jobs are provided on the distribution tape to help you customize these phases. They are:

- IGZWEQC0.Z - Change the VS COBOL II card-device run-time LIOCS phase
- IGZWEQD0.Z - Change the VS COBOL II diskette device run-time LIOCS phase
- IGZWEQP0.Z - Change the VS COBOL II printer device run-time LIOCS phase

An Example of Customizing Run-Time LIOCS Phases

A COBOL program coded with this input SELECT statement

```
SELECT FILE01 ASSIGN TO SYS001-S-SYS001 RESERVE 1 AREA.
```

with the RESERVE 1 AREA clause, would require a new card-device run-time LIOCS phase, omitting the IOAREA2 attribute, as in the following sample TYPE=ENTRY statement of the CEEXCDMD macro:

```
CEEXCDMD TYPE=ENTRY,TYPEFLE=INPUT,DEVICE=3505,                X
      RECFORM=FIXUNB,RDONLY=YES
```

Omitting the IOAREA2 attribute means that one I/O area is used, which is required for the RESERVE 1 AREA clause.

Planning to Customize User Exits

Why Do It

LE/VSE assembler, high-level language (HLL) and abnormal termination user exits perform functions for enclave initialization, normal and abnormal enclave termination, and process termination. When you install LE/VSE, sample user exits are installed by default. You can modify the exits to suit the needs of your site.

Choices to Make Now

You should plan which features of the user exits you want to customize. Refer to "Appendix C. Customizing LE/VSE User Exits" on page 137 for detailed information about the features of the exits, default values, and syntax.

Choose which sample customization jobs to modify and run. Table 12 lists the sample jobs provided on the distribution tape to help you customize LE/VSE user exits. These jobs are included in the PRD2.SCEEBASE sublibrary.

For instructions on how to use these jobs to customize the user exits, see:
- "Changing the Assembler Language User Exit" on page 30
- "Creating a High-Level Language User Exit" on page 33
- "Creating a User-Written Handler for Compatibility with VS COBOL II and DOS PL/I" on page 33
- "Customizing LE/VSE Abnormal Termination Exits" on page 35

Table 12. Customizing Assembler User Exits with Sample Customization Jobs

To	Use this Sample Job
Change installation-wide assembler language user exit.	CEEWDXIT.Z
Change installation-wide CICS assembler language user exit.	CEEWCXIT.Z
Create an application-specific assembler language user exit.	CEEWUXIT.Z
Identify an abnormal termination exit (Batch).	CEEWDEXT.Z
Identify an abnormal termination exit (CICS).	CEEWCEXT.Z
Change high-level language user exit.	CEEWHLLX.Z
Create USRHDLR program for VS COBOL II-only compatibility	CEEWWCHA.Z
Create USRHDLR program for VS COBOL II and DOS PL/I compatibility	CEEWCCHA.Z

Note: These jobs are available in ICCF Library 62.

An Example of Customizing User Exits

If there is an unhandled condition of severity 2 or greater, the default assembler user exit in batch returns to the system. You can change the default assembler user exit so that it forces an abend for unhandled conditions of severity 2 or greater.

Examples of conditions that are severity 2 or greater include:
- Program interrupts: applicable if TRAP=(ON,MAX) is used
- System abends: applicable if TRAP=(ON,MAX) is used
- Conditions detected by LE/VSE: for example, a program load failure

The ABTERMENC(ABEND) run-time option is an alternative way to force an abend for unhandled conditions of severity 2 or greater.

Planning to Install in the Shared Virtual Area

Why Do It

Placing routines in the shared virtual area (SVA) reduces the overall system storage requirement by making the routines shareable and common to all partitions. Also, initialization/termination (init/term) time is reduced for each application, since load time decreases.

Choices to Make Now

Choose which routines to put in the SVA. "Appendix F. Routines Eligible for the Shared Virtual Area" on page 161 contains a complete list of routines you can place in the SVA.

After you have chosen the routines, select the SVA loadlists to use from Table 14. Alternatively, select the sample jobs you will need to use from Table 15. The samples are included in the PRD2.SCEEBASE sublibrary. See "Placing LE/VSE Routines in the Shared Virtual Area (SVA)" on page 42 for instructions on how to use these jobs to load routines into the SVA.

Table 13 contains a list of the loadlists that are *pre-installed*.

Table 13. Loadlists That Are Pre-Installed in the SVA (VSE/ESA 2.5 Onwards)

Description	Loadlist
LE/VSE base routines	$SVACEE
LE/VSE recommended C run-time routines	$SVAEDCM

Table 14. Installing in the SVA with Supplied SVA Loadlists

To	Use this Loadlist
Add recommended COBOL run-time routines to the SVA	$SVAIGZM
Add eligible COBOL run-time routines to the SVA	$SVAIGZ
Add recommended PL/I run-time routines to the SVA	$SVAIBMM
Add eligible PL/I run-time routines to the SVA	$SVAIBM
Add eligible C run-time routines to the SVA	$SVAEDC

Table 15. Installing in the SVA with Sample Customization Jobs

To	Use this Sample Job
Add eligible LE/VSE base routines to the SVA	CEEWMSVA.Z
Add eligible COBOL run-time routines to the SVA	IGZWESV1.Z
Add eligible PL/I run-time routines to the SVA	IBMSVA1.Z
Add eligible C run-time routines to the SVA	EDCWMSV1.Z
Note: These jobs are available in ICCF Library 62.	

Space Required: Allow space in the 24-bit SVA and the 31-bit SVA for the components of LE/VSE that you need. Table 16 provides an estimate of the amount of space used by routines for use with VSE/ESA 2.6, that can be put in the 24-bit and 31-bit SVA.

Table 16. SVA Space Requirements for the Components of LE/VSE

Phase Name	Can Be Used In	24-Bit Members (Size)	31-Bit Members (Size)	Recommended or Optional?
$SVACEE	LE Base	8 (271K)	22 (1047K)	Recommended
$SVAIGZM	LE COBOL	0 (0K)	3 (128K)	Recommended
$SVAIGZ		32 (165K)	15 (344K)	Optional
$SVAIBMM	LE PL/I	1 (36K)	2 (202K)	Recommended
$SVAIBM		1 (36K)	26 (280K)	Optional
$SVAEDCM	LE C	0 (0K)	6 (1773K)	Recommended
$SVAEDC		1 (0.16K)	18 (2368K)	Optional

Notes:

1. $SVACEE and $SVAEDCM are pre-installed in the SVA. All members contained in these load lists are pre-defined to the CICS CSD file via the `USESVACOPY(YES)` parameter. From VSE/ESA 2.6 onwards, the CICS System Initialization Table (SIT) contains a corresponding setting of `SVA=YES`. Also be aware that LE/VSE modules in the SVA are only used by CICS Transaction Server if SIT SVA=YES is specified, plus USESVACOPY(YES). Refer to the CICS Transaction Server documentation for details.
2. These requirements reflect the amount of storage needed if you include all eligible routines from the listed components in your 24-bit and 31-bit SVA. For a list of each routine's storage requirements and the modules recommended for inclusion in the 24-bit or 31-bit SVA, see "Appendix F. Routines Eligible for the Shared Virtual Area" on page 161.
3. The requirements for the C component do not include the space needed for locales and code page converters, as the sizes of these modules may vary significantly. Check the specific locale or converter you plan to use and add this size to the estimates in this table.
4. Number and total size of members are approximate and may change during product cycle.
5. Use only one of the two "language-dependent load books".
6. The use of a load book is "recommended", providing you use the language listed above.
7. If the use of a load book is "optional", you can use it instead of the recommended load book for a language.
8. Make sure that PSIZE31 is large enough to prevent phases with location mode *any* from being loaded into SVA-24.
9. Load books $SVACEE and $SVAEDCM are included in the startup skeleton SKJCL0 located in ICCF library 59. This startup skeleton is directly used during the VSE/ESA 2.5 Base installation. SKJCL0 can also be used after an FSU, by manually including it.

You can obtain further information on LE/VSE performance considerations, by referring to the VSE/ESA Home Page (whose address is given in "Where to Find More Information" on page xvii).

COBOL Users: The system will load the IBM-supplied default COBOL COBPACKs into the 31-bit SVA (above the 16MB line). As distributed, the COBOL COBPACKs can reside above the 16MB line because they include only those routines that have the attribute RMODE(ANY). If you add a routine with RMODE(24) to a COBOL COBPACK, that COBOL COBPACK must reside below the line. See "Appendix D. Using COBOL with LE/VSE" on page 153 for an overview of the contents of the COBOL COBPACKs and a list of their link-edit attributes. "Tailoring the COBOL COBPACKs" on page 45 gives specific instructions on customizing the COBOL COBPACKs.

CICS Users: If your CICS system is 24-bit storage constrained, you should not include unnecessary routines in the 24-bit SVA. Doing so reduces the amount of 24-bit storage that you can allocate to your CICS partition. Therefore, if you use LE/VSE mainly in the CICS environment, only include the CICS initialization routine, CEECCICS, in the 24-bit SVA. Do not include the batch initialization routine, CEEBINIT.

Note: CICS TS users should be aware that to load phases into the SVA, a SIT table setting of `SVA=YES` should be specified. For details, refer to the *CICS Transaction Server for VSE/ESA System Definition Guide*, SC33-1651.

PL/I Users: If you plan to use Debug Tool for VSE/ESA to debug PL/I VSE applications, consider loading the LE/VSE PL/I phases into the SVA. These phases are listed in Table 41 on page 165. They are named IBM3....

An Example of Installing in the Shared Virtual Area

If your SVA space is limited, you can load just the frequently used modules there. For example, load:

CEEBINIT into the 24-bit SVA, if you use LE/VSE mainly in the batch environment

CEECCICS into the 24-bit SVA, if you use LE/VSE mainly in the CICS environment

CEEPLPKA into the 31-bit SVA

Planning to Tailor the COBOL COBPACKs

Why Do It

You might want to customize the COBOL COBPACKs to:

- Shorten the load time for the COBOL COBPACK by reducing its size.
- Minimize the virtual storage required for an application by eliminating seldom-used routines from main storage.
- Reduce the number of loads for application programs by adding frequently used routines to COBOL COBPACKs.
- Reduce the size of the contents of the SVA.

The COBOL COBPACKs are generally shared among several different applications and cannot be tuned for a specific application. Therefore, ideal COBOL COBPACKs contain only library routines that are common to all application programs.

Choices to Make Now

You need to decide whether to modify the COBOL COBPACKs. If you modify the COBOL COBPACKs, you make a trade-off between use of storage and faster performance of application programs. See Table 17.

Table 17. Making the Trade-off: Performance Time versus Storage Use

Type of COBOL COBPACK	Performance Time	Storage Use
Partially loaded	Slower because more routines are loaded individually	Less virtual and SVA storage used
Fully loaded	Faster because no routines loaded individually	More virtual and SVA storage used

You can use the information in the following sections and the tables in "Appendix D. Using COBOL with LE/VSE" on page 153 to decide which modules to include in your COBOL COBPACKs.

Some Examples of Tailoring the COBOL COBPACKs

You can add or remove routines from the COBOL COBPACKs to reflect the requirements of your site. For example, to include only the group of general routines that your location uses most often, eliminate unnecessary routines from the COBOL COBPACK.

If your installation runs only under CICS, you can eliminate the general routines for ACCEPT and DISPLAY, because you cannot use them with CICS.

If you plan to put your COBOL COBPACKs into the SVA and your SVA space is limited, consider reducing the size of your COBOL COBPACKs. All modules eligible to be in the COBOL COBPACKs are reentrant and are therefore eligible to be stored in the SVA.

Planning to Customize C Locale Time Information

Why Do It

The only category of C locale that you can customize at installation is the locale time category. This locale category describes the time zone difference, the time zone name, and Daylight Savings Time start and end dates.

Choices to Make Now

Decide whether or not you should modify C locale time information for your site. For more detailed information, see "Appendix E. Customizing C Locale Time Information" on page 159.

If you decide to modify C locale time information, you can use job EDCLLOCL.Z in the PRD2.SCEEBASE sublibrary to help you.

Chapter 2. Customizing LE/VSE

You can customize LE/VSE only after installation of the product is complete. "Chapter 1. Planning to Customize LE/VSE" on page 1 provides information on what you can modify, and why you might want to customize LE/VSE. This chapter tells *how* to make the modifications or where to find the necessary coding information to customize LE/VSE to the needs of your site. This chapter consists of these main sections:

Overview of Customization

Table 18 shows the names and purpose of the sample customization jobs provided with LE/VSE.

Table 18. Summary of Customization Jobs for LE/VSE

Description		Customization Job	Page
Run-time options			15
	Installation-wide run-time options default (Batch)	CEEWDOPT	16
	Installation-wide run-time options default (CICS)	CEEWCOPT	18
	Application-specific run-time options	CEEWUOPT	20
Run-time LIOCS phases			24
	Card-device run-time LIOCS phase	CEEWDCD0	27
	Diskette-device run-time LIOCS phase	CEEWDDU0	28
	Printer-device run-time LIOCS phase	CEEWDPR0	29
Assembler language user exit			30
	Installation-wide assembler language user exit (Batch)	CEEWDXIT	30
	Installation-wide assembler language user exit (CICS)	CEEWCXIT	31
	Application-specific assembler language user exit	CEEWUXIT	32
High-level language user exit		CEEWHLLX	33
User-written condition handler			33
	Create USRHDLR program for VS COBOL II-only compatibility	CEEWWCHA	33
	Create USRHDLR program for VS COBOL II and DOS PL/I-only compatibility	CEEWCCHA	33
Abnormal termination exit			35
	Identify abnormal termination exit to LE/VSE (Batch)	CEEWDEXT	38
	Identify abnormal termination exit to LE/VSE (CICS)	CEEWCEXT	40
Place LE/VSE routines in the SVA		CEEWMSVA IGZWESV1 IBMSVA1 EDCWMSV1	42
Tailoring the COBOL COBPACKs		IGZWEPAC IGZWEPCO IGZWEPCC	45
Changing the C locale time information		EDCLLOCL	48

Note: These jobs are available in ICCF Library 62.

Changing Run-Time Options Defaults

From VSE/ESA 2.5 onwards, changes you make to the installation-wide run-time options will no longer replace the initialization phases. As a result, the option *regeneration* no longer replaces the CEECCICS and CEEBINIT initialization phases.

You can change the run-time option defaults for batch, for CICS, and for individual applications. Table 19 summarizes the sample jobs IBM provides to help you customize the run-time options.

Table 19. Sample jobs to change run-time options defaults

Set Defaults For	Sample Job Member	Options Member
Installation-wide batch	CEEWDOPT.Z	CEEDOPT.A
Installation-wide CICS	CEEWCOPT.Z	CEECOPT.A
Application-specific	CEEWUOPT.Z	CEEUOPT.A

Note: These jobs are available in ICCF Library 62.

Note: From VSE/ESA 2.6 onwards, the above sample jobs include SET SDL processing to reload the regenerated option phases, into the SVA.

All of the members listed in Table 19 are installed in the PRD2.SCEEBASE sublibrary.

To change the run-time option defaults, copy the corresponding options member into the sample job in place of the comment, and change the parameters on the CEEXOPT macro invocation in the sample job you are using. Make the options in the CEEXOPT macro match the run-time options you have selected for your installation. See Setting Installation-Wide Default Options with the CEEXOPT Macro for sample invocations of CEEXOPT.

At run time you can verify which LE/VSE run-time options are in effect by using the RPTOPTS run-time option. For more information, see “RPTOPTS” on page 95.

Setting Installation-Wide Default Options with the CEEXOPT Macro

When you run the sample jobs CEEWDOPT.Z or CEEWCOPT.Z, they create the CEEDOPT CSECT, an options control block which establishes the defaults for the options. The jobs invoke CEEXOPT during the assembly of the CEEDOPT module. When you modify CEEXOPT to change installation-wide defaults, you must specify each run-time option as either OVR or NONOVR. OVR means that the option can be overridden at run time. NONOVR means that the option cannot be overridden at run time.

To invoke CEEXOPT, adhere to the syntax of the IBM-supplied template for CEEDOPT (see Figure 1 for batch and Figure 2 for CICS). These are samples and should be compared to the actual code before you attempt to use them.

Figure 1 on page 17 shows the source statements used to generate the IBM-supplied version of CEEDOPT, the batch options module, with the default suboption values for each of the options. Figure 2 on page 19 shows the source statements used to generate the IBM-supplied version of CEECOPT, the CICS

options module, with the default suboption values for each of the options. Note that the default sub-options for several options in CEEDOPT differ from those in CEECOPT.

You may have to check whether the system default values (as shown in Figure 1 on page 17 and Figure 2) agree with the values required for your environment.

Changing the Installation-Wide Run-Time Options Default (Batch)

Use the sample job in the PRD2.SCEEBASE sublibrary member CEEWDOPT.Z to change the installation-wide defaults for the LE/VSE run-time options. Use the worksheet in "Planning to Customize LE/VSE Run-Time Options" on page 2 to select your default values and use the information in "Appendix A. LE/VSE Run-Time Options" on page 59for more detail about the options and their syntax.

These defaults apply to applications running with the LE/VSE library. This includes the C Prelinker and the C/VSE Compiler.

Modifying the JCL for CEEWDOPT

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

---

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
3. If necessary, change the name of the sublibrary specified in the ACC SUBLIB=PRD2.SCEEBASE statement to match the sublibrary where you have installed LE/VSE.
4. Copy member CEEDOPT.A from the PRD2.SCEEBASE sublibrary into job CEEWDOPT in place of the comment line.
5. Change the parameters on the CEEXOPT macro statement in CEEDOPT to reflect the values you have chosen for your installation-wide default batch run-time options.

After you modify the CEEWDOPT job, submit it. The job finishes with a return code of 2 if it runs successfully. The return code of 2 indicates there were unresolved weak external references during the link-edit steps. This return code is normal and does not indicate a problem.

Note: From VSE/ESA 2.6 onwards, the CEEWDOPT job contains SET SDL processing to reload the regenerated CEEDOPT.PHASE into the SVA.

Considerations for DB2 and PL/I Users

1. For DB2 users: Check the Usage notes under the run-time option "ABTERMENC" on page 71.
2. PL/I users should be aware of the third STORAGE parameter ('NONE') which provides migration help:
   - If PL/I variables need to be pre-initialized (because the program did not explicitly initialize these variables), you should set NONE to '00'.
   - However, setting NONE to '00' might reduce the performance of other languages, since it uses the LE/VSE stack storage.

```
*/********************************************************************/
*/*    Language Environment/VSE V1 R4 M2                             */
*/*                                                                  */
*/*    LICENSED MATERIALS - PROPERTY OF IBM                          */
*/*                                                                  */
*/*    5686-066-32-65K (C) COPYRIGHT IBM CORP. 1991, 2001            */
*/*    ALL RIGHTS RESERVED.                                          */
*/*    US GOVERNMENT USERS RESTRICTED RIGHTS - USE, DUPLICATION OR   */
*/*    DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE CONTRACT WITH IBM   */
*/*    CORP.                                                         */
*/*                                                                  */
*/*    LE/VSE Version 1 Release 4 Modification Level 1 Changes :     */
*/*  @01  GWH  05/10/99   Update default runtime options.            */
*/*    LE/VSE Version 1 Release 4 Modification Level 2 Changes :     */
*/*  @02  JH   01/03/12   Update TERMTDACT default run time option.  */
*/*                                                                  */
*/********************************************************************/
CEEDOPT  CSECT
CEEDOPT  AMODE ANY
CEEDOPT  RMODE ANY
         CEEXOPT ABPERC=((NONE),OVR),                                  X
               ABTERMENC=((ABEND),OVR),                          @01C X
               AIXBLD=((OFF),OVR),                                    X
               ALL31=((OFF),OVR),                                     X
               ANYHEAP=((16K,8K,ANYWHERE,FREE),OVR),                  X
               BELOWHEAP=((8K,4K,FREE),OVR),                          X
               CBLOPTS=((ON),OVR),                                    X
               CBLPSHPOP=((OFF),OVR),                            @01C X
               CHECK=((OFF),OVR),                                     X
               COUNTRY=((US),OVR),                                    X
               DEBUG=((OFF),OVR),                                     X
               DEPTHCONDLMT=((10),OVR),                               X
               ENVAR=(''),OVR),                                       X
               ERRCOUNT=((20),OVR),                                   X
               HEAP=((32K,32K,ANYWHERE,KEEP,8K,4K),OVR),              X
               HEAPCHK=((OFF,1,0),OVR),                               X
               LIBSTACK=((8K,4K,FREE),OVR),                           X
               MSGFILE=((SYSLST),OVR),                                X
               MSGQ=((15),OVR),                                       X
               NATLANG=((UEN),OVR),                                   X
               NOTEST=((ALL,*,PROMPT,''),OVR),                        X
               NOUSRHDLR=((),OVR),                                    X
               RETZERO=((OFF),OVR),                                   X
               RPTOPTS=((OFF),OVR),                                   X
               RPTSTG=((OFF),OVR),                                    X
               RTEREUS=((OFF),OVR),                                   X
               STACK=((128K,128K,BELOW,KEEP),OVR),                    X
               STORAGE=((00,NONE,NONE,32K),OVR),                 @01C X
               TERMTHDACT=((TRACE,,0),OVR),                      @02C X
               TRACE=((OFF,4K,DUMP,LE=0),OVR),                        X
               TRAP=((ON,MAX),OVR),                              @01C X
               UPSI=((00000000),OVR),                                 X
               XUFLOW=((AUTO),OVR)
         DC    C'5686-066-32-65K (C) COPYRIGHT IBM CORP. 1991, 2001.'
         DC    C'LICENSED MATERIALS - PROPERTY OF IBM'
         END
```

Notes:

1. Xs are in column 72.
2. You can regenerate your batch-wide options using skeleton CEEWDOPT.

Figure 1. Sample Generation of CEEDOPT Object Module (Batch)

Changing the Installation-Wide Run-Time Options Default (CICS)

Use the sample job in the PRD2.SCEEBASE sublibrary member CEEWCOPT.Z to change the installation-wide defaults for the LE/VSE run-time options under CICS. Use the worksheet in "Planning to Customize LE/VSE Run-Time Options" on page 2 to select your default values and use the information in "Appendix A. LE/VSE Run-Time Options" on page 59 for more detail about the options and their syntax.

Modifying the JCL for CEEWCOPT

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

---

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
3. If necessary, change the name of the sublibrary specified in the ACC SUBLIB=PRD2.SCEEBASE statement to match the sublibrary where you have installed LE/VSE.
4. Copy member CEECOPT.A from the PRD2.SCEEBASE sublibrary into job CEEWCOPT in place of the comment line.
5. Change the parameters on the CEEXOPT macro statement in CEECOPT to reflect the values you have chosen for your installation-wide default CICS run-time options.

After you modify the CEEWCOPT job, submit it. The job finishes with a return code of 2 if it runs successfully. The return code of 2 indicates there were unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

Notes:

1. From VSE/ESA 2.6 onwards, the CEEWCOPT job contains SET SDL processing to reload the regenerated CEECOPT.PHASE into the SVA.
2. After running the modified CEEWCOPT JCL and the job completing sucessfully, you must activate the new runtime options while the current CICS system is active. To do this, you need to use the supplied CICS transaction NEWC (or your own defined transaction code) to perform the new-copy function. (Alternatively you can, of course, restart your CICS system to activate the changes).

```
*/*****************************************************************/
*/*    Language Environment/VSE V1 R4 M2                          */
*/*                                                               */
*/*    LICENSED MATERIALS - PROPERTY OF IBM                       */
*/*                                                               */
*/*    5686-066-32-65K (C) COPYRIGHT IBM CORP. 1991, 2001.        */
*/*    ALL RIGHTS RESERVED.                                       */
*/*    US GOVERNMENT USERS RESTRICTED RIGHTS - USE, DUPLICATION OR */
*/*    DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE CONTRACT WITH IBM */
*/*    CORP.                                                      */
*/*                                                               */
*/*   LE/VSE Version 1 Release 4 Modification Level 1 Changes :   */
*/*  @01  GWH  05/10/99   Update default runtime options.         */
*/*                                                               */
*/*   LE/VSE Version 1 Release 4 Modification Level 2 Changes :   */
*/*  @02  JH   12 MAR 01  New TERMTHDACT suboptions DCR A399.12 + */
*/*                       A399.16                                 */
*/*  @03  GWH  25 Jul 01  Update default runtime options.      @03A */
*/*****************************************************************/
CEEDOPT  CSECT
CEEDOPT  AMODE ANY
CEEDOPT  RMODE ANY
         CEEXOPT ABPERC=((NONE),OVR),                                X
               ABTERMENC=((ABEND),OVR),                              X
               AIXBLD=((OFF),OVR),                                   X
               ALL31=((ON),OVR),                                     X
               ANYHEAP=((4K,4080,ANYWHERE,FREE),OVR),            @03C X
               BELOWHEAP=((4K,4080,FREE),OVR),                   @03C X
               CBLOPTS=((ON),OVR),                                   X
               CBLPSHPOP=((ON),OVR),                                 X
               CHECK=((OFF),OVR),                                    X
               COUNTRY=((US),OVR),                                   X
               DEBUG=((OFF),OVR),                                    X
               DEPTHCONDLMT=((10),OVR),                              X
               ENVAR=(''),OVR),                                      X
               ERRCOUNT=((20),OVR),                                  X
               HEAP=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR),       @03C X
               HEAPCHK=((OFF,1,0),OVR),                              X
               LIBSTACK=((4K,4080,FREE),OVR),                    @03C X
               MSGFILE=((CESE),OVR),                             @01C X
               MSGQ=((15),OVR),                                      X
               NATLANG=((UEN),OVR),                                  X
               NOTEST=((ALL,*,PROMPT,''),OVR),                       X
               NOUSRHDLR=(),OVR),                                    X
               RETZERO=((OFF),OVR),                                  X
               RPTOPTS=((OFF),OVR),                                  X
               RPTSTG=((OFF),OVR),                                   X
               RTEREUS=((OFF),OVR),                                  X
               STACK=((4K,4080,ANYWHERE,KEEP),OVR),              @03C X
               STORAGE=((00,NONE,NONE,0K),OVR),                      X
               TERMTHDACT=((TRACE,MSGFL,0),OVR),                 @02C X
               TRACE=((OFF,4K,DUMP,LE=0),OVR),                       X
               TRAP=((ON,MAX),OVR),                              @01C X
               UPSI=((00000000),OVR),                                X
               XUFLOW=((AUTO),OVR)
         DC    C'5686-066-32-65K (C) COPYRIGHT IBM CORP. 1991, 2001.'
         DC    C'LICENSED MATERIALS - PROPERTY OF IBM'
         END
```

Notes:

1. Xs are in column 72.
2. You can regenerate your CICS-wide options using skeleton CEEWCOPT.

Figure 2. Sample Generation of CEECOPT Object Module (CICS)

Creating Application-Specific Options Using the CEEXOPT Macro

You can create an application-specific options module to provide default run-time options for a given application. A sample job to do this is in CEEWUOPT.Z. The name of the CSECT created to establish application-specific defaults is CEEUOPT, and this must then be link-edited with your application program for the defaults to be recognized.

Each run of the job CEEWUOPT can create a new CEEUOPT options module in a user-specified sublibrary. The application programmer can include one of these CEEUOPT modules when link-editing an application. The options in CEEUOPT then override the default options in CEEDOPT or CEECOPT, unless NONOVR was specified for the option when CEEDOPT or CEECOPT was created.

The job invokes the CEEXOPT macro during the assembly of the CEEUOPT module. OVR and NONOVR are not applicable to the creation of an application-specific options module.

Using the CEEUOPT Run-Time Option Module

CEEUOPT is a sample source module (A-book), that is similar to that provided for LE/VSE batch-wide run-time options (CEEDOPT).

You should be careful when specifying application-specific override options, since:

- A CEEUOPT module is intended to be linked with an application, and forces specific application behaviour without changing the general batch- or CICS-wide LE/VSE run-time option settings.
- If you create different levels of CEEUOPT.OBJ modules (for application relink purposes), you can cause various problems if you do not carefully control the use of the various levels.

WARNING: If you plan to use application-specific options with a CICS application, you should review the IBM-supplied values, and ensure they are appropriate for your CICS application. For values that are applicable to CICS, see the storage report produced when using RPTSTG(ON) or the supplied CEECOPT member (shown in Figure 2 on page 19). Otherwise, your CICS system might suffer from performance degradation and/or storage problems.

Modifying the JCL for CEEWUOPT

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
3. Copy member CEEUOPT.A from the PRD2.SCEEBASE sublibrary into job CEEWUOPT in place of the comment line. Change the parameters on the CEEXOPT macro statement in CEEUOPT to reflect the values you have chosen for your application-specific default run-time options. Delete all options you do not wish to change from the system-wide default settings (as shown in Figure 3 below).
4. Change `YOURLIB.YOURSUB` in the ACC SUBLIB statement to the name of the sublibrary into which you want your CEEUOPT module to be cataloged. The new CEEUOPT module replaces any existing CEEUOPT module in the chosen sublibrary.

After you modify the CEEWUOPT job, submit it. The job finishes with a return code of 0 if it runs successfully.

Figure 3 provides an example of the use a CEEUOPT module, in which:

1. An option report is to be generated, by specifying the RPTOPTS(ON) run-time option.
2. The application is prepared to be debugged, by specifying the TEST run-time option.

```
* $$ JOB JNM=SMPUOPT,CLASS=Z
* $$ PUN DISP=I,PRI=6,CLASS=Z
// JOB SMPUOPT
// LIBDEF *,SEARCH=(PRD2.SCEEBASE)
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT
// JOB SMPUOPT
*
*       STEP 2: CATALOG MODULE CEEUOPT.OBJ
*
// EXEC LIBR,PARM='MSHP;ACCESS SUBLIB=PRD2.CONFIG'
* $$ END
*
*       STEP 1: ASSEMBLE LE/VSE APPLICATION UOPTS
*
// OPTION DECK
// EXEC ASMA90,SIZE=ASMA90
   PUNCH ' CATALOG CEEUOPT.OBJ,REPLACE=YES'
   PRINT ON,NOGEN
CEEUOPT  CSECT
CEEUOPT  AMODE ANY
CEEUOPT  RMODE ANY
         CEEXOPT RPTOPTS=(ON),                                          X
               TEST=(ALL,*,PROMPT,*)
         END
/*
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT
/*
#&
$ $$ EOJ
* $$ END
/&
* $$ EOJ
```

Figure 3. Sample Use of the CEEUOPT Run-Time Option Module

Requirements for Coding the CEEXOPT Macro

- **No omissions permitted in CEEDOPT and CEECOPT**. Each option in the CEEDOPT or CEECOPT template must be present, and each of its suboptions must be specified with one of the legal suboption values, except for the

sub-option of the USRHDLR option. The final suboption for each CEEDOPT or CEECOPT option must be OVR or NONOVR. OVR means that the option can be overridden at run time. NONOVR means that the option cannot be overridden at run time.

- **Omissions permitted in CEEUOPT**. You can completely omit the specification of any option in CEEUOPT. Default values are then supplied for each of the missing options.

  In either case, the continuation character (X in this example) must still be present in column 72.

  In CEEUOPT, IBM recommends that you omit any options you do not wish to change. The options you omit from the macro will default to the installation-wide defaults you set in CEEDOPT or CEECOPT.
- **Omission of suboptions in CEEUOPT**. In CEEUOPT, you can use commas to indicate the omission of one or more suboptions for options having more than one suboption. For example, if you wish to specify only the second suboption of the STORAGE option, the omission of the 1st, 3rd, and 4th suboptions can be indicated in any of the following ways:

  ```
  STORAGE=(,NONE),                                                X
  STORAGE=(,NONE,),                                               X
  STORAGE=(,NONE,,),                                              X
  ```

  Because suboptions are positional parameters, do not omit the comma if the corresponding suboption is omitted and another suboption follows.

  Note: If you specify an option in the CEEUOPT, there are special defaults for omitted sub-options. You can find these default values under the "Usage Notes" heading for the related run-time options.
- **Continuing lines of code**. A continuation character (`X` in the source) must be in column 72 on each line of the CEEXOPT invocation except the last line. The continuation line must start in column 16. You can break the coding after any comma.
- **Case sensitivity**. Options and suboptions must be in uppercase. Only suboptions that are strings can be specified in mixed-case or lowercase. For example, both `MSGFILE=(SYSLST)` and `MSGFILE=(syslst)` are acceptable, but `TRACE=((off,4K,DUMP,LE=0),OVR)` is not.
- A comma must end each option except for the final option. If the comma is omitted, everything following the option is treated as a comment.
- **Special characters**. If one of the string suboptions contains a special character (for example, an embedded blank or unmatched right or left parenthesis), the string must be enclosed in single apostrophes ('), not in double quotation marks ("). (You can specify a null string with either contiguous single apostrophes or contiguous double quotation marks.)

  To obtain a single apostrophe (') or a single ampersand (&) within a string, you must specify two contiguous instances of the character. The pair is counted as only one character in determining whether the maximum allowable string length has been exceeded and in setting the effective length of the string.
- **Maximum length**. Macro instruction operands cannot be longer than 255 characters. If the number of characters to the right of the equal sign is greater than 255 for any keyword parameter in the CEEXOPT invocation in CEEDOPT, CEECOPT, or CEEUOPT, a return code of 12 is produced for the assembly, and the options are not parsed properly.

- **Apostrophes**. Avoid unmatched apostrophes in any string that uses apostrophes. The error cannot be captured within CEEXOPT itself; instead, the assembler produces a message such as:

```
ASMA063  *** ERROR ***  NO ENDING APOSTROPHE
```

  However, the assembler does not necessarily produce such a message immediately following the offending suboption. If the assembler detects unmatched apostrophes, none of the options are properly parsed.
- **Options that permit only one suboption**. Options that permit only one suboption do not need to enclose that suboption in parentheses. For example, you can specify the COUNTRY option in CEEUOPT in either of the following ways:

```
COUNTRY=(US),                                                  X
COUNTRY=US,                                                    X
```

Changing Run-Time LIOCS Phases Defaults

You can customize the run-time LIOCS phases for card reader/punch devices, diskette devices, and printer devices. Table 20 summarizes the sample jobs IBM provides to help you customize the run-time options.

Table 20. Sample Jobs to Change Run-Time LIOCS Phases

Change LIOCS Phases For	Sample Job Member	Assembler Source Member
Card reader/punch devices	CEEWDCD0.Z	CEEYCD0.A
Diskette devices	CEEWDDU0.Z	CEEYDU0.A
Printer devices	CEEWDPR0.Z	CEEYPR0.A

Note: These jobs are available in ICCF Library 62.

All of the members listed in Table 20 are installed in the PRD2.SCEEBASE sublibrary.

The LE/VSE macros CEEXCDMD, CEEXDUMD, and CEEXPRMD, described in "Appendix B. LE/VSE Run-Time LIOCS Phases" on page 123, generate card, diskette, and printer LIOCS routines that are suitable for your applications. These macros are front-end macros for the VSE-supplied macros CDMOD, DUMODFI, DUMODFO, and PRMOD. For more information about device LIOCS routines, see the descriptions of macros CDMOD, DUMODFI, DUMODFO, and PRMOD in the *VSE/ESA System Macros Reference*.

To change a run-time LIOCS phase, copy the corresponding assembler source member into the sample job in place of the comment, and change the parameters on the macro (CEEXCDMD, CEEXDUMD, or CEEXPRMD) invocation in the sample job you are using. Make the parameters in the macro invocation match the parameters you have selected for your installation.

Figure 4 on page 25 shows the source statements used to generate the IBM-supplied version of the card reader/punch device run-time LIOCS phase, CEEYCD0, with the file attributes supported.

Figure 5 on page 26 shows the source statements used to generate the IBM-supplied version of the diskette device run-time LIOCS phase, CEEYDU0, with the file attributes supported.

Figure 6 on page 26 shows the source statements used to generate the IBM-supplied version of the printer device run-time LIOCS phase, CEEYPR0, with the file attributes supported.

```
         CEEXCDMD TYPE=START
         CEEXCDMD TYPE=ENTRY,TYPEFLE=INPUT,DEVICE=1442,                X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=INPUT,DEVICE=2520,                X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=INPUT,DEVICE=2540,                X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=INPUT,DEVICE=3505,                X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=INPUT,DEVICE=3525,                X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=INPUT,DEVICE=3881,                X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=1442,               X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2520,               X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES,                  X
               CTLCHR=ASA
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES,                  X
               CTLCHR=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=3525,               X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES


*
* The following are added for C runtime support
*
         CEEXCDMD TYPE=ENTRY,TYPEFLE=INPUT,DEVICE=2540,                X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES,WORKA=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES,WORKA=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES,                  X
               CTLCHR=ASA,WORKA=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=FIXUNB,IOAREA2=YES,RDONLY=YES,                  X
               CTLCHR=YES,WORKA=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=VARUNB,IOAREA2=YES,RDONLY=YES,WORKA=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=VARUNB,IOAREA2=YES,RDONLY=YES,                  X
               CTLCHR=ASA,WORKA=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=VARUNB,IOAREA2=YES,RDONLY=YES,                  X
               CTLCHR=YES,WORKA=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=UNDEF,IOAREA2=YES,RDONLY=YES,WORKA=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=UNDEF,IOAREA2=YES,RDONLY=YES,                   X
               CTLCHR=ASA,WORKA=YES
         CEEXCDMD TYPE=ENTRY,TYPEFLE=OUTPUT,DEVICE=2540,               X
               RECFORM=UNDEF,IOAREA2=YES,RDONLY=YES,                   X
               CTLCHR=YES,WORKA=YES
         CEEXCDMD TYPE=FINAL
         END
```

Note: Xs are in column 72.

Figure 4. Sample Invocation of CEEXCDMD to Generate the CEEYCD0 Phase

```
         CEEXDUMD TYPE=START
         CEEXDUMD TYPE=ENTRY,TYPEFLE=INPUT,RDONLY=YES
*
         CEEXDUMD TYPE=ENTRY,TYPEFLE=OUTPUT,RDONLY=YES
         CEEXDUMD TYPE=FINAL
                  END
```

Figure 5. Sample Invocation of CEEXDUMD to Generate the CEEYDU0 Phase

```
         CEEXPRMD TYPE=START
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=FIXUNB,   X
               RDONLY=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=FIXUNB,   X
               CTLCHR=YES,RDONLY=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=FIXUNB,   X
               CTLCHR=ASA,RDONLY=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=VARUNB,   X
               RDONLY=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=VARUNB,   X
               CTLCHR=YES,RDONLY=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=VARUNB,   X
               CTLCHR=ASA,RDONLY=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=UNDEF,    X
               RDONLY=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=UNDEF,    X
               CTLCHR=YES,RDONLY=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=UNDEF,    X
               CTLCHR=ASA,RDONLY=YES
*
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=FIXUNB,   X
               RDONLY=YES,WORKA=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=FIXUNB,   X
               CTLCHR=YES,RDONLY=YES,WORKA=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=FIXUNB,   X
               CTLCHR=ASA,RDONLY=YES,WORKA=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=VARUNB,   X
               RDONLY=YES,WORKA=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=VARUNB,   X
               CTLCHR=YES,RDONLY=YES,WORKA=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=VARUNB,   X
               CTLCHR=ASA,RDONLY=YES,WORKA=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=UNDEF,    X
               RDONLY=YES,WORKA=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=UNDEF,    X
               CTLCHR=YES,RDONLY=YES,WORKA=YES
         CEEXPRMD TYPE=ENTRY,DEVICE=1403,IOAREA2=YES,RECFORM=UNDEF,    X
               CTLCHR=ASA,RDONLY=YES,WORKA=YES
         CEEXPRMD TYPE=FINAL
                  END
```

Note: Xs are in column 72.

Figure 6. Sample Invocation of CEEXPRMD to Generate the CEEYPR0 Phase

Changing the Card-Device Run-Time LIOCS Phase

Note!:

1. During customization, you are recommended not to delete or change any shipped LE/VSE LIOCS definitions. If LIOCS definitions for dummy card punch, card reader, and printer devices are missing, the startup of the CICS Transaction Server might be severely affected.
2. The Interactive Interface's *Hardware Configuration* dialog also contains definitions for dummy card punch, card reader, and print devices. These definitions reflect the currently-shipped LE/VSE LIOCS. You are recommended not to delete or change any of these definitions.
3. If, however, you do decide to change any of the dummy device defnitions, *ensure you have customized the appropriate LIOCS definitions for these devices!.*

Use the sample job in the PRD2.SCEEBASE sublibrary member CEEWDCD0.Z to replace the IBM-supplied card-device run-time LIOCS phase. Use the information in "Appendix B. LE/VSE Run-Time LIOCS Phases" on page 123 to select your default values.

Modifying the JCL for CEEWDCD0

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries Where you have installed LE/VSE.
3. If necessary, change the name of the sublibrary specified in the ACC SUBLIB=PRD2.SCEEBASE statement to match the sublibrary where you have installed LE/VSE.
4. Copy member CEEYCD0.A from the PRD2.SCEEBASE sublibrary into job CEEWDCD0 in place of the comment line.
5. Add any additional CEEXCDMD TYPE=ENTRY macro calls that you require to describe the additional combinations of file attributes you need to support at your installation.
6. Remove any CEEXCDMD TYPE=ENTRY macros you do not require at your installation. If these are left in it will make the phase slightly larger but will not affect the correct running of your LE/VSE application program.
7. Make sure you do not change the position or format of the CEEXCDMD TYPE=START or CEEXCDMD TYPE=FINAL macro calls in the supplied job.

After you modify the CEEWDCD0 job, submit it. The job finishes with a return code of 0 if it runs successfully.

Note: This sample job will replace the LE/VSE card-device LIOCS phase, CEEYCD0, in the installation sublibrary. If service is applied to this phase you may need to rerun CEEWDCD0.

Changing the Diskette-Device Run-Time LIOCS Phase

Before you start, please read the note at the start of section “Changing the Card-Device Run-Time LIOCS Phase” on page 27!.

Use the sample job in the PRD2.SCEEBASE sublibrary member CEEWDDU0.Z to replace the IBM-supplied diskette-device run-time LIOCS phase. Use the information in “Appendix B. LE/VSE Run-Time LIOCS Phases” on page 123 to select your default values.

Modifying the JCL for CEEWDDU0

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
3. If necessary, change the name of the sublibrary specified in the ACC SUBLIB=PRD2.SCEEBASE statement to match the sublibrary where you have installed LE/VSE.
4. Copy member CEEYDU0.A from the PRD2.SCEEBASE sublibrary into job CEEWDDU0 in place of the comment line.
5. Add any additional CEEXDUMD TYPE=ENTRY macro calls that you require to describe the additional combinations of file attributes you need to support at your installation.
6. Remove any CEEXDUMD TYPE=ENTRY macros you do not require at your installation. If these are left in it will make the phase slightly larger but will not affect the correct running of your LE/VSE application program.
7. Make sure you do not change the position or format of the CEEXDUMD TYPE=START or CEEXDUMD TYPE=FINAL macro calls in the supplied job.

After you modify the CEEWDDU0 job, submit it. The job finishes with a return code of 0 if it runs successfully.

Note: This sample job will replace the LE/VSE diskette-device LIOCS phase, CEEYDU0, in the installation sublibrary. If service is applied to this phase you may need to rerun CEEWDDU0.

Changing the Printer-Device Run-Time LIOCS Phase

Before you start, please read the note at the start of section “Changing the Card-Device Run-Time LIOCS Phase” on page 27!.

Use the sample job in the PRD2.SCEEBASE sublibrary member CEEWDPR0.Z to replace the IBM-supplied printer-device run-time LIOCS phase. Use the information in “Appendix B. LE/VSE Run-Time LIOCS Phases” on page 123 to select your default values.

Modifying the JCL for CEEWDPR0

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
3. If necessary, change the name of the sublibrary specified in the ACC SUBLIB=PRD2.SCEEBASE statement to match the sublibrary where you have installed LE/VSE.
4. Copy member CEEYPR0.A from the PRD2.SCEEBASE sublibrary into job CEEWDPR0 in place of the comment line.
5. Add any additional CEEXPRMD TYPE=ENTRY macro calls that you require to describe the additional combinations of file attributes you need to support at your installation.
6. Remove any CEEXPRMD TYPE=ENTRY macros you do not require at your installation. If these are left in it will make the phase slightly larger but will not affect the correct running of your LE/VSE application program.
7. Make sure you do not change the position or format of the CEEXPRMD TYPE=START or CEEXPRMD TYPE=FINAL macro calls in the supplied job.

After you modify the CEEWDPR0 job, submit it. The job finishes with a return code of 0 if it runs successfully.

Note: This sample job will replace the LE/VSE printer-device LIOCS phase, CEEYPR0, in the installation sublibrary. If service is applied to this phase you may need to rerun CEEWDPR0.

Changing the Assembler Language User Exit

The LE/VSE sublibrary contains three sample jobs to assist you in modifying the assembler language user exit. Two of the jobs replace the IBM-supplied installation-wide assembler user exits. The third sample job creates an application-specific assembler user exit that can be link-edited with those applications that need its functions. You can create several different application-specific user exits, each in a different sublibrary, to satisfy the needs of different application programs. Examples of the source for assembler user exits are provided in the LE/VSE sublibrary.

Table 21 describes these sample assembler user exits.

Table 21. Sample Assembler User Exits for LE/VSE

Example User Exit	Operating System	Language (if Language-Specific)
CEEBXITA	VSE (default)	
CEECXITA	CICS (default)	
CEEBX05A	VSE	VS COBOL II compatibility

Note:

1. CEEBXITA and CEECXITA are the defaults on your system for VSE and CICS, if LE/VSE is installed at your site without modification.
2. The source code for CEEBXITA, CEECXITA, and CEEBX05A can be found in the LE/VSE sublibrary.

Use the information in "Appendix C. Customizing LE/VSE User Exits" on page 137 to assist you in modifying the IBM-supplied user exits or in creating your own.

If you specify run-time options in an assembler language user exit, they override options specified in CEEUOPT. They override options in CEEDOPT or CEECOPT only if OVR was specified for the option in CEEDOPT or CEECOPT.

Changing the Installation-Wide Assembler Language User Exit (Batch)

Use the sample job in the PRD2.SCEEBASE sublibrary member CEEWDXIT.Z to change the installation-wide assembler language user exit. You must replace the comment in CEEWDXIT with your source for CEEBXITA. You can copy the source for the IBM-supplied default installation-wide assembler language user exit from member CEEBXITA.A in the PRD2.SCEEBASE sublibrary and modify it to suit your needs, or you can create your own source for CEEBXITA. Use the information in "Appendix C. Customizing LE/VSE User Exits" on page 137 to guide you in coding your changes.

Modifying the JCL for CEEWDXIT

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
3. If necessary, change the name of the sublibrary specified in the ACC SUBLIB=PRD2.SCEEBASE statement to match the sublibrary where you have installed LE/VSE.
4. Replace the comment in the job with your source program for the installation-wide (batch) assembler language user exit.

After you modify the CEEWDXIT job, submit it. The job finishes with a return code of 2 if it runs successfully. The return code of 2 indicates there were unresolved weak external references during the link-edit steps. This return code is normal and does not indicate a problem.

Note: Remember that if you have the CEEBXITA PHASE loaded in the SVA, you will need to reload this module into the SVA after running the CEEWDXIT job to activate the new Assembler User Exit PHASE.

Changing the Installation-Wide Assembler Language User Exit (CICS)

Use the sample job in the PRD2.SCEEBASE sublibrary member CEEWCXIT.Z to change the CICS installation-wide assembler language user exit. You must replace the comment in CEEWCXIT with your source for CEECXITA. You can copy the source for the IBM-supplied default installation-wide assembler language user exit from the member CEECXITA.A in the PRD2.SCEEBASE sublibrary and modify it to suit your needs, or you can create your own source for CEECXITA.

Note the differences between the IBM-supplied CEEBXITA and the IBM-supplied CEECXITA. You can retain some or all of these differences in your user exit. Use the information in "Appendix C. Customizing LE/VSE User Exits" on page 137 to guide you in coding your changes.

Modifying the JCL for CEEWCXIT

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

---

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
3. If necessary, change the name of the sublibrary specified in the ACC SUBLIB=PRD2.SCEEBASE statement to match the sublibrary where you have installed LE/VSE.
4. Replace the comment in the job with your source program for the installation-wide CICS assembler language user exit.

After you modify the CEEWCXIT job, submit it. The job finishes with a return code of 2 if it runs successfully. The return code of 2 indicates there were unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

Notes:

1. LE/VSE will not use a newly created Assembler User Exit under CICS until the CICS system(s) has been re-started via either a COLD or EMERGENCY startup.

There is no support for dynamically loading a new CEECXITA into an active CICS system. Use of the CICS CEMT facility on CEECXITA is not supported.

2. Remember that if you have the CEECXITA PHASE loaded in the SVA, you will need to reload this module into the SVA after running the CEEWCXIT job to activate the new Assembler User Exit PHASE.

Creating an Application-Specific Assembler Language User Exit

Use the sample job CEEWUXIT in the PRD2.SCEEBASE sublibrary member CEEWUXIT.Z to create as many application-specific assembler language user exits as your site requires. You must replace the comment in CEEWUXIT with your source. You can copy the source for the IBM-supplied default installation-wide assembler language user exit from member CEEBXITA.A (batch) or member CEECXITA.A (CICS) in the PRD2.SCEEBASE sublibrary and modify it to suit your needs, or you can create your own source.

You can run job CEEWUXIT several times to create several different CEEBXITA modules, each in its own user-specified sublibrary. Use the information in "Appendix C. Customizing LE/VSE User Exits" on page 137 to guide you in coding your changes.

Modifying the JCL for CEEWUXIT

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
3. Replace the comment in the job with your source program for the application-specific assembler language user exit.
4. Change `YOURLIB.YOURSUB` in the ACC SUBLIB statement, to the name of the sublibrary into which you want your CEEBXITA module to be cataloged. The new CEEBXITA module replaces any existing CEEBXITA module in the chosen sublibrary.

After you modify the CEEWUXIT job, submit it. The job finishes with a return code of 0 if it runs successfully.

Creating a High-Level Language User Exit

Use the sample job CEEWHLLX in the PRD2.SCEEBASE sublibrary member CEEWHLLX.Z to create as many high-level language user exits as your site requires. The sample job catalogs the object program for the user exit. It does not contain JCL to compile the high-level language source program. You must compile your user exit and use the object program produced by the compiler to replace the comment in the job CEEWHLLX. Refer to "Appendix C. Customizing LE/VSE User Exits" on page 137 for a description of the high-level language user exit interface.

Modifying the JCL for CEEWHLLX

1. Modify the job card as appropriate for your site.
2. Add POWER JECL statements if your site requires them.
3. Replace the comment line in CEEWHLLX with the object program obtained by compiling your high-level language user exit.
4. Change `YOURLIB.YOURSUB` in the ACC SUBLIB parameter of the EXEC LIBR statement, to the name of the sublibrary into which you want your CEEBINT module to be cataloged. The new CEEBINT module replaces any existing CEEBINT module in the chosen sublibrary.

After you modify the CEEWHLLX job, submit it. The job finishes with a return code of 0 if it runs successfully.

Creating a User-Written Handler for Compatibility with VS COBOL II and DOS PL/I

Use the sample code in CEEWUCHA.A in the PRD2.SCEEBASE sub-library, to provide condition handling compatibility under CICS/VSE with VS COBOL II and DOS PL/I. The condition handler you create usingt CEEWUCHA.A is registered at stack frame 0 by the USRHDLR run-time option.

Two sample jobs are provided to assemble and link-edit CEEWUCHA depending upon the requirements of your site. Table 22 summarizes these sample jobs.

Table 22. Sample Jobs to Create a User-Written Condition Handler

Function	Sample Job
Create a phase for VS COBOL II-only compatibility	CEEWWCHA.Z
Create a phase for VS COBOL II and DOS PL/I compatibility	CEEWCCHA.Z

Notes:

1. These jobs are available in ICCF Library 62.
2. VS COBOL II is no longer in service, and is therefore not available from VSE/ESA 2.5 onwards.

Modifying the JCL for CEEWWCHA and CEEWCCHA

1. Modify the job card as appropriate for your site.
2. Add POWER JECL statements if your site requires them.
3. Insert the program CEEWUCHA as described in the JCL.
4. If you are modifying CEEWCCHA.Z then modify the &PLI SETC statement as described in the JCL member, to activate the DOS PL/I compatibility mode.
5. If necessary, change the name of the sub-library specified in the `LIBDEF *,SEARCH` statement to match the sub-library where you have installed LE/VSE.
6. Change `YOURLIB.YOURSUB` in the library `LIBDEF PHASE,CATALOG` statement, to the name of the sub-library into which you want your user-written condition handler phase to be cataloged.

After you modify the job, submit it. The job finishes with a return code less than 4 if it runs successfully.

Customizing LE/VSE Abnormal Termination Exits

If LE/VSE encounters an unhandled condition of severity 2 or greater, it can invoke an abnormal termination exit before it terminates the enclave. If the abnormal termination exit is invoked before the thread is terminated, the abnormal termination exit can collect problem determination data before LE/VSE frees the resources it has acquired.

LE/VSE is shipped with a default abnormal termination exit setting, which you can find in these shipped members:
- CEEBXTAN.A (for batch)
- CEECXTAN.A (for CICS)

The above two shipped members are associated with these job skeletons (available in ICCF library 62):
- CEEWDEXT.Z
- CEEWCEXT.Z

The LE/CICS abnormal termination exit was changed from LE/VSE 1.4.1 onwards, by specifying `TERMXIT=CEEBNATX` instead of `TERMXIT=CEECDATX`. This abnormal termination exit is used when you receive application abends under CICS, and when invoked it ensures immediate return to the caller. From LE/VSE 1.4.1 onwards, the abnormal termination exit for CICS is pre-customized with this null-exit capability (**CEEBNATX**) so that the cause of a CICS *Axxx* abend code is easier to identify.

The LE/VSE library also contains two sample source programs that you can use as examples of how to write an abnormal termination exit:

CEEBBATX.A
A *batch* abnormal termination exit. It produces a system dump when invoked.

Note: If you use runtime options `TERMTHDACT(UADUMP)` and `TRAP(ON,MAX)`, you can also generate a system dump without the need to customize the abnormal termination exits!.

CEECATX1.A
An LE/CICS abnormal termination exit – predump. For details of the `PREDUMP` and `POSTDUMP` functions, refer to "CEEXART Macro" on page 36.

Creating an LE/VSE Abnormal Termination Exit

To create an abnormal termination exit:

1. Create an assembler language routine that conforms to the syntax described in "Abnormal Termination Exit Syntax" on page 150.
2. Assemble and link-edit your exit into a library that LE/VSE can access at run time, such as PRD2.SCEEBASE.
3. Create a CEEEXTAN CSECT containing a CEEXART macro identifying your exit. The CEEXART macro specifies your routine as an abnormal termination exit routine. PRD2.SCEEBASE contains the source modules CEECXTAN.A (for CICS) or CEEBXTAN.A (for non-CICS) which you can use to create a CEEEXTAN CSECT. See "Creating a CEEEXTAN Abnormal Termination Exit CSECT" on page 36 for more information.
4. Replace the existing CEEEXTAN CSECT with the updated CEEEXTAN as described in the sections below.

Creating a CEEEXTAN Abnormal Termination Exit CSECT

CEEEXTAN is a CSECT you create by coding these LE/VSE-provided assembler macros:

CEEXAHD Defines the header of the table

CEEXART Identifies the name of the abnormal termination exit to be invoked

CEEXAST Identifies the end of the list of abnormal termination exits

You can use the above three macros to create these objects decks:
- CEEBXTAN (for batch)
- CEECXTAN (for CICS)

Each object deck has the CSECT name **CEEEXTAN**.

You then link these object decks to create these phases:
- CEEBXTAN.PHASE (for batch)
- CEECXTAN.PHASE (for CICS)

LE/VSE will automatically use the correct phase dependant upon the environment being used.

LE/VSE validates the format of the abnormal termination exit CSECT and issues a load of the phases identified in the table. The LOAD is attempted only for terminations due to unhandled conditions of severity 2 or greater:
- If the LOAD is successful, an abnormal termination exit is invoked according to the interface described below.
- If the LOAD fails (for example, if the phase cannot be found, or there is not enough storage for the routine), no error indication is delivered and either the next name in CEEEXTAN is chosen, or termination continues (if the names have been exhausted).

This allows your sublibrary search chain to either contain or omit the phases.

CEEXAHD Macro

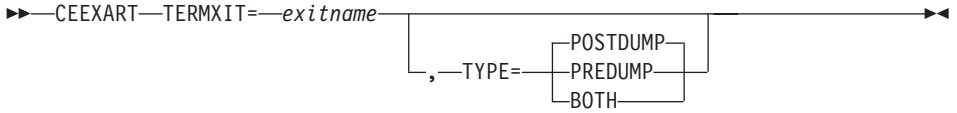
CEEXAHD generates the CSECT statement and any header information required. It has no operands.

CEEXART Macro

CEEXART generates one entry for an abnormal termination exit.

More than one invocation of CEEXART can appear in the CEEEXTAN CSECT, thus allowing multiple abnormal termination exits to be registered. When more than one name is specified, the abnormal termination exits are honored at the specified point in enclave termination in the order found in CEEEXTAN CSECT.

Syntax



TERMXIT
Keyword required to specify the phase name for the abnormal termination exit.

exitname
The phase name for the abnormal termination exit. There is a limit of 8 characters for the phase name, and no validation of the name is performed by the macro.

TYPE
Keyword required to specify the type of abnormal termination exit.

POSTDUMP
The abnormal termination exit is invoked after the LE/VSE dump is generated.

PREDUMP
The abnormal termination exit is invoked before the LE/VSE dump is generated. The abnormal termination exit can pass back to LE/VSE a return code of 8 in register 15 indicating the LE/VSE dump not be generated.

BOTH
The abnormal termination exit is invoked both before and after the LE/VSE dump is generated. When invoked before the LE/VSE dump is generated, the abnormal termination exit can pass back to LE/VSE a return code of 8 in register 15 indicating the LE/VSE dump not be generated.

CEEXAST Macro

CEEXAST generates the trailer for the CEEEXTAN CSECT. It has no parameters.

Installation Jobs to Generate and Modify CEEEXTAN CSECT

You can use two source files to generate CEEEXTAN CSECT, one for CICS and one for Batch. The following members are provided in the PRD2.SCEEBASE sublibrary:

CEEBXTAN.A Source to generate CEEEXTAN CSECT for Batch

CEECXTAN.A Source to generate CEEEXTAN CSECT for CICS

You can use the jobs in the following two PRD2.SCEEBASE sublibrary members to replace CEEEXTAN CSECT:

CEEWDEXT.Z Replaces CEEEXTAN CSECT for Batch

CEEWCEXT.Z Replaces CEEEXTAN CSECT for CICS

Identifying an Abnormal Termination Exit (Batch)

Use the sample job CEEWDEXT in the PRD2.SCEEBASE sublibrary member CEEWDEXT.Z to identify an abnormal termination exit in a Batch environment.

1. Replace the comment in CEEWDEXT with your source for CEEBXTAN.
2. Copy the source from the IBM-supplied default, which invokes the null module CEEBDATX, from member CEEBXTAN.A in the PRD2.SCEEBASE sublibrary and modify it to suit your needs.

Modifying the JCL for CEEWDEXT

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
3. If necessary, change the name of the sublibrary specified in the ACC SUBLIB=PRD2.SCEEBASE statement to match the sublibrary where you have installed LE/VSE.
4. Replace the comment lines in CEEWDEXT with your source program that identifies an abnormal termination exit.

After you modify the CEEWDEXT job, submit it. The job finishes with a return code of 2 if it runs successfully. The return code of 2 indicates there were unresolved weak external references during the link-edit steps. This return code is normal and does not indicate a problem.

Note: This sample job no longer replaces the LE/VSE initialization/termination phases, CEEBINIT and CEEPIPI, in the installation sublibrary.

Figure 7 on page 39 contains the source for the IBM-supplied CEEBXTAN.A:

```
         TITLE 'LE/VSE Abnormal Termination User exit CSECT for BATCH'
*/*****************************************************************/
*/*                                                               */
*/*    Language Environment/VSE V1 R4 M2                          */
*/*                                                               */
*/*    LICENSED MATERIALS - PROPERTY OF IBM                       */
*/*                                                               */
*/*    5686-066-32-65K (C) COPYRIGHT IBM CORPORATION 2001         */
*/*    ALL RIGHTS RESERVED.                                       */
*/*                                                               */
*/*    US Government Users Restricted Rights - Use, duplication or*/
*/*    disclosure restricted by GSA ADP Schedule Contract with IBM*/
*/*    Corp.                                                      */
*/*                                                               */
*/* ************************************************************* */
*/*                    I M P O R T A N T                          */
*/*                                                               */
*/* If you do not use the supplied JCL sample (CEEWDEXT) to assemble */
*/* and linkedit this CSECT, please ensure this module is cataloged  */
*/* as CEEEXTAN.OBJ with a CSECT name of CEEEXTAN. The link-book     */
*/* provided for linkediting the required PHASE is CEE$XTAN.OBJ      */
*/* and is supplied in the LE/VSE Installation sub-library.          */
*/*                                                               */
*/*****************************************************************/
         CEEXAHD        ,User exit header
*
*********************************************************************
* Use the default BATCH null abnormal termination exit.
* You can change the following module name, specified on the TERMXIT
* option, to another user written abnormal termination exit routine.
*********************************************************************
*
         CEEXART  TERMXIT=CEEBNATX
*
*********************************************************************
* To specify an additional abnormal termination exit, change the
* following line where CEEXART is specified:
*  - change the XXXXXXXX to the name of the abnormal termination exit
*  - change the '*' in column 1 to a blank
*********************************************************************
*
*        CEEXART  TERMXIT=XXXXXXXX
*
         CEEXAST        ,Terminate the list
```

Figure 7. Default Member CEEBXTAN.A (for Batch Environment)

Identifying an Abnormal Termination Exit (CICS)

Use the sample job CEEWCEXT in the PRD2.SCEEBASE sublibrary member CEEWCEXT.Z to identify an abnormal termination exit in a CICS environment. Replace the comment in CEEWCEXT with your source for CEECXTAN. Copy the source from the IBM-supplied default, from member CEECXTAN.A in the PRD2.SCEEBASE sublibrary and modify it to suit your needs.

Modifying the JCL for CEEWCEXT

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
3. If necessary, change the name of the sublibrary specified in the ACC SUBLIB=PRD2.SCEEBASE statement to match the sublibrary where you have installed LE/VSE.
4. If necessary, change the name of the sublibrary specified in the EXEC LIBR statement to match the sublibrary where you have installed LE/VSE.
5. Replace the comment lines in CEEWCEXT with your source program that identifies an abnormal termination exit.

After you modify the CEEWCEXT job, submit it. The job finishes with a return code of 2 if it runs successfully. The return code of 2 indicates there were unresolved weak external references during the link-edit steps. This return code is normal and does not indicate a problem.

Figure 8 on page 41 contains the source for the IBM-supplied CEECXTAN.A:

```
        TITLE 'LE/VSE Abnormal Termination User exit CSECT for CICS'
*/*****************************************************************/
*/*                                                               */
*/*    Language Environment/VSE V1 R4 M2                          */
*/*                                                               */
*/*    LICENSED MATERIALS - PROPERTY OF IBM                       */
*/*                                                               */
*/*    5686-066-32-65K (C) COPYRIGHT IBM CORPORATION 1991,  2001  */
*/*    ALL RIGHTS RESERVED.                                       */
*/*                                                               */
*/*     US Government Users Restricted Rights - Use, duplication or  */
*/*     disclosure restricted by GSA ADP Schedule Contract with IBM  */
*/*     Corp.                                                     */
*/* ************************************************************* */
*/*  Change History :                                             */
*/*                                                               */
*/* @01  GWH  02/12/1999  Use supplied 'null' abnormal termination   */
*/*                  exit as the new default.                     */
*/*                                                               */
*/* ************************************************************* */
*/*                    I M P O R T A N T                          */
*/*                                                               */
*/* If you do not use the supplied JCL sample (CEEWCEXT) to assemble */
*/* and linkedit this CSECT, please ensure this module is cataloged  */
*/* as CEECXTAN.OBJ with a CSECT name of CEEEXTAN. The link-book     */
*/* provided for linkediting the required PHASE is CEE$CTAN.OBJ      */
*/* and is supplied in the LE/VSE Installation sub-library.          */
*/*                                                               */
*/*****************************************************************/
        CEEXAHD        ,User exit header
*
*********************************************************************
* To replace the default abnormal termination exit with a your own,
* assemble and link-edit your module and replace CEEBNATX with your
* modules name in the following line where CEEXART is specified.
*********************************************************************
*
*       CEEXART  TERMXIT=CEECDATX                                @01C
        CEEXART  TERMXIT=CEEBNATX                                @01A
*
*********************************************************************
* To specify an additional abnormal termination exit, change the
* following line where CEEXART is specified:
*  - change the XXXXXXXX to the name of the abnormal termination exit
*  - change the '*' in column 1 to a blank
*********************************************************************
*
*       CEEXART  TERMXIT=XXXXXXXX
*
*********************************************************************
* The following line shows an example of specifying the sample
* program CEECATX1 as a pre-dump abnormal termination exit.
* NOTE: valid options for the TYPE parameter are:
*
*       - POSTDUMP: exit invoked after LE dump is generated (default)
*       - PREDUMP : exit invoked before LE dump is generated
*       - BOTH    : exit invoked both before and after LE dump is
*                   generated.
*
*********************************************************************
*       CEEXART  TERMXIT=CEECATX1,TYPE=PREDUMP
*
        CEEXAST        ,Terminate the list
```

Figure 8. Default Member CEECXTAN.A (for CICS Environment)

Note: From LE/VSE 1.4.1 onwards, TERMXIT is set to CEEBNATX by default.

Placing LE/VSE Routines in the Shared Virtual Area (SVA)

Placing routines in the SVA reduces overall system storage requirements. Also, initiate/terminate (init/term) time is reduced for each application, since load time decreases.

Table 16 on page 9 provides an estimate of the amount of space used by routines for use with VSE/ESA 2.5.1, that can be put in the 24-bit and 31-bit SVA.

All of the routines listed in "Appendix F. Routines Eligible for the Shared Virtual Area" on page 161 can be included in the SVA. To include them:

- Modify the SVA statement of the VSE IPL ASI (Automated System Initialization) procedure to allow space for the routines:
  - Increase the SDL parameter by the number of new routines being added to the SVA.
  - Increase the PSIZE parameters by the amount of storage required to contain the new phases being added to the 24-bit SVA and 31-bit SVA.
- Modify the VSE background (BG) ASI procedure to automatically load the selected routines into the SVA:
  - After the SET SDL statement, add a statement:

    phasename,SVA

    or

    LIST=$SVA*xxxx*

    for each routine or list of routines to be loaded into the SVA. For an example, refer to skeleton SKJCK0 in ICCF library 59.
- Shut down and re-IPL your VSE system.

CICS Coexistence Users:

The SIT parameter `SVA=YES` should be specified in order to load phases into the SVA. For details, refer to the *VSE/ESA Administration*.

SVA load lists (phases) are provided with LE/VSE to load either the recommended phases or all SVA-eligible phases, for each component, into the SVA. The modules contained in these load lists are those listed in "Appendix F. Routines Eligible for the Shared Virtual Area" on page 161, omitting the modules for the Japanese national language features. Table 23 lists the names of these load lists and their contents.

Table 23. LE/VSE Supplied SVA load lists

Phase Name	Description
$SVACEE	All LE/VSE base routines eligible for the SVA except callable service stubs and Japanese modules
$SVAIGZM	The LE/VSE COBOL component routines listed as recommended for inclusion in the SVA
$SVAIGZ	All LE/VSE COBOL component routines eligible for the SVA assuming COBPACKs as distributed (COBPACKs reside above the 16MB line) and excepting Japanese modules

Table 23. LE/VSE Supplied SVA load lists (continued)

Phase Name	Description
$SVAIBMM	The LE/VSE PL/I component routines listed as recommended for inclusion in the SVA
$SVAIBM	All LE/VSE PL/I component routines eligible for the SVA except Japanese modules and Debug Tool for VSE/ESA modules
$SVAEDCM	The LE/VSE C component routines listed as recommended for inclusion in the SVA
$SVAEDC	All LE/VSE C component routines eligible for the SVA except Japanese modules, locales and code page converters

To load these lists automatically, modify the VSE BG ASI procedure as follows:

After the SET SDL statement, add the statement:

LIST=$SVA*xxxx*

for each list to be loaded.

LIST statements and *phasename*,SVA statements can both be used in the same execution of SET SDL.

If you wish to tailor your own SVA lists, several members are provided in the PRD2.SCEEBASE sublibrary for you to use as examples in modifying your ASI procedures. Table 24 lists the members and their contents.

Table 24. LE/VSE Sample ASI Procedure Modification Members

Member Name	Description
CEEWMSVA.Z	All LE/VSE base routines eligible for the SVA except callable service stubs
IGZWESV1.Z	All LE/VSE COBOL component routines eligible for the SVA assuming COBPACKs as distributed (COBPACKs reside above the 16MB line)
IBMSVA1.Z	All LE/VSE PL/I component routines eligible for the SVA except Debug Tool for VSE/ESA modules
EDCWMSV1.Z	All LE/VSE C component routines eligible for the SVA except locales and code page converters

Note: These jobs are available in ICCF Library 62.

Examine the lists carefully to ensure that you are installing the correct message modules for the national language support you have installed. Comments in CEEWMSVA, IBMSVA1, and EDCWMSV1 identify the mixed-case U.S. English modules and the Japanese modules. Remove the statements for the national language support you do not require. In IGZWESV1, remove the module name IGZCMGEN if mixed-case U.S. English is not required and add IGZCMGJA if Japanese is installed and you want it to be in the SVA.

For more information on including routines in the SVA, see *VSE/ESA System Control Statements*.

De-Activating LE/VSE Language Components Used By CICS

After installing your LE/VSE base and components, you might decide to de-activate the LE/VSE *LE COBOL* and/or *LE PL/I* components. If you do not require these components, you can save the storage these components require and also the purchase price.

Please be aware!
You should not de-activate the LE/VSE Base or C components, since they are required by the CICS Transaction Server and other base components.

If you do de-activate the LE COBOL and/or LE PL/I components, one or both of these initialization messages (introduced from LE/VSE 1.4.1 onwards) will not be displayed on your system console:

- CEE3551I ("COBOL/LE Run-Time initialized").
- CEE3552I ("PLI/LE Run-Time initialized").

For a detailed description of these messages, refer to the *LE/VSE Debugging Guide and Run-Time Messages*.

To de-activate the LE COBOL and/or LE PL/I components, you should:

1. Run jobs DELLEPLI and DELLECOB, which are contained in ICCF library 59, to delete the programs for the LE COBOL and/or LE PL/I components. You can use the VSE/ESA Interactive Interface to run these jobs.
2. Run job SKLE370 (contained in ICCF library 59) or job CEEWCCSD (contained in ICCF library 62), to modify the appropriate CICS CSD entries used with LE/VSE (see also "Ensuring CICS Coexistence is Set Up Correctly" on page 52). As a result, CICS System Definition file (CSD) resources for LE COBOL and/or LE PL/I components will be removed, and the changes will be effective for all LISTs that have LE/VSE GROUP CEE appended.

   Note: You must also delete LE COBOL and/or LE PL/I components after you have carried out an FSU. For details, refer to the *VSE/ESA Planning* manual.
3. In the CSD (CICS System Definition) for each of your CICS Transaction Server sub-systems, set START=COLD. Then make a COLD start for each of your CICS Transaction Server sub-systems.

   Note: In CICS TS, a COLD start will not take place if you have the parameters JCT=NO and START=AUTO in your SIT. Only a *PARTIAL* cold start will be initiated (and not an explicit COLD start), since an EMERGENCY restart is not possible without enabling journaling.

Tailoring the COBOL COBPACKs

From LE/VSE 1.4.1 onwards, the COBOL/CICS and COBOL/BATCH run-time environments no longer use the same module names. You are therefore *not* required to support *two* LE/VSE installation sub-libraries. As a result, library cleanup activities will automatically be performed if you carry out a Fast Service Upgrade (FSU) from one VSE/ESA release to a later VSE/ESA release.

The COBOL component of the LE/VSE library is shipped with individual routines and with groupings of routines called COBPACKs. A COBPACK is a phase that contains individual library routines packaged together by the linkage editor.

The library routines can be divided into two categories:

General
These routines do not contain system-specific logic.

Environment specific
These routines contain system-specific logic and are known as environment-specific modules (ESMs), of which there are two types:
- One set for use with VSE
- One set for use with CICS

Three COBPACKs are supplied for COBOL support:
IGZCPAC contains general routines that can be used in any operating environment.
IGZCPCO contains routines that are sensitive to the operating environment. This COBPACK is used in a VSE environment.
IGZCPCC also contains routines that are sensitive to the operating environment. This COBPACK is used in a CICS environment.

When you run a program, the general COBPACK and the appropriate system-sensitive COBPACK are loaded into main storage at the start of the run. Any routines not brought in as part of a COBPACK are loaded individually as required. Under CICS, they are loaded once for each CICS system initialization; under VSE they are loaded once for each VSE task (job step).

After installation, all three COBPACKs will reside above the 16MB line. You may wish to modify the COBPACKs to include routines with RMODE 24, or to remove some of the routines distributed in the COBPACKs. If you add one or more RMODE 24 routines to a COBPACK, the system will store that COBPACK below the 16MB line. IBM provides sample jobs to help you add or remove routines in a COBPACK.

Requirements for tailoring or creating a COBPACK:
- If you want your COBPACK to be loaded above the 16MB line, do not include any RMODE 24 routines in it.
- Routines not in your COBPACK are loaded dynamically on an individual basis; thus, you can exclude any routine from the COBACK, even if your application programs use it.
- A COBPACK might be relink-edited as part of the MSHP maintenance procedure. This occurs when a routine being maintained is a routine that was included in an IBM-supplied COBPACK. If you have tailored a COBPACK to remove routines and the COBPACK is relink-edited as part of the maintenance procedure, it may no longer contain the routines you expect. You should therefore rebuild the tailored COBPACK.

- If you have built a COBPACK that is not controlled by MSHP, make sure you rebuild it whenever any routine contained in the COBPACK is maintained by a PTF.
- Routines removed from the COBPACK IGZCPCC must be added to the CICS PPT (Processing Program Table) or the CICS System Definition File.

See "Appendix D. Using COBOL with LE/VSE" on page 153 for information on the contents of each COBPACK.

If you want to alter the contents of COBPACKs, you must modify and run the appropriate job shown in Table 25. The sample jobs listed in Table 25 are supplied in members in the PRD2.SCEEBASE sublibrary.

Table 25. Sample Jobs for Modifying COBPACKs

COBPACK	Library Member
IGZCPAC	IGZWEPAC.Z
IGZCPCO	IGZWEPCO.Z
IGZCPCC	IGZWEPCC.Z

Note: These jobs are available in ICCF Library 62.

Adding and Deleting Routines in a COBPACK

The jobs in Table 25 each contain linkage editor statements to build the COBPACKs as distributed with LE/VSE.

To delete a routine from a COBPACK, you must remove (or replace with a comment statement) the linkage editor INCLUDE statement that specifies the name of the routine you want to remove.

To add a routine to a COBPACK, you must add (or remove the comment indicator from) a linkage editor INCLUDE statement that specifies the name of the routine you want to add.

If you add or delete routines in a COBPACK, and you have loaded any LE/VSE-COBOL routines into the SVA, you may need to modify your SDL procedures and reload the SVA.

Modifying the JCL for Tailoring a COBPACK

1. Modify the job card as appropriate to your site.
2. Add POWER JECL statements if your site requires them.
3. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
4. Add or delete linkage editor INCLUDE statements as required.

After you modify the job, submit it. The job will finish with a return code of 0 or 2. The return code of 2 indicates there were unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

Where to Place the Tailored COBPACKs

The sample jobs provided with LE/VSE tailor the COBPACKs and then link-edit them to replace the resident COBPACKs in the default LE/VSE sublibraries.

Alternatively, you can place them in other sublibraries, provided that the LOADs issued during run-time can find them. You must specify the sublibraries containing the customized COBPACKs ahead of, or instead of, the sublibraries containing the IBM-supplied COBPACKs. You can modify the LIBDEF PHASE,CATALOG job control statement and run the sample jobs in Table 25 on page 46 to link-edit a COBPACK into an alternative sublibrary.

Customizing the COBOL Reusable Run-Time Environment

Customizing the COBOL Reusable Environment

You can customize the COBOL reusable environment behavior, to control how program checks are handled that occur in a non-Language Environment conforming driver. The COBOL reusable environment is established with the RTEREUS run-time option or a call to IGZERRE INIT.

The IBM-supplied default setting for COBOL's reusable environment behavior is IGZERREO with REUSENV=COMPAT. Using this setting, when a program check occurs while the reusable environment is "dormant", standard VSE abends occur. The reusable environment is "dormant" between a GOBACK from a top-level COBOL program to the non-Language Environment conforming assembler driver, and the next call to a COBOL program. This behavior is compatible with the VS COBOL II and DOS/VS COBOL run-times, but it significantly impacts performance when a COBOL/VSE program is invoked repeatedly in a COBOL/LE reusable environment. The performance degradation is caused by Language Environment issuing STXIT requests when the reusable environment becomes dormant and then again upon reentering the reusable environment.

You can customize COBOL's reusable environment behavior (IGZERREO with REUSENV=OPT), so that all program checks are intercepted by Language Environment, even those that occur while the reusable environment is dormant. In this case, a program check that occurs while the reusable environment is dormant, will result in messages CEE3321C/CEE3320C from LE/VSE. However, since Language Environment does not have to issue the STXIT requests between invocations of the COBOL program, this can be faster than using REUSENV=COMPAT.

Customizing the Behaviour of the COBOL Reusable Environment

Use the IGZWARRE sample job to customize the behavior of COBOL's reusable environment. You must modify the IGZRREOP macro invocation, depending on the function that you want.

To run with VS COBOL II and DOS/VS COBOL run-time compatibility mode (that is, the user has control of program checks that occur when the COBOL reusable environment is dormant, resulting in an additional performance cost), use:

```
IGZRREOP REUSENV=COMPAT
```

To run with optimum performance (that is, Language Environment intercepts all program checks that occur when the COBOL reusable environment is dormant and converts them to CEE3321C/CEE3320C, resulting in improved performance), use:

```
IGZRREOP REUSENV=OPT
```

To modify the JCL for IGZWARRE, you should:

1. Copy the IGZERREO member from PRD2.SCEEBASE into IGZWARRE in place of the comment lines.
2. Change the REUSENV parameter on the IGZRREOP macro statement to the desired value.
3. Submit the JCL to create the deisred IGZERREO PHASE.

Note: IGZWARRE should run with a condition code no greater than 2.

Changing the C Locale Time Information

Use the sample job EDCLLOCL.Z to change the C locale time information for your site. See "Appendix E. Customizing C Locale Time Information" on page 159 for information on changing the C locale time information.

Modifying the JCL for EDCLLOCL

This job uses the IESINSRT utility supplied with VSE/ESA and the DISP=I punch facility of VSE/POWER.

1. Modify the POWER JECL and the job card as appropriate for your site.
2. If necessary, change the LIBDEF statements to match the sublibraries where you have installed LE/VSE.
3. If necessary, change the name of the sublibrary specified in the ACC SUBLIB=PRD2.SCEEBASE statement to match the sublibrary where you have installed LE/VSE.
4. Copy member EDCLOCI.A from the PRD2.SCEEBASE sublibrary into the job EDCLLOCL in place of the comment line.
5. Make the required changes to the parameters on the EDCLOCTZ macro call.

After you modify the EDCLLOCL job, submit it. The job finishes with a return code of 2. This return code of 2 indicates there were unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

Note: This sample job will replace the main C event handler phase, CEEEV003, in the installation sublibrary. If service is applied to this phase you may need to rerun EDCLLOCL.

Including the CSD for LE/VSE Support Under CICS

Since there are no PPT or PCT members shipped with LE/VSE, you use the CSD to include LE/VSE support under CICS:

Table 26. Including LE/VSE Support under CICS Using the CSD

To include	Use member	In sublibrary
LE/VSE base (mandatory)*	CEECCSD.Z	PRD2.SCEEBASE
COBOL (optional)*	IGZCCSD.Z	PRD2.SCEEBASE
PL/I (optional)*	IBMCCSD.Z	PRD2.SCEEBASE
C (mandatory)*	EDCCCSD.Z	PRD2.SCEEBASE
C/VSE Code Converter (optional)*	EDCUCSD.Z	PRD2.SCEEBASE

Notes:

1. * These LE/VSE-specific sample books are also referred to via member CEEWCCSD in ICCF Library 62 (or the equivalent member SKLE370 in library 59). If you need to update the CICS Resource Definition File (CSD), one (only) of these members should be edited and executed. The situations in which explicit customization might be required, are discussed on pages 44 and 52.
2. If you use additional LE/VSE SVA loadlists together with CICS TS subsystems, make sure you enable the related `USESVACOPY(YES)` attribute contained in the CICS CSD file. Refer to these skeletons (contained in ICCF library 62) for example of such enablement-support:
   - CEETSCSD
   - EDCTSCSD
   - IGZTSCSD
   - IBMTSCSD

   From VSE/ESA 2.6 onwards, the modules contained in the pre-installed SVA loadlists $SVACEE and $SVAEDCM are already enabled with the `USESVACOPY(YES)` attribute. The IBM-shipped system also has a CICS SIT table setting of `SVA=YES`.

Tailoring the CICS Destination Control Table (Optional)

Tailoring the CICS Destination Control Table (DCT) entries contained in DFHDCT is an *optional* task, and you need to tailor DFHDCT entries only if you do not wish to use the LE/VSE default-implementation.

These CICS transient data queues are specific to LE/VSE:

- CESE (default MSGFILE setting): LE/VSE messages, dumps, and reports are written to this queue. Each record written to the CESE queue has a header with terminal ID, transaction ID, date, and time. This queue is also used by C for `stderr` output and by PL/I for stream output data.
- CESO: C `stdout` stream output is written to this queue. The definition for this queue is required only if you use C. Each record written to the CESO queue has a header with terminal ID and transaction ID.

Figure 9 illustrates the format of a transient data queue entry.

Figure 9. Format of a Transient Data Queue Entry

ASA The American National Standard carriage-control character

Terminal ID A 4-character terminal identifier

Transaction ID
A 4-character transaction identifier

sp A space

Timestamp The date and time displayed in the same format as that returned by the CEELOCT service

Message The message identifier and message text

These queues can each have an intrapartition, extrapartition, or indirect destination. The block size for the transient data queue CESE must be at least 175 and for the transient data queue CESO at least 137. The record format for each should be variable unblocked.

Members That You Use for Your DCT Implementation

The following members support cross-product defaults, as well as optional DCT implementations:

DFHDCT/DFHDCT1.A
The CICS/VSE-supplied sample DCT definitions, used for directing output to SYSLST. This member is used as the default in shipped VSE/ESA systems. This setup is also implemented in member DFHDCTCO (supplied with the VSE/ESA Interactive Interface), for optional CICS coexistence installation. These resource definitions are not applicable for the CICS Transaction Server.

DFH$DCTD.A

The CICS TS-supplied sample DCT definitions, used for directing output to SYSLST. This member is used as the default in shipped VSE/ESA systems.

IESZDCT.A The VSE/ESA-supplied member used for implementing TYPE=INDIRECT implementation (for LE/VSE destinations CESE and CESO) to the VSE/ESA Interactive Interface's "Inspect Message Log" dialog. IESZDCT.A is the default member used with shipped VSE/ESA systems containing CICS TS setup. DFHDCTSP and DFHDCTC2 are the related VSE/ESA Interactive Interface skeletons, which you find in ICCF library 59.

CEECDCT.A (optional)

An LE/VSE-supplied member that you can optionally use in order to direct output to *disk* (rather than SYSLST). It contains both TYPE=SDSCI and TYPE=EXTRA definitions. To avoid compiler errors, before using CEECDCT.A you must first make some environment-specific planning, and appropriate DCT customization changes. Specifically, you must ensure that you are not using double DCT definitions. This is because the copybooks provided by CICS and the VSE/ESA Interactive Interface contain redefinitions, and split TYPE=SDSCI and TYPE=EXTRA into separate members.

Note: If you define CESE and CESO as extrapartition destinations assigned to disk, your CICS startup job must contain the appropriate DLBL, EXTENT, and ASSGN entries.

In addition, to tailor your DFHDCT, refer to *CICS Transaction Server for VSE/ESA Resource Definition (Macro)* for details of the DFHDCT macro and the definitions of the queues and associated buffers.

Ensuring CICS Coexistence is Set Up Correctly

If you perform a Fast Service Upgrade (version upgrade) from VSE/ESA 2.4.x to VSE/ESA 2.5 or later, you are responsible for setting your CICS CSD definitions so that any CICS/VSE installations you have, run correctly with LE/VSE 1.4.1 or later.

Note!
Any CICS/VSE (non-shared) CSD file you have will still hold resource definitions that are related to LE/VSE 1.4.0.

Form LE/VSE 1.4.1 onwards, LE/VSE has been internally restructured to allow you to use LE/COBOL unique naming conventions, single product ship library, and so on. As a result, you must include all related changes in your CICS/VSE CSD. If you do not do so, you will receive LE/VSE initialization errors under CICS/VSE (such as the abend **4093** RC **36**).

To avoid this type of problem, you must edit and execute skeleton SKLE370 (of ICCF lib 59), which points to any CICS/VSE subsystem you use (for example, `GRPLIST` and `// DLBL` references to actual CSD files).

There are two skeletons that you might need to edit and execute:
- SKPREPCO is the skeleton you use for CICS/VSE systems in which a CSD is not shared.
- SKPREPSO is the skeleton you use for CICS systems in which a CSD is shared between CICS TS and CICS/VSE.

From VSE/ESA 2.5 onwards, the CICS Transaction Server is the shipped CICS subsystem of a VSE/ESA base system. Therefore you do *not* need to edit and execute skeleton SKPREPCO, if you perform a Fast Service Upgrade from VSE/ESA 2.4.x to VSE/ESA 2.5 or later.

All related cleanup activities (for removing LE/VSE 1.4.0 and establishing LE/VSE 1.4.1 or later) are performed *automatically*, even if you are performing a version upgrade.

If your CSD file *is* separate from your CSD file for CICS Transaction Server, here is the job you require to update your CSD file for CICS/VSE:

```
* $$ JOB JNM=DFHCSDOL,CLASS=0,DISP=D
// JOB DFHCSDOL UPGRADE THE CSD FILE FOR COEXISTENT CICS
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD2.CICSOLDP,PRD2.SCEEBASE,PRD1.BASE)
// DLBL DFHCSD,'CICSO.CSD',0,VSAM,                                      X
               CAT=VSESPUC
// EXEC DFHCSDUP,SIZE=600K        INIT AND LOAD CICS CSD VSAM FILE
 UPGRADE USING(IESMODEL)
 UPGRADE USING(DFHCU230)
 UPGRADE USING(DFHCU23F)
 MIGRATE TABLE(DFHPPTCO) TOGROUP(VSESPO)
 MIGRATE TABLE(DFHPCTCO) TOGROUP(VSESPT)
 COPY GROUP(VSESPT) TO(VSESPO)
 DELETE GROUP(VSESPT)
 APPEND LIST(DFHLIST) TO(VSELSTO)
 ADD GROUP(VSETYPE) LIST(VSELSTO)
 ADD GROUP(VSETERM) LIST(VSELSTO)
 ADD GROUP(VSETERM1) LIST(VSELSTO)
 ADD GROUP(VSESPO) LIST(VSELSTO)
* $$ SLI MEM=CEECCSD.Z,S=(PRD2.SCEEBASE)
* $$ SLI MEM=IBMCCSD.Z,S=(PRD2.SCEEBASE)
* $$ SLI MEM=IGZCCSD.Z,S=(PRD2.SCEEBASE)
* $$ SLI MEM=EDCCCSD.Z,S=(PRD2.SCEEBASE)
* THESE CODESET CONVERTERS CAN BE OPTIONALLY INCLUDED HERE
* $$ SLI MEM=EDCUCSD.Z,S=(PRD2.SCEEBASE)
   ADD GROUP(CEE) LIST(VSELSTO)
* $$ SLI MEM=IPNCSD.Z,S=(PRD1.BASE)
  ADD GROUP(TCPIP) LIST(VSELSTO)
   LIST ALL
/*
/&
* $$ EOJ
```

Figure 10. Job to Update CSD File for CICS/VSE

Notes:

1. The above job is an extract of the skeleton SKPREPCO, that is provided with the *VSE/ESA Interactive Interface*.
2. Job execution only needs to be considered for a non-shared CICS CSD setup (the CICS coexistence environment).
3. Refer to the skeleton SKPREPCO (contained in ICCF Library 59) for details of how to setup your non-shared CICS CSD file for CICS/VSE.

Chapter 3. Maintaining LE/VSE

This chapter describes how to replace or reinstall LE/VSE, and how to apply service updates to LE/VSE. To effectively use the maintenance procedures, you must have already installed LE/VSE and any required products.

Note: Since LE/VSE is now distributed together with the VSE/ESA Base, the section "Reinstalling LE/VSE" has been removed from this chapter. You must now reinstall LE/VSE using VSE/ESA, either as:
- A Fast Service Upgrade (FSU). Refer to *VSE/ESA System Upgrade and Service* for details.
- An Initial Installation. Refer to *VSE/ESA Installation* for details.

You Should Never...
Remove the LE/VSE Base or LE/VSE C components from your system!.

Separating User-Customized Modules From IBM-Shipped Code

From LE/VSE 1.4.1 onwards, there are changes in the way LE/VSE is serviced. LE/VSE now uses the approach of using the *phase service* wherever possible.

There are, however, some exceptions to this approach: LE/VSE still ships OBJECTs for supporting customization tasks, such as COBPACK tailoring (described on page 45).

In general, LE/VSE attempts to use a hybrid service approach, thereby separating user-customized modules (such as run-time option tailoring, exits, and so on) from IBM-shipped phases. This is especially true for batch and CICS initialization phases CEEBINIT and CEECCICS. As a result, CEEBINIT and CEECCICS initialization phases no longer require that customers re-link such phases, with the resulting danger of errors occuring (for example, when applying PTF service).

These changes should improve the servicability and reliability of LE/VSE.

The changes to the CEEBINIT and CEECCICS initialization phases result in minor changes to several customization jobs, such as run-time option generation, and exit tailoring (CEEWCOPT and CEEWDOPT, CEEWDEXT, and so on).

The initialization phases remain unchanged during the build of the customization option phases.

Related Section:
- "Changing Run-Time Options Defaults" on page 15
- "Customizing LE/VSE Abnormal Termination Exits" on page 35
- "Tailoring the COBOL COBPACKs" on page 45

Applying Service Updates

You might need to apply maintenance or service updates to LE/VSE periodically. There are two types of formally supported software fixes. One is the program temporary fix (PTF) applied as corrective service or as preventive maintenance. The other is the authorized program analysis report (APAR) fix applied as a code replacement in a corrective maintenance mode.

For details of how to apply maintenance or service updates, refer to the *VSE/ESA System Upgrade and Service*.

What You Receive

If you report a problem with LE/VSE to your IBM Support Center, you will receive a tape containing one or more APARs or PTFs which have been created to solve your problem.

You may also receive a list of prerequisite APARs or PTFs which should have been applied to your system before applying the current service. These prerequisite APARs or PTFs may relate to LE/VSE or any other licensed product you have installed, including VSE/ESA.

You apply service to LE/VSE using the *VSE/ESA Interactive Interface*.

Step 1: Check Prerequisite APARs or PTFs

Prerequisite APARs or PTFs are APARs or PTFs that need to be applied to your system before you can apply the current maintenance. These APARs or PTFs may apply to LE/VSE or any licensed program you have installed at your site.

Note: The *VSE/ESA Interactive Interface* provides different types of service dialogs (for example, to check for existing components, or to lookup specific APARs and PTFs). In addition, the corresponding support is provided for analyzing and applying PTFs.

Your IBM Support Center will have given you a list of any relevant prerequisite APARs or PTFs. Most probably they will already be applied to your system. You can verify this by retracing the APARs and PTFs in your system history file. The job shown in Figure 11 shows how to retrace APARs and PTFs in the system history file.

```
// JOB CEEWRETR Retrace APARs and PTFs
// EXEC MSHP,SIZE=900K
RETRACE APARS
RETRACE PTFS
/*
/&
```

Figure 11. Job to Retrace APARs and PTFs

Use the listing produced when you run this job to check that you have already applied any prerequisite APARs or PTFs. If you have not, your IBM Support Center will arrange to send them to you and you should apply them before applying other service.

Step 2: Run the Installation Verification Program (IVP)

After you have applied all the files on the service tape, run the appropriate installation verification programs to ensure that LE/VSE functions properly.

Notes:

1. Some customizing tasks for LE/VSE modify phases which you might have linkedited into your own sublibrary. Examples are the customizing of the options modules and the LIOCS phases (which re-link CEEYCDO, CEEYDUO, and CEEYPRO).
2. If you have linkedited any of the above phases into your own sublibrary, and you apply service which modifies these phases, make sure that the service is also applied to the version of the phase you are using. The most reliable way to do this is to re-run the customizing jobs after the service has been applied.
3. If you apply the service to the phases in the installation sublibrary and do not re-run your customizing jobs (continuing to use the phases from your own sublibrary), then the applied service will not take effect. This is especially important if there are co-requisite PTFs applied to other products (such as CICS/VSE).
4. Similar problems may arise if service is applied to phases which you have loaded into the SVA or into COBOL COBPACKs, and you do not reload the SVA or COBPACK after applying the service.

To Report a Problem with LE/VSE

Report any difficulties you have using this product to your IBM Support Centre. Table 27 identifies the component IDs for LE/VSE.

Table 27. LE/VSE Component IDs and CLCs

Component Id	CLC	Description
5686-066-32	65K	LE Common base, containing information written in: • uppercase and mixed-case U.S. English • Japanese NLF
5686-066-33	65L	LE C-specific base, containing information written in: • uppercase and mixed-case U.S. English • Japanese NLF
5686-066-34	65M	Optional LE DBCS Locale component (see note below)
5686-094-03	6EW	LE COBOL-specific base and CICS, containing information written in: • uppercase U.S. English • Japanese NLF
5686-094-06	6EX	LE PL/I-specific base, containing information written in: • uppercase and mixed-case U.S. English • Japanese NLF

Appendix A. LE/VSE Run-Time Options

This appendix first describes the LE/VSE run-time options in alphabetical sequence. Where noted, some of the run-time options might be used only by a COBOL or a C program. A quick reference table is provided for convenience. In addition, there is a table that maps LE/VSE run-time options to HLL run-time options to help you plan your customization.

The CEEXOPT macro is used to specify installation default run-time options in the CEEDOPT PHASE for batch programs, and the CEECOPT PHASE for CICS. The same macro is used to specify application-specific run-time options in the CEEUOPT CSECT. See "Setting Installation-Wide Default Options with the CEEXOPT Macro" on page 15 for details.

The syntax described here is specific to the *CEEDOPT* form of the file used at installation time. All suboptions must be specified and no abbreviations are permitted in CEEDOPT. IBM-supplied defaults are indicated for planning information only.

This appendix also includes additional CICS-wide run-time option information, under these headings:

Quick Reference Table of LE/VSE Run-Time Options

Table 28. Run-Time Options Quick Reference

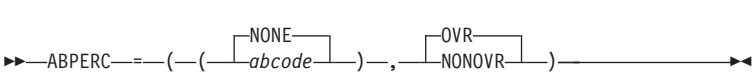
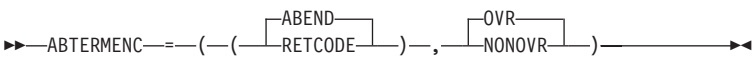
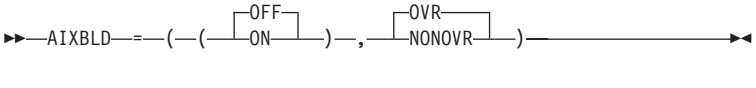
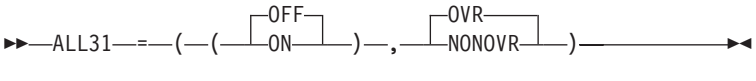
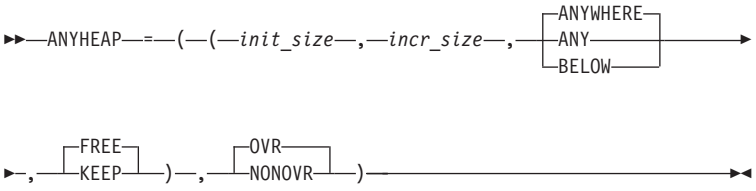
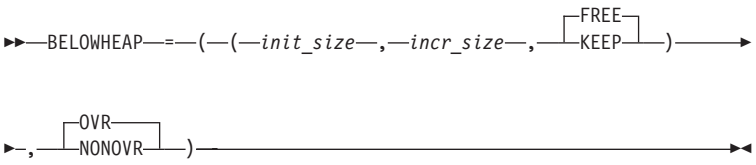
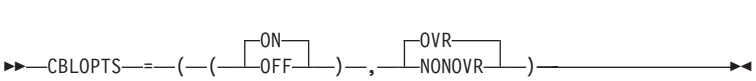
Run-Time Options	Function	Page
	Exempts a specified VSE cancel code, program-interruption code, or user abend code from LE/VSE condition handling.	69
	Sets the enclave termination behavior for an enclave ending with an unhandled condition of severity 2 or greater.	71
	(COBOL only) Invokes the access method services (AMS) for VSAM key-sequenced data sets (KSDS) and relative-record data sets (RRDS) to complete the file and index definition procedures for COBOL routines.	73
	Indicates whether an application does or does not run entirely in AMODE(31).	74
	Controls allocation of library heap storage not restricted to below the 16MB line.	75
	Controls allocation of library heap storage below the 16MB line.	77
	(COBOL only) Specifies the format of the argument string on application invocation when the main program is COBOL.	78

Table 28. Run-Time Options Quick Reference (continued)

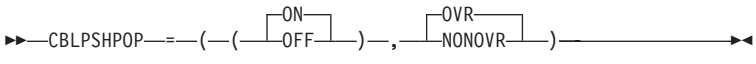
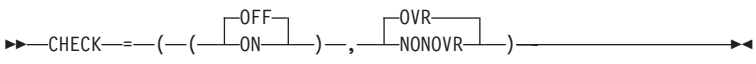
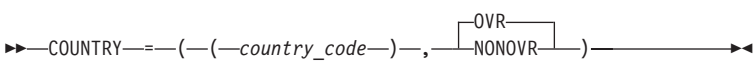
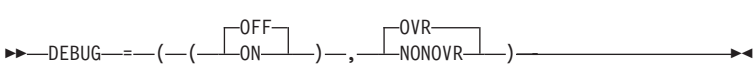
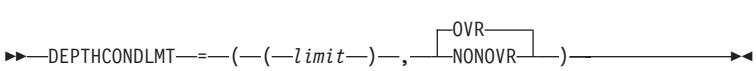
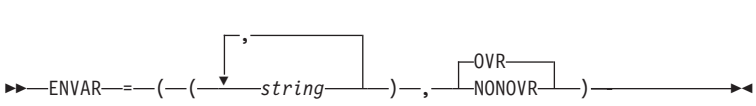
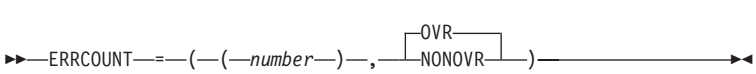
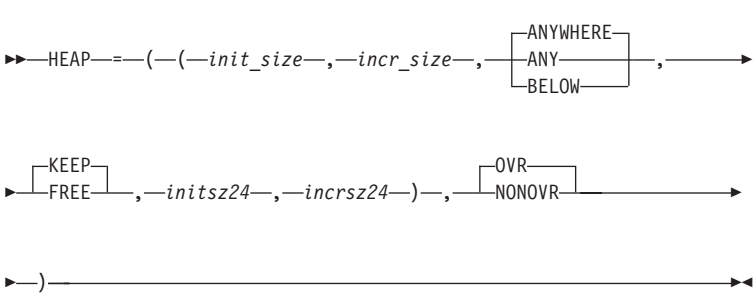
Run-Time Options	Function	Page
	(COBOL only) Controls whether CICS PUSH HANDLE and CICS POP HANDLE commands are issued when a COBOL subprogram is called.	79
	(COBOL only) Indicates whether "checking errors" within an application should be detected.	80
	Specifies the default formats for date, time, currency symbol, decimal separator, and the thousands separator based on a country.	81
	(COBOL only) Activates the COBOL batch debugging features specified by the "debugging lines" or the USE FOR DEBUGGING declarative.	82
	Limits the extent to which conditions can be nested.	83
	(C only) Sets the initial values for the environment variables specified in *string*.	84
	Specifies how many conditions of severity 2, 3, and 4 can occur per thread before an enclave terminates abnormally.	85
	Controls allocation of the heaps.	86

Table 28. Run-Time Options Quick Reference (continued)

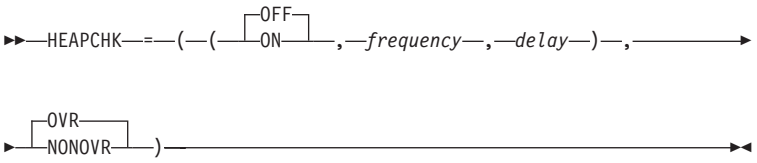
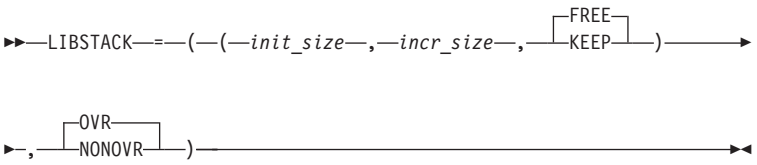
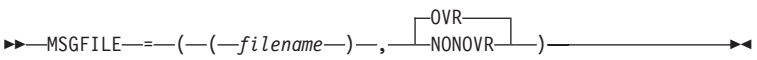
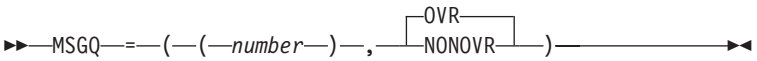
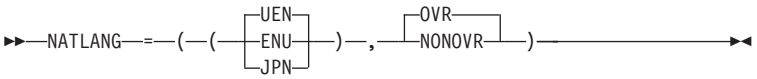
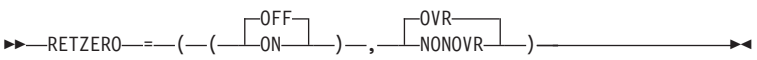
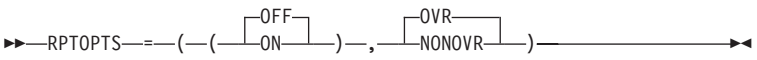
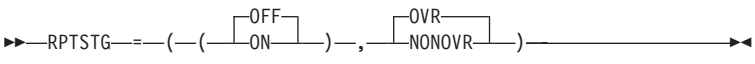
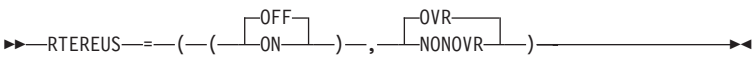
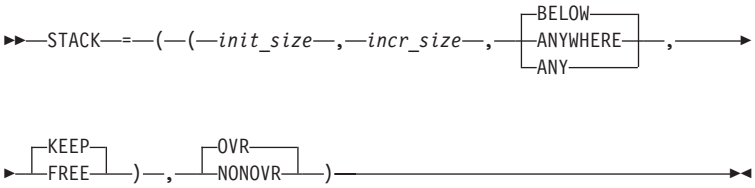
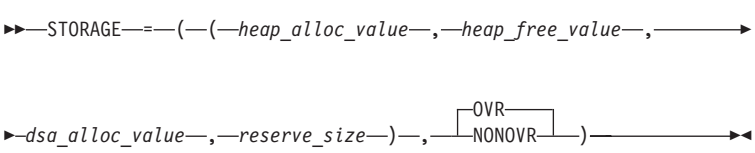
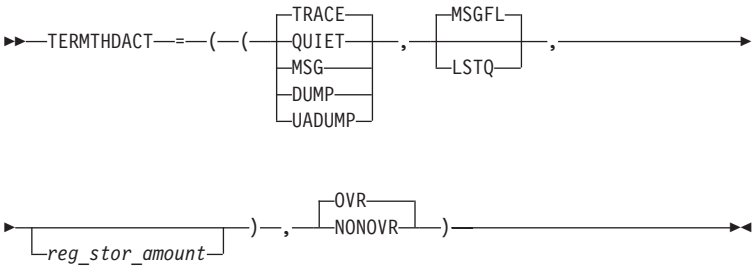
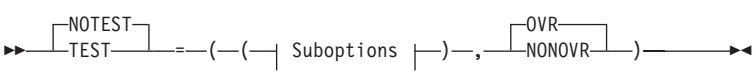
Run-Time Options	Function	Page
	Provides a checking facility to verify that the heap storage has not been damaged.	88
	Controls the allocation of the thread's library stack storage.	89
	Specifies the *filename* of the run-time diagnostics file.	91
	Specifies the number of ISI blocks allocated on a per-thread basis during execution.	92
	Specifies the national language to use for the run-time environment.	93
	(COBOL only) Ensures that, if the run unit does not abend or terminate abnormally, the user return code will be set to zero regardless of the contents of register 15 or the RETURN-CODE special register.	94
	Specifies that a report of the run-time options in use by the application be generated.	95

Table 28. Run-Time Options Quick Reference (continued)

Run-Time Options	Function	Page
	Specifies that a report of the storage used by the application be generated at the end of execution.	97
	Initializes the run-time environment to be reusable when the first COBOL program is invoked.	100
	Controls the allocation and management of thread-level stack storage.	101
	Controls the value of storage that is allocated and freed.	103
	Sets the level of information produced due to an unhandled error of severity 2 or greater.	106
	Specifies that a debug tool is to be given control according to the suboptions specified.	109

Suboptions:

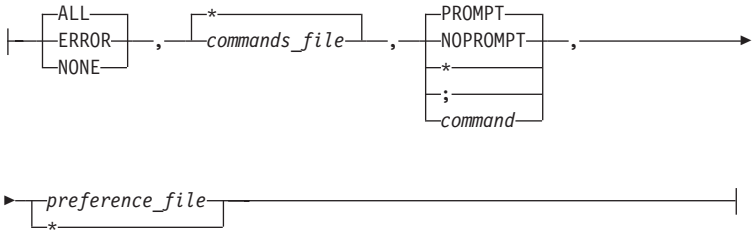
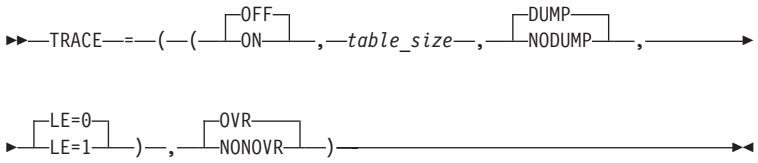
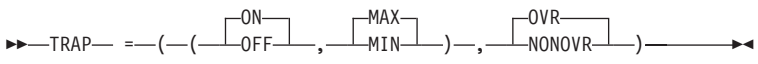
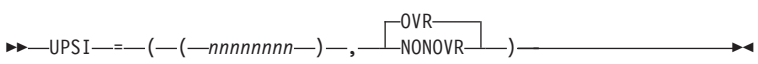
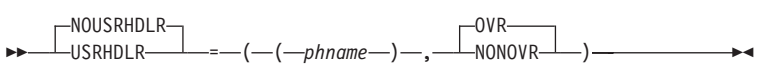
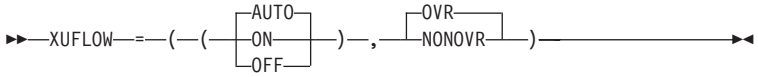


Table 28. Run-Time Options Quick Reference (continued)

Run-Time Options	Function	Page
	Determines whether LE/VSE run-time library tracing is active.	111
	Specifies how LE/VSE routines handle abends and program interrupts.	112
	(COBOL only) Sets the eight UPSI switches on or off. Affects only COBOL programs.	115
	Registers a user condition handler at stack frame 0, allowing you to register a user condition handler without having to include a call to CEEHDLR in your application and then recompile the application.	116
	Specifies whether an exponent underflow causes a program interrupt.	117

Language Run-Time Option Mapping

Table 29. C/370 and LE/VSE Options

C/370 Option	LE/VSE Equivalent	Notes
ISAINC (*incr_size*)	STACK (*incr_size*)	The C/370 ISAINC run-time option is mapped to the LE/VSE STACK run-time option for compatibility. It affects all languages in the enclave.
ISASIZE (*init_size*)	STACK (*init_size*)	The C/370 ISASIZE run-time option is mapped to the LE/VSE STACK run-time option for compatibility. It affects all languages in the enclave.
LANGUAGE	NATLANG	The C/370 LANGUAGE run-time option is mapped to the LE/VSE NATLANG run-time option for compatibility. It affects all languages in the enclave.
REPORT	RPTSTG(ON)	The C/370 REPORT run-time option is mapped to the LE/VSE RPTSTG(ON) run-time option for compatibility. It affects all languages in the enclave.
NOREPORT	RPTSTG(OFF)	The C/370 NOREPORT run-time option is mapped to the LE/VSE RPTSTG(OFF) run-time option for compatibility. It affects all languages in the enclave.

Table 29. C/370 and LE/VSE Options (continued)

C/370 Option	LE/VSE Equivalent	Notes
SPIE	TRAP(ON)	The C/370 SPIE run-time option is mapped to the LE/VSE TRAP(ON) run-time option for compatibility. It affects all languages in the enclave. The mapping of SPIE might differ depending upon other options specified. For more information, see "TRAP" on page 112.
NOSPIE	TRAP(OFF)	The C/370 NOSPIE run-time option is mapped to the LE/VSE TRAP(OFF) run-time option for compatibility. It affects all languages in the enclave. The mapping of NOSPIE might differ depending upon other options specified. For more information, see "TRAP" on page 112.
STAE	TRAP(ON)	The C/370 STAE run-time option is mapped to the LE/VSE TRAP(ON) run-time option for compatibility. It affects all languages in the enclave. The mapping of STAE might differ depending upon other options specified. For more information, see "TRAP" on page 112.
NOSTAE	TRAP(OFF)	The C/370 NOSTAE run-time option is mapped to the LE/VSE TRAP(OFF) run-time option for compatibility. It affects all languages in the enclave. The mapping of NOSTAE might differ depending upon other options specified. For more information, see "TRAP" on page 112.

Table 30. DOS/VS COBOL and LE/VSE Options

DOS/VS COBOL Option	LE/VSE Equivalent	Notes
A (SYSPARM)	AIXBLD	The LE/VSE AIXBLD run-time option is compatible with the DOS/VS COBOL SYSPARM='A' run-time option. It affects only COBOL programs in the enclave.
NA (SYSPARM)	NOAIXBLD	The LE/VSE NOAIXBLD run-time option is compatible with the DOS/VS COBOL SYSPARM='NA' run-time option. It affects only COBOL programs in the enclave.
D (SYSPARM)	DEBUG	The LE/VSE DEBUG run-time option is compatible with the DOS/VS COBOL SYSPARM='D' run-time option. It affects only COBOL programs in the enclave.
ND (SYSPARM)	NODEBUG	The LE/VSE NODEBUG run-time option is compatible with the DOS/VS COBOL SYSPARM='ND' run-time option. It affects only COBOL programs in the enclave.
UPSI	UPSI	The LE/VSE UPSI run-time option replaces the DOS/VS COBOL UPSI run-time option provided by the // UPSI job control statement. The UPSI switches set by the // UPSI job control statement are not available to COBOL programs under LE/VSE.

Table 31. VS COBOL II and LE/VSE Options

VS COBOL II Option	LE/VSE Equivalent	Notes
AIXBLD	AIXBLD	The LE/VSE AIXBLD run-time option is compatible with the VS COBOL II AIXBLD run-time option. It affects only COBOL programs in the enclave.

Table 31. VS COBOL II and LE/VSE Options (continued)

VS COBOL II Option	LE/VSE Equivalent	Notes
NOAIXBLD	NOAIXBLD	The LE/VSE NOAIXBLD run-time option is compatible with the VS COBOL II NOAIXBLD run-time option. It affects only COBOL programs in the enclave.
DEBUG	DEBUG	The LE/VSE DEBUG run-time option is compatible with the VS COBOL II DEBUG run-time option. It affects only COBOL programs in the enclave.
NODEBUG	NODEBUG	The LE/VSE NODEBUG run-time option is compatible with the VS COBOL II NODEBUG run-time option. It affects only COBOL programs in the enclave.
LANGUAGE	NATLANG	The VS COBOL II LANGUAGE run-time option is mapped to the LE/VSE NATLANG run-time option for compatibility. It affects all languages in the enclave.
LIBKEEP	Not applicable	There is no LE/VSE equivalent for the VS COBOL II LIBKEEP run-time option. To obtain similar performance function, use the Library Routine Retention (LRR) feature described in . *LE/VSE Programming Guide*.
NOLIBKEEP	Not applicable	There is no LE/VSE equivalent for the VS COBOL II NOLIBKEEP run-time option.
MIXRES	Not applicable	There is no LE/VSE equivalent for the VS COBOL II MIXRES run-time option. MIXRES applications supported by LE/VSE always exhibit RES behavior.
NOMIXRES	Not applicable	There is no LE/VSE equivalent for the VS COBOL II NOMIXRES run-time option. MIXRES applications supported by LE/VSE always exhibit RES behavior.
RTEREUS	RTEREUS	The LE/VSE RTEREUS run-time option is compatible with the VS COBOL II RTEREUS run-time option. The RTEREUS option is intended for use when the main program of an enclave is a COBOL program. The RTEREUS option can cause problems for HLLs other than COBOL.
NORTEREUS	NORTEREUS	The VS COBOL II NORTEREUS run-time option is compatible with the VS COBOL II NORTEREUS run-time option.
SIMVRD	Not applicable	There is no LE/VSE equivalent for the VS COBOL II SIMVRD run-time option.
NOSIMVRD	Not applicable	There is no LE/VSE equivalent for the VS COBOL II NOSIMVRD run-time option.
SPOUT	RPTOPTS(ON) RPTSTG(ON)	The VS COBOL II SPOUT run-time option is mapped to the LE/VSE RPTOPTS(ON) and RPTSTG(ON) run-time options for compatibility. It affects all languages in the enclave.
NOSPOUT	RPTOPTS(OFF) RPTSTG(OFF)	The VS COBOL II NOSPOUT run-time option is mapped to the LE/VSE RPTOPTS(OFF) and RPTSTG(OFF) run-time options for compatibility. It affects all languages in the enclave.
SSRANGE	CHECK(ON)	The VS COBOL II SSRANGE run-time option is mapped to the LE/VSE CHECK(ON) run-time option for compatibility. It affects only COBOL programs in the enclave.
NOSSRANGE	CHECK(OFF)	The VS COBOL II NOSSRANGE run-time option is mapped to the LE/VSE CHECK(OFF) run-time option for compatibility. It affects only COBOL programs in the enclave.

Table 31. VS COBOL II and LE/VSE Options (continued)

VS COBOL II Option	LE/VSE Equivalent	Notes
STAE	TRAP(ON)	The VS COBOL II STAE run-time option is mapped to the LE/VSE TRAP(ON) run-time option for compatibility. It affects all languages in the enclave. The mapping of STAE might differ depending upon other options specified. For more information, see "TRAP" on page 112. .
NOSTAE	TRAP(OFF)	The VS COBOL II NOSTAE run-time option is mapped to the LE/VSE TRAP(OFF) run-time option for compatibility. It affects all languages in the enclave. The mapping of NOSTAE might differ depending upon other options specified. For more information, see "TRAP" on page 112. .
UPSI	UPSI	The VS COBOL II UPSI option is processed for compatibility.
WSCLEAR	STORAGE(00)	The VS COBOL II WSCLEAR run-time option is not supported under LE/VSE. For behavior similar to that produced by the VS COBOL II WSCLEAR run-time option, use the LE/VSE STORAGE(00) run-time option.
NOWSCLEAR	STORAGE(NONE)	The VS COBOL II NOWSCLEAR run-time option is not supported under LE/VSE. For behavior similar to that produced by the VS COBOL II NOWSCLEAR run-time option, use the LE/VSE STORAGE(NONE) run-time option.

Table 32. DOS PL/I and LE/VSE Options

DOS PL/I Option	LE/VSE Equivalent	Notes
COUNT	Not applicable	There is no LE/VSE equivalent for the DOS PL/I COUNT run-time option.
NOCOUNT	Not applicable	There is no LE/VSE equivalent for the DOS PL/I NOCOUNT run-time option.
FLOW	Not applicable	There is no LE/VSE equivalent for the DOS PL/I FLOW run-time option.
NOFLOW	Not applicable	There is no LE/VSE equivalent for the DOS PL/I NOFLOW run-time option.
ISASIZE (*init_size*)	STACK (*init_size*)	The DOS PL/I ISASIZE run-time option is mapped to the LE/VSE STACK run-time option for compatibility. It affects all languages in the enclave.
REPORT	RPTSTG(ON)	The DOS PL/I REPORT run-time option is mapped to the LE/VSE RPTSTG(ON) run-time option for compatibility. It affects all languages in the enclave.
NOREPORT	RPTSTG(OFF)	The DOS PL/I NOREPORT run-time option is mapped to the LE/VSE RPTSTG(OFF) run-time option for compatibility. It affects all languages in the enclave.
STAE	TRAP(ON)	The DOS PL/I STAE run-time option is mapped to the LE/VSE TRAP(ON) run-time option for compatibility. It affects all languages in the enclave. The mapping of STAE might differ depending upon other options specified. For more information, see "TRAP" on page 112. .

Table 32. DOS PL/I and LE/VSE Options (continued)

DOS PL/I Option	LE/VSE Equivalent	Notes
NOSTAE	TRAP(OFF)	The DOS PL/I NOSTAE run-time option is mapped to the LE/VSE TRAP(OFF) run-time option for compatibility. It affects all languages in the enclave. The mapping of NOSTAE might differ depending upon other options specified. For more information, see "TRAP" on page 112. .

COBOL Compatibility

VS COBOL II supports an order of run-time options and program options that is the reverse of that of LE/VSE: program arguments precede run-time options in COBOL. To ensure compatibility with COBOL, LE/VSE provides the run-time option CBLOPTS, which specifies whether run-time options or program arguments are first in the character parameter.

For example:

CBLOPTS=OFF:

```
// EXEC PGM=program-name,PARM='run-time-options/program-arguments'
// EXEC PGM=program-name,PARM='run-time-options/'
// EXEC PGM=program-name,PARM='/program-arguments'
// EXEC PGM=program-name,PARM='program-arguments'
```

CBLOPTS=ON:

```
// EXEC PGM=program-name,PARM='program-arguments/run-time-options'
// EXEC PGM=program-name,PARM='/run-time-options'
// EXEC PGM=program-name,PARM='program-arguments/'
// EXEC PGM=program-name,PARM='program-arguments'
```

LE/VSE Run-Time Options

The run-time options that can be modified in the CEEDOPT CSECT are described here in detail in the form specific to CEEDOPT.

IBM-supplied default keywords appear **above** the main path or options path in the syntax diagrams. In the parameter list, IBM-supplied default choices are underlined. For a full description of the syntax of LE/VSE run-time options, see *LE/VSE Programming Reference*.

Some of these run-time options descriptions refer to the severity of conditions. The values that can occur as condition token severity codes, and their meanings, are listed here:

0 An informational message (or, if the entire token is zero, no information)

1 An attention message. Service completed, probably correctly.

2 An error message. Correction attempted. Service completed, perhaps incorrectly.

3 A severe error message. Service not completed.

4 A critical error message. Service not completed and condition signaled. A critical error is a condition that jeopardizes the environment. If a critical error occurs during an LE/VSE callable service, it is always signaled to the condition manager instead of being returned synchronously to the caller.

ABPERC

ABPERC exempts a specified VSE cancel code, program-interruption code, or user abend code from LE/VSE condition handling, and causes an operating system request to be issued to terminate the enclave.

The ABPERC option is a debugging aid that can be used by an application that runs with TRAP set to ON. This provides LE/VSE semantics for everything except one VSE cancel, program interruption, or user abend, whose code you specify.

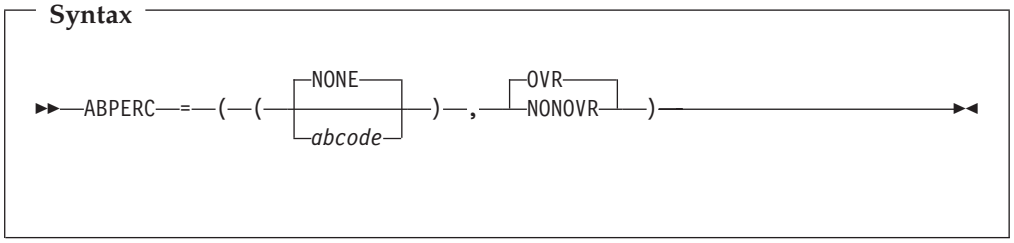
When you run with ABPERC and encounter the specified VSE cancel, interruption, or user abend:

- User condition handlers are not enabled.
- No storage report or run-time options report is generated.
- No LE/VSE messages or LE/VSE dump output is generated.
- The assembler user exit is not driven for enclave termination.
- The abnormal termination exit (if there is one) is not driven.
- Files opened by HLLs are not closed by LE/VSE, so records might be lost.
- Resources acquired by LE/VSE are not freed.
- The debug tool is not notified of the error.

You can also specify a list of VSE cancel codes, interruption codes, and user abend codes in the CEEBXITA assembler user exit for the condition manager to exempt from LE/VSE condition handling.

IBM-Supplied Default: ABPERC=((NONE),OVR)

Syntax



NONE
Specifies that all abnormal terminations are handled according to LE/VSE condition handling semantics.

abcode
Specifies the VSE cancel code, program-interruption code, or user abend code to be exempted from LE/VSE condition handling.

abcode can be specified as:

S*hh* A VSE cancel code where ***hh*** is the hexadecimal cancel code.

I*hh* A VSE interruption code where ***hh*** is the hexadecimal interruption code.

U*dddd* A user abend code where ***dddd*** is a decimal user-issued abend code.

Any 4-character string can also be used as an *abcode*.

You can identify only one VSE cancel code, program-interruption code, or abend code with this option.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- LE/VSE ignores ABPERC=((S20),...). The VSE cancel code 20 indicates a program check has occurred. In this instance, LE/VSE condition handling semantics are in effect. You can, however, specify one program check interruption code, in the form I*hh*, to be exempted from LE/VSE condition handling.
- CICS consideration—ABPERC is ignored under CICS.

For More Information

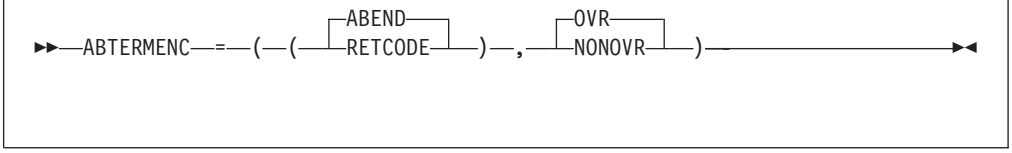
- For more information about the CEEBXITA assembler user exit, see *LE/VSE Programming Guide*.
- For more information about VSE cancel codes, see *VSE/ESA Messages and Codes, Volume 1*.
- For a list of program-interruption codes, see the *Principles of Operations* manual for your machine.

ABTERMENC

ABTERMENC sets the enclave termination behavior for an enclave ending with an unhandled condition of severity 2 or greater. TRAP(ON) must be in effect for ABTERMENC to have an effect when the unhandled condition is a program check or an abend. ABTERMENC is always in effect for unhandled conditions raised by the CEESGL callable service, regardless of the setting of the TRAP option.

IBM-Supplied Default: ABTERMENC=((ABEND),OVR)

Syntax



ABEND

Specifies that LE/VSE terminates the enclave with an abend, regardless of the setting of the CEEAUE_ABND flag by the assembler user exit. In the batch environment, LE/VSE produces run-time message CEE3321C or CEE3322C, and issues an operating system request to terminate the enclave. In the CICS environment, LE/VSE issues an EXEC CICS ABEND. The setting of the CEEAUE_ABND flag affects the abend processing, as follows:

When CEEAUE_ABND is off, the following occurs:

- Abend code: LE/VSE sets an abend code value that depends on the type of unhandled condition.
- Reason code: LE/VSE sets a reason code value that depends on the type of unhandled condition.
- Abend dump attribute: LE/VSE does not request a system dump.

When CEEAUE_ABND is on, LE/VSE uses values set by the assembler user exit to determine abend processing:

- Abend code: Value of the CEEAUE_RETURN parameter of the assembler user exit.
- Reason code: Value of the CEEAUE_REASON parameter of the assembler user exit.
- Abend dump attribute: LE/VSE requests a system dump only if the assembler user exit sets CEEAUE_DUMP to ON.

RETCODE

Specifies that the enclave terminates with a normal return code and reason code.

However, the CEEBXITA assembler user exit can modify this behavior as follows:

- If the assembler user exit does not set the CEEAUE_ABND flag to ON during enclave termination, LE/VSE returns to its caller with a return code and a reason code.
- If the assembler user exit sets the CEEAUE_ABND flag to ON during enclave termination, LE/VSE terminates the enclave with an abend. In the batch environment, LE/VSE produces the run-time message CEE3322C, and issues an operating system request to terminate the enclave. In the CICS environment, LE/VSE issues an EXEC CICS ABEND.

LE/VSE sets the abend and reason code for the abend to equal the values of assembler-user-exit parameters, as follows:

- Abend code: Value of the CEEAUE_RETURN parameter of the assembler user exit. If the assembler user exit does not modify the CEEAUE_RETURN value, LE/VSE sets an abend code that maps to the severity of the condition and to the user return code.
- Reason code: Value of the CEEAUE_REASON parameter of the assembler user exit.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- COBOL consideration—For compatibility with pre-LE/VSE- conforming COBOL, ABEND is the recommended setting for COBOL customers.
- PL/I consideration–For compatibility with DOS PL/I, ABEND is the recommended setting for PL/I customers.
- DB2 and DL/I Considerations – ABEND is the recommended setting for SQL and DL/I users. For SQL, for example, this ensures that error conditions are mirrored back to SQL to enable SQL/DS ROLLBACK. See also Chapter 23 "Running Applications with SQL/DS" of the *LE/VSE Programming Guide*.

For More Information

- For information about return code calculation, CEEAUA_RETURN, CEEAUE_ABND, and CEEBXITA assembler user exit processing, see *LE/VSE Programming Guide*.
- For a list of abend code values and reason code values, see *LE/VSE Debugging Guide and Run-Time Messages*.

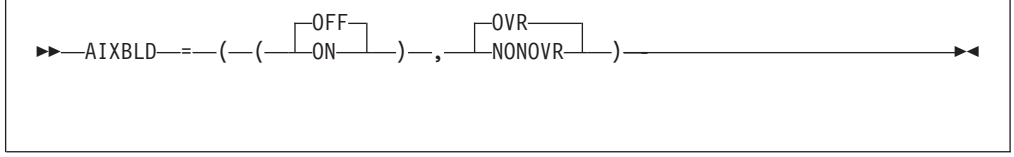
AIXBLD (COBOL Only)

AIXBLD invokes the access method services (AMS) for VSAM key-sequenced (KSDS) and relative-record data sets (RRDS) to complete the file and index definition procedures for COBOL routines.

AIXBLD conforms to the ANSI 1985 COBOL standard.

IBM-Supplied Default: AIXBLD=((OFF),OVR)

Syntax



OFF
Does not invoke the access method services for VSAM key-sequenced and relative-record datasets.

ON
Invokes the access method services for VSAM key-sequenced and relative-record datasets.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- CICS consideration—AIXBLD is ignored under CICS.
- VSE consideration—Access method services messages are directed to the MSGFILE *filename* or, if the file identified by *filename* is unavailable, to SYSLST.

Performance Considerations

Running your program under AIXBLD requires more storage, which can degrade performance. Therefore, use AIXBLD only during application development to build alternate indexes. Use AIXBLD=((OFF),...) when you have already defined your VSAM data sets.

For More Information

- For more information about AIXBLD, see *LE/VSE Programming Guide*.
- For more information about the MSGFILE run-time option, see "MSGFILE" on page 91.

ALL31

ALL31 specifies whether an application can run entirely in AMODE 31 or whether the application has one or more AMODE 24 routines.

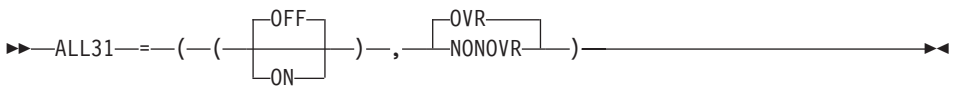
This option does not implicitly alter storage, in particular storage managed by the STACK and HEAP run-time options. However, you must be aware of your application's requirements for stack and heap storage, because such storage can potentially be allocated above the line while running in AMODE 24.

ALL31 should have the same setting for all enclaves in the process, because LE/VSE does not support the invocation of a nested enclave requiring ALL31(OFF) from an enclave running with ALL31(ON).

IBM-Supplied Default for CICS: ALL31=((ON),OVR)

IBM-Supplied Default for Batch: ALL31=((OFF),OVR)

Syntax



OFF
Indicates that one or more routines of an LE/VSE application are AMODE 24.

With ALL31(OFF) specified:

- AMODE switching across calls to LE/VSE common run-time routines is performed. For example, AMODE switching is performed on calls to LE/VSE callable services.
- In COBOL, EXTERNAL data is allocated in storage below the 16MB line.

If you use the default setting ALL31=((OFF),...), you must also use the BELOW suboption of the STACK option. AMODE 24 routines usually require stack storage below the 16MB line.

ON
Indicates that no user routines of an LE/VSE application are AMODE 24.

With ALL31(ON) specified:

- AMODE switching across calls to LE/VSE common run-time routines is minimized. For example, no AMODE switching is performed on calls to LE/VSE callable services.
- In COBOL, EXTERNAL data is allocated in unrestricted storage.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- COBOL consideration—When you link-edit a COBOL program compiled with the NORENT compiler option, the default addressing mode of the link-edited

phase is AMODE(ANY). This might result in your program being invoked in 24-bit addressing mode. In order to specify ALL31(ON), your program must be invoked in 31-bit addressing mode. Therefore, you should link-edit your application as AMODE(31). You can use the MODE linkage editor control statement to override the default addressing mode.

Performance Consideration

If your application consists entirely of AMODE (31) routines, it might run faster with ALL31(ON) than with ALL31(OFF) because mode switching code is not required.

Automatic AMODE detection is available under CICS using EXEC CICS calls to other LE-enabled applications. However, if an installation uses dynamic calls from an AMODE31 to an AMODE24 program, they must still use an installation default of ALL31(OFF) or use a specific override using CEEUOPT or exits. AMODE24 autodetection will not work for dynamically-called programs.

For More Information

For more information about the STACK run-time option, see "STACK" on page 101.

ANYHEAP

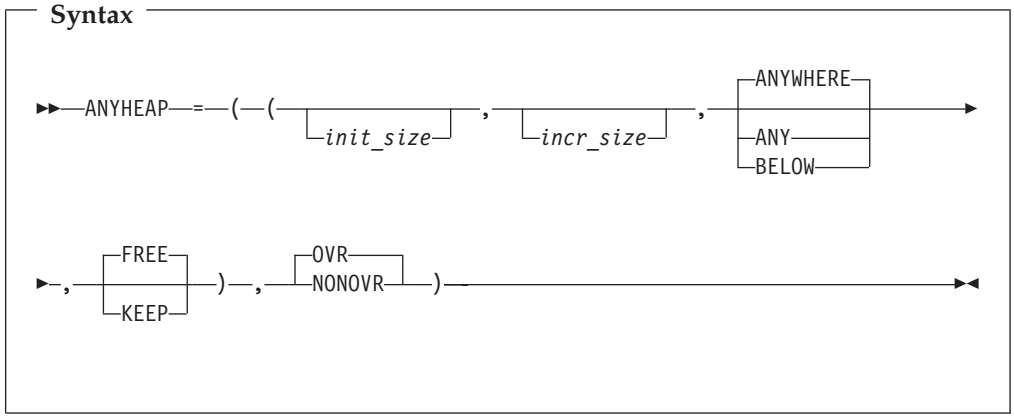
ANYHEAP controls the allocation of library heap storage that is not restricted to a location below the 16MB line.

The ANYHEAP option is always in effect. If you do not specify ANYHEAP or if you specify ANYHEAP(0), LE/VSE allocates the IBM-supplied default value of 16K when a call is made to obtain heap storage.

IBM-Supplied Default for CICS: ANYHEAP=((4K,4080,ANYWHERE,FREE),OVR)

IBM-Supplied Default for Batch: ANYHEAP=((16K,8K,ANYWHERE,FREE),OVR)

Syntax



init_size
Determines the minimum initial size of the anywhere heap storage. This value can be specified as *n*, *n*K, or *n*M bytes of storage. The actual amount of allocated storage is rounded up to the nearest multiple of 8 bytes.

incr_size
Determines the minimum size of any subsequent increment to the anywhere heap area, and is specified in *n*, *n*K, or *n*M bytes of storage. This value is rounded up to the nearest multiple of 8 bytes.

ANYWHERE | ANY
Specifies that heap storage can be allocated anywhere in storage. On systems that support bimodal addressing, storage can be allocated either above or below the 16MB line. If there is no storage available above the line, storage is acquired below the line.

BELOW
Specifies that heap storage must be allocated below the 16MB line in storage that is accessible to 24-bit addressing.

FREE
Specifies that storage allocated to ANYHEAP increments is released when the last of the storage is freed.

KEEP
Specifies that storage allocated to ANYHEAP increments is **not** released when the last of the storage is freed.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- CICS consideration—Both the initial size and the increment size are rounded up to the nearest multiple of 8 bytes. The minimum is 4K for initial size, and 4080 bytes for increment size.

  Under CICS/VSE 2.3, if ANYHEAP(,,BELOW) is in effect, the maximum initial and increment size for ANYHEAP is 65,504 bytes. If ANYHEAP(,,ANYWHERE) is in effect, the maximum initial and increment size for ANYHEAP is 1 gigabyte (1024M).
- CEEUOPT consideration—If you specify the ANYHEAP run-time option in CEEUOPT, the following default values are used for omitted suboptions:
  init_size
  32K
  incr_size
  16K

Performance Considerations

The ANYHEAP option improves performance when you specify values that minimize the number of times the operating system allocates storage. The RPTSTG run-time option generates a report of the storage the application uses while running; you can use this report to help determine what values to specify.

For More Information

- For more information about LE/VSE heap storage, see *LE/VSE Programming Guide*.
- For more information about the RPTSTG run-time option, see "RPTSTG" on page 97.
- For more information about using the storage report generated by the RPTSTG run-time option to tune your application, see *LE/VSE Programming Guide*.
- For more information about CEEUOPT, see *LE/VSE Programming Guide*.

BELOWHEAP

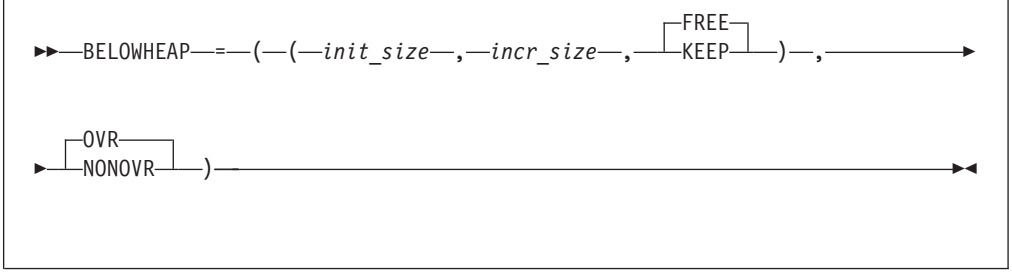
BELOWHEAP controls the allocation of library heap storage that must be located below the 16MB line. The heap controlled by BELOWHEAP is intended for items such as control blocks used for I/O.

The BELOWHEAP option is always in effect. If you do not specify BELOWHEAP or if you specify BELOWHEAP(0), the IBM-supplied default value of 8K is allocated when a call is made to obtain heap storage.

IBM-Supplied Default for CICS: BELOWHEAP=((4K,4080,FREE),OVR)

IBM-Supplied Default for Batch: BELOWHEAP=((8K,4K,FREE),OVR)

Syntax



init_size
Determines the minimum initial size of the below heap storage. This value can be specified as *n*, *n*K, or *n*M bytes of storage. The actual amount of allocated storage is rounded up to the nearest multiple of 8 bytes.

incr_size
Determines the minimum size of any subsequent increment to the area below the 16MB line, and is specified in *n*, *n*K, or *n*M bytes of storage. This value is rounded up to the nearest multiple of 8 bytes.

FREE
Specifies that storage allocated to BELOWHEAP increments is released when the last of the storage is freed.

KEEP
Specifies that storage allocated to BELOWHEAP increments is **not** released when the last of the storage is freed.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- CICS considerations—Both the initial size and the increment size are rounded to the nearest multiple of 8 bytes. The minimum is 4K for initial size, and 4080 bytes for increment size. The maximum initial and increment size for BELOWHEAP under CICS/VSE 2.3 is 65,504 bytes.
- CEEUOPT consideration—If you specify the BELOWHEAP run-time option in CEEUOPT, the following default values are used for omitted suboptions:
  init_size
  32K

incr_size
16K

Performance Considerations

BELOWHEAP improves performance when you specify values that minimize the number of times that the operating system allocates storage. The RPTSTG run-time option generates a report of storage your application uses while running. You can use this report to help determine what values to specify.

For More Information

- For more information about LE/VSE heap storage, see *LE/VSE Programming Guide*.
- For more information about the RPTSTG run-time option, see "RPTSTG" on page 97.
- For more information about tuning your application, see *LE/VSE Programming Guide*.
- For more information about CEEUOPT, see *LE/VSE Programming Guide*.

CBLOPTS (COBOL Only)

CBLOPTS specifies the format of the parameter string on application invocation when the main routine is COBOL. CBLOPTS determines whether run-time options or program arguments appear first in the parameter string.

You can specify this option only in CEEUOPT or CEEDOPT at initialization.

When you specify the ON suboption of CBLOPTS in CEEUOPT or CEEDOPT, the run-time arguments and program arguments specified in the JCL are honored in the following order:

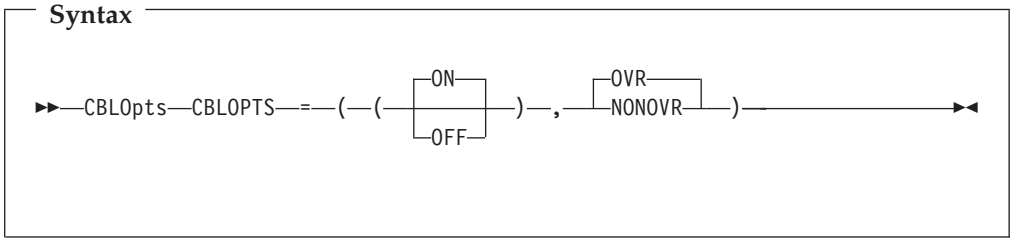
```
program arguments/run-time options
```

This order is the reverse of that normally honored by LE/VSE.

CBLOPTS(ON) allows the existing COBOL format of the invocation character string to continue working (user parameters followed by run-time options). CBLOPTS(ON) is valid only for applications whose main program is COBOL.

IBM-Supplied Default: CBLOPTS=((ON),OVR)

Syntax



ON
Specifies that program arguments appear first in the parameter string.

OFF
Specifies that run-time options appear first in the parameter string.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

For More Information
For more information about CEEUOPT, see *LE/VSE Programming Guide*.

CBLPSHPOP (COBOL Only)

CBLPSHPOP controls whether CICS PUSH HANDLE and CICS POP HANDLE commands are issued when a COBOL (VS COBOL II or COBOL/VSE) subroutine is called.

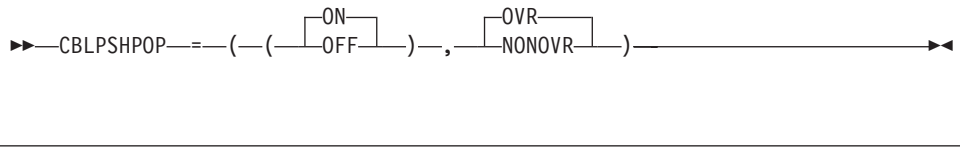
Specify CBLPSHPOP=((ON),...) to avoid compatibility problems when calling COBOL/VSE or VS COBOL II subroutines that contain CICS CONDITION, AID, or ABEND condition handling commands.

You can set the CBLPSHPOP run-time option on a transaction by transaction basis using CEEUOPT.

IBM-Supplied Default for CICS: CBLPSHPOP=((ON),OVR)

IBM-Supplied Default for Batch: CBLPSHPOP=((OFF),OVR)

Syntax



ON
Automatically issues the following when a COBOL subroutine is called:
- An EXEC CICS PUSH HANDLE command as part of the routine initialization
- An EXEC CICS POP HANDLE command as part of the routine termination

OFF
Does not issue CICS PUSH HANDLE and CICS POP HANDLE commands on a call to a COBOL subroutine.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Performance Consideration
- If your application calls COBOL subroutines under CICS, performance is better with CBLPSHPOP(OFF) than with CBLPSHPOP(ON).

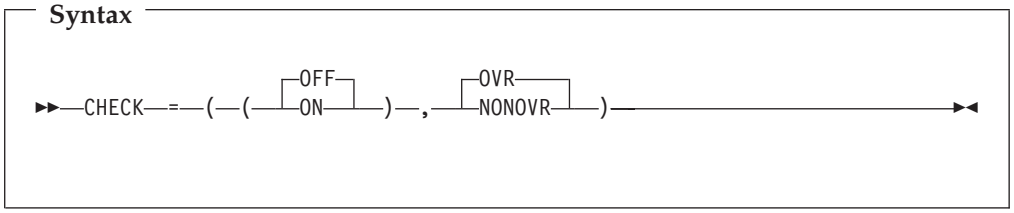
For More Information
For more information about CEEUOPT, see *LE/VSE Programming Guide*.

CHECK (COBOL Only)

CHECK flags checking errors within an application. In COBOL, index, subscript, and reference modification ranges are checking errors. COBOL is the only language that uses the CHECK option.

IBM-Supplied Default: CHECK=((OFF),OVR)

Syntax



OFF
Specifies that run-time checking is not performed.

ON
Specifies that run-time checking is performed.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Note

CHECK=((ON),...) has no effect if NOSSRANGE was in effect at compile time.

Performance Consideration

1. Please be aware that CHECK(ON) is required to ensure that the COBOL Compile option SSRANGE takes effect. This may be required for debugging purposes and would, for example, enable storage boundary checking.
2. If your COBOL program was compiled with SSRANGE, and you are not testing or debugging an application, performance improves when you specify CHECK(OFF).

COUNTRY

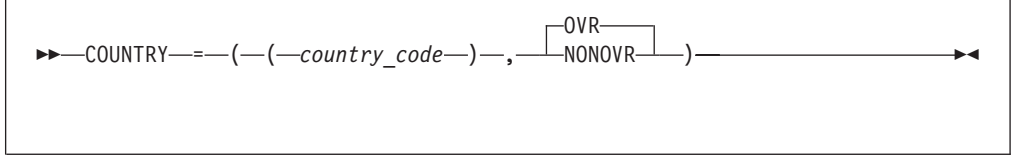
COUNTRY sets the country code, which affects the date and time formats, the currency symbol, the decimal separator, and the thousands separator, based on a specified country. COUNTRY does not change the default settings for the language currency symbol, decimal point, thousands separator, and date and time picture strings set by CEESETL or `setlocale()`. COUNTRY affects only the LE/VSE NLS services, not the LE/VSE locale callable services.

You can set the country value using the run-time option COUNTRY or the callable service CEE5CTY.

The COUNTRY setting affects the format of the date and time in the reports generated by the RPTOPTS and RPTSTG run-time options.

IBM-Supplied Default: COUNTRY=((US),OVR)

Syntax



country_code
A 2-character code that indicates to LE/VSE the country on which to base the default settings.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- If you specify a *country_code* that is not available on your system, LE/VSE accepts the value, issues informational message CEE3616I, and uses a default generic country code. This is not the same as the installation-supplied default US country code. For more information about the settings of this default country code, see Appendix A in *LE/VSE Programming Reference*.

  CEEUOPT and CEEDOPT permit the specification of an unavailable country code, but give a return code of 4 and a warning message.
- C consideration—LE/VSE provides locales used in C to establish default formats for the locale-sensitive functions and locale callable services, such as date and time formatting, sorting, and currency symbols. To change the locale, you can use the `setlocale()` library function or the CEESETL callable service.

  The settings of CEESETL or `setlocale()` do not affect the setting of the COUNTRY run-time option. COUNTRY affects only LE/VSE NLS and date and time services. `setlocale()` and CEESETL affect only C locale-sensitive functions and LE/VSE locale callable services.

  To ensure that all settings are correct for your country, use COUNTRY and either CEESETL or `setlocale()`.

For More Information

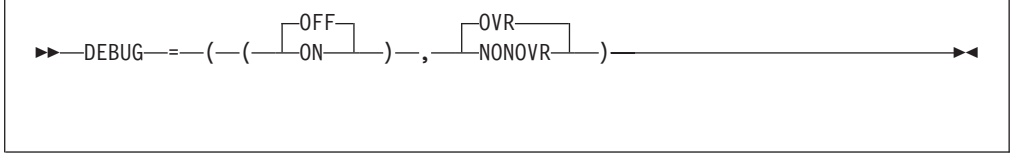
- For a list of countries and their codes, see "Appendix G. LE/VSE National Language Support Country Codes" on page 173.
- For more information about the CEE5CTY callable service, see *LE/VSE Programming Reference*.
- For more information about the RPTOPTS and RPTSTG run-time options, see "RPTOPTS" on page 95 and "RPTSTG" on page 97.
- For more information about the CEESETL callable service, see *LE/VSE Programming Reference*.
- For more information on `setlocale()`, see *LE/VSE C Run-Time Programming Guide*.

DEBUG (COBOL Only)

DEBUG activates the COBOL batch debugging features specified by the USE FOR DEBUGGING declarative.

IBM-Supplied Default: DEBUG=((OFF),OVR)

Syntax



OFF
Suppresses the COBOL batch debugging features.

ON
Activates the COBOL batch debugging features.

You must have the WITH DEBUGGING MODE clause in the environment division of your application in order to compile the debugging sections.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Performance Consideration

Because DEBUG(ON) gives worse run-time performance than DEBUG(OFF), you should use it only during application development or debugging.

For More Information

For more information on the USE FOR DEBUGGING declarative, see *LE/VSE Programming Guide*.

DEPTHCONDLMT

DEPTHCONDLMT specifies the extent to which conditions can be nested. Figure 12 illustrates the effect of DEPTHCONDLMT(3) on condition handling. The initial condition and two nested conditions are handled in this example. The third nested condition is not handled.

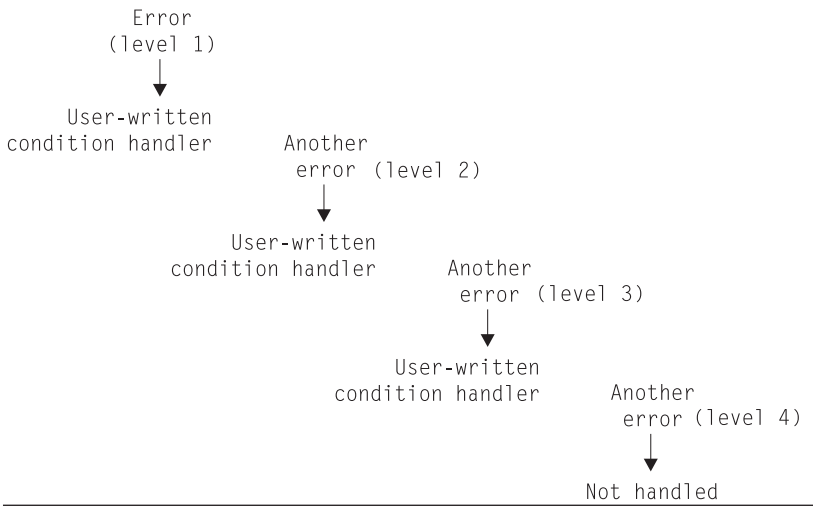
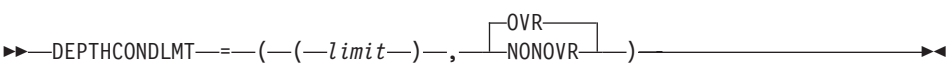


Figure 12. Effect of DEPTHCONDLMT(3) on Condition Handling

IBM-Supplied Default: DEPTHCONDLMT=((10),OVR)

Syntax



limit

An integer of 0 or greater value. It is the depth of condition handling allowed. An unlimited depth of condition handling is allowed if you specify 0.

A value of 1 specifies handling of the initial condition, but does not allow handling of nested conditions that occur while handling a condition. With a value of 5, for example, the initial condition and four nested conditions are processed, but there can be no further nesting of conditions.

If the number of nested conditions exceeds the limit, the application terminates with abend 4091 and reason code 21 (X'15').

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

Usage Notes

PL/I consideration—DEPTHCONDLMT(0) provides compatibility with previous releases of the DOS PL/I Optimizing Compiler.

For More Information

For more information on nested conditions, see *LE/VSE Programming Guide*.

ENVAR (C Only)

ENVAR sets the initial values for the environment variables specified in *string*. With ENVAR, you can pass into the application switches or tagged information that can then be accessed using the C functions `getenv`, `setenv`, and `clearenv`.

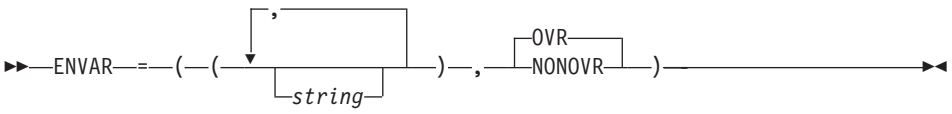
When the run-time options are merged, ENVAR strings are appended in the order encountered during the merge. Thus, the set of environment variables established by the end of run-time option processing reflects all the various sources where environment variables are specified (rather than just the one source with the highest precedence). However, if a setting for the same environment variable is specified in more than one source, the last setting is used.

Environment variables in effect at the time of the `system` function are copied to the new environment. The copied environment variables are treated the same as those found in the ENVAR run-time option on the command level, with respect to the merge of the run-time options from their various sources.

When you have specified the RPTOPTS run-time option, you receive a list of the merged ENVAR run-time options. The output for the ENVAR run-time options contains a separate entry for each source where ENVAR was specified with the environment variables from that source.

IBM-Supplied Default: ENVAR(("),OVR)

Syntax



string
Is of the form *name=value*, where *name* and *value* are sequences of characters that do not contain null bytes or equal signs. The string *name* is an environment variable, and *value* is its value.

Blanks are significant in both the *name*= and the *value* characters.

You can enclose the *string* in either single or double quotation marks to distinguish it from other strings. *string* cannot contain DBCS characters. It can have a maximum of 250 characters.

You can specify multiple environment variables, separating the *name=value* pairs with commas. Quotation marks are required when specifying multiple variables.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

C consideration—An application can access the environment variables using C function `getenv`.

HLLs can access the environment variables through standard C functions at enclave initialization and throughout the application's run. Access remains until the HLL returns from enclave termination.

For More Information

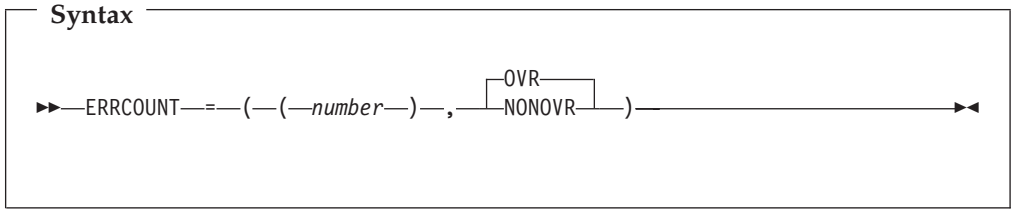
- For more information about the RPTOPTS run-time option, see "RPTOPTS" on page 95.
- For more information about `getenv`, `setenv`, and `clearenv`, see *LE/VSE C Run-Time Programming Guide*.

ERRCOUNT

ERRCOUNT specifies how many conditions of severity 2, 3, and 4 can occur before the enclave terminates abnormally. After the number specified in ERRCOUNT is reached, no further LE/VSE condition management, including CEEHDLR management, is honored.

IBM-Supplied Default: ERRCOUNT=((20),OVR)

Syntax



number
The number of severity 2, 3, and 4 conditions that can occur while this enclave is running. If the number of conditions exceeds *number*, the enclave terminates abnormally.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- ERRCOUNT(0) means the number of conditions that can occur is unlimited. This setting can cause an infinite loop or a runaway task.
- COBOL consideration—LE/VSE counts severity 1 messages with the facility ID IGZ. When the limit is reached, additional severity 1 messages are suppressed.
- PL/I consideration—You should use ERRCOUNT(0) if you are using PL/I.

For More Information

- For more information about the CEEHDLR callable service, see *LE/VSE Programming Reference*.
- For more information about facility IDs, see *LE/VSE Programming Guide*.

HEAP

HEAP controls the allocation of the initial heap, controls allocation of additional heaps created with the CEECRHP callable service, and specifies how that storage is managed.

Heaps are storage areas where you allocate memory for user-controlled dynamically allocated variables such as:

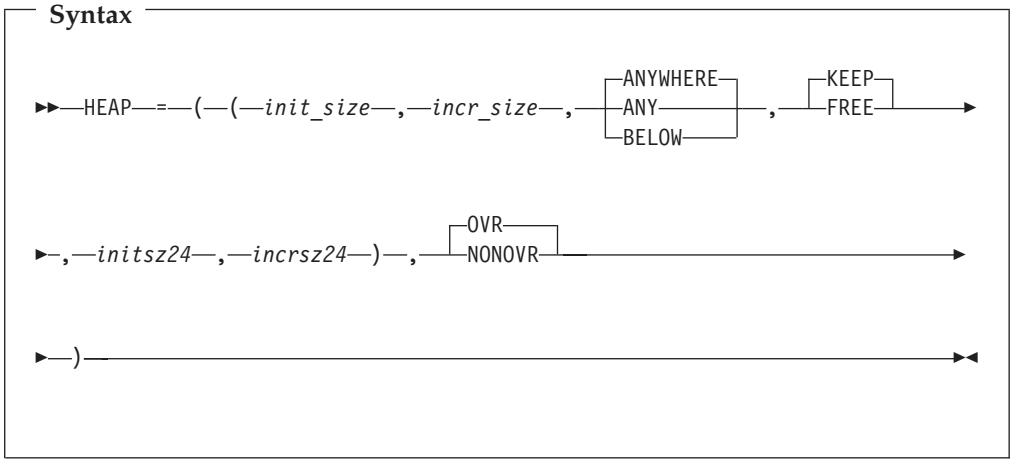
- C variables allocated as a result of the `malloc()`, `calloc()`, and `realloc()` functions
- COBOL WORKING-STORAGE data items
- PL/I variables with the storage class CONTROLLED, or the storage class BASED

LE/VSE does not allocate heap storage until the first call to obtain heap storage is made. You can obtain heap storage by using language constructs or by making a call to CEEGTST.

IBM-Supplied Default for CICS:
HEAP=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR)

IBM-Supplied Default for Batch:
HEAP=((32K,32K,ANYWHERE,KEEP,8K,4K),OVR)

Syntax



init_size
Determines the minimum initial allocation of heap storage. Specify this value as *n*, *n*K, or *n*M bytes of storage. The actual amount of allocated storage is rounded up to the nearest multiple of 8 bytes.

incr_size
Determines the minimum size of any subsequent increment to the heap storage. Specify this value as *n*, *n*K, or *n*M bytes of storage. The actual amount of allocated storage is rounded up to the nearest multiple of 8 bytes.

ANYWHERE | ANY
Specifies that you can allocate heap storage anywhere in storage. On systems that support bimodal addressing, you can allocate storage either above or below the 16MB line. If there is no available storage above the line, storage is acquired below the line.

BELOW
Specifies that you must allocate heap storage below the 16MB line in storage that is accessible to 24-bit addressing.

KEEP
Specifies that storage allocated to HEAP increments is not released when the last of the storage is freed.

FREE
Specifies that storage allocated to HEAP increments is released when the last of the storage is freed.

initsz24
Determines the minimum initial size of the heap storage that is obtained below the 16MB line for applications running with ALL31(OFF) when these applications specify ANYWHERE in the HEAP run-time option. Specify *initsz24* as *n*, *n*K, or *n*M number of bytes. The amount of storage is rounded up to the nearest multiple of 8 bytes.

initsz24 applies to all heaps that are not allocated strictly below the 16MB line.

incrsz24
Determines the minimum size of any subsequent increment to the heap area that is obtained below the 16MB line for applications running with ALL31(OFF) when these applications specify ANYWHERE in the HEAP run-time option. Specify *incrsz24* as *n*, *n*K, or *n*M number of bytes. The amount of storage is rounded up to the nearest multiple of 8 bytes.

incrsz24 applies to all heaps that are not allocated strictly below the 16MB line.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- Applications running in AMODE 24 that request heap storage get the storage below the 16MB line regardless of the setting of ANYWHERE | BELOW.
- COBOL consideration—You can use the HEAP option to provide some of the function provided by the VS COBOL II space management tuning table.
- PL/I consideration—For PL/I, the only case in which storage is allocated above the line is when all of the following conditions exist:
  - The user routine requesting the storage is running in 31-bit addressing mode.
  - HEAP(,,ANYWHERE) is in effect.
  - The main routine is AMODE 31.
- CICS consideration—Both the initial HEAP allocation and HEAP increments are rounded to the next higher multiple of 8 bytes. The minimum is 4K for initial size, and 4080 bytes for increment size.

  Under CICS/VSE 2.3, if HEAP(,,BELOW) is in effect, the maximum size of a heap segment is 65,504 bytes. If too large a value is specified, the application fails at the first attempt to allocate heap storage. If HEAP(,,ANYWHERE) is in effect, the maximum size of a heap segment is 1 gigabyte (1024M). These restrictions are subject to change from one release of CICS to another.
- CEEUOPT consideration—If you specify the HEAP run-time option in CEEUOPT, the following default values are used for omitted suboptions:
  init_size
  64K

incr_size
64K
initsz24
16K
incrsz24
16K

Performance Considerations

The RPTSTG run-time option generates a report of storage your application uses while running. To improve performance, use the information in this report as an aid in setting the initial and increment sizes for HEAP.

For More Information

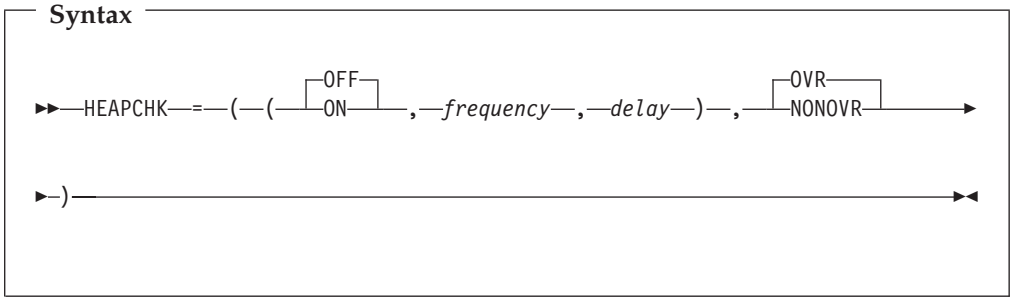
- For more information about LE/VSE heap storage, see *LE/VSE Programming Guide*
- For more information about the CEECRHP and CEEGTST callable services, see *LE/VSE Programming Reference*
- For more information about the RPTSTG run-time option, see "RPTSTG" on page 97.
- For more information about using the storage report generated by the RPTSTG run-time option to tune your application, see *LE/VSE Programming Guide*.

HEAPCHK

HEAPCHK provides a checking facility to verify that the heap storage has not been damaged.

IBM-Supplied Default: HEAPCHK((OFF,1,0)OVR)

Syntax



OFF
Specifies that no heap checking will be done.

ON
Specifies that heap checking will be activated and controlled by the *frequency* and *delay* parameters.

frequency
Determines the event frequency at which heap checking is to occur. This specifies that heap storage will be checked for damage on every *n*th call to an LE/VSE storage management service. Specify this value as *n*, *n*K, or *n*M.

delay
Determines the number of calls to LE/VSE storage management services that will be made before activating the heap checking mechanism. Specify this value as *n*, *n*K, or *n*M.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- When specifying values for *frequency* and *delay*, remember that storage management services are called by LE/VSE's internal routines in addition to your application calls.
- Certain language constructs will also call LE/VSE storage management services. For example, PL/I ALLOCATE and FREE statements for variables and aggregates that are not within a PL/I AREA, and the C `malloc()` and `free()` library functions.
- EXEC CICS GETMAIN and FREEMAIN do not use LE/VSE storage management services.

Performance Considerations

HEAPCHK is intended to be used in a test environment only!. Use HEAPCHK in production only when necessary, as it will use extra CPU resources and degrade performance.

For More Information

For more information about LE/VSE's storage management services, see the description of CEEGTST and CEEFRST in the *LE/VSE Programming Reference*.

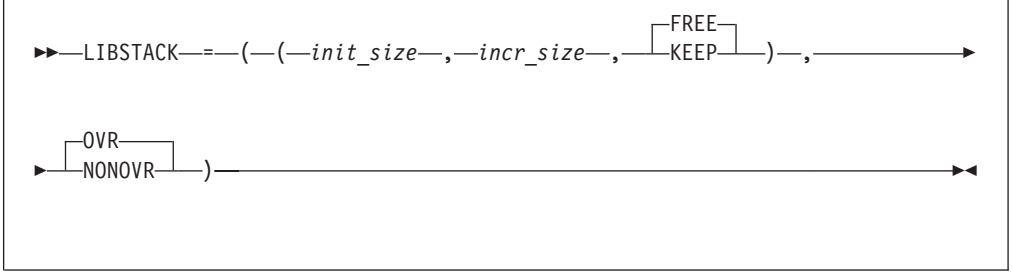
LIBSTACK

LIBSTACK controls the allocation of the thread's library stack storage. This stack is used by LE/VSE and HLL library routines that require save areas below the 16MB line.

IBM-Supplied Default for CICS: LIBSTACK=((4K,4080,FREE),OVR)

IBM-Supplied Default for Batch: LIBSTACK=((8K,4K,FREE),OVR)

Syntax



init_size
Determines the size of the initial library stack segment. The storage is contiguous.

Specify *init_size* as *n*, *n*K, or *n*M bytes of storage. *init_size* can be preceded by a minus sign. In the batch environment, if you specify a negative number, all available storage minus the amount specified is used for the initial stack segment.

In the batch environment, an *init_size* of 0 or −0 requests half of the largest block of contiguous storage below the 16MB line.

At initialization, LE/VSE allocates the storage rounded up to the nearest multiple of 8 bytes.

incr_size
Determines the minimum size of any subsequent increment to the library stack area. Specify this value as *n*, *n*K, or *n*M bytes of storage. The actual amount of allocated storage is the larger of 2 values— *incr_size* or the requested size—rounded up to the nearest multiple of 8 bytes.

If you do not specify *incr_size*, LE/VSE uses the IBM-supplied default setting of 4K. If *incr_size*=0, LE/VSE obtains only the amount of storage needed at the time of the request, rounded up to the nearest multiple of 8 bytes.

The requested size is the amount of storage a routine needs for a stack frame. For example, if the requested size is 9000 bytes, *incr_size* is specified as 8K and the initial stack segment is full, LE/VSE obtains a 9000-byte stack increment from the operating system to satisfy the request. If the requested size is smaller than 8K, LE/VSE obtains an 8K stack increment from the operating system.

FREE
Specifies that LE/VSE releases storage allocated to LIBSTACK increments when the last of the storage in the library stack is freed. The initial library stack segment is not released until the enclave terminates.

KEEP
Specifies that LE/VSE does not release storage allocated to LIBSTACK increments when the last of the storage is freed.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- CICS consideration—Both the initial and increment sizes are rounded up to the next multiple of 8 bytes. The minimum is 4K for initial size, and 4080 bytes for increment size.

  Under CICS, the maximum initial and increment size for LIBSTACK is 65,504 bytes.
- CEEUOPT consideration—If you specify the LIBSTACK run-time option in CEEUOPT, the following default values are used for omitted suboptions:
  init_size
  32K
  incr_size
  16K

Performance Considerations

The RPTSTG run-time option generates a report of storage your application uses while running. To improve performance, use the information in this report as an aid in setting the initial and increment sizes for LIBSTACK.

For More Information

- For more information about the RPTSTG run-time option, see “RPTSTG” on page 97.

- For more information about using the storage report generated by the RPTSTG run-time option to tune your application, see *LE/VSE Programming Guide*.

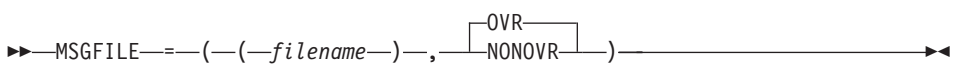
MSGFILE

MSGFILE specifies the *filename* of the file where all run-time diagnostics and reports generated by the RPTOPTS and RPTSTG run-time options are written. MSGFILE also specifies the *filename* for CEEMSG and CEEMOUT callable services.

IBM-Supplied Default for CICS: MSGFILE=((CESE),OVR)

IBM-Supplied Default for Batch: MSGFILE=((SYSLST),OVR)

Syntax



filename
The filename of the run-time diagnostics file.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- CICS considerations – The MSGFILE option defaults to the CESE transient data queue. Specification of a different transient data queue for MSGFILE is possible. However, it is the user's responsibility to ensure that this transient data queue is available in the CICS system. If an option other than CESE is specified for MSGFILE and this transient data queue becomes unusable or unavailable, LE/VSE will default to the CESE transient data queue. The CESE transient data queue must always be available either as TYPE=INDIRECT or TYPE=EXTRA in the CICS DCT definition.
  The MSGFILE destination name under CICS must not exceed 4 characters in length. Truncation will occur on the MSGFILE destination if the name used is greater than 4 characters in length. The supplied definition of CEEMSG in CEECDCT.A in PRD2.SCEEBASE should be used as an example for any other TYPE=SDSCI destinations being used as a Disk File destination for MSGFILE. Note that if a DISK file is being used as a final destination, you must remember to add 8 bytes to the BLKSIZE specified in your DCT definition. Any MSGFILE destination used must support a blksize of at least 175 bytes (inclusive of the 8 bytes required for LIOCS output files if DISK is used). The VSE system console is not a supported destination for MSGFILE either directly or indirectly.
- HLL compile-time options can affect whether your run-time output goes to MSGFILE *filename*.
- LE/VSE does not check the validity of the MSGFILE *filename*. An invalid *filename* generates an error condition on the first attempt to issue a message.
- C consideration—C `perror()` messages and output directed to `stderr` go to the MSGFILE destination.

- PL/I consideration—Run-time messages in PL/I routines are directed to the file specified by the LE/VSE MSGFILE run-time option, instead of to the PL/I SYSPRINT STREAM PRINT file.

  User-specified output is still directed to the PL/I SYSPRINT STREAM PRINT file. If you want LE/VSE to handle this output, specify MSGFILE(SYSPRINT). When you specify MSGFILE(SYSPRINT), all PL/I run-time messages and user-specified output are directed to SYSLST.

For More Information

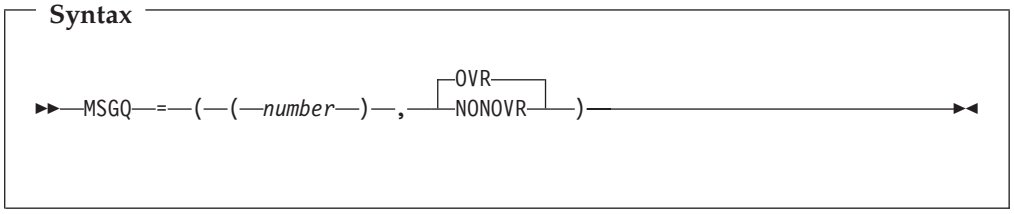
- For more information about the RPTOPTS and RPTSTG run-time options, see "RPTOPTS" on page 95 and "RPTSTG" on page 97.
- For more information about the CEEMSG and CEEMOUT callable services, see *LE/VSE Programming Reference*.
- For details on how HLL compiler options affect messages, see information on HLL I/O statements and message handling in *LE/VSE Programming Guide*.
- For more information about the CESE transient data queue, see *LE/VSE Programming Guide*.
- For more information about `perror()` and `stderr`, see C message output information in *LE/VSE Programming Guide*.

MSGQ

MSGQ specifies the number of ISI blocks that LE/VSE allocates on a per thread basis for use by the application. The ISI contains information that LE/VSE uses to identify and react to conditions, provide access to q_data tokens, and assign space for message inserts used with user-created messages. When an ISI is needed and one is not available, LE/VSE takes the least recently used ISI for reuse. CEECMI allocates storage for the ISI, if necessary.

IBM-Supplied Default: MSGQ=((15),OVR)

Syntax



number
An integer that specifies the number of ISIs to be maintained per thread within an enclave.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

For More Information

- For more information about the CEECMI callable service, see *LE/VSE Programming Reference*.
- For more information about the ISI, see *LE/VSE Programming Guide*.

NATLANG

NATLANG specifies the initial national language to be used for the run-time environment, including error messages, month names, and day-of-the week names. Message translations are provided for Japanese and (uppercase and mixed-case) U.S. English. NATLANG also determines how the message facility formats messages.

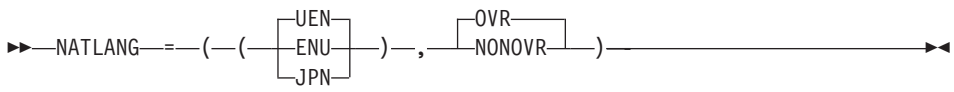
NATLANG affects only the LE/VSE NLS and date and time services, not the LE/VSE locale callable services.

You can set the national language by using the NATLANG run-time option or the SET option of the CEE5LNG callable service. LE/VSE maintains one current language at the enclave level. This current language remains in effect until one of the above changes it. For example, if you specify JPN in the NATLANG run-time option, but subsequently specify ENU using the CEE5LNG callable service, ENU becomes the current national language.

LE/VSE writes certain parts of storage and options reports and dump output only in mixed-case U.S. English, and certain abnormal termination messages only in uppercase U.S. English.

IBM-Supplied Default: NATLANG=((UEN),OVR)

Syntax



UEN
A 3-character ID specifying uppercase U.S. English.

Message text consists of SBCS (single-byte character set) characters and includes only uppercase letters.

ENU
A 3-character ID specifying mixed-case U.S. English.

Message text consists of SBCS characters and includes both uppercase and lowercase letters.

JPN
A 3-character ID specifying Japanese.

Message text can contain a mixture of SBCS and DBCS (double-byte character set) characters.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- If you specify a national language that is not available on your system, LE/VSE uses the IBM-supplied default UEN (uppercase U.S. English).

CEEUOPT and CEEDOPT can specify an unknown national language code, but give a return code of 4 and a warning message.

- C consideration—LE/VSE provides locales used in C to establish default formats for the locale-sensitive functions and locale callable services, such as date and time formatting, sorting, and currency symbols. To change the locale, you can use the `setlocale()` library function or the CEESETL callable service.

  The settings of CEESETL or `setlocale()` do not affect the setting of the NATLANG run-time option. NATLANG affects only LE/VSE NLS and date and time services. `setlocale()` and CEESETL affect only C locale-sensitive functions and LE/VSE locale callable services.

  To ensure that all settings are correct for your country, use NATLANG and either CEESETL or `setlocale()`.

For More Information

- For more information about the CEE5LNG and CEESETL callable services, see *LE/VSE Programming Reference*.
- For more information about `setlocale()`, see *LE/VSE C Run-Time Programming Guide*.

NOTEST

See "TEST" on page 109.

NOUSRHDLR

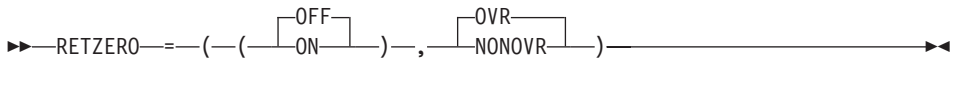
See "USRHDLR" on page 116.

RETZERO (COBOL Only)

RETZERO will ensure that, if the run unit does not abend or terminate abnormally, the user return code will be set to zero regardless of the contents of register 15 or the RETURN-CODE special register.

IBM-Supplied Default: RETZERO((OFF),OVR)

Syntax



OFF
Specifies that the user return code will be unchanged.

ON
Specifies that LE/VSE will set the user return code to zero before terminating the run unit.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

This option is intended for COBOL programs that call Assembler or other non-COBOL/VSE subroutines, where the subroutine does not clear register 15 before returning to the calling program. Under normal circumstances (with RETZERO(OFF)), the contents of register 15 will become the contents of the RETURN-CODE special register, and if the COBOL program does not subsequently change this, it will be used as the user return code for the enclave. As this is an unpredictable value (possibly a virtual storage address), it can cause errors in the batch job stream. With RETZERO(ON), the user return code will be forced to zero before the run unit ends.

RPTOPTS

RPTOPTS generates, after an application has run, a report of the run-time options in effect while the application was running. LE/VSE writes options reports only in mixed-case U.S. English.

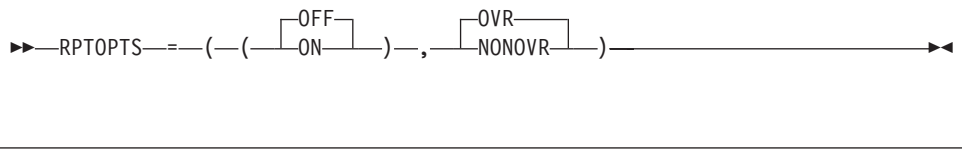
In the batch environment, LE/VSE directs the report to the *filename* specified in the MSGFILE run-time option. In the CICS environment, LE/VSE directs the report to the CESE transient data queue.

Figure 13 on page 96 shows the sample output when RPTOPTS is set to ON. RPTOPTS(ON) lists the declared run-time options in alphabetic order. The report lists the option names and shows where each option obtained its current setting. The report heading displayed at the top of the options report is set by CEE5RPH. The date and time formats are affected by the country code set by the COUNTRY run-time option or the CEE5CTY callable service.

The LAST WHERE SET column in the report shows the last place where the options were referenced, even if no suboptions or subsets of the options were changed. “Default setting” in the report indicates that you cannot specify the option in CEEDOPT or CEEUOPT. “Programmer default” includes any options specified with C `#pragma runopts`, PL/I PLIXOPT, and CEEUOPT.

IBM-Supplied Default: RPTOPTS=((OFF),OVR)

Syntax



OFF
Does not generate a report of the run-time options in effect while the application was running.

ON
Generates a report of the run-time options in effect while the application was running.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

In most instances, RPTOPTS does not generate the options report if an application terminates abnormally.

Performance Considerations

This option increases the time it takes for the application to run. Therefore, use it only as an aid to application development.

```
Options Report for Enclave CBLDATE 07/26/01 3:52:13 PM
Language Environment for VSE/ESA V1 R4.2

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

Figure 13. Options Report Produced by LE/VSE Run-Time Option RPTOPTS(ON)

For More Information

- For more information about the MSGFILE and COUNTRY run-time options, see "MSGFILE" on page 91 and "COUNTRY" on page 81.
- For more information about the CEE5RPH and CEE5CTY callable services, see *LE/VSE Programming Reference*.

RPTSTG

RPTSTG generates, after an application has run, a report of the storage the application used. In the batch environment, the report is directed to the *filename* specified in the MSGFILE run-time option. In the CICS environment, the report is directed to the CESE transient data queue.

Figure 14 on page 99 shows a sample report created with the RPTSTG option set to ON.

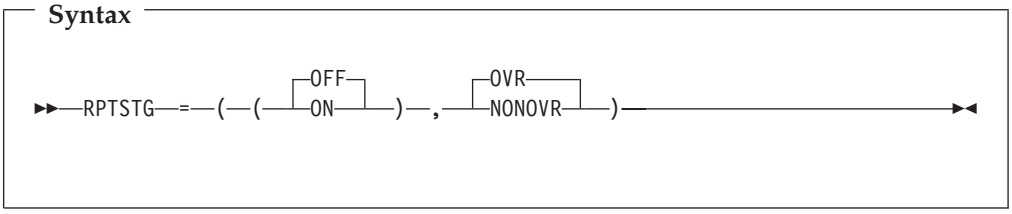
The storage report heading is set by CEE5RPH. The date and time formats, in the RPTSTG generated reports, are affected by the country code set by the COUNTRY run-time option or the CEE5CTY callable service.

You can use the storage report information to adjust the ANYHEAP, BELOWHEAP, HEAP, LIBSTACK, and STACK run-time options.

LE/VSE writes storage reports only in mixed-case U.S. English.

IBM-Supplied Default: RPTSTG=((OFF),OVR)

Syntax



OFF
Does not generate a report of the storage used while the application was running.

ON
Generates a report of the storage used while the application was running.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- In most instances, RPTSTG does not generate a storage report if your application terminates abnormally.
- The phrases “Number of segments allocated” and “Number of segments freed” represent the following:
  - In the batch environment, the number of GETVIS and FREEVIS requests, respectively.
  - Under CICS, the number of EXEC CICS GETMAIN and EXEC CICS FREEMAIN requests, respectively.
- The statistics for initial and incremental allocations of storage types that have a corresponding run-time option differ from the run-time option settings when (1) their values have been rounded up by the implementation, or (2) when allocations larger than the amounts specified were required during execution. All of the following are rounded up to an integral number of double-words:

  Initial STACK allocations

Initial allocations of all types of heap

Incremental allocations of all types of stack and heap

Performance Considerations

This option increases the time it takes for an application to run. Therefore, use it only as an aid to application development.

The storage report generated by RPTSTG(ON) shows the number of system-level get storage calls that were required while the application was running. To improve performance, use the information in the storage report generated by the RPTSTG option as an aid in setting the initial and increment sizes for STACK and HEAP. This reduces the number of times that the LE/VSE storage manager makes requests to acquire storage. For example, you can use the storage report to set appropriate values in the HEAP and STACK *init_size* and *incr_size* fields for allocating storage.

```
Storage Report for Enclave ABC 05/13/96 11:17:50 AM

    STACK statistics:
      Initial size:                                      131072
      Increment size:                                    131072
      Total stack storage used (sugg. initial size):       9504
      Number of segments allocated:                           1
      Number of segments freed:                               0
    LIBSTACK statistics:
      Initial size:                                       8192
      Increment size:                                     4096
      Total stack storage used (sugg. initial size):      1152
      Number of segments allocated:                           1
      Number of segments freed:                               0
    HEAP statistics:
      Initial size:                                      32768
      Increment size:                                    32768
      Total heap storage used (sugg. initial size):          0
      Successful Get Heap requests:                          0
      Successful Free Heap requests:                         0
      Number of segments allocated:                          0
      Number of segments freed:                              0
    ANYHEAP statistics:
      Initial size:                                      16384
      Increment size:                                     8192
      Total heap storage used (sugg. initial size):       7144
      Successful Get Heap requests:                         61
      Successful Free Heap requests:                         0
      Number of segments allocated:                          1
      Number of segments freed:                              0
    BELOWHEAP statistics:
      Initial size:                                       8192
      Increment size:                                     4096
      Total heap storage used (sugg. initial size):        936
      Successful Get Heap requests:                          5
      Successful Free Heap requests:                         4
      Number of segments allocated:                          1
      Number of segments freed:                              0
    Additional Heap statistics:
      Successful Create Heap requests:                       0
      Successful Discard Heap requests:                      0
      Total heap storage used:                               0
      Successful Get Heap requests:                          0
      Successful Free Heap requests:                         0
      Number of segments allocated:                          0
      Number of segments freed:                              0
End of Storage Report
```

Figure 14. Storage Report Produced by LE/VSE Run-Time Option RPTSTG(ON)

For More Information

- For more information about the MSGFILE and COUNTRY run-time options, see "MSGFILE" on page 91 and "COUNTRY" on page 81.
- For more information about the CEE5RPH and CEE5CTY callable services, see *LE/VSE Programming Reference*.
- For more information about the BELOWHEAP and HEAP run-time options, see "BELOWHEAP" on page 77 and "HEAP" on page 86.
- For more information about the LIBSTACK and STACK run-time options, see "LIBSTACK" on page 89 and "STACK" on page 101.

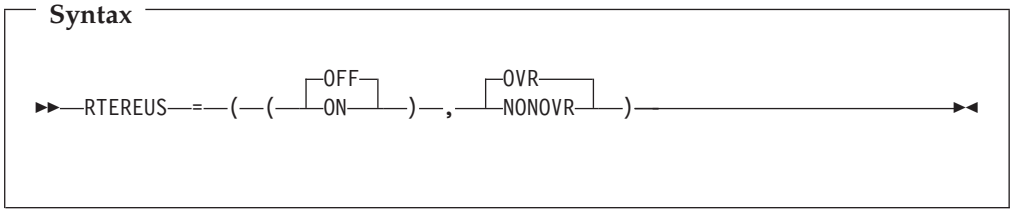
- For more information about using the storage report generated by the RPTSTG run-time option to tune your application, see *LE/VSE Programming Guide*.

RTEREUS (COBOL Only)

RTEREUS implicitly initializes the run-time environment to be reusable when the main program for the thread is a COBOL program. This option is valid only when used with CEEDOPT or CEEUOPT.

IBM-Supplied Default: RTEREUS=((OFF),OVR)

Syntax



NORTEREUS
Does not initialize the run-time environment to be reusable when the first COBOL routine is invoked.

RTEREUS
Initializes the run-time environment to be reusable when the first COBOL routine is invoked.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- Avoid using RTEREUS=((ON),...) as an installation default, because doing so can cause problems for other HLLs such as C and PL/I.
- CICS consideration—RTEREUS is ignored under CICS.
- The IGZERREO CSECT affects the handling of program checks in the non-Language Environment conforming driver that repeatedly invokes COBOL programs.

Performance Considerations

You must change STOP RUN statements to GOBACK statements in order to gain the benefits of RTEREUS. STOP RUN terminates the reusable environment. If you specify RTEREUS, LE/VSE recreates the reusable environment on the next invocation of COBOL. Doing this repeatedly degrades performance, because a reusable environment takes longer to create than does a normal environment.

Notes:

1. The IGZERREO CSECT affects the performance of running with RTEREUS.
2. LE/VSE also offers preinitialization support in addition to RTEREUS.

For More Information

- For more information about CEEUOPT, see *LE/VSE Programming Guide*.
- For more information about preinitialization, see *LE/VSE Programming Guide*.
- For more information about IGZERREO, see "Customizing the COBOL Reusable Environment" on page 47.

STACK

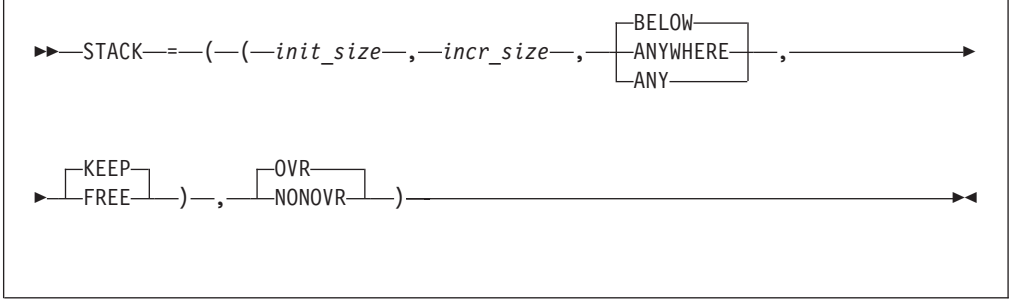
STACK controls the allocation of the thread's stack storage. Typical items residing in the stack are C or PL/I automatic variables, and temporary work areas for COBOL library routines.

Storage required for the common anchor area (CAA) and other control blocks is allocated separately from, and prior to, the allocation of the initial stack segment and the initial heap.

IBM-Supplied Default for CICS: STACK=((4K,4080,ANYWHERE,KEEP),OVR)

IBM-Supplied Default for Batch: STACK=((128K,128K,BELOW,KEEP),OVR)

Syntax



init_size
Determines the size of the initial stack segment. The storage is contiguous. You specify the *init_size* value as *n*, *n*K, or *n*M bytes of storage. The actual amount of allocated storage is rounded up to the nearest multiple of 8 bytes.

init_size can be preceded by a minus sign. In the batch environment, if you specify a negative number LE/VSE uses all available storage minus the amount specified for the initial stack segment.

A size of "0" or "–0" requests half of the largest block of contiguous storage available below the 16MB line. Behavior under CICS is described in the Usage Notes for this run-time option.

incr_size
Determines the minimum size of any subsequent increment to the stack area. You can specify this value as *n*, *n*K, or *n*M bytes of storage. The actual amount of allocated storage is the larger of two values—*incr_size* or the requested size—rounded up to the nearest multiple of 8 bytes.

If you specify *incr_size* as 0, only the amount of the storage needed at the time of the request, rounded up to the nearest 4K, is obtained.

The requested size is the amount of storage a routine needs for a stack frame. For example, if the requested size is 9000 bytes, *incr_size* is specified as 8K, and the initial stack segment is full, LE/VSE obtains a 9000-byte stack increment from the operating system to satisfy the request. If the requested size is smaller than 8K, LE/VSE obtains an 8K stack increment from the operating system.

BELOW
Specifies that the stack storage must be allocated below the 16MB line, in storage that is accessible to 24-bit addressing.

ANYWHERE | ANY
Specifies that stack storage can be allocated anywhere in storage. On systems that support bimodal addressing, storage can be allocated either above or below the 16MB line. If there is no storage available above the line, LE/VSE acquires storage below the line.

KEEP
Specifies that storage allocated to STACK increments is not released when the last of the storage in the stack increment is freed.

FREE
Specifies that storage allocated to STACK increments is released when the last of the storage in the stack is freed. The initial stack segment is never released until the enclave terminates.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- Applications running with ALL31(OFF) must specify the BELOW suboption of STACK to ensure that stack storage is addressable by the application.
- CICS consideration—the maximum initial and increment size for STACK below 16MB is 65,504 bytes. The maximum initial and increment size for STACK above 16MB is 1 gigabyte (1024M). This restriction is subject to change from one release of CICS to another.

  Both the initial size and the increment size are rounded up to the nearest multiple of 8 bytes. The initial size minimum is 4K, the increment size minimum is 4080 bytes.

  If you do not specify STACK, LE/VSE assumes the default value of 4K. Under CICS, STACK(0), STACK (–0), and STACK (–n) are all interpreted as STACK(4K).
- COBOL consideration—When you link-edit a COBOL program compiled with the NORENT compiler option, the default addressing mode of the link-edited phase is AMODE(ANY). This might result in your program being invoked in 24-bit addressing mode. In order to specify STACK(ANY), your program must be invoked in 31-bit addressing mode. Therefore, you should link-edit your application as AMODE(31). You can use the MODE linkage editor control statement to override the default addressing mode.
- PL/I consideration—PL/I automatic storage above the 16MB line is supported under control of the LE/VSE STACK option. When the LE/VSE stack is above, PL/I temporaries (dummy arguments) and parameter lists (for reentrant/recursive blocks) also reside above.

  The stack frame size for an individual block is constrained to 16MB. Stack frame extensions are also constrained to 16MB. Therefore, the size of an automatic aggregate, temporary variable, or dummy argument cannot exceed 16MB. Violation of this constraint might have unpredictable results.
- CEEUOPT consideration—If you specify the STACK run-time option in CEEUOPT, the following default values are used for omitted suboptions:
  init_size
  512K
  incr_size
  512K

Performance Considerations

To improve performance, use the information in the storage report generated by the RPTSTG run-time option as an aid in setting the initial and increment sizes for STACK.

For More Information

- For more information about the ALL31 run-time option, see "ALL31" on page 74.
- For more information about the RPTSTG run-time option, see "RPTSTG" on page 97.
- For more information about using the storage report generated by the RPTSTG run-time option to tune your application, see *LE/VSE Programming Guide*.

STORAGE

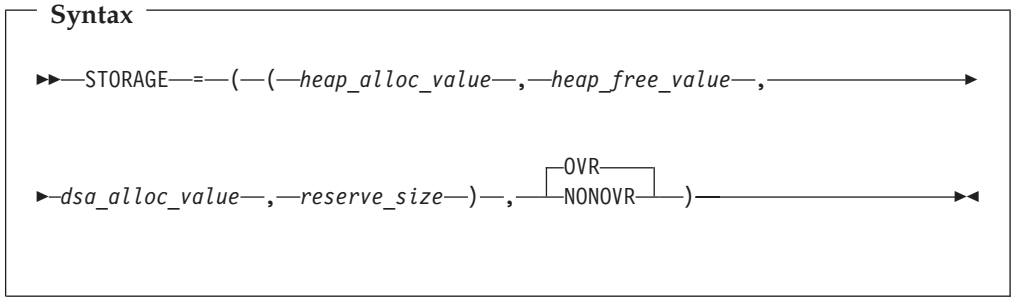
STORAGE controls the initial content of storage when allocated and freed, and the amount of storage that is reserved for the out-of-storage condition. If you specify one of the parameters in the STORAGE run-time option, all allocated storage processed by the parameter is initialized to that value. Otherwise, it is left uninitialized.

You can use the STORAGE option to identify uninitialized application variables, or prevent the accidental use of previously freed storage. STORAGE is also useful in data security. For example, storage containing sensitive data can be cleared when it is freed.

IBM-Supplied Default for CICS: STORAGE=((00,NONE,NONE,0K),OVR)

IBM-Supplied Default for Batch: STORAGE=((00,NONE,NONE,32K),OVR)

Syntax



heap_alloc_value

The initialized value of any heap storage allocated by the storage manager. You can specify *heap_alloc_value* as:

- A single character enclosed in quotes. If you specify a single character, every byte of heap storage allocated by the storage manager is initialized to that character's EBCDIC equivalent. For example, if you specify 'a' as the *heap_alloc_value*, heap storage is initialized to X'818181...81' or 'aaa...a'.
- Two hex digits without quotes. If you specify two hex digits, every byte of the allocated heap storage is initialized to that value. For example, If you specify FE as the *heap_alloc_value*, heap storage is initialized to X'FEFEFE...FE'. A *heap_alloc_value* of 00 initializes heap storage to X'0000...00'.
- NONE. If you specify the default value of NONE, the allocated heap storage is not initialized.

heap_free_value
The value with which any heap storage freed by the storage manager is overwritten. You can specify *heap_free_value* as:

- A single character enclosed in quotes. For example, a *heap_free_value* of 'f' overwrites freed heap storage to X'868686...86'; 'B' overwrites freed heap storage to X'C2'.
- Two hex digits without quotes. A *heap_free_value* of FE overwrites freed heap storage with X'FEFEFE...FE'.
- NONE. If you specify the default value of NONE, the freed heap storage is not overwritten.

dsa_alloc_value
The initialized value of stack frames from the LE/VSE stack. A stack frame is dynamically-acquired storage that is composed of a standard register save area and the area available for automatic storage.

If specified, all LE/VSE stack storage including automatic variable storage is initialized to *dsa_alloc_value*. Stack frames allocated outside the LE/VSE stack are never initialized.

You can specify *dsa_alloc_value* as:

- A single character enclosed in quotes. If you specify a single character, any dynamically-acquired stack storage allocated by the storage manager is initialized to that character's EBCDIC equivalent. For example, if you specify 'A' as the *dsa_alloc_value*, stack storage is initialized to X'C1'. A *dsa_alloc_value* of 'F' initializes stack storage to X'C6', 'd' to X'84'.
- Two hex digits without quotes. If you specify two hex digits, any dynamically acquired stack storage is initialized to that value. For example, if you specify FE as the *dsa_alloc_value*, stack storage is initialized to X'FE'. A *dsa_alloc_value* of 00 initializes stack storage to X'00', FF to X'FF'.
- NONE. If you specify the default value of NONE, the stack storage is not initialized.

reserve_size
The amount of storage for the LE/VSE storage manager to reserve in the event of an out-of-storage condition. You can specify the *reserve_size* value as *n*, *n*K, or *n*M bytes of storage. The amount of storage is rounded up to the nearest multiple of 8 bytes.

If you specify *reserve_size* as 0, no reserve segment is allocated. If you do not specify a reserve segment and your application runs out of storage, the application abends with a return code of 4088 and a reason code of 1004.

If you specify a *reserve_size* that is greater than 0 in the batch environment, LE/VSE does not immediately abend when your application runs out of storage. Instead, when the stack overflows, LE/VSE attempts to get another stack segment and add it to the stack.

If unsuccessful, LE/VSE temporarily adds the reserve stack segment to the overflowing stack, and signals the out-of-storage condition. This allows a user-written condition handler to gain control and release storage. If the reserve stack segment overflows while this is happening, LE/VSE abends with a return code of 4088 and reason code of 1004.

To avoid such an overflow, increase the size of the reserve stack segment with the STORAGE(,,,*reserve_size*) run-time option. The reserve stack segment is not freed until thread termination.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- *heap_alloc_value*, *heap_free_value*, and *dsa_alloc_value* can all be enclosed in quotes. To initialize heap storage to the EBCDIC equivalent of a single quote, double it within the string delimited by single quotes or surround it with a pair of double quotes. Both of the following are correct ways to specify a single quote:

  ```
  STORAGE('''')
  STORAGE("'")
  ```

  Similarly, double quotes must be doubled within a string delimited by double quotes, or surrounded by a pair of single quotes. The following are correct ways to specify a double quote:

  ```
  STORAGE("""")
  STORAGE('"')
  ```

- CICS consideration— The out-of-storage condition is not raised under CICS.

  If a reserved segment size is specified, either as a default or an override under CICS, this storage size will be allocated but never used. This results in wasted 24-bit storage. You are recommended to always use a reserve segment size of 0k under CICS.
- CEEUOPT consideration— If you specify the STORAGE run-time option in CEEUOPT, the default value 8K is used if the *reserve_size* suboption is omitted.
- PL/I considerations— To provide similar storage initialization as DOS/PL1, use STORAGE=(00,NONE,00,32K) as an installation default setting. Please note that a performance degradation may result from using this setting.

Performance Considerations

Using STORAGE to control initial values can increase program run time. If you specify a *dsa_alloc_value*, performance is likely to be poor. Therefore, use the *dsa_alloc_value* option only for debugging, not to initialize automatic variables or data structures.

Use STORAGE= ((00,NONE,NONE,..),...) when you are not debugging.

TERMTHDACT

TERMTHDACT sets the level of information that is produced when LE/VSE percolates a condition of severity 2 or greater beyond the first routine's stack frame. The UADUMP suboption has been added to TERMTHDACT as part of these enhancements to provide a comprehensive dump in the event of an abnormal termination.

The LE/VSE service CEE5DMP is called for the TRACE, DUMP, and UADUMP suboptions of TERMTHDACT.

The following CEE5DMP options are passed for TRACE:

NOENTRY CONDITION TRACEBACK THREAD(ALL) NOBLOCKS NOSTORAGE NOVARIABLES NOFILES STACKFRAME(ALL) PAGESIZE(60) FNAME(CEEDUMP)

The following options are passed for DUMP and UADUMP:

NOENTRY CONDITION TRACEBACK THREAD(ALL) BLOCKS STORAGE VARIABLES FILES STACKFRAME(ALL) PAGESIZE(60) FNAME(CEEDUMP)

If a message is printed (based upon the TERMTHDACT(MSG) run-time option), the message is for the active condition immediately prior to the termination imminent step. In addition, if that active condition is a promoted condition (was not the original condition), the original condition's message is printed.

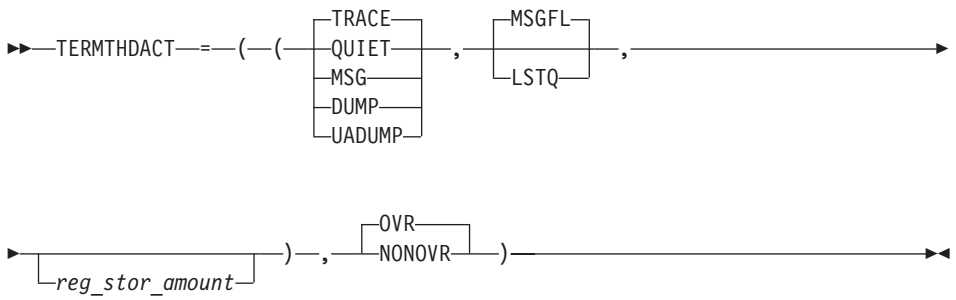
If the TRACE run-time option is specified with the DUMP suboption, a dump containing the trace table, at a minimum, is produced. The contents of the dump depend on the values set in the TERMTHDACT run-time option.

- Under abnormal termination, the following dump contents are generated:
  - TERMTHDACT(QUIET)—generates a dump containing the trace table only
  - TERMTHDACT(MSG)—generates a dump containing the trace table only
  - TERMTHDACT(TRACE)—generates a dump containing the trace table and the traceback
  - TERMTHDACT(DUMP)—generates a dump containing thread/enclave/process storage and control blocks (the trace table is included as an enclave control block)
  - TERMTHDACT(UADUMP)—generates a CEE5DMP, which is the same as TERMTHDACT(DUMP), as well as producing a complete VSE system dump.
- Under normal termination, the following dump contents are generated:
  - Independent of the TERMTHDACT setting, LE/VSE generates a dump containing the trace table only.

IBM-Supplied Default for CICS: TERMTHDACT((TRACE,MSGFL,0),OVR)

IBM-Supplied Default for Batch: TERMTHDACT((TRACE,,0),OVR)

Syntax



TRACE
Specifies that when a thread terminates due to an unhandled condition of severity 2 or greater, LE/VSE generates a message indicating the cause of the termination and a trace of the active routines on the activation stack.

QUIET
Specifies that LE/VSE does not generate a message when a thread terminates due to an unhandled condition of severity 2 or greater.

MSG
Specifies that when a thread terminates due to an unhandled condition of severity 2 or greater, LE/VSE generates a message indicating the cause of the termination.

DUMP
Specifies that when a thread terminates due to an unhandled condition of severity 2 or greater, LE/VSE generates a message indicating the cause of the termination, a trace of the active routines on the activation stack, and an LE/VSE dump.

A currently active run-time options report will also be sent to the dump destination for problem diagnosis assistance, even if RPTOPTS(OFF) has been previously specified. If RPTOPTS(ON) has been specified and a dump is produced in response to a failure only a single run-time options report will be generated within the dump output.

UADUMP
Specifies that when a thread terminates due to an unhandled condition of severity 2 or greater, LE/VSE generates a message indicating the cause of the termination, a trace of the active routines on the activation stack, and an LE/VSE dump, and, depending on the JCL OPTION DUMP setting, a system dump of the user partition.

If abnormal termination occurs while TERMTHDACT(UADUMP) is set, if possible LE/VSE will attempt to re-execute the failing instruction, to produce a system dump. If this is not possible, LE/VSE will issue an JDUMP macro to terminate the enclave with a system dump.

A currently active run-time options report will also be sent to the dump destination for problem diagnosis assistance, even if RPTOPTS(OFF) has been

previously specified. If RPTOPTS(ON) has been specified and a dump is produced in response to a failure only a single run-time options report will be generated within the dump output.

MSGFL
Causes all dump output to be sent to the output destination specified by the Run-Time option MSGFILE.

Usage Notes:

1. This sub-option is only applicable under CICS. MSGFL is aways used in a BATCH environment.
2. A run-time options report in the BATCH environment will not report on the dump destination setting as its usage is only applicable to the CICS environment.

LSTQ
Causes all dump output to be sent to the VSE/POWER LSTQ. This setting is only available in the CICS environment.

Usage Notes:

1. This sub-option is ignored if set in a BATCH environment.
2. The LSTQ option will only take effect when a condition of severity 2 or greater goes unhandled and a dump is requested.
3. Callable services, such as CEE5DMP, will still have dump output sent to the destination specified in the MSGFILE runtime option.
4. Abend messages and LE/VSE reports will still be sent to the MSGFILE destination even if LSTQ is set. The exception to this is when DUMP or UADUMP are set and a run-time options report is generated as part of the dump output. In this case, the options report will be send to the LSTQ member along with the dump output.
5. A TERMTHDACT level setting of TRACE or more must be in effect for dump output to be produced. Otherwise all output will still be sent to the MSGFILE destination.
6. Output sent to the VSE/POWER LSTQ will be stored on CLASS=L with DISP=D attributes. These defaults can be modified by way of a supplied member CEEWLSTQ.Z in the LE/VSE installation sub-library or ICCF Library 62.
7. A run-time options report in the BATCH environment will not report on the dump destination setting as its usage is only applicable to the CICS environment.
8. The VSE/POWER LSTQ entry name is built by using the failing CICS transaction identification name and the CICS terminal number that the user was signed-on to at the time of the failure. For example, if transaction `MENU` fails on terminal A001, the LSTQ entry name will be `MENUA001`.

reg_stor_amount
Controls the amount of storage to be dumped around registers. The *reg_stor_amount* variable must be in the range 0-256 and indicates the storage in bytes to be dumped around each register. The *reg_stor_amount* value will be rounded up to the nearest multiple of 32. If you call the CEE5DMP service and do not require a dump storage around registers (regardless of the *reg_stor_amount* value), you must specify `REGSTor(0)` as a CEE5DMP option.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- PL/I considerations—After a normal return from a PL/I ERROR ON-unit or from a PL/I FINISH ON-unit, LE/VSE considers the condition unhandled. If a GOTO is not performed and the resume cursor is not moved, the thread terminates. The TERMTHDACT setting guides the amount of information that is produced. The message is not presented twice.
- Batch considerations—The recommended setting for debugging under Batch is TERMTHDACT(UADUMP).
- CICS considerations—The recommended setting for debugging under CICS is TERMTHDACT(DUMP).

For More Information

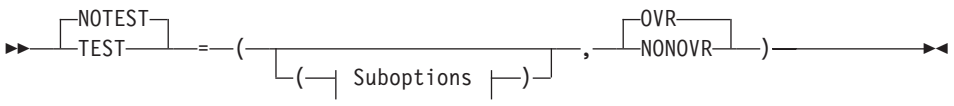
- For more information about the TRACE and USRHDLR run-time options, see the run-time option descriptions in the *LE/VSE Programming Reference*.
- For more information about the CEE5DMP service and its parameters, see the description of the CEE5DMP callable service in the *LE/VSE Programming Reference*.
- For more information about the TERMTHDACT run-time option and condition messages, see *LE/VSE Programming Guide*.
- For more information about the CESE transient data queue, see *LE/VSE Programming Guide*.

TEST

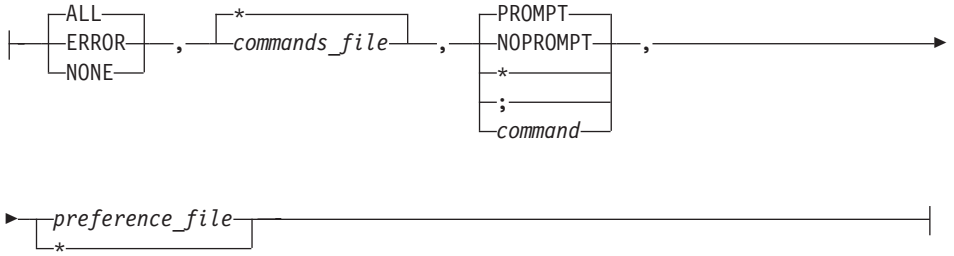
TEST specifies the conditions under which a debug tool (such as Debug Tool for VSE/ESA) assumes control when the user application is being initialized. Parameters of the TEST and NOTEST run-time options are merged as one set of parameters.

IBM-Supplied Default: NOTEST=((ALL,*,PROMPT,''),OVR)

Syntax



Suboptions:



ALL
Specifies that either of the following will cause the debug tool to gain control even without a defined AT OCCURRENCE for a particular condition or AT TERMINATION:
- Any LE/VSE condition of severity 1 or above
- Application termination

ERROR
Specifies that only one of the following will cause the debug tool to gain control without a defined AT OCCURRENCE for a particular condition or AT TERMINATION:
- Any LE/VSE-defined error condition of severity 2 or higher
- Application termination

NONE
Specifies that no condition will cause the debug tool to gain control without a defined AT OCCURRENCE for a particular condition or AT TERMINATION.

commands_file
A character string, up to 80 characters long, that you use to pass to the debug tool the filename of the primary commands file for this run. The syntax of the character string is defined by the debug tool you are using.

You can enclose *commands_file* in single or double quotes to distinguish it from the rest of the TEST | NOTEST suboption list.

***** (asterisk—in place of *commands_file*)
Specifies that no *commands_file* is supplied. The default device, as defined by your debug tool, is used as the source of the debug tool commands.

PROMPT
Specifies that the debug tool is invoked at LE/VSE initialization.

NOPROMPT
Specifies that the debug tool is not invoked at LE/VSE initialization.

***** (asterisk—in place of PROMPT/NOPROMPT)
Specifies that the debug tool is not invoked at LE/VSE initialization; equivalent to NOPROMPT.

; (semicolon—in place of PROMPT/NOPROMPT)
Specifies that the debug tool is invoked at LE/VSE initialization; equivalent to PROMPT.

command
A character string, up to 250 characters long, that specifies a valid debug tool command. The command string can be enclosed in single or double quotes to distinguish it from the rest of the TEST parameter list; it cannot contain DBCS characters. Quotes are needed whenever the command string contains embedded blanks, commas, semicolons, or parentheses. The list can have a maximum of 250 characters.

preference_file
A character string, up to 80 characters long, that you use to pass to the debug tool the filename of the preference file to be used. A preference file is a type of commands file that you can use to specify settings for your debugging environment. The syntax of the character string is defined by the debug tool you are using.

You can enclose *preference_file* in single or double quotes to distinguish it from the rest of the TEST parameter list.

* (asterisk—in place of *preference_file*)
  Specifies that no *preference_file* is supplied.

Usage Notes

You can specify parameters on the NOTEST option. If NOTEST is in effect when the application gains control, it is interpreted as TEST(NONE,,*,). If Debug Tool for VSE/ESA is initialized using a CALL CEETEST or equivalent, the initial test level, the initial *commands_file*, and the initial *preference_file* are taken from the NOTEST run-time option setting.

Performance Consideration

To improve performance, use this option only while debugging.

For More Information

- For more information about the syntax of the TEST run-time option when using Debug Tool for VSE/ESA, see *Debug Tool for VSE/ESA User's Guide and Reference*.
- For more information about creating and using a commands and a preference file for Debug Tool for VSE/ESA, see *Debug Tool for VSE/ESA User's Guide and Reference*.

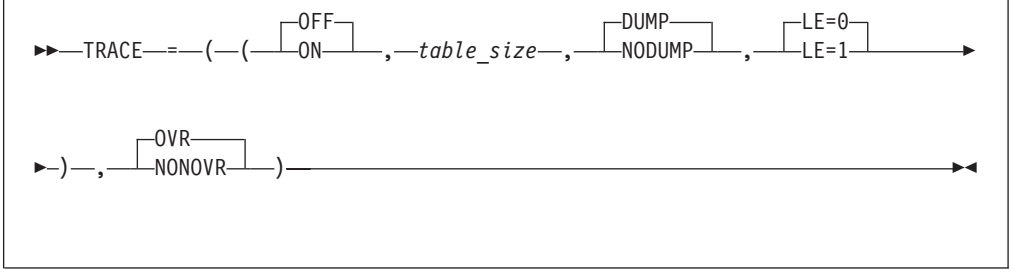
TRACE

TRACE controls run-time library tracing activity, the size of the in-storage trace table, the type of trace events to record, and it determines whether a dump containing, at a minimum, the trace table should be unconditionally taken when the application terminates. When you specify TRACE(ON), user-requested trace entries are intermixed with LE/VSE trace entries in the trace table.

Under normal termination conditions, if TRACE is active and you specify DUMP, only the trace table is written to the dump report, independent of the TERMTHDACT setting. Only one dump is taken for each termination. Under abnormal termination conditions, the type of dump taken (if one is taken) depends on the value of the TERMTHDACT run-time option and whether TRACE is active and the DUMP suboption is specified.

IBM-Supplied Default: TRACE=((OFF,4K,DUMP,LE=0),OVR)

Syntax



OFF
  Indicates that the tracing facility is inactive.

ON
  Indicates that the tracing facility is active.

table_size
  Determines the size of the tracing table as specified in bytes (*n*K or *n*M). The upper limit is 16M.

DUMP
Requests that an LE/VSE-formatted dump (containing the trace table) be produced at program termination regardless of the setting of the TERMTHDACT run-time option.

NODUMP
Requests that an LE/VSE-formatted dump not be produced at program termination.

LE=0
Specifies that no trace events be recorded.

LE=1
Specifies that entry to and exit from LE/VSE member libraries be recorded (such as, in the case of C, entry and exit of the `printf()` library function).

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

For More Information

- For more information about the dump contents, see "TERMTHDACT" on page 106.
- For more information about using the tracing facility, see *LE/VSE Debugging Guide and Run-Time Messages*.

TRAP

Trap specifies the level of condition handling that LE/VSE performs for user abends and program interrupts.

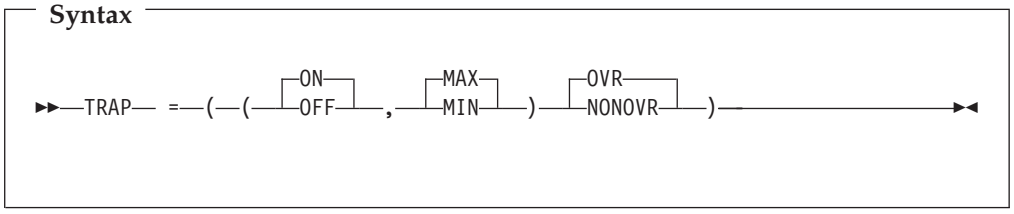
You must specify at least TRAP=((ON,MIN),..) in order for applications to run successfully.

TRAP=((ON,MAX),..) must be in effect for the ABTERMENC or ABPERC run-time options to have effect.

The use of the CEESGL callable service is not affected by this option.

IBM-Supplied Default: TRAP=((ON,MAX),OVR)

Syntax



ON
Enables LE/VSE condition handling.

OFF
Disables LE/VSE condition handling. The MIN|MAX option is ignored when TRAP(OFF) is used.

MAX
Instructs LE/VSE to activate full condition handling. This will involve the use

of both STXIT AB and STXIT PC processing. TRAP sub-option ON must also be specified for this option to have any effect.

MIN
Instructs LE/VSE not to use any STXIT AB processing for LE/VSE condition handling and to only use STXIT PC condition handling. This is required for internal LE/VSE failures that are part of application abend reporting and dump producing functions. TRAP sub-option ON must also be specified for this option to have any effect.

Note: Each of the HLLs will still issue STXIT AB's as required regardless of the TRAP run-time option setting.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

During normal processing, LE/VSE expects TRAP(ON,..) to be in effect for the application to run successfully. Use TRAP(ON,MIN) when a program exception needs to be analyzed. The use of TRAP(OFF,...) is not recommended and can cause unpredictable results to occur. Alternatively, an abnormal termination exit can be used to analyze the failure and report on the problem using information provided by LE/VSE to the exit.

Specifying TRAP(ON) without MIN or MAX is equivilent to specifying TRAP(ON,MAX).

The use of TRAP(ON,MIN) when an abnormal condition occurs within a user application will result in:

- Any user registered condition handlers will not be called. This includes handlers specified via the USRHDLR run-time option.
- LE DUMP processing will not be performed.
- ABTERMENC run-time option has no effect.
- ABPERC run-time option has no effect.
- No resources acquired by LE/VSE are released.
- Files opened by HLLs are not closed and this may result in the loss of data.
- The abnormal termination exit is not driven.
- The assembler user exit is not called for enclave termination.
- Debug Tool is not notified of the error.
- No storage or run-time options reports are generated.
- No LE/VSE messages are produced.
- For a user program interrupt, or an unexpected LE/VSE program interrupt, LE/VSE will try to re-execute the failing instruction in an attempt to invoke normal VSE abnormal termination semanics. If this is not possible, a JDUMP macro will be issued to terminate the enclave and to produce a system dump.

  Note: LE/VSE internal condition handling is enabled but any user program interrupts or abends are not handled by LE/VSE condition management.

Running with TRAP(OFF) can cause many side effects because LE/VSE requires internal condition handling to successfully execute. If you run with TRAP(OFF), you can get failures even if you do not encounter a software-raised condition, program check or abend. If you do encounter a program check or abend with TRAP(OFF) in effect, the following will occur:

- The ABTERMENC run-time option will have no effect.

- The ABPERC run-time option will have no effect.
- Resources acquired by LE/VSE are not freed.
- Files opened by HLLs are not closed so data may be lost.
- The abnormal termination exit is not driven for enclave termination.
- The assembler user exit is not driven for enclave termination.
- User condition handlers are not enabled.
- The Debug Tool is not notified of the error.
- No storage report or run-time options report is generated.

The enclave terminates abnormally if such conditions are raised.

TRAP(ON,MAX) must be in effect when you want to use the CEEBXITA assembler user exit for enclave initialization to specify a list of VSE cancel codes, program interruption code and user abend codes that LE/VSE exempts from normal condition handling.
When TRAP(ON,MAX) is in effect and an abnormal condition occurs, if the VSE cancel code, program-interruption code, or user abend code is in the CEEAUE_CODES list in CEEBXITA, LE/VSE exempts the condition from normal condition handling. Normal LE/VSE condition handling is never invoked to handle these conditions. This feature is useful when you do not want LE/VSE condition handling to intervene for certain abnormal conditions, or when you want to prevent invocation of the abnormal termination exit for such conditions.
When TRAP(OFF) or TRAP(ON,MIN) are set and there is a program interrupt, or an abend, the assembler user exit is not driven at termination. Any information provided in CEEAUE_CODES is ignored.

CICS Considerations

The MIN|MAX option is ignored under CICS. If you specify TRAP(OFF) in a CICS environment, LE/VSE does not produce any messages or dumps for conditions raised by program interrupts or transaction abends. The standard CICS system action occurs. However abends not caused by the application program, can occur, as internal LE/VSE condition handling has been disabled by the use of TRAP(OFF). Therefore, this is not a recommended setting to use under CICS.

Performance Considerations

When using COBOL internal sorts, with a SORT product's STXIT option activated either by default or by design, performance benefits can be achieved by using TRAP(ON,MIN) for the application. Alternatively, if possible, set the SORT product's option to NOSTXIT (or to MINSTXIT in the case of DFSORT/VSE), which will allow the use of TRAP(ON,MAX) without impacting performance.

Fore More Information

For more information about the:

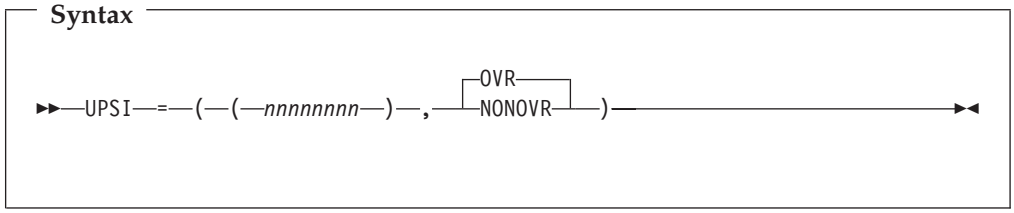
- ABPERC or ABTERMENC run-time options, see "ABPERC" on page 69 and "ABTERMENC" on page 71 respectively.
- CEESGL callable service, refer to the *LE/VSE Programming Reference*.
- CEEBXITA assembler user exit, refer to the *LE/VSE Programming Guide*.

UPSI (COBOL Only)

UPSI sets the eight UPSI switches on or off for applications that use COBOL routines.

IBM-Supplied Default: UPSI=((00000000),OVR)

Syntax



nnnnnnnn
n represents one UPSI switch between 0 and 7, the leftmost *n* representing the first switch. Each *n* can either be 0 (off) or 1 (on).

The IBM-supplied default setting is UPSI(00000000).

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

Do not confuse the LE/VSE UPSI run-time option with the job control UPSI statement. The UPSI switches set by the job control UPSI statement are not available to COBOL routines under LE/VSE.

For More Information

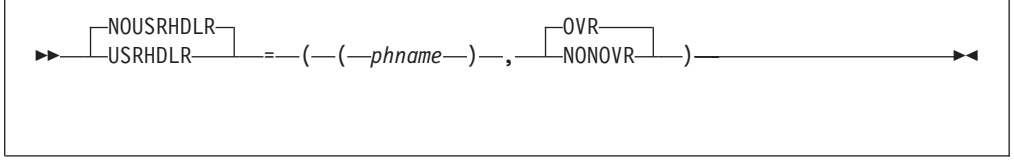
For more information on how COBOL routines access the UPSI switches, see *LE/VSE Programming Guide*.

USRHDLR

USRHDLR registers a user condition handler at stack frame 0, allowing you to register a user condition handler without having to include a call to CEEHDLR in your application and then recompile the application.

IBM-Supplied Default: NOUSRHDLR=(),OVR)

Syntax



NOUSRHDLR
Does not register a user condition handler without recompiling an application to include a call to CEEHDLR.

USRHDLR
Registers a user condition handler without recompiling an application to include a call to CEEHDLR.

phname
The name of a phase that contains the user condition handler that is to be registered at stack frame 0.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

- User Handler module names of YES, NO, ON and OFF, are no longer permitted.
- The user condition handler specified by the USRHDLR run-time option must be in a separate phase rather than be linkedited with the rest of the application.
- The user condition handler *phname* is invoked for conditions that are still unhandled after being presented to condition handlers for the main program.
- Restriction - if USRHDLR is in effect, you cannot resume execution in the program in which the condition occurs. This includes calls in the condition handler to CEEMRCR and CEEMRCE.
- You can use a user condition handler registered with the USRHDLR run-time option to return any of the result codes allowed for a user condition handler registered with the CEEHDLR callable service.
- A condition that is percolated or promoted by a user condition handler registered with the USRHDLR run-time option, is not presented to any other user condition handler.
- The loading of the user condition handler *phname* occurs only when that user condition handler needs to be invoked the first time.
- PL/I Consideration - Although PL/I cannot use the CEEHDLR callable service to register a user-written condition handler, PL/I can use the USRHDLR run-time option.
- CICS Consideration - *phname* must be defined in the CICS System Definition File (CSD).

For More Information

For more information on registering a user condition handler, see CEEHDLR callable service in the *LE/VSE Programming Reference*.

XUFLOW

XUFLOW specifies whether an exponent underflow causes a program interrupt. An exponent underflow occurs when a floating point number becomes too small to be represented.

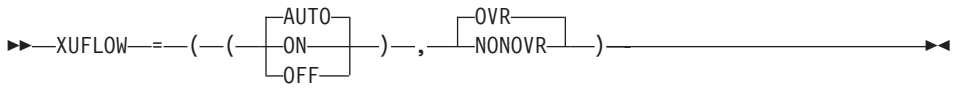
The underflow setting is determined at enclave initialization and is updated when new languages are introduced into the application (via fetch or dynamic call, for example). Otherwise, it does not vary while the application is running.

LE/VSE preserves the language semantics for C and COBOL regardless of the XUFLOW setting. LE/VSE preserves the language semantics for PL/I only when XUFLOW is set to AUTO or ON. LE/VSE does not preserve the language semantics for PL/I when XUFLOW is set to OFF.

An exponent underflow caused by a C or COBOL routine does not cause a condition to be raised.

IBM-Supplied Default: XUFLOW=((AUTO),OVR)

Syntax



AUTO

An exponent underflow causes or does not cause a program interrupt dynamically, based upon the HLLs that make up the application. Enablement is determined without user intervention.

XUFLOW(AUTO) causes condition management to process underflows only in those applications where the semantics of the application languages require it. Normally, XUFLOW(AUTO) provides the best efficiency while meeting language semantics.

ON

An exponent underflow causes a program interrupt.

XUFLOW(ON) causes condition management to process underflows regardless of the mix of languages; therefore, this setting might be less efficient in applications that consist of languages not requiring underflows to be processed by condition management.

OFF

An exponent underflow does not cause a program interrupt; the hardware takes care of the underflow.

When you set XUFLOW to OFF, the hardware processes exponent underflows. This is more efficient than condition handling to process the underflow.

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

Usage Notes

PL/I consideration—You should use XUFLOW=((AUTO),...) or XUFLOW=((ON),...) for PL/I.

Activating Changed CICS-Wide Run-Time Options

This function allows you to activate changed LE/VSE CICS-wide default runtime options, without the need to re-cycle the CICS system.

This means, you can change your installation CICS-wide default runtime options by editing and running the supplied CEEWCOPT job, and activate these new runtime options by executing the supplied CICS transid NEWC. This is done while the CICS system is still processing online transactions.

The function is available for both CICS Transaction Server 1.1.1 and CICS/VSE 2.3 systems.

You may change the transaction name used to perform this function to one of your choice. This functionality is provided by program EDCCNEWC and the NEWC transaction, which you use to establish new option settings.

Notes:

1. If you decide to change the transaction name as described above, before using this transaction you must ensure that BSM-security definitions have been established for your transaction name.
2. Each transaction used with the CICS Transaction Server must be "security-enabled" before it can be executed. The *Interactive Interface* provides this support via the *Define Transaction Security* dialog. To access this dialog, you use the selection path **Resource Definition — Define Transaction Security**.

Installing Function for Activating Changed CICS-Wide Options

To install the new function for activating changed CICS-wide options, follow these steps:

1. Edit JCL member CEEWCOPT (supplied in ICCF LIB 59 and in the LE/VSE installation library as CEEWCOPT.Z).
2. Submit the modified JCL for execution. Ensure the job completes with a return code *not greater* than 2 (RC<=2).
3. If you have CEECOPT.PHASE loaded in the SVA, you will need to re-load this module using the **SET SDL** command from BG partition. For example,

```
0 // LIBDEF PHASE,SEARCH=PRD2.SCEEBASE (or other Lib where CEECOPT.PHASE is cataloged)
0 SET SDL
0 CEECOPT,SVA
0 /*
```

4. Sign-on to your CICS system. Exit to a 'blank' CICS session (if using IUI, press PF6).
5. Enter **NEWC** (or whatever transaction code you have defined). You should receive these messages on the CICS terminal:

```
CEE3553I CICS Options Newcopy Started
CEE3554I CICS Options Newcopy Complete
```

   If the options module *newcopy* fails to complete in some way, you will receive this message:

```
CEE3555E CICS Options Newcopy Failed
```

   Keep the CICS output log for problem determination, and report the failure to your Systems Programmer or IBM Support Center.
6. If all completed sucessfully, your new installation default CICS-wide runtime options are now activated for this CICS system.

Notes:

1. If you have more than one CICS system for which you want the new options activated, you will need to repeat steps 4-6 for each of the CICS systems. Otherwise, the new default options will not be activated until the CICS systems are re-started (with a COLD/EMERGENCY start-up).
2. Transaction NEWC is shipped with “security enabled” (using BSM) and ready for you to use.
3. You can perform an immediate verification of your LE/CICS run-time option changes, by using the transaction `ROPC`. An updated options report is available at destination MSGFILE immediately upon completion of the NEWC transaction.

Printing CICS-Wide Run-Time Options to Console

This function allows you to print LE/VSE CICS-wide run-time options to your VSE/ESA console. This provides an alternative to the global CICS-wide setting that is possible using the LE/VSE run-time option `RPTOPTS(ON)`. This function allows you to avoid producing large output on the LE/VSE destination `CESE` (in contrast to the `RPTOPTS(ON)` run-time option which might fill up this queue, if defined as a file, and generate message CEE3492S).

From VSE/ESA 2.5 onwards, LE/VSE program EDCYCROP is shipped with the CICS transid already set to ‘ROPC’. Transaction ROPC is already BSM security-enabled, and can be used immediately. When this transaction has been invoked on a CICS terminal, LE/CICS-wide default run-time options (CEECOPT) appear on the console.

Notes:

1. If you have set the ENVAR run-time option of the CICS-wide Assember User Exit, you can list this option in the CICS-Wide Options report sent to your VSE/ESA console. In the example report shown in Figure 15 on page 121, the ENVAR run-time option has been included.
2. The function is available with both CICS/VSE 2.3 and CICS Transaction Server.
3. For VSE systems before VSE/ESA 2.5, you can specify your own CICS transid for LE/VSE program EDCYCROP.

```
F2 0103 Options Report for Enclave EDCYCROP 03/09/01 4:24:01 PM
F2 0103
F2 0103 LAST WHERE SET          OPTION
F2 0103 ------------------------------------------------------------------------
F2 0103 LE/CICS-wide default      ABPERC(NONE)
F2 0103 LE/CICS-wide default      ABTERMENC(ABEND)
F2 0103 LE/CICS-wide default    NOAIXBLD
F2 0103 LE/CICS-wide default      ALL31(ON)
F2 0103 LE/CICS-wide default      ANYHEAP(4096,4080,ANYWHERE,FREE)
F2 0103 LE/CICS-wide default      BELOWHEAP(4096,4080,FREE)
F2 0103 LE/CICS-wide default      CBLOPTS(ON)
F2 0103 LE/CICS-wide default      CBLPSHPOP(ON)
F2 0103 LE/CICS-wide default      CHECK(OFF)
F2 0103 LE/CICS-wide default      COUNTRY(US)
F2 0103 LE/CICS-wide default    NODEBUG
F2 0103 LE/CICS-wide default      DEPTHCONDLMT(10)
F2 0103 LE/CICS-wide default      ENVAR("")
F2 0103 LE/CICS-wide default      ERRCOUNT(20)
F2 0103 LE/CICS-wide default      HEAP(4096,4080,ANYWHERE,KEEP,4096,4080)
F2 0103 LE/CICS-wide default      HEAPCHK(OFF,1,0)
F2 0103 LE/CICS-wide default      LIBSTACK(4096,4080,FREE)
F2 0103 LE/CICS-wide default      MSGFILE(CESE)
F2 0103 LE/CICS-wide default      MSGQ(15)
F2 0103 LE/CICS-wide default      NATLANG(UEN)
F2 0103 LE/CICS-wide default      RETZERO(OFF)
F2 0103 LE/CICS-wide default      RPTOPTS(OFF)
F2 0103 LE/CICS-wide default      RPTSTG(OFF)
F2 0103 LE/CICS-wide default    NORTEREUS
F2 0103 LE/CICS-wide default      STACK(4096,4080,ANYWHERE,KEEP)
F2 0103 LE/CICS-wide default      STORAGE(00,NONE,NONE,0)
F2 0103 LE/CICS-wide default      TERMTHDACT(TRACE,MSGFL,0)
F2 0103 LE/CICS-wide default    NOTEST(ALL,"*","PROMPT","")
F2 0103 LE/CICS-wide default      TRACE(OFF,4096,DUMP,LE=0)
F2 0103 LE/CICS-wide default      TRAP(ON,MAX)
F2 0103 LE/CICS-wide default      UPSI(00000000)
F2 0103 LE/CICS-wide default    NOUSRHDLR()
F2 0103 LE/CICS-wide default      XUFLOW(AUTO)
```

Figure 15. Sample of LE/CICS-Wide Options Printed to Console

Appendix B. LE/VSE Run-Time LIOCS Phases

This appendix includes descriptions of the macros supplied with LE/VSE to generate the run-time LIOCS phases for card, diskette, and printer devices. The run-time LIOCS phases contain logic modules used by VSE input/output services when processing files assigned to the these device types.

If the logic module required by VSE input/output services cannot be found, message CEE3751S is issued. This may mean you need to customize the LE/VSE-supplied phase. The phases that you can customize are:

- CEEYCD0 - LIOCS routines for card reader/punch devices
- CEEYDU0 - LIOCS routines for diskette devices
- CEEYPR0 - LIOCS routines for printer devices.

Each LIOCS logic module in the LIOCS phases has a unique name in the format IJ*xxxxxx*, where *xxxxxx* is determined by the file attributes specified in the LE/VSE-supplied macros, CEEXCDMD, CEEXDUMD and CEEXPRMD. These macros are the front-ends to the VSE system macros, CDMOD, DUMODFI, DUMODFO and PRMOD. For more information about these VSE system macros and the names of the modules, see *VSE/ESA System Macros Reference*.

Note!:

1. During customization, you are recommended not to delete or change any shipped LE/VSE LIOCS definitions. If LIOCS definitions for dummy card punch, card reader, and printer devices are missing, the startup of the CICS Transaction Server might be severely affected.
2. The Interactive Interface's *Hardware Configuration* dialog also contains definitions for dummy card punch, card reader, and print devices. These definitions reflect the currently-shipped LE/VSE LIOCS. You are recommended not to delete or change any of these definitions.
3. If, however, you do decide to change any of the dummy device defnitions, *ensure you have customized the appropriate LIOCS definitions for these devices!.*

CEEYCD0—Card Device Run-Time LIOCS Phase

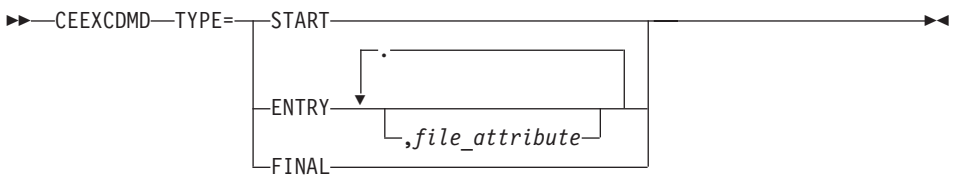
The run-time LIOCS phase CEEYCD0 contains the LIOCS logic modules required for processing files assigned to card reader and card punch devices. For information about the types of card files supported by the IBM-supplied CEEYCD0 phase, see Figure 4 on page 25.

If you plan to run an application that processes a card file other than those supported by the IBM-supplied CEEYCD0 phase, use the CEEXCDMD macro to generate the required logic module.

Each LIOCS logic module in CEEYCD0 has a unique name in the format IJC*xxxxx*, where *xxxxx* is determined by the file attributes specified in the CEEXCDMD macro. The CEEXCDMD macro is a front-end to the VSE system macro, CDMOD.

The syntax of the CEEXCDMD macro is as follows:

Syntax



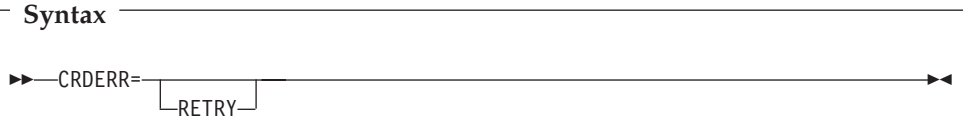
The following sections describe the file attributes that you can specify on the CEEXCDMD macro. Refer to the description of CDMOD in *VSE/ESA System Macros Reference* to determine the values you should supply for these attributes.

Note: The CONTROL operand of CDMOD is not applicable to CEEYCD0.

CRDERR

CRDERR specifies whether error retry routines for the IBM 2520 and 2540 are included in the generated logic module.

Syntax



RETRY
Specifies that the generated logic module contains error retry routines for the IBM 2520 and 2540 punch-equipment check.

Default
If you do not specify this file attribute, the IBM-supplied default is that the generated logic module does not contain error retry routines.

Usage Notes

Do not specify this file attribute for input or combined files.

CTLCHR

CTLCHR specifies whether or not the first character of each record written to this card punch file contains a stacker selection control character.

Syntax

ASA

Specifies that the first character of each record written to this card punch file is an American National Standard stacker selection character. Specify CTLCHR=ASA if the file you want to process is either:

- Defined in a VS COBOL II or COBOL/VSE program as a non-EXTERNAL file, and written using the AFTER ADVANCING phrase of the WRITE statement to control stacker selection
- Declared in a PL/I VSE program and declared with the CTLASA ENVIRONMENT option to control stacker selection

YES

Specifies that the first character of each record written to this card punch file is a System/370 stacker selection character. Specify CTLCHR=YES if the file you want to process is either:

- Defined in a VS COBOL II or COBOL/VSE program as an EXTERNAL file, and written using the AFTER ADVANCING phrase of the WRITE statement to control stacker selection
- Declared in a PL/I VSE program and declared with the CTL360 ENVIRONMENT option to control stacker selection

Default

If you do not specify this file attribute, the IBM-supplied default is that records written to this file do not contain stacker selection control characters.

Usage Notes

This file attribute should only be specified for card punch devices.

DEVICE

DEVICE specifies the device code of the IBM device the generated logic module supports.

Syntax

1442
Generates code for an IBM 1442 card reader/punch.

2520
Generates code for an IBM 2520 card reader/punch.

2540
Generates code for an IBM 2540 card reader/punch.

3505
Generates code for an IBM 3505 card reader/punch.

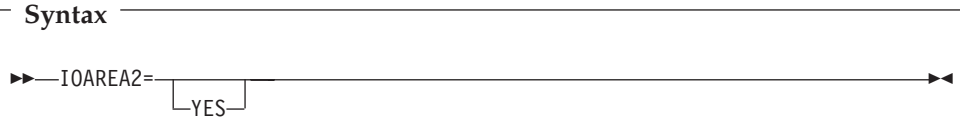
3525
Generates code for an IBM 3525 multifunction card unit.

3881
Generates code for an IBM 3881 card reader.

IOAREA2

IOAREA2 specifies whether or not a second I/O area is used for this file.

Syntax



YES
Specifies that two I/O areas are used for this file. Specify IOAREA2=YES if the file you want to process is either:

- Defined in a VS COBOL II or COBOL/VSE program, and defined without the RESERVE 1 AREA clause of the FILE-CONTROL paragraph
- Declared in a PL/I VSE program, and declared without the BUFFERS(1) .

Default
If you do not specify this file attribute, the IBM-supplied default is that one I/O area is used for this file.

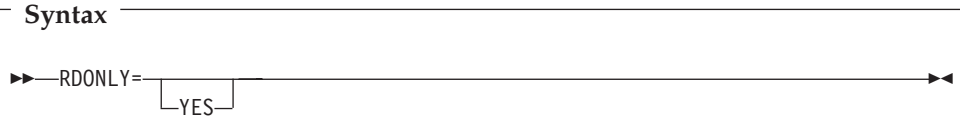
Usage Notes

Do not specify this file attribute for combined files.

RDONLY

RDONLY specifies whether or not the generated logic module is a read-only module.

Syntax



YES
Specifies that the generated logic module is read-only. Specify RDONLY=YES for all LE/VSE-conforming HLL applications.

Default
If you do not specify this file attribute, the IBM-supplied default is that the generated logic module is not read-only.

Usage Notes

If all the logic modules in the LIOCS phase CEEYCD0 are generated with the RDONLY=YES attribute, the phase is eligible for inclusion in the SVA.

RECFORM

RECFORM specifies the record format of this file.

Syntax

FIXUNB
Specifies a record format of fixed length, unblocked. If you specify any of the attributes TYPEFLE=INPUT, TYPEFLE=CMBND, or DEVICE=3881, you must specify RECFORM=FIXUNB.

UNDEF
Specifies a record format of undefined.

VARUNB
Specifies a record format of variable length, unblocked.

TYPEFLE

TYPEFLE specifies whether the generated logic module is for an input file or an output file.

Syntax

INPUT
Generates a logic module for an input file. If you specify DEVICE=3881, you must specify TYPEFLE=INPUT.

OUTPUT
Generates a logic module for an output file.

CMBND
Generates a logic module for a combined input and output file.

WORKA

WORKA specifies whether or not records are processed in work areas instead of I/O areas.

Syntax

YES
Specifies that records are processed in work areas instead of I/O areas. WORKA=YES is not valid for the IBM 3881.

Default
If you do not specify this file attribute, the IBM-supplied default is that records are processed in I/O areas.

CEEYDU0—Diskette Device Run-Time LIOCS Phase

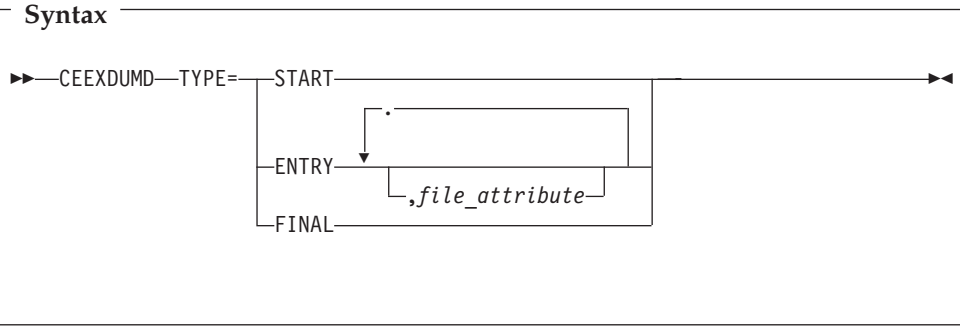
The run-time LIOCS phase CEEYDU0 contains the LIOCS logic modules required for processing files assigned to diskette devices. For information about the types of diskette files supported by the IBM-supplied CEEYDU0 phase, see Figure 5 on page 26.

If you plan to run an application that processes a diskette file other than those supported by the IBM-supplied CEEYDU0 phase, use the CEEXDUMD macro to generate the required logic module.

Each LIOCS logic module in CEEYDU0 has a unique name in the format IJND*xxxx*, where *xxxx* is determined by the file attributes specified in the CEEXDUMD macro. The CEEXDUMD macro is a front-end to the VSE system macros, DUMODFI and DUMODFO.

The syntax of the CEEXDUMD macro is as follows:

Syntax



The following sections describe the file attributes that you can specify on the CEEXDUMD macro. Refer to the description of DUMODFI and DUMODFO in *VSE/ESA System Macros Reference* to determine the values you should supply for these file attributes.

RDONLY

RDONLY specifies whether or not the generated logic module is a read-only module.

Syntax

YES
Specifies that the generated logic module is read-only. Specify RDONLY=YES for all LE/VSE-conforming HLL applications.

Default
If you do not specify this file attribute, the IBM-supplied default is that the generated logic module is not read-only.

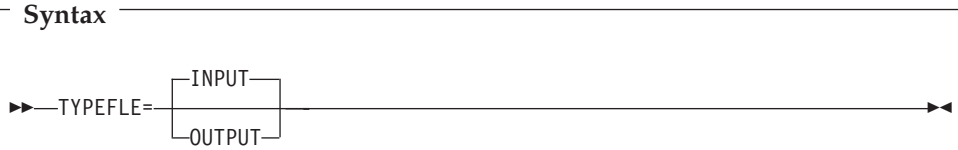
Usage Notes

Even if all the logic modules in the LIOCS phase CEEYDU0 are generated with the RDONLY=YES attribute, the phase is not eligible for inclusion in the SVA.

TYPEFLE

TYPEFLE specifies whether the generated logic module is for an input file or an output file.

Syntax



INPUT
Generates a logic module for an input file.

OUTPUT
Generates a logic module for an output file.

CEEYPR0—Printer Device Run-Time LIOCS Phase

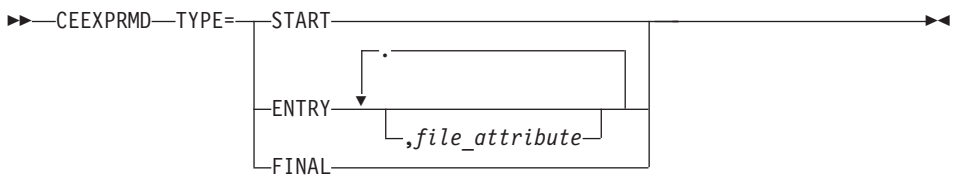
The run-time LIOCS phase CEEYPR0 contains the LIOCS logic modules required for processing files assigned to printer devices. For information about the types of printer files supported by the IBM-supplied CEEYPR0 phase, see Figure 6 on page 26.

If you plan to run an application that processes a printer file other than those supported by the IBM-supplied CEEYPR0 phase, use the CEEXPRMD macro to generate the required logic module.

Each LIOCS logic module in CEEYPR0 has a unique name in the format IJD*xxxxx*, where *xxxxx* is determined by the file attributes specified in the CEEXPRMD macro. The CEEXPRMD macro is a front-end to the VSE system macro, PRMOD.

The syntax of the CEEXPRMD macro is as follows:

Syntax



The following sections describe the file attributes that you can specify on the CEEXPRMD macro. Refer to the description of PRMOD in *VSE/ESA System Macros Reference* for the values you should supply for these file attributes.

Note: The CONTROL operand of PRMOD is not applicable to CEEYPR0.

CTLCHR

CTLCHR specifies whether or not the first character of each record written to this printer file contains a print control character.

Syntax

ASA

Specifies that the first character of each record written to this printer file is an American National Standard print character. Specify CTLCHR=ASA if the file you want to process is either:

- Defined in a VS COBOL II or COBOL/VSE program as a non-EXTERNAL file, and written using any of the following language constructs to control printing:
  - The LINAGE clause of the FD entry
  - The AFTER ADVANCING phrase of the WRITE statement
  - The BEFORE ADVANCING phrase of the WRITE statement for a file defined with the CODE-SET clause of the FD entry
- Declared in a PL/I VSE program and declared with the CTLASA ENVIRONMENT option to control printing

YES

Specifies that the first character of each record written to this printer file is a System/370 print control character. Specify CTLCHR=YES if the file you want to process is either:

- Defined in a VS COBOL II or COBOL/VSE program and written using any of the following language constructs to control printing:
  - For an EXTERNAL file:
    - The LINAGE clause of the FD entry
    - The AFTER ADVANCING phrase of the WRITE statement
    - The BEFORE ADVANCING phrase of the WRITE statement
  - For a non-EXTERNAL file, the BEFORE ADVANCING phrase of the WRITE statement for a file defined without the CODE-SET clause of the FD entry
- Declared in a PL/I VSE program and declared with the CTL360 ENVIRONMENT option to control printing

Default

If you do not specify this file attribute, the IBM-supplied default is that records written to this file do not contain print control characters.

DEVICE

DEVICE specifies the device code of the IBM device the generated logic module supports.

Syntax

1403
Generates code for an IBM 1403 printer.

3203
Generates code for an IBM 3203 printer.

3525
Generates code for an IBM 3525 multifunction card unit.

IOAREA2

IOAREA2 specifies whether or not a second I/O area is used for this file.

Syntax

YES
Specifies that two I/O areas are used for this file. Specify IOAREA2=YES if the file you want to process is either:

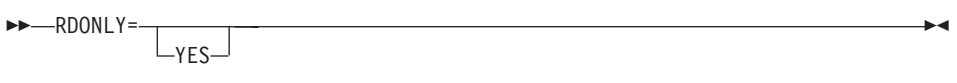
- Defined in a VS COBOL II or COBOL/VSE program, and defined without the RESERVE 1 AREA clause of the FILE-CONTROL paragraph
- Declared in a PL/I VSE program, and declared without the BUFFERS(1) characteristic.

Default
If you do not specify this file attribute, the IBM-supplied default is that one I/O area is used for this file.

RDONLY

RDONLY specifies whether or not the generated logic module is a read-only module.

Syntax



YES
Specifies that the generated logic module is read-only. Specify RDONLY=YES for all LE/VSE-conforming HLL applications.

Default
If you do not specify this file attribute, the IBM-supplied default is that the generated logic module is not read-only.

Usage Notes

If all the logic modules in the LIOCS phase CEEYPR0 are generated with the RDONLY=YES attribute, the phase is eligible for inclusion in the SVA.

RECFORM

RECFORM specifies the record format of files assigned to this device.

Syntax

FIXUNB
Specifies a record format of fixed length, unblocked.

UNDEF
Specifies a record format of undefined.

VARUNB
Specifies a record format of variable length, unblocked.

STLIST

STLIST specifies whether or not the IBM 1403 selective tape listing feature is used.

Syntax

YES
Generates a logic module that contains code to support the IBM 1403 selective tape feature.

Default
If you do not specify this file attribute, the IBM-supplied default is that the generated logic module does not contain code to support the IBM 1403 selective tape feature.

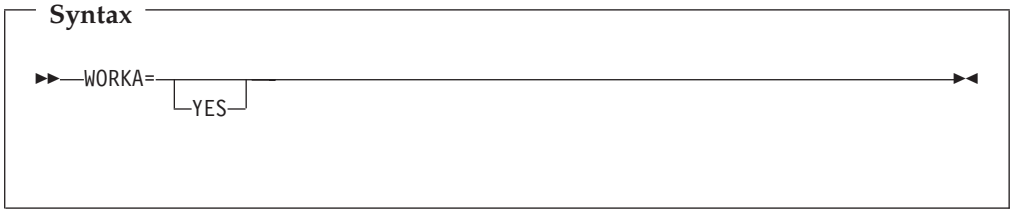
Usage Notes

- Do not specify this file attribute for LE/VSE-conforming HLL applications.
- This file attribute should only be specified if DEVICE=1403 is specified.
- If you specify this file attribute, you must also specify RECFORM=FIXUNB.
- If you specify this file attribute, the CTLCHR file attribute is not valid and is ignored if specified.

WORKA

WORKA specifies whether or not records are processed in work areas instead of I/O areas.

Syntax



YES
Specifies that records are processed in work areas instead of I/O areas.

Default
If you do not specify this file attribute, the IBM-supplied default is that records are processed in I/O areas.

Appendix C. Customizing LE/VSE User Exits

IBM offers a default version of the CEEBXITA assembler user exit that you can customize during your LE/VSE installation and use on a global or installation-wide basis. After installation, you can again customize CEEBXITA and link it directly to applications to use on an application-specific basis.

IBM also provides an HLL user exit, CEEBINT, that you can modify and use after installation. The HLL user exit is used during enclave initialization. LE/VSE supplies an IBM-supplied default HLL user exit, or you can code one in C, PL/I, or LE/VSE-conforming assembler language. You cannot write one in COBOL.

After the enclave has been established, the HLL user exit is invoked and passed a parameter list that conforms to the LE/VSE definition. The parameter list is described in *LE/VSE Programming Guide*.

You can use the sample assembler user exit programs distributed with LE/VSE to modify the code for the requirements of your application. Choose a sample program appropriate for your application. The following assembler user exit programs are delivered with LE/VSE:

Table 33. Sample Assembler User Exits for LE/VSE

Example User Exit Member Name	Operating Environment	Language (if Language-Specific)
CEEBXITA.A	VSE (default)	
CEECXITA.A	CICS (default)	
CEEBX05A.A	VSE	VS COBOL II compatibility

Note:

1. If LE/VSE is installed at your site without modification, then CEEBXITA and CEECXITA are the defaults on your system for VSE and CICS, respectively.

If LE/VSE is installed in the default sublibraries, you can find the source code for CEEBXITA, CEECXITA, and CEEBX05A in the PRD2.SCEEBASE sublibrary.

The assembler user exit CEEBXITA performs functions for enclave initialization, normal and abnormal enclave termination, and process termination. CEEBXITA must be written in assembler language, because an HLL environment might not be established when the exit is invoked.

You can set up user exits for tasks such as:
- Installation accounting and charge back
- Installation audit controls
- Programming standard enforcement
- Common application run-time support

shows the timing of the invocations of the user exits at initialization and termination processing.

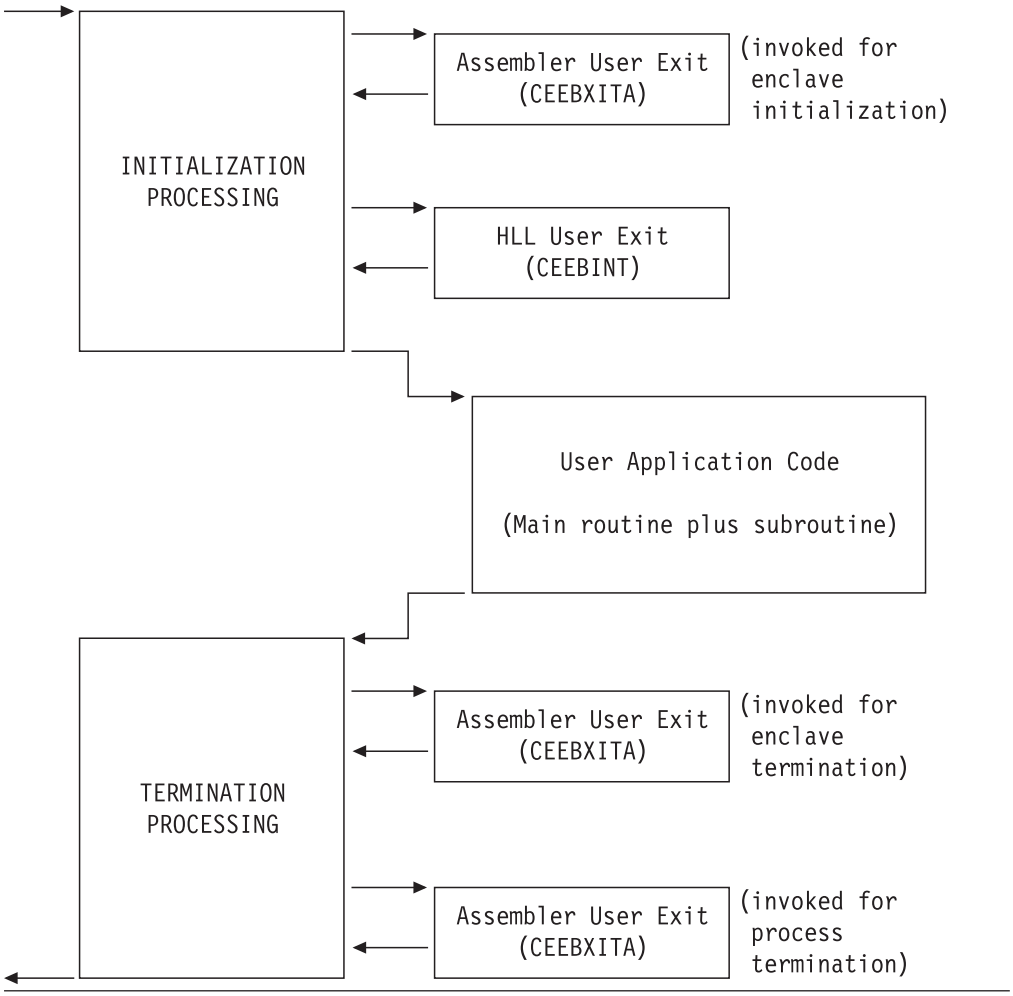


Figure 16. Location of User Exits

In Figure 16, run-time user exits are invoked in the following sequence:

1. Assembler user exit is invoked for enclave initialization
2. Environment is established
3. HLL user exit is invoked
4. Main routine is invoked
5. Main routine returns control to caller
6. Assembler user exit is invoked for termination of the enclave

   CEEBXITA is invoked for enclave termination processing after all application code in the enclave has completed, but prior to any enclave termination activity.
7. Environment is terminated
8. Assembler user exit is invoked for termination of the process

   CEEBXITA is invoked again when the LE/VSE process terminates.

LE/VSE provides the CEEBXITA assembler user exit for termination but does not provide a corresponding HLL termination user exit.

CEEBXITA behaves differently, depending upon when it is invoked, as described in the following sections.

CEEBXITA Behavior During Enclave Initialization

The CEEBXITA assembler user exit is invoked before enclave initialization is performed. You can use CEEBXITA to help establish your application run time environment. For example, in the assembler user exit you can specify the stack and heap run-time options. You can also use the user exit to interrogate program parameters supplied in the JCL and change them if you want. In addition, you can specify run-time options in the user exit by using the CEEAUE_OPTION field of the assembler interface.

CEEBXITA Behavior During Enclave Termination

The CEEBXITA assembler exit is invoked after the user code for the enclave has completed, but prior to the occurrence of any enclave termination activity. In other words, the assembler user exit for termination is invoked when the environment is still active. For example, CEEBXITA is invoked before the storage report is produced (if you requested one), files are closed, and the debug tool is invoked for enclave termination.

The assembler user exits permit you to request an abend. You can also request a dump to assist in problem diagnosis. Because termination activities have not yet begun when the user exit is invoked, the majority of storage has not been modified when the dump is produced.

You can request the abend and dump in the assembler user exit for all enclave-terminating events including:

- The situation that occurs in PL/I when the ON condition (including ERROR or FINISH) is raised and one of the following conditions is true:
  The program does not have an appropriate ON-unit.
  The ON-unit does not terminate with a GOTO.
  The GOTO is not allowed.

  This rule applies only to the conditions that cause termination of the program.
- Return from the main routine
- A debug tool QUIT command
- An HLL stop statement such as:
  C `exit()`
  COBOL STOP RUN
  PL/I STOP or EXIT
- An unhandled condition of severity 2 or above

CEEBXITA Behavior During Process Termination

The CEEBXITA assembler exit is invoked after:
All enclaves have terminated
The enclave resources have been relinquished
Any LE/VSE-managed files have been closed
The debug tool has terminated

At this time you can free allocated files and request an abend.

During termination, CEEBXITA can interrogate the LE/VSE reason and return codes and, if necessary, request an abend with or without a dump. This can be done at either enclave or process termination.

Specifying Abnormal Conditions to Be Exempted from Condition Handling

The assembler user exit, when invoked for initialization in the batch environment, can return a list of VSE cancel codes, program-interruption codes, and user abend codes (contained in the CEEAUE_CODES field of the assembler user exit interface—see "CEEBXITA Assembler User Exit Interface" on page 141) that are to be exempted from LE/VSE condition handling.

When an abend or program interrupt occurs in your application, and TRAP(ON) is in effect, and the VSE cancel code, program-interruption code, or user abend code is in the CEEAUE_CODES list, LE/VSE produces an abnormal termination message and issues an abend to terminate the enclave. Normal LE/VSE condition handling is never invoked to handle these conditions. This feature is useful when you do not want LE/VSE condition handling to intervene for certain abends, and when you want to produce a system dump.

When TRAP(OFF) is specified and there is a program interrupt, the user exit for termination is not driven.

Actions Taken for Errors That Occur within the Assembler User Exit

If any errors occur during the enclave initialization user exit, the standard system action occurs because LE/VSE condition handling has not yet been established.

Any errors occurring during the enclave termination user exit lead to abnormal termination (through an abend) of the LE/VSE environment.

If there is a program check during the enclave termination user exit and TRAP(ON) is in effect, the application ends abnormally with message CEE3322C and user abend code 4094 and reason code 44. If there is a program check during the enclave termination user exit and TRAP(OFF) has been specified, the application ends abnormally without additional error checking support. LE/VSE performs no condition handling; error handling is performed by the operating system.

If there is a program check during the process termination user exit, the application ends abnormally without additional error checking support, regardless of the setting of the TRAP run-time option. LE/VSE performs no condition handling; error handling is performed by the operating system.

CEEBXITA Assembler User Exit Interface

You can modify CEEBXITA to perform any function you need, but the exit must have the following attributes after you modify it at installation:

- The user-supplied exit must be named CEEBXITA.
- The exit must be reentrant.
- The exit must be capable of executing in AMODE(ANY) and RMODE(ANY).

If a user exit is modified, you are responsible for conforming to the interface shown in Figure 17. Note that this user exit **must** be written in assembler.

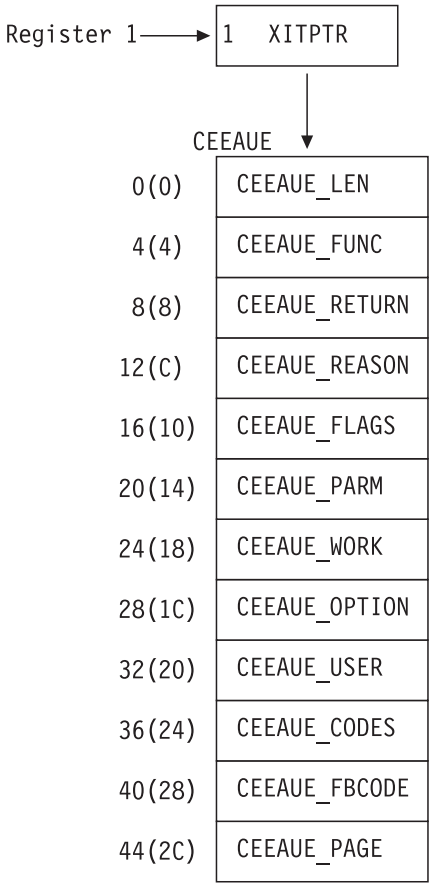


Figure 17. Interface for CEEBXITA Assembler User Exit

When the user exit is called, register 1 points to a word that contains the address of the CEEAUE control block. The high-order bit is on.

The CEEAUE control block contains the following fullwords:

CEEAUE_LEN (input parameter)
A fullword integer that specifies the total length of this control block. For LE/VSE, the length is 48 bytes.

CEEAUE_FUNC (input parameter)
A fullword integer that specifies the function code. LE/VSE supports the following function codes:

1 Initialization of the first enclave within a process.
2 Termination of the first enclave within a process.

3 Nested enclave initialization.
4 Nested enclave termination.
5 Process termination.

The user exit should ignore function codes other than those numbered from 1 through 5.

CEEAUE_RETURN (input/output parameter)
A fullword integer that specifies the return or abend code. CEEAUE_RETURN has different meanings, depending on whether it is an input parameter or an output parameter:

- As an input parameter, CEEAUE_RETURN is the enclave return code.
- As an output parameter, CEEAUE_RETURN has different meanings, depending on the flag CEEAUE_ABND (see below):
  - If the flag CEEAUE_ABND is off, CEEAUE_RETURN is interpreted as the LE/VSE return code placed in register 15.
  - If the flag CEEAUE_ABND is on, CEEAUE_RETURN is interpreted as an abend code used when an abend is issued. (In batch, run-time message CEE3322C is produced and an operating system request is issued to terminate the enclave; in CICS, an EXEC CICS ABEND is issued.)

CEEAUE_REASON (input/output parameter)
A fullword integer that specifies the reason code for CEEAUE_RETURN. CEEAUE_REASON has different meanings, depending on whether it is an input parameter or an output parameter:

- As an input parameter, CEEAUE_REASON is the LE/VSE return code modifier.
- As an output parameter, CEEAUE_REASON has different meanings, depending on the flag CEEAUE_ABND (see below):
  - If the flag CEEAUE_ABND is off, CEEAUE_REASON is interpreted as the LE/VSE return code modifier placed in register 0.
  - If the flag CEEAUE_ABND is on, CEEAUE_REASON is interpreted as an abend reason code used when an abend is issued. (CEEAUE_REASON is used in the batch abnormal-termination run-time message CEE3322C, but is ignored in the CICS environments when an EXEC CICS ABEND is issued.)

CEEAUE_FLAGS
Contains four 1-byte flags. CEEBXITA uses only the first byte but reserves the remaining flags. All unspecified bits and bytes must be 0. The layout of these flags is shown in Figure 18 on page 143:

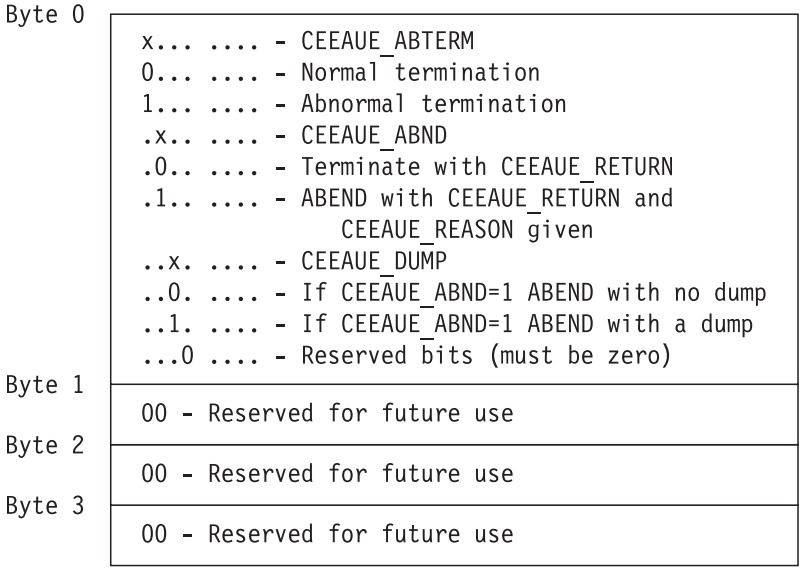


Figure 18. CEEAUE_FLAGS Format

Byte 0 (CEEAUE_FLAG1) has the following meaning:

CEEAUE_ABTERM (input parameter)

OFF Indicates that the enclave is terminating normally (severity 0 or 1 condition).

ON Indicates that the enclave is terminating with an LE/VSE return code modifier of 2 or greater. This could, for example, indicate that a severity 2 or greater condition was raised but not handled.

CEEAUE_ABND (input/output parameter)

OFF Indicates that the enclave should terminate without an abend being issued. Thus, CEEAUE_RETURN and CEEAUE_REASON are placed into register 15 and register 0 respectively and returned to the enclave creator.

ON Indicates that the enclave terminates with an abend. Thus, CEEAUE_RETURN and CEEAUE_REASON are used by LE/VSE in the invocation of the abend. When running in the batch environment, run-time message CEE3322C is produced and an operating system request is issued to terminate the enclave. When running under CICS, an EXEC CICS ABEND command is issued using the abend code contained in CEEAUE_RETURN. CEEAUE_REASON is ignored under CICS.

The TRAP run-time option does not affect the setting of CEEAUE_ABND.

When the ABTERMENC(ABEND) run-time option is specified, the enclave always terminates with an abend when there is an unhandled condition of severity 2 or greater, regardless of the setting of the CEEAUE_ABND flag. However, if you want a system dump to be produced when the enclave terminates with an abend, you must set CEEAUE_ABND and CEEAUE_DUMP to ON. See “ABTERMENC” on page 71 for a detailed explanation of how the CEEAUE_ABND parameter can affect the behavior of the ABTERMENC run-time option.

CEEAUE_DUMP (output parameter)

OFF Indicates that when you request an abend, by setting CEEAUE_ABND to ON, an abend is issued without requesting a dump.

ON Indicates that when you request an abend by setting CEEAUE_ABND to ON, an abend requesting a dump is issued. You must also specify the VSE DUMP option if you want a system dump to be produced when you request an abend.

CEEAUE_PARM (input/output parameter)
A fullword pointer to the parameter address list of the application program.

As an input parameter, CEEAUE_PARM contains the register 1 value passed to the main routine. The exit can modify this value, and the value is then passed to the main routine. If run-time options are present in the PARM parameter of the JCL EXEC statement, they are stripped off before the exit is called.

If the parameter inbound to the main routine is a character string, CEEAUE_PARM contains the address of a fullword address that points to a halfword prefixed string. The halfword prefix contains the length of the string. If no program arguments are specified in the PARM parameter of the JCL EXEC statement, the halfword prefix contains zero.

If this string is altered by the user exit, the string must not be extended in place. Before the string is extended, it must be copied to an area of user storage large enough for the extended string.

CEEAUE_WORK (input parameter)
A fullword pointer to a 256-byte work area that the exit can use. On entry it contains binary zeros and is doubleword-aligned.

This area does not persist across exits.

CEEAUE_OPTION (output parameter)
Upon return, CEEAUE_OPTION contains a fullword pointer to the address of a halfword-length prefixed character string that contains run-time options. These options are honored only during the initialization of the first enclave. When invoked for enclave termination, CEEAUE_OPTION is ignored.

These run-time options override all other sources of run-time options except those that are specified as NONOVR in the installation default run-time options.

Under CICS, the STACK run-time option cannot be modified with the assembler user exit.

CEEAUE_USER (input/output parameter)
A fullword whose value is maintained without alteration and passed to every user exit. Upon entry to the enclave initialization user exit, it is zero. Thereafter, the value of the user word is not altered by LE/VSE or any member libraries. The user exit might change the value of CEEAUE_USER, and LE/VSE maintains that value. This allows the initialization user exit to acquire and initialize a work area, save its address in CEEAUE_USER, and pass the work area address to subsequent user exits. The work area might be freed by the termination user exit.

CEEAUE_CODES (output parameter)
During the initialization exit, CEEAUE_CODES contains the fullword address of a table of VSE cancel codes, program-interruption codes, and user-abend codes that the LE/VSE condition handler exempts from normal condition handling. Therefore, the application is not given the opportunity to field the abend. The table consists of:

- A fullword count of the number of cancel codes, program-interruption codes, and abend codes that are to be exempted from LE/VSE condition handling, and passed to the operating system.
- A fullword for each of the particular cancel codes, program-interruption codes, or abend codes that are to be exempted from LE/VSE condition handling, and passed to the operating system.
  - User abend codes are specified as F'uuuu'. For example, if you want user abend 7777 to be exempted from LE/VSE condition handling, code F'7777'.
  - VSE cancel codes are specified as X'000000cc'. Avoid specifying the value X'00000020', which indicates a program check has occurred. If you specify the value X'00000020', LE/VSE ignores it, and normal LE/VSE condition handling semantics take effect. If you want to exempt specific program checks from LE/VSE condition handling, specify the program-interruption codes.
  - Program-interruption codes are specified as X'800000ii'. For example, if you want an operation exception to be exempted from LE/VSE condition handling, code X'80000001'.

This function is not enabled under CICS.

CEEAUE_FBCODE (input parameter)
Contains a fullword address of the condition token with which the enclave terminated. If the enclave terminates normally (that is, not due to a condition), the condition token is zero.

CEEAUE_PAGE (input parameter)
This parameter indicates whether PL/I BASED variables that are allocated storage outside of AREAs are allocated on a 4K-page boundary. You can specify in the field the minimum number of bytes of storage that must be allocated. Your allocation request must be an exact multiple of 4K.

The IBM-supplied default setting for CEEAUE_PAGE is 32768 (32K).

If CEEAUE_PAGE is set to zero, PL/I BASED variables can be placed on other than 4K-page boundaries.

CEEAUE_PAGE is honored only during enclave initialization, that is, when CEEAUE_FUNC is 1 or 3.

Parameter Values in the Assembler User Exit

The parameters described in “CEEBXITA Assembler User Exit Interface” on page 141 contain different values depending on how the user exit is used. Table 34 on page 146 and Table 35 on page 148 describe the possible values for the parameters based on how the assembler user exit is invoked.

Table 34. Parameter Values in the Assembler User Exit (Part 1). The assembler user exit contains these parameter values depending on when it is invoked.

When Invoked	CEEAUE_ LEN	CEEAUE_RETURN	CEEAUE_REASON (See Note 1)	CEEAUE_FLAGS	CEEAUE_PARM
First Enclave within Process Initialization — Entry CEEAUE_FUNC = 1	48	0	0	0	The address of a fullword that points to a string of user parameters prefixed by a halfword length. If no parameters are present, the halfword length contains zero. You can alter the string in a user exit. Upon return, the CEEAUE_PARM is processed and merged as the invocation string.
First Enclave within Process Initialization — Return		0, or abend code if CEEAUE_ABND = 1	0, or reason code for CEEAUE_RETURN if CEEAUE_ABND = 1	See Note 2 on page 147.	The address of a fullword that points to an optionally altered string of user parameters prefixed by a halfword length. If no parameters are present, the halfword length contains zero. Upon return, the CEEAUE_PARM is processed and merged as the invocation string.
First Enclave within Process Termination — Entry CEEAUE_FUNC = 2	48	Return code issued by application that is terminating.	Reason code that accompanies CEEAUE_RETURN.	See Note 3 on page 147.	
First Enclave within Process Termination — Return		If CEEAUE_ABND = 0, the return code placed into register 15 when the enclave terminates. If CEEAUE_ABND = 1, the abend code.	If CEEAUE_ABND = 0, the enclave reason code. If CEEAUE_ABND = 1, the abend reason code.	See Note 2 on page 147.	
Nested Enclave Initialization — Entry CEEAUE_FUNC = 3	48	0	0	0	The address of a fullword that points to a string of user parameters prefixed by a halfword length. If no parameters are present, the halfword length contains zero. You can alter the string in a user exit. Upon return, the CEEAUE_PARM is processed and merged as the invocation string.
Nested Enclave Initialization — Return		0, or if CEEAUE_ABND = 1, the abend code.	0, or if CEEAUE_ABND = 1, reason code for CEEAUE_RETURN.	See Note 2 on page 147.	The address of a fullword that points to an optionally altered string of user parameters prefixed by a halfword length. If no parameters are present, the halfword length contains zero. Upon return, the CEEAUE_PARM is processed and merged as the invocation string.
Nested Enclave Termination — Entry CEEAUE_FUNC = 4	48	Return code issued by enclave that is terminating.	Reason code accompanying CEEAUE_RETURN.	See Note 3 on page 147.	
Nested Enclave Termination — Return		If CEEAUE_ABND = 0, the return code from the enclave. If CEEAUE_ABND = 1, the abend code.	If CEEAUE_ABND = 0, the enclave reason code. If CEEAUE_ABND = 1, the enclave reason code.	See Note 2 on page 147.	
Process Termination — Entry Function Code = 5	48	Return code presented to the invoking system in register 15 that reflects the value returned from the “first enclave within process termination”.	Reason code accompanying CEEAUE_RETURN that is presented to the invoking system in register 0 and reflects the value returned from the “first enclave within process termination”.	See Note 4 on page 147.	

Table 34. Parameter Values in the Assembler User Exit (Part 1) (continued). The assembler user exit contains these parameter values depending on when it is invoked.

When Invoked	CEEAUE_ LEN	CEEAUE_RETURN	CEEAUE_REASON (See Note 1)	CEEAUE_FLAGS	CEEAUE_PARM
Process Termination — Return		If CEEAUE_ABND = 0, return code from the process. If CEEAUE_ABND = 1, the abend code.	If CEEAUE_ABND = 0, the reason code for CEEAUE_RETURN from the process. If CEEAUE_ABND = 1, reason code for the CEEAUE_RETURN abend reason code.	See Note 2.	

Notes:

1. CEEAUE_REASON is ignored under CICS when CEEAUE_ABND = 1.
2. **CEEAUE_FLAGS:**
   CEEAUE_ABND = 1 if an abend is requested, or 0 if the enclave should continue with termination processing
   CEEAUE_DUMP = 1 if the abend should request a dump
3. **CEEAUE_FLAGS:**
   CEEAUE_ABTERM = 1 if the application is terminating with an LE/VSE return code modifier of 2 or greater, or 0 otherwise
   CEEAUE_ABND = 1 if an abend is requested, or 0 if the enclave should continue with termination processing
   CEEAUE_DUMP = 0
4. **CEEAUE_FLAGS:**
   CEEAUE_ABTERM = 1 if the last enclave is terminating abnormally (that is, an LE/VSE return code modifier is 2 or greater). This reflects the value returned from the "first enclave within process termination".
   CEEAUE_ABND = 1 if an abend is requested, or 0 if the enclave should continue with termination processing "first enclave within process termination" (function code 2).
   CEEAUE_DUMP = 0

Table 35. Parameter Values in the Assembler User Exit (Part 2). The assembler user exit contains these parameter values depending on when it is invoked.

When Invoked	CEEAUE_WORK	CEEAUE_OPTION	CEEAUE_USER	CEEAUE_CODES	CEEAUE_FBCODE	CEEAUE_PAGE
First Enclave within Process Initialization — Entry CEEAUE_FUNC = 1	Address of a 256-byte work area of binary zeros.		0		0	Minimum number of storage bytes to be allocated for PL/I BASED variables (default = 32768).
First Enclave within Process Initialization — Return		Pointer to address of a halfword prefixed character string containing run-time options, or 0.	Value of CEEAUE_USER for all subsequent exits.	Pointer to the abend codes table, or 0.		User specified PAGE value. Minimum number of storage bytes to be allocated for PL/I BASED variables (default = 32768).
First Enclave within Process Termination — Entry CEEAUE_FUNC = 2	Address of a 256-byte area of binary zeros.		Return value from previous exit.		Feedback code causing termination.	
First Enclave within Process Termination — Return			The value of CEEAUE_USER for all subsequent exits.			
Nested Enclave Initialization — Entry CEEAUE_FUNC = 3	Address of a 256-byte work area of binary zeros.		Return value from previous exit.		0	Minimum number of storage bytes to be allocated for PL/I BASED variables (default = 32768).
Nested Enclave Initialization — Return		Pointer to fullword address that points to a halfword prefixed length string containing run-time options, or 0.	The value of CEEAUE_USER for all subsequent exits.	Pointer to abend codes table, or 0.		User specified PAGE value. Minimum number of storage bytes to be allocated for PL/I BASED variables (default = 32768).
Nested Enclave Termination — Entry CEEAUE_FUNC = 4	Address of a 256-byte work area of binary zeros.		Return value from previous exit.		Feedback code causing termination.	
Nested Enclave Termination — Return			Value of CEEAUE_USER for all subsequent exits.			
Process Termination — Entry CEEAUE_FUNC = 5	Address of a 256-byte work area of binary zeros.		Return value from previous exit.		Feedback code causing termination.	

Table 35. Parameter Values in the Assembler User Exit (Part 2) (continued). The assembler user exit contains these parameter values depending on when it is invoked.

When Invoked	CEEAUE_WORK	CEEAUE_OPTION	CEEAUE_USER	CEEAUE_ CODES	CEEAUE_ FBCODE	CEEAUE_PAGE
Process Termination — Return			Value of CEEAUE_USER for all subsequent exits.			

Abnormal Termination Exit Syntax

The abnormal termination exits in CEEEXTAN are invoked during the termination of an enclave due to an unhandled condition of severity 2 or greater. An abnormal termination exit is invoked in AMODE(31), with register 12 pointing to the CAA and register 13 pointing to a DSA with a valid NAB.

For more information about creating and using abnormal termination exits, see "Creating a CEEEXTAN Abnormal Termination Exit CSECT" on page 36.

Syntax

```
Abnormal_Termination_Exit (CIBPTR)
```

CIBPTR (INPUT)
A pointer to the condition information block for the current condition.

Usage notes:

1. The abnormal termination exit must be written in assembler. If you write an abnormal termination exit in LE/VSE-enabled assembler, be sure to specify MAIN=NO in the CEENTRY macro.
2. The abnormal termination exit cannot call any HLL programs.
3. The abnormal termination exit cannot create an LE/VSE enclave.
4. The abnormal termination exit can use the following LE/VSE callable services if the feedback code is passed as a parameter:
   - Date and time callable services
   - Dynamic storage callable services
   - Message handling callable services
   - National language support callable services
   - A subset of the general callable services: CEE5DMP, CEE5GRC, CEE5PRM
   - A subset of the condition handling callable services: CEE5GRN, CEEDCOD, CEEGPID, CEEGQDT, CEEITOK, CEENCOD

   In addition, observe the restrictions on the use of system services as described in *LE/VSE Programming Guide*.
5. LE/VSE issues a system-dependent LOAD for one of the names contained in CEEEXTAN. If the load is successful, the abnormal termination exit is invoked.
6. Upon return from the abnormal termination exit, LE/VSE deletes the routine. A return code is not provided, because LE/VSE takes no action (beyond deleting the routine) for a non-zero return code.
7. If LE/VSE intercepts a program check, an abend, or a CEESGL while an abnormal termination exit is in control, LE/VSE issues an ABEND to terminate the enclave with the abend code 4087 reason code 10.
8. Entry conditions into the abnormal termination exit are:

   Register 1
   Has a standard OS parameter list as described above.

   Register 12
   Points to the CAA.

   Register 13
   Points to an LE/VSE DSA with a valid NAB. (You can use it as a standard 18-fullword save area.)

Register 14
Contains the return address.

Register 15
Contains the entry point address.

AMODE
Is 31.

9. Exit conditions from the abnormal termination exit are:

Register 1
Undefined.

Registers 2–13
Are unchanged.

Register 14
Is the return point.

Register 15
For an abnormal termination exit invoked before the LE/VSE dump is generated, can contain a return code of 8 indicating the LE/VSE dump is not to be generated. Otherwise undefined.

AMODE
Is 31.

Appendix D. Using COBOL with LE/VSE

This appendix contains diagnosis, modification, or tuning information.

This appendix provides information for tuning and customizing your LE/VSE COBOL support run-time routines within the LE/VSE environment.

The customization information in this appendix is intended to help you enhance system performance.

Contents of the General COBPACK (IGZCPAC)

Table 36 lists routines you can include in the general COBPACK (IGZCPAC) and briefly describes each to help you determine which to include in your tailored COBPACK.

Table 36 also indicates which routines are included in the IBM-supplied COBPACK.

Table 36. Routines Eligible for Inclusion in General COBPACK (IGZCPAC)

Name	Description	VSE/ CICS	In IBM-Supplied COBPACK?	Link-Edited AMODE	Link-Edited RMODE
IGZCACP	ACCEPT and STOP literal	VSE	Yes	31	ANY
IGZCACS	Alternate collating sequence comparison	Both	Yes	31	ANY
IGZCANE	Alphanumeric editing	Both	Yes	31	ANY
IGZCANF	Format with figurative constant	Both	Yes	31	ANY
IGZCBID	Binary to internal decimal	Both	Yes	31	ANY
IGZCBUG[4]	Used for debugging	Both	No	31	24
IGZCCAL	Call intercept routine	Both	Yes	31	ANY
IGZCCLS	Class test	Both	Yes	31	ANY
IGZCCTL	Interface to debug tool	Both	Yes	31	ANY
IGZCCVB	Numeric conversion	Both	Yes	31	ANY
IGZCDSP	DISPLAY	VSE	Yes	31	ANY
IGZCFDP[3]	Formatted FDUMP	Both	Yes	31	ANY
IGZCFDW	TRUNC floating point to binary conversion	Both	Yes	31	ANY
IGZCFPW	Exponentiates double precision floating-point numbers	Both	Yes	31	ANY
IGZCGDR	Segment refresh	Both	Yes	31	ANY
IGZCHCM	Condition management events handler	Both	Yes	31	ANY
IGZCIDB	Internal decimal to binary	Both	Yes	31	ANY
IGZCINS	INSPECT	Both	Yes	31	ANY
IGZCIN1	INSPECT library	Both	Yes	31	ANY
IGZCIN2	INSPECT library	Both	Yes	31	ANY
IGZCIPS	Initialization for internal program setup	Both	Yes	31	ANY
IGZCIVL	Comparison with figurative constant	Both	Yes	31	ANY

Table 36. Routines Eligible for Inclusion in General COBPACK (IGZCPAC) (continued)

Name	Description	VSE/ CICS	In IBM-Supplied COBPACK?	Link-Edited AMODE	Link-Edited RMODE
IGZCKCL	Kanji class test	Both	Yes	31	ANY
IGZCLDL	Load/delete subroutines	Both	Yes	31	ANY
IGZCLDR[1]	Partition loader (COBLDR)	Both	Yes	31	ANY
IGZCLLM[2]	Load list manager	Both	Yes	31	ANY
IGZCLNC[4]	Linkage manager for IGZBRDGE (dynamic call and cancel)	Both	No	31	24
IGZCLNK[4]	Linkage manager for VS COBOL II and COBOL/VSE (dynamic call and cancel)	Both	No	31	24
IGZCMED	Median function processor	Both	Yes	31	ANY
IGZCMLT[3]	Message table	Both	Yes	31	ANY
IGZCMSG	Message process control routine	Both	Yes	31	ANY
IGZCNMV	NUMVAL/NUMVAL-C function processor	Both	Yes	31	ANY
IGZCONV	Conversion routine for floating point	Both	Yes	31	ANY
IGZCPPL[4]	Linkage manager for procedure-pointers	Both	No	31	24
IGZCPRC[2]	Program cleanup	Both	Yes	31	ANY
IGZCPRS[2]	Program setup	Both	Yes	31	ANY
IGZCRCL[2]	Run unit cleanup	Both	Yes	31	ANY
IGZCREV	Reverse function processor	Both	Yes	31	ANY
IGZCRSU[2]	Run unit setup	Both	Yes	31	ANY
IGZCSCH	Binary search of table	Both	Yes	31	ANY
IGZCSMV	Move right-justified	Both	Yes	31	ANY
IGZCSPA	Printer spacing	VSE	Yes	31	ANY
IGZCSPC	Call by content	Both	Yes	31	ANY
IGZCSPM	Space manager	Both	Yes	31	ANY
IGZCSSN	Separate sign numeric	Both	Yes	31	ANY
IGZCSSR	SSRANGE compile-time option	Both	Yes	31	ANY
IGZCSTA	Statistical routine function processor	Both	Yes	31	ANY
IGZCSTG	STRING	Both	Yes	31	ANY
IGZCULE[4]	User I/O logic error handler	VSE	No	31	24
IGZCUPL	Upper and lowercase function	Both	Yes	31	ANY
IGZCUST	UNSTRING	Both	Yes	31	ANY
IGZCVDP[3]	Variable dump routine 1	Both	Yes	31	ANY
IGZCVIN	VSAM initialization	VSE	Yes	31	ANY
IGZCVLD[2]	Verify loader	Both	Yes	31	ANY
IGZCVMO	Variable length move	Both	Yes	31	ANY
IGZCXDI	Double precision division	Both	Yes	31	ANY
IGZCXFR[4]	I/O declarative transfer	VSE	No	31	24
IGZCXMU	Double precision multiplication	Both	Yes	31	ANY
IGZCXPR	Decimal fixed-point exponentiation	Both	Yes	31	ANY

Table 36. Routines Eligible for Inclusion in General COBPACK (IGZCPAC) (continued)

Name	Description	VSE/ CICS	In IBM-Supplied COBPACK?	Link-Edited AMODE	Link-Edited RMODE
IGZIBPC	Compile Unit Control Table Builder	Both	Yes	31	ANY
IGZICUD	Descibe Compile Unit	Both	Yes	31	ANY

Notes to Routines Eligible for inclusion in General COBPACK (IGZCPAC):

1. Highly recommended for inclusion in a partially loaded COBPACK.
2. Highly recommended for inclusion in the general COBPACK, regardless of whether the location is above or below the 16MB address line.
3. If IGZCFDP is included in the COBPACK, you should also include routines IGZCMLT and IGZCVDP.
4. This routine is not included in the IBM-supplied COBPACK IGZCPAC so that IGZCPAC is RMODE(ANY) and will load above the 16MB line.

Contents of the Environment-Specific COBPACK (IGZCPCO)

Table 37 lists routines you can include in the environment-specific COBPACK (IGZCPCO) and describes each to help you determine which to include in your tailored COBPACK. Table 37 also indicates which routines are included in the IBM-supplied COBPACK.

Table 37. Routines Eligible for Inclusion in the Environment-Specific COBPACK (IGZCPCO)

Name	Description	In IBM- Supplied COBPACK?	Link- Edited AMODE	Link- Edited RMODE
CEEARLU[4]	Anchor lookup	Yes	31	ANY
CEEBLLST[4]	Language list CSECT	Yes	31	ANY
CEEBPIRA[4]	Common initialization	Yes	31	ANY
CEEBPUBT[4]	Common product signature	Yes	31	ANY
CEEBTRM[4]	Common termination	Yes	31	ANY
CEESG005[4]	COBOL signature	Yes	31	ANY
CEEYDBCP[5,6]	DTFCP builder	No	31	ANY
CEEYCP0[5,6]	CPMOD LIOCS module	No	24	24
IGZCBET[4]	Common table CSECT	Yes	31	ANY
IGZECKP[6]	Checkpoint	No	31	24
IGZEDMR[6]	Reusable environment deactivation	No	31	24
IGZEDTE	Date, day, and time of day	Yes	31	ANY
IGZEDTG[5,6]	Get storage for DTF	No	31	ANY
IGZEINI[2,3,4,6]	Environment initialization	No	31	24
IGZEINP[5,6]	Accept input reader	No	31	24
IGZEMSG	Object-time message writer	Yes	31	ANY
IGZENRT	NORES termination	Yes	31	ANY
IGZEOUT[5,6]	Display output writer	No	31	24
IGZEQBL[6]	SAM initialization transmission verbs, error exits	No	31	24
IGZEQOC[6]	SAM OPEN/CLOSE	No	31	24
IGZESCD[5,6]	SORT-CONTROL I/O handling routine	No	31	24

Table 37. Routines Eligible for Inclusion in the Environment-Specific COBPACK (IGZCPCO) (continued)

Name	Description	In IBM- Supplied COBPACK?	Link- Edited AMODE	Link- Edited RMODE
IGZESMG[6]	Sort/Merge interface	No	31	24
IGZETCL[1]	Thread cleanup	Yes	31	ANY
IGZETRM[6]	Environment termination	No	31	24
IGZETSU[1]	Thread setup	Yes	31	ANY
IGZEVAM[6]	VSAM-to-IDCAMS interface	No	31	24
IGZEVEX[6]	VSAM exit routine for SYNAD and LERAD	No	31	24
IGZEVIO[6]	VSAM input/output	No	31	24
IGZEVOC[6]	VSAM OPEN/CLOSE	No	31	24
IGZEVOP[6]	VSAM OPEN interface for variable length records	No	31	24
IGZEVO2[6]	VSAM OPEN	No	31	24

Notes to Routines Eligible for Inclusion in the Environment-Specific COBPACK(IGZCPCO):

1. Highly recommended for inclusion in a COBPACK, regardless of whether it is located above or below the 16MB address line.
2. Must exist outside the VSE ESM COBPACK, even if it also exists in it.
3. Highly recommended for inclusion in a COBPACK if it is located below the 16MB address line.
4. If IGZEINI is included in the COBPACK, the following routines must also be included: CEEARLU, CEEBLLST, CEEBPIRA, CEEBPUBT, CEEBTRM, CEESG005, and IGZCBET.
5. If IGZEINP, IGZEOUT, or IGZESCD is included in the COBPACK, the following routines must also be included: CEEYDBCP, CEEYCP0, and IGZEDTG.
6. This routine is not included in the IBM-supplied COBPACK IGZCPCO so that IGZCPCO is RMODE(ANY) and will load above the 16MB line. This routine is either link-edited RMODE(24) or is exclusively called by RMODE(24) routines.

Contents of the CICS ESM COBPACK (IGZCPCC)

Table 38 lists routines you can include in the CICS ESM COBPACK (IGZCPCC) and describes each to help you determine which to include in your tailored COBPACK. Table 38 also indicates which routines are included in the IBM-supplied COBPACK.

Table 38. Routines Eligible for Inclusion in the CICS ESM COBPACK (IGZCPCC)

Name	Description	In IBM- Supplied COBPACK?	Link- Edited AMODE	Link- Edited RMODE
CEEARLU[3]	Anchor lookup	Yes	31	ANY
DFHEAI[4]	CICS DFHEAI module	Yes	31	ANY
IGZ9DCM	Diagnose calls to DOS/VS COBOL	Yes	31	ANY
IGZ9INI[1,2,3]	Environment initialization	Yes	31	ANY
IGZ9MSG[1]	Object-time message writer	Yes	31	ANY
IGZ9POP	Perform CICS pop	Yes	31	ANY
IGZ9SMG[2]	Sort/merge interface	Yes	31	ANY
IGZ9TCL[1]	Thread cleanup	Yes	31	ANY
IGZ9TRM[1]	Environment termination	Yes	31	ANY
IGZ9TSU[1]	Thread setup	Yes	31	ANY

Notes to Routines Eligible for Inclusion in the CICS ESM COBPACK:

1. Highly recommended for inclusion in the CICS ESM COBPACK.
2. These routines have AMODE (31), RMODE (ANY) in CICS only.
3. If IGZ9INI is included in the COBPACK, you must also include routine CEEARLU.
4. DFHEAI can be removed from the COBPACK only if you remove all of the other routines.

Appendix E. Customizing C Locale Time Information

This section describes the time information options that you can change at installation time for the default C locale. When C initializes its environment, it uses EDC$S370 as its default locale. The only category of EDC$S370 that you can change at installation time is the LC_TOD category. See *LE/VSE C Run-Time Programming Guide* for information on how to create new locales.

The LC_TOD category defines variables that describe time zone difference, time zone name, and Daylight Savings Time (DST) start and end. The LC_TOD variables are used by the mktime and localtime functions for determining local time. The time functions use the time zone difference from the system as the default.

Customizing Locale

After installing LE/VSE and its C-specific component, you can set up your default run-time environment and customize time information for your installation's default locale. You can use the supplied sample job EDCLLOCL.Z to help you make changes to the time zone and Daylight Savings Time parameters listed below. To set these time parameters, make your changes to the EDCLLOCL sample job, and run the modified job stream.

Figure 19 is a hypothetical example. The time zone name in the example is EST. The time zone difference (TZDIFF) is 300 (minutes), which means EST is 5 hours west of Greenwich mean time (5 hours must be added to EST to obtain Greenwich mean time.) If TZDIFF is greater than 1440, the time zone difference from Greenwich mean time is obtained from the system. The Daylight Savings Time (DST) information in the example is:

- DST starts in April, in the second week on Sunday.
- DST begins at 2 AM.
- DST shifts 1 hour.
- DST ends in October, in the second week on Sunday.
- DST ends at 2 AM.
- DST time zone name is EDT.

```
TZDIFF=300,TNAME='EST',
DSTSTM=4,DSTSTW=2,DSTSTD=0,STARTTM=7200,SHIFT=3600,
DSTENM=10,DSTENW=2,DSTEND=0,ENDTM=7200,DSTNAME='EDT',
```

Figure 19. Example Time Zone and Daylight Savings Time Information

Time Information Options Reference

TZDIFF Time zone difference expressed in minutes. This value is added to the local time to obtain Greenwich mean time. If the local time zone is west of the Greenwich Meridian, this value must be positive. If the local time zone is east of the Greenwich Meridian, this value must be negative. If the absolute value given for this field is greater than 720 then the results may not be as expected, except that an absolute value greater than 1440 (the number of minutes in a day) tells the C Library to get the time zone difference from the system.

Note: You should ensure that the value you specify for TZDIFF corresponds with the VSE system time zone in effect, otherwise C date and time functions may produce incorrect results.

TNAME Time zone name such as PST (Pacific Standard Time) specified within quotation marks. The default for this field is a NULL string.

DSTNAME
A Daylight Savings Time zone name such as PDT (Pacific Daylight Time) specified within quotation marks, if available. If DST information is not available, this is set to `NULL`, which is also the IBM-supplied default. This field must have a value if Daylight Savings Time information (as provided by the other fields) is to be taken into account by the `mktime` and `localtime` functions. These functions ignore DST if this field is set to `NULL`.

DSTSTM Month of the year when DST (Daylight Savings Time) comes into effect. This value ranges from 1 through 12 inclusive, with 1 corresponding to January and 12 corresponding to December. If DST is not applicable to a locale, this is set to 0, which is also the IBM-supplied default.

DSTENM Month of the year when Daylight Savings Time ceases to be in effect. Semantics similar to `DSTSTM`.

DSTSTW Week of the month when DST comes into effect. Acceptable values range from -4 to +4. A value of 4 means the fourth week of the month, while a value of -4 means fourth week of the month, counting back from the end of the month. Sunday is considered the start of the week. If DST is not applicable to a locale, `DSTSTW` is set to 0, which is also the IBM-supplied default.

DSTENW Week of the month when DST ceases to be in effect. Semantics similar to `DSTSTW`.

Note: `DSTSTW` and `DSTENW` need not be used. The `DSTSTD` and `DSTEND` fields can specify either day of week or day of month. If day of month is specified, `DSTSTW` and `DSTENW` become redundant.

DSTSTD Dependent on the value of `DSTSTW`. If `DSTSTW` is not equal to 0, this is the day of the week when DST comes into effect. It ranges from 0 through 6 inclusive, with 0 corresponding to Sunday and 6 corresponding to Saturday. If `DSTSTW` equals 0, `DSTSTD` is the day of the month (for the current year) when DST comes into effect. It ranges from 1 to the last day of the month inclusive. The last day of the month is 31 for January, March, May, July, August, October, and December. It is 30 for April, June, September, and November. For February, it is 28 on non-leap years and 29 on leap years. If DST is not applicable to a locale, `DSTSTD` is set to 0, which is also the IBM-supplied default.

DSTEND The day of the week or the day of the month when DST ceases to be in effect. Semantics similar to `DSTSTD`.

STARTTM
Seconds after 12 midnight, local standard time, when DST comes into effect. For example, if DST is to start at 2:00 AM, `STARTTM` is assigned the value 7200; for 12:00 AM (midnight), `STARTTM` is 0; for 1:00 AM, it is 3600.

ENDTM Seconds after 12 midnight, local standard time, when DST ceases to be in effect. Semantics similar to `STARTTM`.

SHIFT DST time shift, expressed in seconds. Default is 3600, for 1 hour.

Appendix F. Routines Eligible for the Shared Virtual Area

This appendix contains diagnosis, modification, or tuning information.

This appendix provides information for loading LE/VSE routines into the shared virtual area (SVA).

The routines (or, more correctly, phases containing routines) listed in Table 39, Table 40 on page 163, and Table 41 on page 165 can be put in the 24-bit SVA or the 31-bit SVA, depending on their RMODE:

- If the RMODE is ANY, the routine can reside in the 24-bit SVA or the 31-bit SVA.
- If the RMODE is 24, the routine can reside only in 24-bit SVA.

The sizes indicated in these tables are approximate.

The specific HLL sections contain both a table of routines eligible for the SVA and a listing of what routines are recommended. You do not need to include recommended routines if they contain functions your installation does not use.

LE/VSE Base Routines

Table 39. LE/VSE Routines Eligible for Inclusion in the SVA

LE/VSE Routine Name	Description	Decimal Size	RMODE
CEEBINIT	Initialization/termination for batch	52,760	24
CEEBLIBM	Library routine retention initialization/ termination	19,968	24
CEEBLRR	Library routine retention interface	920	24
CEEBNATX	Null Abnormal termination exit	2	ANY
CEEBXTAN	BATCH Abnormal termination exit table	274	ANY
CEECCICS	CICS library support routines	49,696	24
CEECOPP	Compiler options parsing program	49,360	ANY
CEECOPT	CICS Installation-wide default runtime options	26,120	ANY
CEECXTAN	CICS Abnormal Termination Exit table	274	ANY
CEEDOPT	BATCH Installation-wide default runtime options	26,120	ANY
CEEKDS	Contains dump services	89,496	ANY
CEELCLE	Contains locale services	10,936	ANY
CEELRRIN	Library routine retention initialization interface	352	ANY
CEELRRTR	Library routine retention termination interface	344	ANY
CEEMENU0	Message file with mixed-case English; messages 000-999	9,456	ANY
CEEMENU2	Message file with mixed-case English; messages 2000-2999	10,384	ANY
CEEMENU3	Message file with mixed-case English; messages 3000-3999	32,344	ANY
CEEMENU4	Message file with mixed-case English; messages 4000-4999	1,032	ANY

Table 39. LE/VSE Routines Eligible for Inclusion in the SVA (continued)

LE/VSE Routine Name	Description	Decimal Size	RMODE
CEEMENU5	Message file with mixed-case English; messages 5000-5999	696	ANY
CEEMJPN0	Message file with Japanese; messages 000-999	9,536	ANY
CEEMJPN2	Message file with Japanese; messages 2000-2999	10,488	ANY
CEEMJPN3	Message file with Japanese; messages 3000-3999	32,720	ANY
CEEMJPN4	Message file with Japanese; messages 4000-4999	1,024	ANY
CEEMJPN5	Message file with Japanese; messages 5000-5999	648	ANY
CEEMMS	Contains messages services	70,416	ANY
CEEMUEN0	Message file with uppercase English; messages 000-999	9,456	ANY
CEEMUEN2	Message file with uppercase English; messages 2000-2999	10,384	ANY
CEEMUEN3	Message file with uppercase English; messages 3000-3999	32,344	ANY
CEEMUEN4	Message file with uppercase English; messages 4000-4999	1,032	ANY
CEEMUEN5	Message file with uppercase English; messages 5000-5999	696	ANY
CEEPIPI	Initialization/termination routines for the LE/VSE pre-initialization facility	66,296	24
CEEPLPKA	Other library routines that can reside above the line	326,216	ANY
CEEPLPKD	VSE system-specific routines	76,112	24
CEEQMATH	Contains math library services	314,760	ANY
CEEYCD0	Card device LIOCS routines	6,856	24
CEEYDTS	Contains date/time library services	49,040	ANY
CEEYPR0	Printer device LIOCS routines	4,528	24
EDCCNEWC	Newcopy CICS Installation-wide default runtime options	10,152	ANY
EDCROPT	Report new CICS Installation-wide default runtime options to MSGFILE	7,960	ANY

LE/VSE COBOL Component Routines

The base and COBOL component routines recommended for inclusion in the SVA are:

CEEBINIT
CEECCICS
CEEEV005
CEEPIPI
CEEPLPKA
CEEPLPKD
CEEQMATH
CEEYDTS
IGZCPAC
IGZCPCO

Table 40. COBOL/VSE Routines Eligible for Inclusion in the SVA. (Assuming COBPACKs as distributed)

COBOL/VSE Routine Name	Description	Decimal Size	RMODE
CEEEV005	COBOL event handler	15,603	ANY
IGZCA2D	DBCS data manipulation	1,992	ANY
IGZCBUG[1]	Used for debugging	1,840	24
IGZCD2A	DBCS data manipulation	1,112	ANY
IGZCLNC[1]	Linkage manager for IGZBRDGE (dynamic call and cancel)	2,712	24
IGZCLNK[1]	Linkage manager for VS COBOL II and COBOL/VSE (dynamic call and cancel)	3,136	24
IGZCMGEN	COBOL (IGZ) messages in English	17,960	ANY
IGZCMGJA	COBOL (IGZ) messages in Japanese	18,376	ANY
IGZCMGUE	COBOL (IGZ) messages in uppercase English	17,960	ANY
IGZCMTUE	COBOL WTO error messages	504	ANY
IGZCPAC[3]	COBPACK	108,728	ANY
IGZCPCC[3]	COBPACK	10,904	ANY
IGZCPCO[3]	COBPACK	6,272	ANY
IGZCPCR[1]	Partition dump routine	3,392	ANY
IGZCPPL[1]	Linkage manager for procedure-pointers	2,192	24
IGZCPSU[1]	Partition setup routine	3,136	ANY
IGZCULE[1]	User I/O logic error handler	1,456	24
IGZCWTO	COBOL write error message	4,496	ANY
IGZCXFR[1]	I/O declarative transfer	2,400	24
IGZECKP[1]	Checkpoint	3,192	24
IGZEDMR[1]	Reusable environment deactivation	568	24
IGZEINI[2]	Environment initialization	12,960	24
IGZEINP[1]	Accept input reader	10,984	24
IGZEOUT[1]	Display output writer	9,664	24
IGZEPCL	COBOL termination	512	24
IGZEPLF	COBOL environment initialization	2,800	24
IGZEQBL[1]	SAM initialization transmission verbs, error exits	7,968	24

Table 40. COBOL/VSE Routines Eligible for Inclusion in the SVA (continued). (Assuming COBPACKs as distributed)

COBOL/VSE Routine Name	Description	Decimal Size	RMODE
IGZEQCD[1]	Routine to build DTFCD	4,056	24
IGZEQC0[1]	IGZCDMOD	2,824	24
IGZEQDU[1]	Routine to build DTFDU	2,096	24
IGZEQD0[1]	IGZDUMOD	2,192	24
IGZEQMT[1]	Routine to build DTFMT	2,696	24
IGZEQOC[1]	SAM OPEN/CLOSE	15,664	24
IGZEQPR[1]	Routine to build DTRPR	2,368	24
IGZEQP0[1]	IGZPRMOD	1,072	24
IGZEQSD[1]	Routine to build DTFSD	3,248	24
IGZERRE	COBOL reusable environment	6,520	ANY
IGZESCD[1]	SORT-CONTROL I/O handling routine	9,816	24
IGZESMG[2]	Sort/Merge interface	21,592	24
IGZETRM[2]	Environment termination	2,096	24
IGZEVAM[1]	VSAM-to-IDCAMS interface	3,248	24
IGZEVEX[1]	VSAM exit module for SYNAD and LERAD	872	24
IGZEVIO[1]	VSAM input/output	12,800	24
IGZEVOC[1]	VSAM OPEN/CLOSE	4,024	24
IGZEVOP[1]	VSAM OPEN interface for variable length records	9,336	24
IGZEVO2[1]	VSAM OPEN	5,856	24
IGZEWTO	COBOL: write error message to operator's console	1,088	ANY
IGZINSH[1]	COBOL DT/VSE Code handler	163,216	ANY
IGZMSGT	COBOL message tables	152	ANY

Notes to COBOL/VSE Routines Eligible for the SVA:

1. This module is not included in an IBM-supplied COBPACK. If you customize your COBPACKs and include this module, do not load it separately in the SVA.
2. A module of this name is included in the IBM-supplied CICS COBPACK (IGZCPCC), but not in any IBM-supplied batch COBPACK. If you load the CICS COBPACK into the SVA do not load this module separately into the SVA.
3. Size is as shipped by IBM. The size will vary based on how you customize it.

LE/VSE PL/I Component Routines

The base and PL/I component routines recommended for inclusion in the SVA are:

CEEBINIT
CEECCICS
CEECOPP
CEEEV010
CEEPIPI
CEEPLPKA
CEEPLPKD
CEEQMATH
CEEYDTS
IBMRLIB1
IBMRSAP

Table 41 lists the PL/I VSE phases eligible for inclusion in the SVA.

Phases named IBM3... are used by Debug Tool for VSE/ESA for debugging PL/I VSE applications. If you do not plan to use Debug Tool for VSE/ESA with LE/VSE PL/I, do not load these phases into the SVA.

Table 41. PL/I VSE Routines Eligible for Inclusion in the SVA

PL/I Routine Name	Description	Decimal Size	RMODE
CEEEV010	PL/I event handler	200,872	ANY
IBMMSGT	Message table	200	ANY
IBMRBCGA	CHAR built-in	368	ANY
IBMRBCIA	INDEX (character strings)	270	ANY
IBMRBCTA	TRANSLATE (character string)	886	ANY
IBMRBCVA	VERIFY (character strings)	278	ANY
IBMRBEIA	Graphic string index	280	ANY
IBMRBGBA	BOOL (bit strings)	1,304	ANY
IBMRBGCA	COMPARE (general bit strings)	392	ANY
IBMRBGIA	INDEX (bit strings)	518	ANY
IBMRBGVA	VERIFY (bit strings)	496	ANY
IBMRBMPA	MPSTR built-in	1,156	ANY
IBMRCCLA	Conversion director (complex strings)	2,020	24
IBMRCCRA	Conversion director (non-complex strings)	1,145	24
IBMRCOMP	Conversion routines vector	15,400	24
IBMRDMPJ	Dump formatter for Japanese	2,216	ANY
IBMRDMPM	Dump formatter for mixed-case U.S. English	2,088	ANY
IBMRDMPU	Dump formatter for uppercase English	2,072	ANY
IBMREOCA	ON-code routine / ON-code calculator	920	ANY
IBMREOLA	ONLOC built-in function	243	ANY
IBMRJDDA	DATETIME built-in	560	ANY
IBMRJDTA	DATE built-in	560	ANY

Table 41. PL/I VSE Routines Eligible for Inclusion in the SVA (continued)

PL/I Routine Name	Description	Decimal Size	RMODE
IBMRJTTA	TIME built-in	560	ANY
IBMRKDMA	Dump bootstrap	384	ANY
IBMRKMRA	Main dump control routine	20	24
IBMRLANA	Language table (mixed-case U.S. English)	1,880	24
IBMRLANN	Language table (Japanese)	1,880	24
IBMRLANU	Language table (uppercase English)	1,880	24
IBMRLIB1	Lib pack (below the line)	35,880	24
IBMRMCTA	ERF/ERFC (extended float)	1,216	24
IBMROCAA	Close routine	2,448	24
IBMROPEA	Open routine (VSAM)	3,816	24
IBMROPZA	Direct output file formatter	528	24
IBMRPDBA	Debugger interface routine	140	24
IBMRPTLA	Transient library level data	8	24
IBMRRAAA	IBMRRAI: regional sequential output	1,160	24
IBMRRABA	REG(1) sequential unbuffered transmitter	1,424	24
IBMRRACA	BSAM LOAD REG(2) buffered F-format transmitter	1,224	24
IBMRRADA	REG(2) SEQ. unbuffered transmitter	1,512	24
IBMRRAEA	REG(3) buffered F-format transmitter	1,128	24
IBMRRAFA	REG(3) sequential unbuffered F-format transmitter	1,336	24
IBMRRAGA	REG(3) buffered U+V-format transmitter	1,008	24
IBMRRAHA	REG(3) sequential unbuffered U+V-format transmitter	1,280	24
IBMRRBAA	BSAM REG(1) buffered F-format transmitter	1,072	24
IBMRRBBA	BSAM REG(1) unbuffered F-format transmitter	1,528	24
IBMRRBEA	BSAM REG(3) buffered U+V-format transmitter	1,112	24
IBMRRBFA	BSAM REG(3) update U+V-format transmitter	1,624	24
IBMRRCAA	BSAM (consecutive) F-format transmitter	1,520	24
IBMRRDAA	REG(1) direct F-format transmitter	1,152	24
IBMRRDBA	REG(2)+(3) direct F-format transmitter	2,104	24
IBMRRDCA	REG(3) direct U-format transmitter	1,960	24
IBMRRDDA	REG(3) direct V+VS-format transmitter	2,136	24
IBMRREAA	Consecutive buffered record I/O error routines	700	24
IBMRRECA	REG+SEQ+T.P. files record I/O error routines	796	24
IBMRREEA	VSAM record I/O error routines	1,030	24
IBMRREFA	Record I/O endfile routine	346	24
IBMRRQAA	SAM F-format transmitter	1,312	24
IBMRRQBA	SAM V-format transmitter	1,440	24
IBMRRQCA	SAM U-format transmitter	1,256	24
IBMRRQEA	Buffered consecutive spanned record format input	1,112	24
IBMRRQFA	Buffered consecutive spanned record format output	424	24
IBMRRQGA	Buffered consecutive record format update	896	24

Table 41. PL/I VSE Routines Eligible for Inclusion in the SVA (continued)

PL/I Routine Name	Description	Decimal Size	RMODE
IBMRRVAA	ESDS transmitter	1,894	24
IBMRRVGA	KSDS sequential output	1,100	24
IBMRRVHA	KSDS or PATH input/update/direct	2,638	24
IBMRRVIA	VSAM RRDS	2,182	24
IBMRRVJA	VSAM VRDS	2,250	24
IBMRSAP	CICS bootstrap	5,176	ANY
IBMRSOFA	Output file transmitter (F-format)	488	24
IBMRSOUA	Output file transmitter (U-format)	448	24
IBMRSOVA	Output file transmitter (V-format)	560	24
IBMRSTFA	Print file transmitter (F-record)	608	24
IBMRSTIA	Input file transmitter	456	24
IBMRSTUA	Print file transmitter (U-record)	640	24
IBMRSTVA	Print file transmitter (V-record)	640	24
IBM3ABF	Arith BIF evaluator abstraction	2,392	ANY
IBM3AMI	Add message inserts	1,144	ANY
IBM3ANL	Analyze execution abstraction	1,264	ANY
IBM3ANX	API expression analysis	1,936	ANY
IBM3ASD	Perform data assignment	1,352	ANY
IBM3ASK	Check assign compatibility	1,104	ANY
IBM3ASN	Assignment processor	1,944	ANY
IBM3BGE	Bit/String routine Xtrns	320	ANY
IBM3BIF	BIF evaluation abstraction	15,136	ANY
IBM3BLK	Block abstraction	5,176	ANY
IBM3BRN	Build root node information	792	ANY
IBM3CCD	Conversion director	4,176	ANY
IBM3CMD	Command controller	5,576	ANY
IBM3COG	Cleanup on GO/GOTO	1,096	ANY
IBM3CSV	Clear session variable	648	ANY
IBM3CUS	CU abstraction	7,600	ANY
IBM3DCD	Describe condition	7,824	ANY
IBM3DCL	Declare processing	14,008	ANY
IBM3DCU	Describe CU	3,208	ANY
IBM3DED	DED data abstraction	1,304	ANY
IBM3DEV	Describe environment	1,224	ANY
IBM3DLL	Describe list location	704	ANY
IBM3DPA	Describe paths	1,256	ANY
IBM3DSF	Stack frame abstraction	1,000	ANY
IBM3DSL	Describe symbol location	1,432	ANY
IBM3DST	Describe statements	1,024	ANY
IBM3DTP	Converts DT tree to PL/I tree	2,280	ANY

Table 41. PL/I VSE Routines Eligible for Inclusion in the SVA (continued)

PL/I Routine Name	Description	Decimal Size	RMODE
IBM3EAB	Array BIF evaluator	3,464	ANY
IBM3ECO	Comparison evaluator	5,776	ANY
IBM3EEX	Exponentiation evaluator	4,824	ANY
IBM3EFB	Fixed binary arithmetic	2,528	ANY
IBM3EFD	Fixed decimal arithmetic	2,552	ANY
IBM3EMC	Equates and messages collector	728	ANY
IBM3ESL	Extract source listing	1,976	ANY
IBM3EVB	Variable BIF evaluator	6,280	ANY
IBM3EVC	Expression value converter	6,144	ANY
IBM3EVX	Evaluate expression	576	ANY
IBM3EXP	Expression evaluation	15,512	ANY
IBM3FER	Free expression internal representation	584	ANY
IBM3FLT	Float arithmetic	2,072	ANY
IBM3FTA	Format tree attributes	5,512	ANY
IBM3FTI	Free storage of type result information	864	ANY
IBM3FTV	Format tree value	3,336	ANY
IBM3GGE	Graphic Xtrns	184	ANY
IBM3GLL	Get list location	560	ANY
IBM3GNLA	General item equivalents	272	ANY
IBM3GNLN	General item equivalents (JPN)	272	ANY
IBM3GNLU	General item equivalents	272	ANY
IBM3GQD	Get qualifying data handle	936	ANY
IBM3GQN	Get qualifying name	680	ANY
IBM3GRW	Get reserved words	536	ANY
IBM3GVH	Get variable name	544	ANY
IBM3ICU	Identify CU	840	ANY
IBM3IDB	Identify and describe blocks	1,440	ANY
IBM3IDC	Interpret declaration	920	ANY
IBM3IDE	Identify entry	928	ANY
IBM3IDO	Provide label name(s) for a given offset	912	ANY
IBM3IDV	Identify version	1,496	ANY
IBM3IEE	Identify expression error	528	ANY
IBM3ILB	Request offset of a label	1,168	ANY
IBM3IMC	Identify module change	544	ANY
IBM3IMH	Identify module handle	544	ANY
IBM3INP	Command input services	3,208	ANY
IBM3IRC	identify referenced CUs	672	ANY
IBM3ISL	Identify source and listing	528	ANY
IBM3IVH	Identify variable handle	1,152	ANY
IBM3IXE	Identify expression error	1,856	ANY

Table 41. PL/I VSE Routines Eligible for Inclusion in the SVA (continued)

PL/I Routine Name	Description	Decimal Size	RMODE
IBM3LEXA	LEXEME equivalents	3,352	ANY
IBM3LEXN	LEXEME equivalents (JPN)	3,344	ANY
IBM3LEXU	LEXEME equivalents	3,352	ANY
IBM3LNK	Linked list abstraction	1,080	ANY
IBM3MBA	Math BIF Xtrns for ATAN routines	360	ANY
IBM3MBE	Math BIF Xtrns for extended float routines	1,488	ANY
IBM3MBF	Math BIF evaluator abstraction	3,832	ANY
IBM3MBL	Math BIF Xtrns for long float routines	1,488	ANY
IBM3MBP	Math power Xtrns	312	ANY
IBM3MBS	Math BIF Xtrns for short float routines	1,488	ANY
IBM3MBX	Fixed mathematical Xtrns	264	ANY
IBM3NRM	Normalize input records	6,624	ANY
IBM3OPE	Operation evaluator	5,656	ANY
IBM3PEX	Perform an assignment	560	ANY
IBM3PRD	Parser director	38,000	ANY
IBM3QAE	Query array element information	2,280	ANY
IBM3QAI	Query array information	704	ANY
IBM3QNS	List names execution	3,296	ANY
IBM3QRT	Query result type	1,592	ANY
IBM3QSE	Query structure element information	1,232	ANY
IBM3QSI	Query structure information	984	ANY
IBM3REP	Reset expression internal representation	520	ANY
IBM3RIR	Reset expression internal representation	528	ANY
IBM3RSV	Register session variable	936	ANY
IBM3SBE	SUBSTR Xtrns	216	ANY
IBM3STR	string abstraction	720	ANY
IBM3STT	Statement number table routine	4,952	ANY
IBM3TEX	Test expression	1,096	ANY
IBM3UTV	Update tree value	528	ANY
IBM3VAR	Access program variables	14,552	ANY
IBM3VAT	Display attributes	1,112	ANY
IBM3VEX	Validate an expression tree	1,808	ANY
IBM3VGT	Validate and prepare for GOTO	832	ANY
IBM3VIA	Variable information	4,320	ANY
IBM3VOCA	PLITEST keyword equivalents	8,304	ANY
IBM3VOCN	PLITEST keyword equivalents (JPN)	8,272	ANY
IBM3VOCU	PLITEST keyword equivalents	8,304	ANY
IBM9LMSA	NLS mixed-case message source	21,968	ANY
IBM9LMSN	NLS Japanese message source	24,176	ANY
IBM9LMSU	NLS uppercase message source	20,112	ANY

Table 41. PL/I VSE Routines Eligible for Inclusion in the SVA (continued)

PL/I Routine Name	Description	Decimal Size	RMODE
IBM9LM2A	NLS mixed-case message	11,456	ANY
IBM9LM2N	NLS Japanese message	12,008	ANY
IBM9LM2U	NLS uppercase English message	10,448	ANY

LE/VSE C Component Routines

The base and C component modules recommended for inclusion in the SVA are:

- CEEBINIT
- CEECCICS
- CEECOPP
- CEEEV003
- CEEPIPI
- CEEPLPKA
- CEEPLPKD
- CEEQMATH
- CEEYDTS
- EDC$LCNM
- EDC$S370
- EDCMSGT
- EDCUCSNM
- EDCZ24

Table 42. C Routines Eligible for Inclusion in the SVA

C Routine Name	Description	Decimal Size	RMODE
CEEEV003[1]	Main C event handler; base library	996,464	ANY
EDC$...[4]	Locales	–[5]	ANY
EDC$LCNM[1]	Locale name table	1,418	ANY
EDC$S370	Default locale	2,856	ANY
EDCGMENU[3]	`genxlt` utility messages - mixed-case English	3,800	ANY
EDCGMJPN[3]	`genxlt` utility messages - Japanese	3,864	ANY
EDCGMUEN[3]	`genxlt` utility messages - uppercase English	3,800	ANY
EDCIMENU[3]	`iconv` utility messages - mixed-case English	4,008	ANY
EDCIMJPN[3]	`iconv` utility messages - Japanese	4,096	ANY
EDCIMUEN[3]	`iconv` utility messages - uppercase English	4,008	ANY
EDCLMENU[3]	`localedef` utility messages - mixed-case English	5,992	ANY
EDCLMJPN[3]	`localedef` utility messages - Japanese	6,048	ANY
EDCLMUEN[3]	`localedef` utility messages - uppercase English	5,992	ANY
EDCMSGT	C/370 message table	192	ANY
EDCNINSP	Interface to Debug Tool	187,112	ANY
EDCPRLK[2]	Prelink utility	340,176	ANY
EDCPVLNK	Turn off COMREG link bit.	160	24
EDCU...[4]	Code page converters	–[5]	ANY
EDCUCSNM[1]	`iconv` codeset converter name table	2,716	ANY
EDCZEMSG	Mixed-case U.S. English messages	24,920	ANY
EDCZJMSG	Japanese messages	27,912	ANY
EDCZUMSG	Uppercase English messages	24,920	ANY
EDCZ24[1]	I/O extensions	811,576	ANY
EDDDMENU[3]	`DSECT` utility messages - mixed-case English	2112	ANY

Table 42. C Routines Eligible for Inclusion in the SVA (continued)

C Routine Name	Description	Decimal Size	RMODE
EDDDMJPN[3]	DSECT utility messages - Japanese	2280	ANY
EDDDMUEN[3]	DSECT utility messages - uppercase English	2112	ANY

Notes to C Routines Eligible for the SVA:

1. Highly recommended for inclusion in the SVA, especially when using Debug Tool/VSE.
2. Highly recommended for inclusion in the SVA if the prelink utility is heavily used.
3. Highly recommended for inclusion in the SVA if the national language resource utilities are heavily used.
4. The default code page converters or locale modules, or customized code page converters or locale modules (the ones applicable for the user's country), should be included in the SVA.
5. Sizes might vary significantly, so check the specific locale or converter you plan to use.

Appendix G. LE/VSE National Language Support Country Codes

The following table contains valid country / region identifiers along with their respective countries:

Table 43. Country / Region Codes

Code	Country / Region	Code	Country / Region
AD	Andorra	AE	United Arab Emirates
AF	Afghanistan	AG	Antigua and Barbuda
AL	Albania	AN	Netherlands Antilles
AO	Angola	AR	Argentina
AT	Austria	AU	Australia
BA	Bosnia/ Herzegovina	BB	Barbados
BD	Bangladesh	BE	Belgium
BF	Burkina Faso (Upper Volta)	BG	Bulgaria
BH	Bahrain	BI	Burundi
BJ	Benin	BM	Bermuda
BN	Brunei Darussalam	BO	Bolivia
BR	Brazil	BS	Bahamas
BU	Burma	BW	Botswana
CA	Canada	CF	Central African Republic
CG	Congo	CH	Switzerland
CI	Ivory Coast	CL	Chile
CM	Cameroon	CN	People's Republic of China
CO	Colombia	CR	Costa Rica
CS	Czechoslovakia	CU	Cuba
CY	Cyprus	CZ	Czech Republic
DE	Germany	DK	Denmark
DO	Dominican Republic	DZ	Algeria
EC	Ecuador	EE	Estonia
EG	Egypt	ES	Spain
ET	Ethiopia	FI	Finland
FR	France	GA	Gabon
GB	United Kingdom	GH	Ghana
GM	Gambia	GN	Guinea
GR	Greece	GT	Guatemala
GW	Guinea-Bissau	GY	Guyana
HK	Hong Kong	HN	Honduras
HR	Croatia	HT	Haiti
HU	Hungary	ID	Indonesia
IE	Ireland	IL	Israel
IN	India	IQ	Iraq
IR	Iran	IS	Iceland
IT	Italy	JM	Jamaica
JO	Jordan	JP	Japan
KE	Kenya	KR	Korea, Republic of
KW	Kuwait	KY	Cayman Islands
LB	Lebanon	LC	Saint Lucia
LI	Liechtenstein	LT	Lithuania
LR	Liberia	LK	Sri Lanka
LS	Lesotho	LU	Luxembourg

Table 43. Country / Region Codes (continued)

Code	Country / Region	Code	Country / Region
LV	Latvia	LY	Libya
MA	Morocco	MC	Monaco
MG	Madagascar	MK	Macedonia
ML	Mali	MO	Macau
MR	Mauritania	MT	Malta
MU	Mauritius	MW	Malawi
MX	Mexico	MY	Malaysia
MZ	Mozambique	NA	Namibia
NC	New Caledonia	NG	Nigeria
NE	Niger	NI	Nicaragua
NL	Netherlands	NO	Norway
NZ	New Zealand	OM	Oman
PA	Panama	PE	Peru
PG	Papua New Guinea	PH	Philippines
PK	Pakistan	PL	Poland
PR	Puerto Rico	PT	Portugal
PY	Paraguay	QA	Qatar
RO	Romania	RU	Russian Federation
SA	Saudi Arabia	SC	Seychelles
SD	Sudan	SE	Sweden
SG	Singapore	SI	Slovenia
SK	Slovakia	SL	Sierra Leone
SN	Senegal	SO	Somalia
SR	Surinam	SU	Union of Soviet Socialist Republics
SV	El Salvador	SY	Syria
SZ	Swaziland	TD	Chad
TG	Togo	TH	Thailand
TN	Tunisia	TR	Turkey
TT	Trinidad and Tobago	TW	Taiwan
TZ	Tanzania	UG	Uganda
US	United States	UY	Uruguay
VE	Venezuela	VU	Vanuatu
WS	Western Samoa	YE	Yemen
YU	Yugoslavia	ZA	South Africa
ZM	Zambia	ZR	Zaire
ZW	Zimbabwe		

Notes on the Country Codes:

In other versions of Language Environment, country code CS was previously used for Czechoslovakia. Instead of CS you should use either the Czech Republic country code CZ, or the Slovakia country code SK.

In other versions of Language Environment, country code DE was previously used for the Federal Republic of Germany.

In other versions of Language Environment, country code SU was previously used for the Union of Soviet Socialist Republics. Instead of RU you should use the following country codes for the appropriate country: Estonia, EE; Latvia, LV; Lithuania, LT; Russian Federation, RU, etc.

Appendix H. Program and Service Level Information

This appendix provides a list of the APARs included in LE/VSE 1.4.2.

Notes:

1. All APARs *previous* to those listed below are also included in the LE/VSE 1.4.2 GA code or ISD Level code.
2. Since the APARs listed below are integrated into LE/VSE 1.4.2, they are not visible in the MSHP History File.

Service Updates to the LE/VSE Base

These are the APARs that have been included in the LE/VSE 1.4.2 Base component:

Table 44. APARs against the LE/VSE Base Component

PQ06256*	PQ23918*	PQ47866
PQ06598*	PQ39636*	PQ48405
PQ09262*	PQ42344*	PQ51170

Note: * indicates the APAR has been routed from LE OS/390, and included in the base code for LE/VSE 1.4.2 .

Service Updates to the C Component of LE/VSE

These are the APARs that have been included in the LE/VSE 1.4.2 C component:

Table 45. APARs against the C Component of LE/VSE

PQ45676	PQ45681	PQ47358

Service Updates to the COBOL Component of LE/VSE

These are the APARs that have been included in the LE/VSE 1.4.2 COBOL component:

Table 46. APARs against the COBOL Component of LE/VSE

PQ49487	PQ50085	PQ51144

Service Updates to the PL/I Component of LE/VSE

This APAR has been included in the LE/VSE 1.4.2 PL/1 component:

Table 47. APAR against the PL/I Component of LE/VSE

PQ48684

Index

F

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Readers' Comments — We'd Like to Hear from You

IBM Language Environment for VSE/ESA
Customization Guide
Version 1 Release 4

Publication No. SC33-6682-03

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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

Name _____ Address _____

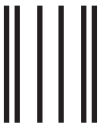
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC33-6682-03

Cut or Fold
Along Line

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



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

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

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

Cut or Fold
Along Line

File Number: S370/S390-34
Program Number: 5686-094

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-6682-03

Spine information:

LE/VSE Customization Guide Version 1 Release 4 SC33-6682-03