



**Program Directory for
IBM Language Environment VM
With US English and Japanese
National Language Support**

Release 08.00

Program Number 5654-030

for Use with
VM/ESA Version 2 Release 3.0

Document Date: March 1998

G110-4676-00

Note!

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

This program directory, dated March 1998, applies to IBM Language Environment VM Release 08.00 (Language Environment or LE), Program Number 5654-030.

A form for reader's comments appears at the back of this publication. When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1991, 1998. All rights reserved.**

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks and Service Marks	ix
1.0 Introduction	1
1.1 Overview of Language Environment	2
1.2 Summary of Changes	2
1.2.1 C Component	2
1.2.2 CEL Component	2
2.0 Program Materials	3
2.1 Basic Machine-Readable Material	3
2.2 Program Publications	3
2.2.1 Displayable Softcopy Publications	3
2.2.2 Basic Unlicensed Publications	3
2.2.3 Optional Program Publications	3
2.3 Microfiche Support	4
2.4 Publications Useful During Installation	4
3.0 Program Support	5
3.1 Preventive Service Planning	5
3.2 Statement of Support Procedures	5
4.0 Program and Service Level Information	7
4.1 Service Level Information for IBM Language Environment VM Release 08.00	7
4.2 Program Level Information for the PL/I Component of IBM Language Environment VM	7
4.3 Program Level Information for the C Component of IBM Language Environment VM	8
4.4 Program Level Information for the CEL Component of IBM Language Environment VM	8
4.5 Program Level Information for the COBOL Component of IBM Language Environment VM	9
4.6 Cumulative Service Tape	9
5.0 Installation Requirements and Considerations	10
5.1 Hardware Requirements	10
5.2 Program Considerations	10
5.2.1 Operating System Requirements	10
5.2.2 Other Program Product Requirements	10
5.2.3 Program Installation / Service Considerations	10
5.3 DASD Storage and USER ID Requirements	11
6.0 Installation Instructions	14
6.1 Language Environment Installation and Customization Overview	14
6.2 Review the Language Environment Default Installation Environment	15
6.3 (Optional) Installing Kanji National Language	15

6.4 Place Language Environment Into Production	16
6.4.1 Copy Language Environment Files Into Production	17
6.4.2 Customizing Language Environment	17
6.4.3 Redefining and Building Saved Segments for Language Environment	18
6.4.4 (Optional) Re-define Saved Segments Using VMSES/E	18
6.4.5 Build the Language Environment Saved Segments	23
7.0 Service Instructions	26
7.1 VMSES/E Service Process Overview	26
7.2 Servicing Language Environment	27
7.2.1 Prepare to Receive Service	27
7.2.2 Receive the Service	29
7.2.3 Apply the Service	29
7.2.4 Update the Build Status Table	30
7.2.5 Build Serviced Objects	32
7.3 Place the New Language Environment Service Into Production	32
7.3.1 Rebuild the Saved Segments	32
7.3.2 Copy the New Language Environment Serviced Files Into Production	33
8.0 Customizing Language Environment	34
8.1 Updating Run-Time Options	34
8.2 Updating User Exit Options	35
8.3 Updating COBOL Component COBPACKs	35
8.4 Updating COBOL Component Reusable Environment	35
8.4.1 Modifying the Behavior of the COBOL Reusable Environment	36
8.5 C Component Locale Time Information	36
8.6 Updating Saved Segments	37
8.7 Installing in Saved Segments	39
Appendix A. Applying a Recommended Service Upgrade (RSU) Tape For Language Environment	40
A.1 Apply Language Environment RSU	41
A.1.1 Prepare Your System for Service Refresh	41
A.1.2 Receive the Preapplied, Prebuilt Service	43
A.1.3 Process Additional Service	44
A.1.4 Build the New Service Level and Place Into Production	46
A.2 Determine RSU Service Level	46
Appendix B. Moving Language Environment to SFS Directories and Creating PPF Override	47
B.1 Setup SFS File Pool and Directories	48
B.2 Copy Minidisk (or SFS) Files to SFS Directories	49
B.3 Create Product Parameter File (PPF) Override	50
Appendix C. Contents of COBPACKs (IGZCPAC/IGZPCO)	52
C.1 Contents of General COBPACK - IGZCPAC	52
C.2 Contents of the Environment-Specific COBPACK (IGZPCO)	56

Appendix D. Segment Build Lists (CEEBSGA/CEEBSGB)	58
D.1 CEEBSGB	58
D.2 CEEBSGA	59
Appendix E. IBM Language Environment VM Run-time Options	61
E.1 Quick Reference Table of Language Environment Run-Time Options	61
E.2 Language Run-Time Option Mapping	65
E.2.1 COBOL Compatibility	71
E.3 Language Environment Run-Time Options	72
E.3.1 ABPERC	72
E.3.1.1 Usage Notes	73
E.3.1.2 For More Information	74
E.3.2 ABTERMENC	74
E.3.2.1 Usage Notes	75
E.3.2.2 For More Information	75
E.3.3 AIXBLD (COBOL Only)	75
E.3.3.1 Usage Notes	76
E.3.3.2 Performance Considerations	76
E.3.3.3 For More Information	76
E.3.4 ALL31	76
E.3.4.1 Usage Notes	77
E.3.4.2 Performance Consideration	78
E.3.4.3 For More Information	78
E.3.5 ANYHEAP	78
E.3.5.1 Usage Notes	79
E.3.5.2 Performance Considerations	79
E.3.5.3 For More Information	79
E.3.6 AUTOTASK NOAUTOTASK (Fortran Only)	80
E.3.7 BELOWHEAP	80
E.3.7.1 Usage Notes	81
E.3.7.2 Performance Considerations	81
E.3.7.3 For More Information	81
E.3.8 CBLOPTS (COBOL Only)	82
E.3.8.1 For More Information	82
E.3.9 CBLPSHPOP (COBOL Only)	83
E.3.9.1 Performance Consideration	83
E.3.9.2 For More Information	83
E.3.10 CBLQDA (COBOL Only)	84
E.3.10.1 Usage Note	84
E.3.11 CHECK (COBOL Only)	84
E.3.11.1 Usage Note	85
E.3.11.2 Performance Consideration	85
E.3.12 COUNTRY	85
E.3.12.1 Usage Notes	86
E.3.12.2 For More Information	86
E.3.13 DEBUG (COBOL Only)	86

E.3.13.1 Usage Note	87
E.3.13.2 Performance Consideration	87
E.3.13.3 For More Information	87
E.3.14 DEPTHCONDLMT	87
E.3.14.1 Usage Notes	88
E.3.14.2 For More Information	89
E.3.15 ENVAR	89
E.3.15.1 Usage Notes	90
E.3.15.2 For More Information	90
E.3.16 ERRCOUNT	90
E.3.16.1 Usage Notes	91
E.3.16.2 For More Information	91
E.3.17 ERRUNIT (Fortran Only)	91
E.3.18 FILEHIST (Fortran Only)	92
E.3.19 HEAP	93
E.3.19.1 Usage Notes	94
E.3.19.2 Performance Considerations	95
E.3.19.3 For More Information	95
E.3.20 INQPCOPN (Fortran Only)	95
E.3.21 INTERRUPT	96
E.3.21.1 Usage Notes	97
E.3.21.2 For More Information	97
E.3.22 LIBSTACK	97
E.3.22.1 Usage Notes	98
E.3.22.2 Performance Considerations	98
E.3.22.3 For More Information	99
E.3.23 MSGFILE	99
E.3.23.1 Usage Notes	100
E.3.23.2 For More Information	101
E.3.24 MSGQ	102
E.3.24.1 Usage Notes	102
E.3.24.2 For More Information	102
E.3.25 NATLANG	102
E.3.25.1 Usage Notes	103
E.3.25.2 For More Information	104
E.3.26 NONIPTSTACK NONONIPTSTACK	104
E.3.26.1 Usage Notes	105
E.3.26.2 For More Information	106
E.3.27 OCSTATUS (Fortran Only)	106
E.3.28 PC (Fortran Only)	107
E.3.29 PLITASKCOUNT (PL/I Only)	107
E.3.29.1 Usage Notes	108
E.3.30 POSIX	108
E.3.30.1 Usage Notes	109
E.3.30.2 For More Information	109
E.3.31 PRTUNIT (Fortran Only)	109

E.3.32 PUNUNIT (Fortran Only)	110
E.3.33 RDRUNIT (Fortran Only)	110
E.3.34 RECPAD (Fortran Only)	111
E.3.34.1 Usage Notes	112
E.3.35 RPTOPTS	112
E.3.35.1 Usage Note	113
E.3.35.2 Performance Considerations	113
E.3.35.3 For More Information	115
E.3.36 RPTSTG	115
E.3.36.1 Usage Notes	116
E.3.36.2 Performance Considerations	116
E.3.36.3 For More Information	118
E.3.37 RTEREUS (COBOL Only)	118
E.3.37.1 Usage Notes	119
E.3.37.2 Performance Considerations	119
E.3.37.3 For More Information	120
E.3.38 SIMVRD (COBOL Only)	120
E.3.38.1 Usage Notes	120
E.3.38.2 For More Information	120
E.3.39 STACK	121
E.3.39.1 Usage Notes	122
E.3.39.2 Performance Considerations	123
E.3.39.3 For More Information	123
E.3.40 STORAGE	123
E.3.40.1 Usage Notes	125
E.3.40.2 Performance Considerations	126
E.3.41 TERMTHDACT	126
E.3.41.1 Usage Notes	128
E.3.41.2 For More Information	128
E.3.42 TEST NOTEST	128
E.3.42.1 Usage Notes	130
E.3.42.2 Performance Consideration	131
E.3.42.3 For More Information	131
E.3.43 THREADHEAP	131
E.3.43.1 Usage Notes	132
E.3.44 TRACE	133
E.3.44.1 Usage Note	134
E.3.44.2 For More Information	134
E.3.45 TRAP	134
E.3.45.1 Usage Notes	135
E.3.45.2 For More Information	137
E.3.46 UPSI (COBOL Only)	137
E.3.46.1 Usage Note	137
E.3.46.2 For More Information	137
E.3.47 USRHDLR NOUSRHDLR	138
E.3.47.1 Usage Notes	138

E.3.47.2 For More Information	139
E.3.48 VCTRSAVE	139
E.3.48.1 Usage Note	139
E.3.48.2 Performance Considerations	139
E.3.49 XUFLOW	140
E.3.49.1 Usage Notes	141
Appendix F. Language Environment National Language Support Country Codes	142
Reader's Comments	145

Figures

1. Optional Material: Program Publications	3
2. Optional Material: Program Publications - Other	4
3. Publications Useful During Installation / Service on VM/ESA Version 2	4
4. PSP UPGRADE and SUBSET ID	5
5. Component IDs	5
6. DASD Storage Requirements for Target Minidisks	13
7. Sample KANJI install console.	16
8. Segment Map panel example.	19
9. SCEE "Change Segment Definition" panel example.	20
10. SCEEX "Change Segment Definition" panel example.	21
11. Segment Map panel with SCEE/SCEEX Segments	22
12. Sample Console output for SCEE Segment Load	24
13. Sample Console output for SCEEX Segment Load	24
14. Customization EXEC - Panel 1	34
15. Component Module Size	39
16. Routines Eligible for Inclusion in General COBPACK (IGZCPAC)	52
17. Routines Eligible for Inclusion in the Environment-Specific COBPACK (IGZPCO)	56
18. Contents of CEEBLSGB Build List	58
19. Contents of CEEBLSGA Build List	59
20. Run-Time Options Quick Reference	61
21. C and Language Environment Options	65
22. COBOL and Language Environment Options	66
23. Fortran and Language Environment Options	68
24. PL/I and Language Environment Options	70
25. Effect of DEPTHCONDLMT(3) on Condition Handling	88
26. Options Report Produced by Language Environment Run-Time Option RPTOPTS(ON)	114
27. Storage Report Produced by Language Environment Run-Time Option RPTSTG(ON)	117
28. TRAP Run-Time Option Settings	134
29. Country Codes	142

Notices

References in this document 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 IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe on any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

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
500 Columbus Avenue
Thornwood, NY 10594

Trademarks and Service Marks

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

IBM
Language Environment
Virtual Machine/Enterprise System Architecture (VM/ESA)
AD/Cycle
System Application Architecture (SAA)
CustomPac
FunctionPac
SystemPac
C/370
IBMLink (ServiceLink)
COBOL/370
Customer Information Control System (CICS)

1.0 Introduction

This program directory is intended for the system programmer responsible for program installation and maintenance. It contains information concerning the material and procedures associated with the installation of Language Environment. You should read all of this program directory before installing the program and then keep it for future reference.

The program directory contains the following sections:

- 2.0, “Program Materials” on page 3 identifies the basic and optional program materials and documentation for Language Environment.
- 3.0, “Program Support” on page 5 describes the IBM support available for Language Environment.
- 4.0, “Program and Service Level Information” on page 7 lists the APARs (program level) and PTFs (service level) incorporated into Language Environment.
- 5.0, “Installation Requirements and Considerations” on page 10 identifies the resources and considerations for installing and using Language Environment.
- 6.0, “Installation Instructions” on page 14 provides detailed installation instructions for Language Environment.
- 7.0, “Service Instructions” on page 26 provides detailed servicing instructions for Language Environment.
- 8.0, “Customizing Language Environment” on page 34 provides instructions on how to customize Language Environment.
- Appendix A, “Applying a Recommended Service Upgrade (RSU) Tape For Language Environment” on page 40 describes how to apply a Recommended Service Upgrade (RSU) to Language Environment.
- Appendix B, “Moving Language Environment to SFS Directories and Creating PPF Override” on page 47 describes how to move Language Environment into SFS and how to create a Product Parameter File (PPF) override.
- Appendix C, “Contents of COBPACKs (IGZCPAC/IGZPCO)” on page 52 lists and describes routines eligible for inclusion in COBPACKs.
- Appendix D, “Segment Build Lists (CEEBSGA/CEEBSGB)” on page 58 describes build lists for Saved Segments for the full IBM Language Environment VM product.
- Appendix E, “IBM Language Environment VM Run-time Options” on page 61 describes Run-time options for IBM Language Environment VM
- Appendix F, “Language Environment National Language Support Country Codes” on page 142 describes IBM Language Environment VM National Language Support Country Codes.

Before installing Language Environment, read 3.1, “Preventive Service Planning” on page 5. This section tells you how to find any updates to the information and procedures in this program directory.

1.1 Overview of Language Environment

Language Environment provides a common set of services in a single run-time environment while enhancing the run-time environment with additional support for emerging application development technologies such as object-oriented, distributed client/server, and open standards.

1.2 Summary of Changes

This section lists the major changes that have been made to the Language Environment product since Release 6.

1.2.1 C Component

- Serviceability enhancements have been added to enhance problem isolation and program source identification.
- Support is added for fetching without loading. This allows a routine to be fetched when previously loaded with an alternative mechanism.
- ISO/ANSI MSE Support for wide character array handling. Wide character functions have been enhanced to add wide character locate compare, copy, move and set functions. This allows processing of arrays of wide characters.
- Support is added for the sharing of C memory files.

1.2.2 CEL Component

- Serviceability enhancements have been added to enhance problem isolation and program source identification. They are:
 - Dump enhancements to assist with finding the problem more quickly, thus increasing productivity. This enhancement adds the assembler module name and module service level to the dump, thus reducing the time needed to identify a failing module.
 - Selected messages enhanced to make the text of the message more meaningful.
 - Specificity of ABEND codes have been improved.
- Storage utilization is improved to include dynamic allocation of control blocks first time they are used.
- One main program will load a second main program and then link to it. This prevents crossing a request block boundary.
- Support is added for fetching without loading. This allows a routine to be fetched when previously loaded with a alternative mechanism.
- Multiple compatibility pre-initialization environments allow multiple environments to be initialized.

2.0 Program Materials

An IBM program is identified by a program number. The program number for Language Environment is 5654-030.

The program announcement material describes the features supported by Language Environment. Ask your IBM marketing representative for this information if you have not already received a copy.

The following sections identify the basic and optional program materials available with this program.

2.1 Basic Machine-Readable Material

IBM Language Environment VM is distributed pre-installed on the VM/ESA Version 2 Release 3.0 System DDR. If necessary, see the *VM/ESA Program Directory* for more information about the distribution media for VM/ESA Version 2 Release 3.0.

2.2 Program Publications

2.2.1 Displayable Softcopy Publications

IBM Language Environment VM Release 8.0 publications are offered in displayable softcopy form on CD-ROM. The files are shipped on CD-ROM in the IBM Online Library VM Collection (SK2T-2067).

These displayable manuals can be used with the BookManager(TM) READ licensed programs in any of the supported environments. Terms and conditions for use of the machine-readable files are shipped with the Online Product Library.

2.2.2 Basic Unlicensed Publications

There are no basic unlicensed publications for IBM Language Environment VM.

2.2.3 Optional Program Publications

Figure 1 identifies the optional unlicensed publications for IBM Language Environment VM. The first copy of the following optional unlicensed program publications are available at no charge to licensers of the basic material for VM/ESA Version 2 Release 3.0 by ordering the 7215 Feature Number. For additional copies, contact your IBM representative.

Figure 1 (Page 1 of 2). Optional Material: Program Publications

Publication Title	Form Number
OS/390 Language Environment for OS/390 & VM Programming Guide	SC28-1939

Figure 1 (Page 2 of 2). Optional Material: Program Publications

Publication Title	Form Number
OS/390 Language Environment for OS/390 & VM Debugging Guide and Run-Time Messages	SC28-1942
OS/390 Language Environment for OS/390 & VM Writing Interlanguage Communication Applications	SC28-1943
OS/390 Language Environment for OS/390 & VM Run-Time Migration Guide	SC28-1944
OS/390 Language Environment for OS/390 & VM Concepts Guide	GC28-1945
OS/390 Language Environment for OS/390 & VM Programming Reference	SC28-1940

You may order the following publications from your IBM representative, through the System Library Subscription Service (SLSS) or by direct order.

Figure 2. Optional Material: Program Publications - Other

Publication Title	Form Number
OS/390 Language Environment for OS/390 & VM Vendor Interfaces (available in softcopy)	LY28-1152
COBOL/370 and COBOL for MVS & VM Compiler and Run-Time Migration Guide	GC26-4764
PLI for MVS & VM Compiler and Run-Time Migration Guide	GC26-3118

2.3 Microfiche Support

There is no microfiche for Language Environment.

2.4 Publications Useful During Installation

The publications listed in Figure 3 may be useful during the installation of Language Environment. To order copies, contact your IBM representative.

Figure 3. Publications Useful During Installation / Service on VM/ESA Version 2

Publication Title	Form Number
VMSES/E Introduction and Reference	GC24-5837
VM/ESA Service Guide	GC24-5838
VM/ESA CP Planning and Administration	SC24-5750
VM/ESA CMS Command Reference	SC24-5776
VM/ESA CMS File Pool Planning, Administration, and Operation	SC24-5751
VM/ESA System Messages and Codes	GC24-5841
VM/ESA: Installation Guide	GC24-5836

3.0 Program Support

This section describes the IBM support available for Language Environment.

3.1 Preventive Service Planning

Before installing Language Environment, check with your IBM Support Center or use IBMLink (ServiceLink) to see whether there is additional Preventive Service Planning (PSP) information. To obtain this information, specify the following **UPGRADE** and **SUBSET** values as shown below.

Figure 4. PSP UPGRADE and SUBSET ID

UPGRADE	SUBSET
VMLE180	See Figure 5 below
VMLE180	yynnRSU
VMLE180	RSU-BY-LVL
VMLE180	RSU-BY-APAR
VMLE180	RSU-BY-PTF

3.2 Statement of Support Procedures

Report any difficulties you have using this program to your IBM Support Center. If an APAR is required, the Support Center will provide the address to which any needed documentation can be sent.

Figure 5 identifies the component ID (COMPID), Retain Release, and PSP SUBSET values for Language Environment.

Figure 5 (Page 1 of 2). Component IDs

COMPID	Retain		PSP SUBSET
	Release	Component Name	
568819801	81N	Language Environment Base	VM/81N
568819801	81G	Language Environment Base - Mixed Case English	VM/81G
568819801	81J	Language Environment Base - Japanese	VM/81J
568819802	82N	Language Environment COBOL - Base	VM/82N
568819802	82G	Language Environment COBOL - Mixed Case English	VM/82G
568819802	82J	Language Environment COBOL - Japanese	VM/82J
568819803	83N	Language Environment PL/I - Base	VM/83N
568819803	83G	Language Environment PL/I - Mixed Case English	VM/83G
568819803	83J	Language Environment PL/I - Japanese	VM/83J

Figure 5 (Page 2 of 2). Component IDs

Retain			PSP SUBSET
COMPID	Release	Component Name	
568819805	81N	Language Environment "C" - Base	VM/81N
568819805	85G	Language Environment "C" - Mixed Case English	VM/85G
568819805	85J	Language Environment "C" - Japanese	VM/85J
568819805	85C	Language Environment "C" - Japanese Msgs	VM/85C

4.0 Program and Service Level Information

This section identifies the program and any relevant service levels of IBM Language Environment VM. The program level refers to the APAR fixes incorporated into the program. The service level refers to the PTFs shipped with this product. Information about the cumulative service tape is also provided.

4.1 Service Level Information for IBM Language Environment VM Release 08.00

The following IBM Language Environment VM Release 08.00 PTFs, against the components of LE, were applied to IBM Language Environment VM Release 08.00 as shipped on the VM/ESA Version 2 Release 3.0 System DDR. These PTFs constitute a service level of 9712.

UQ04381	UQ05817	UQ07746	UQ08689	UQ09273	UQ10412
UQ04618	UQ05829	UQ08169	UQ08729	UQ09282	UQ10424
UQ04621	UQ05939	UQ08182	UQ08786	UQ09325	UQ10457
UQ04803	UQ06197	UQ08186	UQ08893	UQ09350	UQ10460
UQ04810	UQ06206	UQ08337	UQ08965	UQ09499	UQ10509
UQ04821	UQ06210	UQ08342	UQ08969	UQ09553	UQ10591
UQ04825	UQ06364	UQ08354	UQ08979	UQ09554	UQ10605
UQ05021	UQ06618	UQ08360	UQ08989	UQ09555	UQ10730
UQ05023	UQ06776	UQ08362	UQ08993	UQ09868	UQ10749
UQ05029	UQ06788	UQ08375	UQ09038	UQ09970	UQ10787
UQ05049	UQ07047	UQ08393	UQ09048	UQ10033	UQ10806
UQ05070	UQ07052	UQ08426	UQ09114	UQ10054	UQ10807
UQ05078	UQ07148	UQ08427	UQ09121	UQ10069	UQ10926
UQ05083	UQ07158	UQ08505	UQ09128	UQ10072	UQ10953
UQ05165	UQ07162	UQ08512	UQ09130	UQ10074	UQ11052
UQ05167	UQ07430	UQ08537	UQ09192	UQ10093	UQ11384
UQ05231	UQ07550	UQ08565	UQ09247	UQ10103	UQ11408
UQ05325	UQ07732	UQ08569	UQ09261	UQ10108	UQ11435
UQ05575	UQ07733	UQ08659	UQ09270	UQ10257	

4.2 Program Level Information for the PL/I Component of IBM Language Environment VM

The following APAR fixes, for the "PL/I" component of IBM Language Environment VM, against the previous release of IBM Language Environment VM have been incorporated into this release.

PN60271	PN75689	PN77143	PN82912	PN90392
PN74449	PN77102	PN81772	PN85553	PN92581

4.3 Program Level Information for the C Component of IBM Language Environment VM

The following APAR fixes, for the “C” Component of IBM Language Environment VM, against the previous release of IBM Language Environment VM have been incorporated into this release.

PN81203	PN85059	PN88288	PN91289	PN91396	PQ00166
PN81407	PN85540	PN88340	PN91290	PN91400	PQ00167
PN81817	PN85673	PN88377	PN91291	PN91402	PQ00324
PN81886	PN86526	PN88538	PN91292	PN91405	PQ00532
PN82174	PN86617	PN89356	PN91294	PN91490	PQ00702
PN82916	PN86790	PN89381	PN91299	PN91812	PQ00719
PN83962	PN86809	PN90592	PN91300	PN92103	PQ00805
PN84724	PN87803	PN91224	PN91306	PN92303	PQ00928
PN84815	PN87937	PN91280	PN91308	PN92454	PQ01167
PN84863	PN88212	PN91285	PN91365	PN92852	PQ01724
PN85058					

4.4 Program Level Information for the CEL Component of IBM Language Environment VM

The following APAR fixes, for the “CEL” Component of IBM Language Environment VM, against the previous release of IBM Language Environment VM have been incorporated into this release.

PN68187	PN82144	PN85055	PN86850	PN90765	PQ00172
PN69119	PN82427	PN85056	PN86963	PN91284	PQ00414
PN70708	PN82508	PN85057	PN87524	PN91301	PQ00588
PN70395	PN82510	PN85078	PN87646	PN91303	PQ00623
PN71474	PN82806	PN85241	PN87874	PN91305	PQ00696
PN74747	PN83523	PN85304	PN88424	PN91321	PQ00707
PN75354	PN83525	PN85307	PN89393	PN91337	PQ00770
PN76475	PN83771	PN85371	PN89847	PN91576	PQ00794
PN76984	PN83993	PN85668	PN82933	PN91606	PQ00813
PN79307	PN84073	PN85845	PN84164	PN92107	PQ00874
PN80130	PN84209	PN85950	PN85668	PN92349	PQ00996
PN80958	PN84228	PN85976	PN88555	PN92353	PQ01591
PN81688	PN84382	PN86297	PN90478	PN92493	PQ01597
PN82010	PN84398	PN86614	PN90731	PQ00168	PQ01884
PN82012	PN85053	PN86726			

4.5 Program Level Information for the COBOL Component of IBM Language Environment VM

The following APAR fixes, for the “COBOL” Component of IBM Language Environment VM, against the previous release of IBM Language Environment VM have been incorporated into this release.

PN75790	PN82576	PN84072	PN89392	PN91348	PQ00331
PN77057	PN82795	PN84844	PN89517	PN91763	PQ00336
PN80318	PN83385	PN87216	PN90452	PN92661	PQ00372
PN80628	PN83619	PN88745	PN91347	PN92858	PQ00380
PN81799					

4.6 Cumulative Service Tape

Cumulative service for this Language Environment is available through a periodic Recommended Service Upgrade (RSU) tape.

See upgrade bucket VMLE180 subset yynnRSU (where yynn is the service level) for the latest RSU tape available. For the list of PTF's included on the RSU tape, see the service memo from the tape or the PSP subsets sorted by PTF, APAR, or RSU level (LVL).

The RSU can be ordered using the PTF number UQ99180 (APAR PQ12470).

5.0 Installation Requirements and Considerations

The following sections identify the system requirements for installing and activating Language Environment.

5.1 Hardware Requirements

There are no special hardware requirements for Language Environment.

5.2 Program Considerations

The following sections list the programming considerations for installing Language Environment and activating its functions.

5.2.1 Operating System Requirements

IBM Language Environment VM 1.8.0 supports the following VM operating system:

- VM/ESA(R) Version 2 Release 3 Modification 0

5.2.2 Other Program Product Requirements

IBM Language Environment VM requires the following:

- High Level Assembler/MVS & VM & VSE Release 1 or above (5696-234)

5.2.3 Program Installation / Service Considerations

This section describes items that should be considered before you install or service Language Environment.

- A LDRTBLS value of 24 is needed. The PROFILE EXEC for the installation user ID P688198H has this in it.
- When installing into saved segments the 'virtual machine' storage should be at least 1 meg larger than the segment into which you are installing. This size is calculated from the 'END' address of the saved segment. The virtual machine storage is set to 64M for the installation user ID P688198H.
- VMSES/E is required to install and service this product.
- If multiple users install and maintain licensed products on your system, there might be a problem getting the necessary access to MAINT's 51D disk. If you find that there is contention for write access to the 51D disk, you can eliminate it by converting the Software Inventory from minidisk to Shared File System (SFS). See the *VMSES/E Introduction and Reference* manual, section 'Changing the Software Inventory to an SFS Directory', for information on how to make this change.

- You can no longer install and service Language Environment using the MAINT user ID, but use the new user ID “P688198H” which is the IBM suggested user ID name. You are free to change this to any user ID name you wish, however, a PPF override must be created. Refer to B.3, “Create Product Parameter File (PPF) Override” on page 50 and the *VMSES/E Introduction and Reference* manual for information on creating a PPF override.

CAUTION

If you plan to have C, COBOL or PL/I products such as C/370, COBOL/370*, VS COBOL II, OS/VS COBOL or OS PL/I V1 or V2 on the same operating system, you should install the products on separate minidisks to ensure that the library routines for each product maintain their integrity. The IBM Language Environment VM library contains library routines having names identical to those of the other C, COBOL or PL/I library routines.

While you run IBM Language Environment VM you should not have these other products accessed. If you must use a disk that contains these other products while running IBM Language Environment VM then access them **AFTER** you access IBM Language Environment VM.

5.3 DASD Storage and USER ID Requirements

Figure 6 on page 13 lists the User ID and minidisks that are used to install and service Language Environment.

Important Installation Notes:

- The user ID(s) and minidisks shown in Figure 6 on page 13 are already defined when you install VM/ESA 2.3.0 because Language Environment is pre-installed on the VM/ESA 2.3.0 System DDRs.
- If you plan on customizing the “C” Run time options, the “191” disk size must have at least 20 cylinders on a 3390, or other DASD equivalent, of unused (free) space. This is due to the fact that this option will cause a rebuild of the SCEERUN LOADLIB and the “A” disk is used as an interim work disk while building the loadlib.
- P688198H is a default user ID and can be changed. If you choose to change the name of the installation user ID you need to create an override of the Product Parameter File (PPF) to do this. Refer to B.3, “Create Product Parameter File (PPF) Override” on page 50 and the *VMSES/E Introduction and Reference* manual for information on creating a PPF override.
- If you choose to change any of the default minidisks or SFS directories then you will need to create an override of the Product Parameter File (PPF) to do this. Refer to B.3, “Create Product Parameter File (PPF) Override” on page 50 and the *VMSES/E Introduction and Reference* manual for information on creating a PPF override.
- If you choose to move Language Environment from minidisk to Shared File System (SFS), if you did not do it at initial VM/ESA 2.3.0 install time or if you want to change from the default SFS file

pool or directories, then you will need to follow the steps in Appendix B, “Moving Language Environment to SFS Directories and Creating PPF Override” on page 47. These steps will have you create the SFS directories, move the files to the SFS directories and create a PPF override. You can also refer to the *VMSES/E Introduction and Reference* manual for more information on creating a PPF override.

Figure 6. DASD Storage Requirements for Target Minidisks

Minidisk owner (user ID)	Default Address	Storage in Cylinders		FB-512 Blocks	SFS 4K Blocks	Usage
		DASD	CYLS			
P688198H	2B2	9345 3390 3380 3375 3350	80 65 80 125 100	96000	12000	Contains all base code shipped with Language Environment <u>SFS Name:</u> VMSYS:P688198H.LE370.OBJECT
P688198H	2C2	9345 3390 3380 3375 3350	6 5 6 10 8	7200	900	Contains sample files and user local modifications for Language Environment <u>SFS Name:</u> VMSYS:P688198H.LE370.LOCAL
P688198H	2D2	9345 3390 3380 3375 3350	50 42 50 84 50	60000	7500	Contains serviced files <u>SFS Name:</u> VMSYS:P688198H.LE370.DELTA
P688198H	2A6	9345 3390 3380 3375 3350	3 2 3 4 3	3600	450	Contains AUX files and version vector table that represent your test level of Language Environment <u>SFS Name:</u> VMSYS:P688198H.LE370.ALTAPPLY
P688198H	2A2	9345 3390 3380 3375 3350	3 2 3 4 3	3600	450	Contains AUX file and version vector table that represent your production level of Language Environment <u>SFS Name:</u> VMSYS:P688198H.LE370.PRODAPPLY
P688198H	29E	9345 3390 3380 3375 3350	54 45 54 85 68	64800	N/A	Test build disk. If this disk is to be copied to MAINT's 19E disk, make sure the 19E disk is large enough to hold entire contents of 29E disk.
P688198H	191	9345 3390 3380 3375 3350	45 38 45 65 55	42000	5250	P688198H user ID's 191 minidisk NOTE: See Section 5.3 "DASD Storage and User ID Requirements" if customizing "C" runtime options. <u>SFS Name:</u> VMSYS:P688198H.

Note: Cylinder values defined in this table are based on a 4k block size. FB-512 block and SFS values are derived from the 3380 cylinder values in this table.

6.0 Installation Instructions

This chapter describes the installation methods and the step-by-step procedures to complete the installation and activation of Language Environment. Language Environment was pre-installed, using VMSES/E, on the VM/ESA Version 2 Release 3.0 System DDRs. You still **NEED TO** finish the installation and do customization as documented in this chapter.

The step-by-step procedures are in two column format. The steps to be performed are in bold large numbers. Commands for these steps are on the left-hand side of the page in bold print. Any additional information for a command is to the right of the command. For more information about the two column format see 'Understanding Dialogs with the System' in the *VM/ESA Installation Guide*.

Each step of the installation instructions must be followed. Do not skip any step unless directed otherwise.

Throughout these instructions, the use of IBM-supplied default minidisk addresses and user IDs is assumed. If you use different user IDs, minidisk addresses, or SFS directories to install Language Environment, adapt these instructions as needed for your environment.

Note!

The sample console output presented throughout these instructions was produced on a VM/ESA 2.3.0 system and assumes you are installing to minidisks using the default PPF and component names.

6.1 Language Environment Installation and Customization Overview

The following is a brief description of the main steps to complete the installation of Language Environment using VMSES/E.

- Review Your Installation Environment

While resources have been allocated for Language Environment as part of the VM/ESA Version 2 Release 3.0 System DDR that you installed, this default environment should be reviewed and, if necessary, modified for your installation.

- Optionally Install National Languages

The VMFBLD command is used to build the NLS message modules and SCEERUN LOADLIB.

- Placing the Language Environment Files into Production

Once the product files have been tailored and the operation of Language Environment is satisfactory, the product files are copied from the test BUILD disk(s) to production BUILD.

- Customize Language Environment

Use the customization exec to customize runtime options, user exits, COBPACKs, locale time information, saved segments, etc..

- Rebuild the Language Environment Saved Segments
Uses VMFBLD to rebuild the SCEE and SCEEX segments.

For a complete description of all VMSES/E installation options refer to:

- *VMSES/E Introduction and Reference*

6.2 Review the Language Environment Default Installation Environment

IBM Language Environment VM has been installed on the VM/ESA Version 2 Release 3.0 System DDRs using VMSES/E. This means that several of the VMSES/E installation steps have been performed on your behalf. Among these are the:

- addition of the Language Environment installation user ID, P688198H, entry to the VM/ESA 2.3.0 CP system directory
- creation of the P688198H user ID PROFILE EXEC. This PROFILE EXEC contains the accesses to the VMSES/E code disk, MAINT 5E5, and the VMSES/E Software Inventory disk, MAINT 51D. It also sets the LDRTBLS to 24.
- allocation of the Language Environment minidisks
- loading of the Language Environment product files to the minidisks, using VMSES/E commands
- defined SCEE and SCEEX saved segments
- installation of the Language Environment initial RSU.

If you want to change any of the default Language Environment user IDs, minidisks or SFS directories you should do so at this time. In order to do this you will need to create an override to the Product Parameter File (PPF).

If you want to use SFS for Language Environment, because you did not choose it during initial VM/ESA 2.3.0 install, you will have to create your SFS directories, move the code to those directories and reflect these new directories in your PPF override.

Refer to Appendix B, “Moving Language Environment to SFS Directories and Creating PPF Override” on page 47 for instructions on moving Language Environment into SFS and creating a PPF override.

6.3 (Optional) Installing Kanji National Language

If you do not plan on using KANJI then skip this step.

Language Environment's KANJI NLS feature will be installed through VMSES/E support using the user ID, P688198H and the following command:

1 Install KANJI NLS help and message files.

VMFBLD PPF 5688198H LE370KANJI EDCBLHPJ (ALL SETUP)

```
VMFBLD PPF 5688198H LE370KANJI EDCBLHPJ (ALL
VMFBLD2760I VMFBLD processing started
VMFBLD1851I Reading build lists
VMFBLD2182I Identifying new build requirements
VMFBLD2182I No new build requirements identified
VMFBLD1851I (1 of 1) VMFBDCOM processing EDCBLHPJ EXEC
VMFBDC2219I Processing object CMOD.HELPCMS
VMFBDC2219I Processing object CPLINK.HELPCMS
VMFBDC2219I Processing object C370LIB.HELPCMS
VMFBDC2219I Processing object LINKLOAD.HELPCMS
VMFBDC2219I Processing object GENXLT.HELPCMS
VMFBDC2219I Processing object ICONV.HELPCMS
VMFBDC2219I Processing object EDCPMSGKE.MSGS
VMFBDC2219I Processing object EDCUME.TXTKANJI
VMFBLD1851I (1 of 1) VMFBDCOM completed with return code 0
VMFBLD2180I There are 0 build requirements remaining
VMFBLD2760I VMFBLD processing completed successfully
Ready; T=114.45/115.74 11:01:21
```

Figure 7. Sample KANJI install console.

If, in the future, you need to return back to Mixed-Case English help and messages files simply rebuild the required language function messages by issuing the following command:

```
VMFBLD PPF 5688198H LE370 EDCBLHPE (ALL SETUP
```

Note: Selecting a Language Environment national language to be used, other than mixed-case English, is discussed in section 6.4.2, “Customizing Language Environment” on page 17. When you get to that section you can change the language that LE uses, if you need to.

6.4 Place Language Environment Into Production

In this step you will:

- Copy the Language Environment files to the production minidisk
- Customize Language Environment
- Rebuild the Language Environment saved segments.

6.4.1 Copy Language Environment Files Into Production

NOTE!

During initial install of VM/ESA 2.3.0 a Recommended Service Upgrade (RSU) was applied to Language Environment. This means that you **MUST** place the new serviced Language Environment files into production with this step.

- 1 Logon to MAINT if you plan to put Language Environment general use code on the 'Y' disk (MAINT's 19E disk). Or logon to the owner of the disk that will contain the 'production' level of the Language Environment code.

link P688198H 29e 29e rr

access 29e e

access 19e f

vmfcopy * * e = = f2 (prodid 5688198H%LE370 olddate replace

The VMFCOPY command will update the VMSES PARTCAT file on the 19E disk.

6.4.2 Customizing Language Environment

See Chapter 8.0, "Customizing Language Environment" on page 34 for a full description on how to customize Language Environment. After you have finished your customization you need to return back to 6.4.3, "Redefining and Building Saved Segments for Language Environment" on page 18 in order to rebuild the Language Environment saved segments.

When installing IBM Language Environment VM you can choose the national language to be used for things such as system and IBM Language Environment VM text, run-time messages, IBM Language Environment VM reports, and output of such IBM Language Environment VM services as date and time services. Mixed-Case English is the default for the run-time language option NATLANG. If you need a language other than mixed-case English as the default for your system, you can change to uppercase English or Japanese, depending on what national language support language you have installed, when you customize Language Environment. See Appendix E, "IBM Language Environment VM Run-time Options" on page 61 for a description of the NATLANG runtime option.

NOTE!

Whether you did or did not do any customization you **need** to rebuild the saved segments.

6.4.3 Redefining and Building Saved Segments for Language Environment

The Language Environment segments, SCEE and SCEEX, were pre-defined and pre-built on the VM/ESA 2.3.0 System DDRs. They were defined using the segment mapping tool VMFSGMAP and built with VMFBLD.

NOTE!

During initial install of VM/ESA 2.3.0 a Recommended Service Upgrade (RSU) was applied to Language Environment. This means that the segments **MUST** be rebuilt at this time. Optionally you can re-define the segments before you rebuild them.

When you use Saved segments for IBM Language Environment VM remember that the order in loading is:

1. Nucleus extension
2. Saved segments
3. Relocatable modules
4. OS simulation load

Notes:

1. The redefining and building of the Language Environment saved segments should be performed from the installation user ID. If you move any segments that are currently defined on your system you must ensure that they are rebuilt from the user ID that maintains them.
2. Care should be taken that Saved Segments from previous releases of Language Environment are not accessed during installation. If they are, unpredictable results could occur.

For more information on using VMSES/E for saved segments, review the chapter, 'Using VMSES/E to Define, Build, and Manage Saved Segments in the *VM/ESA Planning and Administration* manual.

6.4.4 (Optional) Re-define Saved Segments Using VMSES/E

Note

If you do not need to re-define either Language Environment segment then continue with 6.4.5, "Build the Language Environment Saved Segments" on page 23.

1 Logon to the installation user ID **P688198H**.

2 Establish write access to the VMSES/E and software inventory disks.

**link maint 51D 51D mr
access 51D D**

3 Update Language Environment segment object definitions to the SEGBLIST EXC00000 build list.

vmfsgmap segbld esasegs segblst

This command displays a panel for making segment updates. See Figure 8 for an example of the Segment Map panel that will be displayed.

```

VMFSGMAP - Segment Map
More: +
Lines 1 to nn of nn

Meg          008-MB      009-MB      00A-MB      00B-MB
St Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
DOSBAM      SPA 8.....9.....A.....=====
CMSBAM      MEM 8.....9.....A.....RRRR.....
CMSDOS      MEM 8.....9.....A.....R.....
DOSINST     DCS 8.....R-----A.....B.....
SCEE        DCS 8.....RRRRRRRRRRRRRA.....B.....

Meg          00C-MB      00D-MB      00E-MB      00F-MB
St Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
HELPSEG     DCS RRRRRRRRRRRRRRD.....E.....F.....
M CMS       SYS C.....D.....E.....RRRRRRRRRRRRR>

===== 16-MB Line =====

Meg          010-MB      011-MB      012-MB      013-MB
St Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
M CMS       SYS >RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
M GCS       SYS RNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN2.....3.....

F1=Help    F2=Chk Obj  F3=Exit    F4=Chg Obj  F5=File    F6=Save
F7=Bkwd    F8=Fwd     F9=Retrieve F10=Add Obj F11=Del Obj F12=Cancel
====> _

```

Figure 8. Segment Map panel example.

4 Obtain the Current Language Environment Segment Definition

a Place cursor on the SCEE or SCEEX segment.

Note: You can only change one segment at a time, so if you need to change both segments then you will have to repeat steps 4 and 5 for each segment. The 'Change Segment Definition' panel is shown for each segment but they do not come up at the same time.

b Hit the F4 Chg Obj key

F4

F4 will take you from the Segment Map panel to the Change Segment Definition panel.

See Figure 9 to see the Change Segment Definition panel that will be displayed for the SCEE segment.

See Figure 10 on page 21 to see the Change Segment Definition panel that will be displayed for the SCEEX segment.

```
Change Segment Definition                               Lines 1 to nn of nn

OBJNAME.....: SCEE
DEFPARMS....: 0900-09FF SR
SPACE.....:
TYPE.....: PSEG
OBJDESC....: SCEE SEGMENT BELOW 16 MEG
OBJINFO....:
GT_16MB....: NO
DISKS.....:
SEGREQ.....:
PRODID.....: 5688198H LE370
BLDPARMS...: PPF(5688198H LE370 CEEBLSGB)

F1=Help      F2=Get Obj   F3=Exit      F4=Add Line  F5=Map       F6=Chk MEM
F7=Bkwd      F8=Fwd       F9=Retrieve   F10=Seginfo F11=Adj MEM  F12=Cancel
====>
```

Figure 9. SCEE "Change Segment Definition" panel example.

```

Change Segment Definition                               Lines 1 to nn of nn

OBJNAME....: SCEEX
DEFPARMS...: 1A00-1EFF SR
SPACE.....:
TYPE.....: PSEG
OBJDESC....: SCEEX SEGMENT ABOVE 16 MEG
OBJINFO....:
GT_16MB....: YES
DISKS.....:
SEGREQ.....:
PRODID.....: 5688198H LE370
BLDPARMS...: PPF(5688198H LE370 CEEBLSGA)

F1=Help      F2=Get Obj   F3=Exit      F4=Add Line  F5=Map       F6=Chk MEM
F7=Bkwd      F8=Fwd       F9=Retrieve   F10=Seginfo F11=Adj MEM  F12=Cancel
====>

```

Figure 10. SCEEX "Change Segment Definition" panel example.

5 Update Segment Definition

- a** Change any of the information displayed on the Change Segment Definition Panel.

Note: If you are using a different PPF name than the default then you must change the BLDPARMS field to reflect your PPF name. If you have changed to using SFS then you must update the BLDPARMS field to reflect the SFS component name (default is LE370SFS).

- b** To complete the change hit the F5 Map key.

F5

Confirm information shown on panel is correct and press **F5** to return to the Segment Map panel. See Figure 11 on page 22 for the refreshed Segment Map panel that will be displayed. Since this example did not really change any information the refreshed Segment Map panel shows the original definitions.

```

VMFSGMAP - Segment Map
More: +
Lines 1 to nn of nn

SCEE DCS 8.....RRRRRRRRRRRRRA.....B.....
M DOSINST DCS 8.....R-----A.....B.....

Meg          00C-MB          00D-MB          00E-MB          00F-MB
St Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
M CMS      SYS C.....D.....E.....RRRRRRRRRRRRRRRR
M HELPINST DCS RRRRRRRRRRRRRRRD.....E.....F.....

===== 16-MB Line =====

Meg          010-MB          011-MB          012-MB          013-MB
St Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
M CMS      SYS >RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR

Meg          014-MB          015-MB          016-MB          017-MB
St Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
M CMSVLIB DCS 4.....5.....6.....RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR

Meg          018-MB          019-MB          01A-MB          01B-MB
St Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
SCEEX      DCS 8.....9.....RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR>

Meg          01C-MB          01D-MB          01E-MB          01F-MB
St Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
SCEEX      DCS >RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRF.....

===== End Segment Map =====

F1=Help    F2=Chk Obj  F3=Exit    F4=Chg Obj  F5=File    F6=Save
F7=Bkwd    F8=Fwd     F9=Retrieve F10=Add Obj F11=Del Obj F12=Cancel
====>

```

Figure 11. Segment Map panel with SCEE/SCEEEX Segments

6 Save new information and exit from the Segment Map panel.

F5

Ready; T=nn.nn/nn.nn hh:mm:ss

F5 saves all changed information and exits the map panel.

6.4.5 Build the Language Environment Saved Segments

1 Prepare to build the segments.

a Logon to the installation user ID **P688198H**.

b Make sure that there are no outstanding build requirements for the segments.

vmfbld ppf 5688198H {LE370|LE370SFS} (serviced setup

c IPL CMS to clear the virtual storage

ipl 190 clear parm nosprof instseg no

**** DO NOT press ENTER at the VMREAD!****

access (nosprof

IPL 190 to clear your virtual machine. This command bypasses the execution of the system profile (SYSPROF EXEC) and without loading the installation saved segment (CMSINST).

Bypass the execution of the PROFILE EXEC.

d Access the VMSES/E code

access 5E5 B

e Establish write access the Software Inventory Disk

link maint 51D 51D mr

access 51d D

f Reset Loader Tables back to 24

Set LDRTBLS 24

2 Issue VMFBLD command to build the Language Environment segments.

vmfbld ppf segbld esasegs segblist SCEE (all

vmfbld ppf segbld esasegs segblist SCEEX (all

```

Ready; T=1.39/1.53 05:56:57
vmfblld ppf segblld esasegs segblld scee (all
VMFBLD2760I VMFBLD processing started
VMFBLD1851I Reading build lists
VMFBLD2182I Identifying new build requirements
VMFBLD2182I New build requirements identified
VMFBLD1851I (1 of 1) VMFBDESEG processing SEGBLIST EXC00000 D, target is BUILD
51D (D)
VMFBDS2115I Validating segment SCEE
VMFBDS2002I A DEFSEG command will be issued for 1 segment(s).
VMFBDS2219I Processing object SCEE.SEGMENT
HCPNSS440I Saved segment SCEE was successfully saved in fileid 1129.
VMFBDS2003W The SYSTEM SEGID D(51D) file has been changed and must be moved to
the S disk.
VMFBLD1851I (1 of 1) VMFBDESEG completed with return code 4
VMFBLD2180I There are 1 build requirements remaining
VMFBLD2760I VMFBLD processing completed with warnings
Ready(00004); T=22.05/23.26 05:57:59

```

Figure 12. Sample Console output for SCEE Segment Load

```

Ready; T=0.01/0.01 11:40:56
VMFBLD PPF SEGBLD ESASEGS SEGBLIST SCEEX (ALL
VMFBLD2760I VMFBLD processing started
VMFBLD1851I Reading build lists
VMFBLD2182I Identifying new build requirements
VMFBLD2182I New build requirements identified
VMFBLD1851I (1 of 1) VMFBDESEG processing SEGBLIST EXC00000 D, target is BUILD
51D (D)
VMFBDS2115I Validating segment SCEEX
VMFBDS2002I A DEFSEG command will be issued for 1 segment(s).
VMFBDS2219I Processing object SCEEX.SEGMENT
HCPNSS440I Saved segment SCEEX was successfully saved in fileid 1577.
VMFBDS2003W The SYSTEM SEGID D(51D) file has been changed and must be moved to
the S disk.
VMFBLD1851I (1 of 1) VMFBDESEG completed with return code 4
VMFBLD2180I There are 0 build requirements remaining
VMFBLD2760I VMFBLD processing completed with warnings
Ready(00004); T=22.62/24.50 11:42:15
vmfview build
Ready; T=0.27/0.29 11:44:40

```

Figure 13. Sample Console output for SCEEX Segment Load

- 3 Use VMFVIEW to review the build message log (\$VMFBLD \$MSGLOG). If necessary, correct any problems before going on.

vmfview build

4 If you received the message:

VMFBDS2003W The SYSTEM SEGID D(51D) file has been changed and must be moved to the S disk

then the SYSTEM SEGID must be copied over to the S-disk in order to stay in sync with the system's SEGID. Remember to re-SAVE CMS to avoid the Shared S-STAT not available message.

Language Environment is now installed, customized and built on your system.

7.0 Service Instructions

RSU Alert

If applying a Language Environment RSU, go to the instructions in Appendix A, "Applying a Recommended Service Upgrade (RSU) Tape For Language Environment" on page 40. You will return to a step in this chapter as specified in the RSU appendix.

This section of the Program Directory contains the procedure to install CORrective service to Language Environment. VMSES/E is used to install service for Language Environment.

To become more familiar with service using VMSES/E, you should read the introductory chapters in:

- *VMSES/E Introduction and Reference*

This manual also contains the command syntax for the VMSES/E commands listed in the procedure.

Note: Each step of the servicing instructions must be followed. Do not skip any step unless directed to. All instructions showing accessing of disks assume the use of default minidisk addresses. If different minidisk addresses are used, or if using a shared file system, change the instructions appropriately.

7.1 VMSES/E Service Process Overview

The following is a brief description of the main steps in servicing Language Environment using VMSES/E.

- Merge Service

Use the VMFMRDSK command to clear the alternate apply disk before receiving new service. This allows you to easily remove the new service if a serious problem is found.

- Receive Service

The VMFREC command receives service from the delivery media and places it on the Delta disk.

- Apply Service

The VMFAPPLY command updates the version vector table (VVT), which identifies the service level of all the serviced parts. In addition, AUX files are generated from the VVT for parts that require them.

- Reapply Local Service (if applicable)

All local service must be entered into the software inventory to allow VMSES/E to track the changes and build them into the system. Refer to Chapter 7 in the *VM/ESA Service Guide* for this procedure.

- Build New Levels

The build task generates the serviced level of an object and places the new object on a test BUILD disk.

- Place the New Service into Production

Once the service is satisfactorily tested it should be put into production by copying the new service to the production disk, re-saving the Saved Segments or DCSS (Discontiguous Saved Segments), etc.

7.2 Servicing Language Environment

RSU Alert

If applying a Language Environment RSU, go to the instructions in Appendix A, "Applying a Recommended Service Upgrade (RSU) Tape For Language Environment" on page 40. You will return to a step in this chapter as specified in RSU appendix.

7.2.1 Prepare to Receive Service

The *ppfname* used throughout these servicing instructions is **5688198H**, which assumes you are using the PPF supplied by IBM for Language Environment. If you have your own PPF override file for Language Environment, you should use your file's *ppfname* instead of **5688198H**. The *ppfname* you use should be used **throughout** the rest of this procedure, unless otherwise stated differently.

1 Log onto Language Environment service user ID **P688198H**

2 Establish access to the software inventory disk.

Note: If the MAINT 51D minidisk was accessed R/O, you will need to have the user that has it accessed R/W link it R/O. You then can issue the following commands to obtain R/W access to it.

```
link maint 51D 51D mr
access 51D D
```

The 51D minidisk is where the VMSES/E Software Inventory files and other product dependent files reside.

3 Have the Language Environment CORrective service tape mounted and attached to **P688198H**.

4 Establish the correct minidisk access order.

```
vmfsetup 5688198H {LE370|LE370SFS}
```

5688198H is the PPF that was shipped with the product. If you have your own PPF override you should substitute your PPF name for 5688198H.

Use **LE370** if the product is installed on minidisk or **LE370SFS** if the product is installed in SFS.

- 5 Receive the documentation. VMFREC, with the INFO option, loads the documentation and displays a list of all the products on the tape.

Electronic Service

If you are receiving service from ServiceLink (electronic service) see Appendix A, 'Receiving Service for VMSES Envelopes', section "Receive the Service" in the *VM/ESA Service Guide*. Then continue with step 7.

vmfrec info

This command will load the service memo to the 191 disk.

- 6 Check the receive message log (\$VMFREC \$MSGLOG) for warning and error messages.

vmfview receive

Also make note of which products and components have service on the tape. To do this, use the PF5 key to show all status messages which identify the products on the tape.

- 7 Clear the alternate APPLY disk to ensure that you have a clean disk for new service.

vmfmrdsd 5688198H {LE370 | LE370SFS} apply

Use **LE370** for installing on minidisks or **LE370SFS** for installing in Shared File System directories.

This command clears the alternate APPLY disk.

- 8 Review the merge message log (\$VMFMRD \$MSGLOG). If necessary, correct any problems before going on. For information about handling specific build messages, see *VM/ESA: System Messages and Codes*, or use online HELP.

vmfview mrd

7.2.2 Receive the Service

Electronic Service

If you are receiving service from ServiceLink (electronic service) see Appendix A, 'Receiving Service for VMSES Envelopes', in *VM/ESA Service Guide*. Then continue with 7.2.3 "Apply the Service."

- 1 Receive the service.

vmfrec ppf 5688198H {LE370|LE370SFS}

Use **LE370** for installing on minidisks or **LE370SFS** for installing on Shared File System directories.

This command receives service from your service tape. All new service is loaded to the alternate DELTA disk.

- 2 Review the receive message log (\$VMFREC \$MSGLOG). If necessary, correct any problems before going on. For information about handling specific build messages, see *VM/ESA: System Messages and Codes*, or use online HELP.

vmfview receive

7.2.3 Apply the Service

- 1 Apply the new service.

vmfapply ppf 5688198H {LE370|LE370SFS}

Use **LE370** for installing on minidisks or **LE370SFS** for installing in Shared File System directories.

This command applies the service that you just received. The version vector table (VVT) is updated with all serviced parts and all necessary AUX files are generated on the alternate apply disk.

You must review the VMFAPPLY message log if you receive a return code (RC) of a 4, as this may indicate that you have local modifications that need to be reworked.

- 2 Review the apply message log (\$VMFAPP \$MSGLOG). If necessary, correct any problems before going on. For information about handling specific build messages, see *VM/ESA: System Messages and Codes*, or use online HELP.

vmfview apply

Note

If you get the message VMFAPP2120W then re-apply any local modifications before building the new Language Environment. Refer to Chapter 7 in *VM/ESA Service Guide*. Follow the steps that are applicable to your local modification.

The following substitutions need to be made:

- **esalcl** should be **5688198H**
- **esa** should be **5688198H**
- *compname* should be **LE370** or **LE370SFS** (minidisk or SFS)
- *appid* should be **5688198H**
- *fm-local* should be the fm of 2C2
- *fm-applyalt* should be the fm of 2A6

Keep in mind that, when you get to the "Rebuilding Objects" step in the *Service Guide*, you should return to this program directory at 7.2.4, "Update the Build Status Table."

7.2.4 Update the Build Status Table

- 1 Update the Build Status Table with serviced parts.

vmfbld ppf 5688198H {LE370|LE370SFS} (status

Use **LE370** for installing on minidisks or **LE370SFS** for installing on Shared File System directories.

This command updates the Build Status Table.

Note

If the \$PPF files have been serviced you will get the following prompt:

VMFBLD2185R The following source product parameter files have been serviced:

VMFBLD2185R 5688198H \$PPF

VMFBLD2185R When source product parameter files are serviced, all product parameter files built from them must be recompiled using VMFPPF before VMFBLD can be run.

VMFBLD2185R Enter zero (0) to have the serviced source product parameter files built to you A-disk and exit VMFBLD so you can recompile your product parameter files with VMFPPF

VMFBLD2185R Enter one (1) to continue only if you have already recompiled your product parameter files with VMFPPF

0

Enter a 0 and complete the following steps before you continue.

VMFBLD2188I Building 5688198H \$PPF on 191 (A) from level \$PFnnnnn

vmfppf 5688198H *

Note: If you've created your own PPF override, use your PPF name instead of 5688198H.

An '*' will rebuild all components in the specified PPF.

copyfile 5688198H \$PPF a = = d (olddate replace erase 5688198H \$PPF a

Note: Do not use your own PPF name in place of 5688198H for the COPYFILE and ERASE commands.

vmfbl d ppf 5688198H {LE370 | LE370SFS} (status 1

Reissue VMFBLD to complete updating the build status table. When you receive the prompt that was previously displayed, enter a 1 to continue.

Use **LE370** for installing on minidisks or **LE370SFS** for installing on Shared File System directories.

- 2 Use VMFVIEW to review the build status messages, and see what objects need to be built.

vmfview build

7.2.5 Build Serviced Objects

- 1 Rebuild Language Environment serviced parts.

vmfbld ppf 5688198H {LE370|LE370SFS} (serviced

Use **LE370** for installing on minidisks or LE370SFS for installing on Shared File System directories.

Note: If the warning message VMFLB1966W (with return code 4) is issued for objects ILBONBL, ILBONTR, ILBOREC, ILBORNT, ILBOSND, ILBOSNT or ILBOSSN it can be **ignored**.

- 2 Review the build message log (\$VMFBLD \$MSGLOG). If necessary, correct any problems before going on. For information about handling specific build messages, see *VM/ESA: System Messages and Codes*, or use online HELP.

vmfview build

7.3 Place the New Language Environment Service Into Production

7.3.1 Rebuild the Saved Segments

- 1 If installing IBM Language Environment VM on a system that uses VMSES/E to save the shared segments, then depending on which saved segment you have installed, issue the following command(s):

vmfbld ppf segbld esasegs segblist SCEE (serviced
vmfbld ppf segbld esasegs segblist SCEEX (serviced

- 2 Review the build message log (\$VMFBLD \$MSGLOG). If necessary, correct any problems before going on. For information about handling specific error messages, see VM/ESA: System Messages and Codes, or use on-line HELP.

vmfview build

7.3.2 Copy the New Language Environment Serviced Files Into Production

- 1 Logon to MAINT if you plan to put Language Environment general use code on the 'Y' disk (MAINT's 19E disk). Or logon to the owner of the disk that will contain the 'production' level of the Language Environment code.

```
link P688198H 29e 29e rr
access 29e e
access 19e f
vmfcopy * * e = = f2 (prodid 5688198H%LE370 olddate replace
```

The VMFCOPY command will update the VMSES PARTCAT file on the 19E disk.

You have now finished servicing Language Environment.

8.0 Customizing Language Environment

Once the product has been installed, it can be customized using the 'C5688198' EXEC. This EXEC will do the following:

1. Prompt you for the area you wish to customize:
 - IBM Language Environment VM Runtime Options
 - IBM Language Environment VM User Exit
 - COBOL COBPACKs
 - 'C' Component Locale Time Information
 - Saved Segments Components
 - COBOL Reusable environment
2. Invoke an 'XEDIT' session for the specific customization component requested;
3. Re-assemble, if required, component customized;
4. Rebuild required modules using the specific VMSES/E part handler

```
IBM Language Environment for MVS & VM
Version 1 Release 8 Mod 0

1) Run Time Options
2) User Exits
3) "C" Locale Time Information
4) Saved Segments
5) COBOL COBPACKs
6) COBOL Reusable Environment

Enter number of option you wish to change or
Enter 'END' or 'QUIT' to exit EXEC
```

Figure 14. Customization EXEC - Panel 1

8.1 Updating Run-Time Options

IBM Language Environment VM run-time options are updated by invoking the customization EXEC which puts you into an XEDIT session of CEEDOPT ASSEMBLE. After you update and file CEEDOPT, the EXEC assembles it (using HASM) and if the assembly is successful, will then prompt you to see if you want to rebuild the modules in which it is included. Modules which will be rebuilt are CEEBINIT, CEEBPICI and CEEPIPI all of which are in Build List "CEEBLMOD." See Appendix E, "IBM Language Environment VM Run-time Options" on page 61 for a complete description of the run-time options.

8.2 Updating User Exit Options

The assembler user exit is updated by invoking the customization EXEC which puts you into an XEDIT session of CEEBXITB ASSEMBLE. After you update and file CEEBXITB, the EXEC assembles it (using HASM) and if the assembly is successful, will then prompt you to see if you want to rebuild the component in which it is included. Modules which will be rebuilt are CEEBINIT, CEEBPICI and CEEPIPI all of which are in Build List "CEEBLMOD."

8.3 Updating COBOL Component COBPACKS

COBPACKs are updated by the editing of file IGZBLPAC Exec using the C5688198 Customization EXEC. After IGZBLPAC is updated and filed, the EXEC will rebuild the IGZCPAC and IGZCPCO MODULEs.

As this is the control file defining the structure of these two relocatable LOAD MODULEs, it is important that it is updated very carefully. Updates to this file should be made by commenting out those ":PARTID." statements of the text files you do not wish to have in the COBPACKs. Do NOT delete any lines. For the IGZCPCO COBPACK you must, at a minimum leave in the ":PARTID.IGZCPCO " statement. For the IGZCPAC COBPACK you must leave the ":PARTID.IGZCPAC" statement. If any lines other than the ":PARTID." are commented out, or if the IGZCPCO or IGZCPAC basic ":PARTID." statements are commented out, or the order of any of the lines has been changed, the update could fail with a control file error.

The COBPACKs can also be tailored to run above the line during customization. This is accomplished by commenting out all of the text files built with "RMODE 24" and the COBOL COBPACKs in the PSEG files. Modules which will be rebuilt are IGZCPAC and IGZCPCO which are in Build List "IGZBLPAC."

8.4 Updating COBOL Component Reusable Environment

COBOL's reusable environment behavior is updated by invoking the customization EXEC which puts you into an XEDIT session of IGZERREO ASSEMBLE. After you update and file IGZERREO, the EXEC assembles it (using HASM) and if the assembly is successful, will then prompt you to see if you want to rebuild the component in which it is included. The module that will be rebuilt is CEEEV005 which is in Build List "IGZBLMOD".

The COBOL reusable environment behavior can be modified 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 either ILBOSTP0 or IGZERRE INIT.

With the IBM supplied default setting for COBOL's reusable environment behavior (IGZERREO with REUSENV=COMPAT), when a program check occurs while the reusable environment is dormant (i.e. 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), a S0Cx abend will occur. This behavior is compatible with the VS COBOL II and OS/VS COBOL run-times, but it significantly impacts the

performance when a COBOL/370 or COBOL for MVS & VM program is invoked repeatedly in a COBOL reusable environment. The performance degradation is caused by Language Environment issuing an ESPIE RESET when the reusable environment becomes dormant and then an ESPIE SET upon reentering the reusable environment.

COBOL's reusable environment behavior can be modified (IGZERREO with REUSENV=OPT) so that all program checks will be 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 a 4036 abend from Language Environment. However, since Language Environment does not have to issue the ESPIE RESET and ESPIE SET between invocations of the COBOL program, this can be faster than using REUSENV=COMPAT.

8.4.1 Modifying the Behavior of the COBOL Reusable Environment

Modify the IGZRREOP macro invocation, depending on the function that you want.

To run with VS COBOL II and OS/VS COBOL run-time compatibility mode (i.e., 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 (i.e., Language Environment intercepts all program checks that occur when the COBOL reusable environment is dormant and converts them to a 4036 abend, resulting in improved performance), use:

```
IGZRREOP REUSENV=OPT
```

8.5 C Component Locale Time Information

Note

Due to the size and having to rebuild the SCEERUN LOADLIB for this option, your "A" disk, which VM uses as an interim work disk during the rebuild, must have at least 20 cylinders on a 3390, or equivalent, of unused (free) space.

C locale time information is used for options such as Time Zone name and Daylight Savings Time starting dates.

Locale time is updated by editing a file named 'EDCLOCI'. The EXEC will put you into an XEDIT session of EDCLOCI ASSEMBLE and after updates are completed it is filed and then assembled using HASM. Once successfully assembled, the EXEC will rebuild the required components and the C locale time is updated.

8.6 Updating Saved Segments

After successfully installing IBM Language Environment VM, you can load certain routines into *Saved Segments* on VM/ESA. Placing routines into Saved Segments reduces overall system storage requirements by making the routines sharable and also, initiation/termination (init/term) time is reduced for each application, since load time decreases.

Included with IBM Language Environment VM are two build lists, CEEBLSGA and CEEBLSGB, plus the necessary LSEG files required to install specific routines of IBM Language Environment VM into segments. By selecting option 5 in the Customization exec, these individual build lists can be tailored to load only specific routines of the IBM Language Environment VM product (i.e commonly used COBOL, PL/I, or C routines) into segments. Each build list contains comments that identify these routines and to help tailor the segment install.

Customizing can be accomplished by either commenting or uncommenting the *LOADFUNC* component statement(s) you wish to take action on or by adding a correct "LOADFUNC" statement into the build list.

A "*" inserted in the first column of any *LOADFUNC* statement will eliminate that component from being included while deleting the "*" from the first column will include the component. In the example shown below, the COBOL COBPACKs (IGZCPAC and IGCCPCO) which are normally installed below the line, and thus included in the CEEBLSGB build list, will be eliminated from the saved segment environment.

```
*****
*           IBM Language Environment for MVS & VM           *
*           Version 1 Release 8 Modification 0             *
*                                                         *
*           Licensed Materials -- Property of IBM         *
*           5688-198 (C) Copyright IBM Corporation 1997  *
*           All Rights Reserved                          *
*****
* Buildlist CEEBLSGB for Saved Segment (Below line 16M Line) *
*****
*
:FORMAT. 2
*
:OBJNAME. SCEE.SEGMENT
:BLDREQ.  CEEBLMOD.CEEBINIT.MODULE
          CEEBLMOD.CEEBLIIA.MODULE
          CEEBLMOD.CEEPIPI.MODULE
          CEEBLMOD.CEEBPICI.MODULE
          EDCBLSP2
          IBMBLMOD.IBMRCOMP.MODULE
          IBMBLMOD.IBMRLIB1.MODULE
          IBMBLMOD.IBMRPTLA.MODULE
:GLOBAL. TXTLIB SCEESPC
:OPTIONS. LOADFUNC ( LSEG CEEBINIT )
          LOADFUNC ( LSEG CEEBLIIA )
          LOADFUNC ( LSEG CEEPIPI )
```

```
        LOADFUNC ( LSEG CEEBPICI )
        LOADFUNC ( LSEG IBMRLIB1 )
        LOADFUNC ( LSEG IBMRCOMP )
*       LOADFUNC ( LSEG IGZCPAC  )
*       LOADFUNC ( LSEG IGZPCO   )
:EOBJNAME.
*
```

By using this method, should you decide at a later date to reinstate these routines in the saved segments simply remove the asterisk "*" and regenerate the segments. If you wish to include other routines into saved segments simply add the correct "LOADFUNC" statement into the respective build list.

See Appendix D, "Segment Build Lists (CEEBSGA/CEEBSGB)" on page 58 for a full description of the segment build lists.

For more information on the defining/loading of the Saved segments for IBM Language Environment VM, see 6.4.3, "Redefining and Building Saved Segments for Language Environment" on page 18.

8.7 Installing in Saved Segments

All IBM Language Environment VM routines that can be installed into saved segments are shown below with their approximate sizes in hex bytes.

SCEE (CEEBSGB) (Below Line)		SCEEX (CEEBSGA) (Above Line)	
Mod Name	Size	Mod Name	Size
** LE Components **		** LE Components **	
CEEBINIT	A740	CEECOPP	BB68
CEEBLIIA	1548	CEEPLPKA	A87F0
CEEPIPI	CF88	CEEQMATH	843A8
CEEBPICI	AD20	CEEMUEN0	2890
		CEEMUEN2	2DB8
		CEEMUEN3	7420
** COBOL Components **		CEEMUEN4	0460
IGZCPAC	1AF18	CEEMUEN5	3598
IGZPCO	BA90	CEEMENU0	2890
		CEEMENU2	2DB8
** PL/I Components **		CEEMENU3	7420
IBMRLIB1	BDA0	CEEMENU4	0400
IBMRCOMP	3818	CEEMENU5	3598
IBMRPTLA	0008		
		** PL/I Components **	
		CEEEV010	343A0
		** COBOL Components **	
		IIGZMSGT	0088
		IGZINSH	27A00
		IGZCMGEN	40D0
		CEEEV005	33A8
		** 'C' Components **	
		EDCNSS01	16F840 (CEEEV003 Module)
		EDCNSS02	8E958 (EDCZ24 Module)
		EDCNSS03	47DF8 (EDCNINSP Module)
		EDCZUMSG	6D70
		EDCZEMSG	6D70
		** 'NLS' Components **	
		EDCZJMSG	
		IGZCMGJA	4278
		CEEMJPN0	2848
		CEEMJPN2	2E70
		CEEMJPN3	76D8
		CEEMJPN4	03F8
		CEEMJPN5	3728

Figure 15. Component Module Size

Appendix A. Applying a Recommended Service Upgrade (RSU) Tape For Language Environment

The RSU Tape is structured to install all PTFs included on the tape plus the tape files containing the preapplied service and prebuilt objects. All PTF-related files are loaded to the delta disk. The tape file containing the preapplied service, i.e. containing the results of VMFAPPLY, is loaded to the alternate apply disk and the contents of the tape files containing prebuilt objects are loaded to the appropriate build disks.

Points to consider about using the Product Service Upgrade procedure are:

- This process will not alter any of your tailored flat files in any way. Local modifiable files you may have updated will have to be re-worked to include any new service to these files.
- Planning must be done (such as determining disk sizes, and determining what service, if any, on your existing system is not contained on the RSU Tape) prior to actually loading the service from the RSU Tape. These tasks will be discussed.

The following outline is an overview of what tasks need to be performed during the Preventive Service procedure (PSU) using the RSU tape:

- Prepare System

In this task, you will receive the documentation contained on the RSU Tape and determine the DASD required to install the RSU Tape.

- Merge Service

Use the VMFMRDSK command to clear the alternate apply disk before receiving the RSU Tape. This allows you to easily remove the new service if a serious problem is found.

- Receive Service

The VMFINS command receives service from the RSU Tape and places it on the Delta disk.

- Apply Additional Service

The VMFAPPLY command updates the version vector table (VVT), which identifies the service level of all the serviced parts. In addition, AUX files are generated from the VVT for parts that require them. These steps are used to reapply service that was not contained on the RSU tape that was already installed for Language Environment.

- Reapplying Local Modifications (if applicable)

All local modifications must be entered into the software inventory to allow VMSES/E to track the changes and build them into the system.

- Building New Levels

The build tasks generates the serviced level of an object and places the new object on a BUILD disk.

- Placing the New Service into Production

Once the service is satisfactorily tested it should be put into production by copying the new service to the production disk, re-saving the NSS (Named Saved System) or DCSS (Discontiguous Saved Segments), etc.

A.1 Apply Language Environment RSU

A.1.1 Prepare Your System for Service Refresh

The *ppfname* used throughout these instructions is **5688198H**, which assumes you are using the PPF supplied by IBM for Language Environment. If you have your own PPF override file for Language Environment you should use your file's *ppfname* instead of **5688198H**. The *ppfname* you use should be used **throughout** the rest of this procedure.

Electronic Service (envelope file)

If you have received the RSU as an envelope file, follow the procedures in the *ServiceLink User's Guide* to retrieve and decompact the envelope file to your A-disk. This is currently done by using the VMDPACK EXEC (which uses the DETERSE MODULE) supplied by ServiceLink. The file name of the decompacted file will be of the format RPTFnnnn. The file type will be SERVLINK. You will need to enter the file name on the VMFINS commands.

- 1** Read through the latest RSU information hard copy memo. This is a memo that should have come with the RSU tape.
- 2** Logon to the IBM Language Environment VM service user ID P688198H
- 3** If the Software Inventory disk (51D) was accessed R/O (read only) then establish write access to the Software Inventory disk.

Note: If the MAINT 51D minidisk was accessed R/O, you will need to have the user who has it linked R/W link it as R/O. You then can issue the following commands to obtain write access to it. Do not use *mw* mode.

**link MAINT 51d 51d mr
access 51d d**

The MAINT 51D disk is where the VMSES/E system level software inventory files reside.

- 4** Mount the RSU tape on the tape drive as virtual device 181. You must use 181.

5 Receive the documentation

Receive the documentation on the tape for the RSU Tape. This step will also load the cumulative Apply Status Table (SRVAPPS) which identifies all preapplied service contained on the tape. These files are loaded to the 51D disk.

a If receiving the RSU from tape

vmfins install info (nomemo

b If receiving the RSU from an envelope file

vmfins install info (nomemo env rptf*nnnn*

6 Determine DASD sizes for disks to receive service

In order to receive the service from the RSU Tape, you need to have adequate space available on the alternate APPLY and DELTA disks. The required sizes are identified in the Language Environment documentation (5688198H MEMO D) received in the previous step.

7 Setup the correct minidisk access order

vmfsetup 5688198H {LE370 | LE370SFS}

Use **LE370** for installing on minidisks or **LE370SFS** for installing in Shared File System directories.

8 Merge the APPLY disks for Language Environment

Next, you must prepare your system to receive the service from the RSU Tape. To do this, you must first clear the alternate apply disk for receipt of the service from the RSU Tape.

Enter the VMFMRDSK command to merge the alternate apply disk to the apply disk. This will clear the alternate apply disk.

vmfmrdsk 5688198H {LE370 | LE370SFS} apply

Use **LE370** for installing on minidisks or **LE370SFS** for installing in Shared File System directories.

- 9 Obtain additional information about the service on the RSU and how it will affect your local modifications by invoking the VMFPSU command. This command creates an output file, **appid PSUPLAN**, which you can review. See *VM/ESA Service Guide* for an explanation of this file.

vmfpsu 5688198H {LE370 | LE370SFS}

Use **LE370** for installing on minidisks or **LE370SFS** for installing in Shared File System directories.

This command produces an output file that contains information about the service on the RSU compared against the service and local modifications on your system. The file name is **appid PSUPLAN**, where **appid** is specified in the PPF file.

A.1.2 Receive the Preapplied, Prebuilt Service

- 1 Refresh the Language Environment service disks by loading new service from the RSU Tape

a If receiving the RSU from tape

vmfins install ppf 5688198H {LE370 | LE370SFS} (nomemo nolink

Use **LE370** for installing on minidisks or **LE370SFS** for installing in Shared File System directories.

b If receiving the RSU from an envelope file

vmfins install ppf 5688198H {LE370 | LE370SFS} (nomemo nolink env rptfnnnn

Use **LE370** for installing on minidisks or **LE370SFS** for installing in Shared File System directories.

```

VMFINS2767I Reading VMFINS DEFAULTS B for additional options
VMFINS2760I VMFINS processing started
VMFINS2601R Do you want to create an override for :PPF 5688198H
LE370 :PRODID 5688198H%LE370?
Enter 0 (No), 1 (Yes) or 2 (Exit)
0
VMFINS2603I Processing product :PPF 5688198H LE370 :PRODID
5688198H%LE370
VMFREQ2805I Product :PPF 5688198H LE370 :PRODID 5688198H%LE370
has passed requisite checking
VMFINT2603I Installing product :PPF 5688198H LE370 :PRODID
5688198H%LE370
VMFSET2760I VMFSETUP processing started for 5688198H LE370
VMFUTL2205I Minidisk|Directory Assignments:
String Mode Stat Vdev Label/Directory
VMFUTL2205I LOCALSAM E R/W 2C2 SES2C2
VMFUTL2205I APPLY F R/W 2A6 SES2A6
VMFUTL2205I G R/W 2A2 SES2A2
VMFUTL2205I DELTA H R/W 2D2 SES2D2
VMFUTL2205I BUILDO I R/W 29E SES29E
VMFUTL2205I BASE1 J R/W 2B2 SES2B2
VMFUTL2205I ----- A R/W 191 USR191
VMFUTL2205I ----- B R/O 5E5 MNT5E5
VMFUTL2205I ----- D R/W 51D MNT51D
VMFUTL2205I ----- S R/O 190 MNT190
VMFUTL2205I ----- Y/S R/O 19E MNT19E
VMFSET2760I VMFSETUP processing completed successfully
VMFREC2760I VMFREC processing started
VMFREC1852I Volume 1 of 1 of INS TAPE yynn
:
VMFREC2760I VMFREC processing completed successfully
VMFINT2603I Product Installed
VMFINS2760I VMFINS processing completed successfully
Ready;

```

- 2** Check the receive message log (\$VMFREC \$MSGLOG) for warning and error messages. If necessary, correct any problems before going on. For information about handling specific receive messages, see *VM/ESA: System Messages and Codes*, or use online HELP.

vmfview install

A.1.3 Process Additional Service

- 1** Apply additional service

The VMFAPPLY command is used to reapply service that was not contained on the refresh tape that was already installed for the component.

Applying service with preapplied, prebuilt service will reapply any reach-ahead service that may be on the system **or** indicate that there are no reach-ahead PTFs to be applied.

vmfapply ppf 5688198H {LE370 | LE370SFS}

Use **LE370** for installing on minidisks or **LE370SFS** for installing in Shared File System directories.

Messages VMFAPP2122E and VMFAPP2109R will be displayed only if you have reach-ahead service that needs to be reapplied. If you receive these messages, enter **1** in reply to VMFAPP2109R to reapply the reach-ahead service (as shown in the example below).

VMFAPP2122E The set of PTFs in the Apply Status Table (5688198H SRVAPPS) on the 2A2 (G) disk is not a subset of the PTFs in the highest Level Apply Status Table on the 2A6 (F) disk. This is an inconsistent state.

VMFAPP2109R VMFAPPLY will automatically correct the problem identified by message 2122E by including the missing PTFs in the current Apply List. Enter (1) to continue; (0) to quit.

1

Enter **1** for VMFAPPLY to reapply the reach-ahead service.

- 2** Check the apply message log (\$VMFAPP \$MSGLOG) for warning and error messages. If necessary, correct any problems before going on. For information about handling specific apply messages, see *VM/ESA: System Messages and Codes*, or use online HELP.

vmfview apply

- 3** If necessary, rework **local modifications**.

The output from the VMFPSU command (which was run in an earlier step), *appid* **PSUPLAN** file, can be used to indicate what local service or mods are affected by the RSU Tape. If a PTF is applied and it contains service to a part for which you have a local modification, you will need to rework the local modification. Refer to the *VM/ESA Service Guide*.

A \$PSU\$ \$SELECT file might have been created on the A-disk. This file has in it the local modifications affected by service; whether they required rework or just a rebuild. You need to append this file to the top of the 5688198H \$SELECT file on the alternate apply disk. This ensures that your local modifications get rebuilt.

A.1.4 Build the New Service Level and Place Into Production

To rebuild all objects that were affected by reach-ahead service that was reapplied, local modifications, and saved segments continue with the instructions in 7.2.4, "Update the Build Status Table" on page 30. This will also lead you into the steps to place Language Environment into production.

A.2 Determine RSU Service Level

The service contained on each RSU constitutes a new service level. Use this service level when ordering corrective service. The service level is updated in the system inventory when the RSU is installed.

The following command is used to query the current service level of the system.

vmfsim query vm sysrecls tdata :ppf ppfname :stat The output from this command is similar to the following console log. The last part of the status line indicates the RSU service level: 9801

```
VMFSIP2408I RESULTS FOR
      TDATA :PPF ppfname :STAT
:PPF ppfname compname
:STAT RECEIVED.mm/dd/yy.hh:mm:ss.userid.RSU-9801
```

Appendix B. Moving Language Environment to SFS Directories and Creating PPF Override

By default, IBM Language Environment VM is installed to minidisks, as part of the VM/ESA Version 2 Release 3.0 System DDR; unless you choose to move Language Environment to SFS during the initial VM/ESA 2.3.0 install. However, you can move some of the Language Environment minidisks to SFS directories. Refer to the figure Figure 6 on page 13 to see which minidisks can reside in SFS. You can use either the default file pool, VMSYS, or a file pool of your choice.

A summary of the steps that are needed to place Language Environment into SFS are:

- Allocate space in the user-defined file pool
- Provide the installation user ID, P688198H, access to the file pool
- Create the necessary Language Environment subdirectories
- Copy files from the minidisks to the new SFS subdirectories
- Create a PPF override, if not using the default file pool or subdirectory names.

Where to next

You need to do one of the following:

- If you want to place Language Environment into SFS directories **using the defaults defined in the 5688198H PPF file** you need to continue with the instructions provided in the *VM/ESA Installation Guide*, in the Appendix titled "Moving Components to SFS Directories".
- If you want to place Language Environment into your **own user-defined file pool or SFS directories**, continue with the instructions in B.1, "Setup SFS File Pool and Directories" on page 48.
- If you need to **just create a PPF override**, continue with the instructions in B.3, "Create Product Parameter File (PPF) Override" on page 50.

After you have chosen one of the previous options and completed the steps required (whether in this appendix or in the *VM/ESA Installation Guide*) you need to return to the section that pointed you to this appendix and continue from there (if you were pointed to this appendix).

B.1 Setup SFS File Pool and Directories

Notes:

1. The steps that follow help you determine the Language Environment file pool space requirements, file pool enrollment, and SFS directory definitions. If not all of these steps are required - for example, you are changing only SFS directory or subdirectory names - adapt these instructions as needed for your environment.
2. The steps that follow assume that a user-defined SFS file pool name is being used. If you will be using the VM/ESA default VMSYS file pool name, substitute it in any place you see 'user-defined file pool' or *userfilepool*.
3. For more information about planning for, generating and managing a file pool and server, see *VM/ESA File Pool Planning, Administration and Operation*.

- 1** Determine the number of 4k blocks that are required for SFS directories by adding up the 4K blocks required for each SFS directory you plan to use.

If you intend to use all of the default Language Environment SFS directories, the 4K block requirements for each directory are summarized in Figure 6 on page 13.

This information is used when enrolling the P688198H to the user-defined file pool.

- 2** Enroll user P688198H in the user-defined file pool using the ENROLL USER command:

```
ENROLL USER P688198H userfilepool: (BLOCKS blocks)
```

where *blocks* is the number of 4k blocks that you calculated in the previous step.

Note: This must be done from a user ID that is an administrator for the user-defined file pool.

- 3** Determine if there are enough blocks available in the file pool to install Language Environment. This information can be obtained from the QUERY FILEPOOL STATUS command. Near the end of the output from this command is a list of minidisks in the file pool and the number of blocks free. If the number of blocks free is smaller than the total 4k blocks needed to install Language Environment you need to add space to the file pool. See *VM/ESA SFS and CRR Planning, Administration, and Operation* manual for information on adding space to a file pool.

- 4 Create the necessary subdirectories using the CREATE DIRECTORY command.

A complete list of default Language Environment SFS directories is provided in Figure 6 on page 13.

```
set filepool userfilepool:
create directory dirid
```

dirid is the name of the SFS directory you're creating, such as the default names:

```
create directory userfilepool:P688198H.LE370
create directory userfilepool:P688198H.LE370.OBJECT
create directory userfilepool:P688198H.LE370.LOCAL
create directory userfilepool:P688198H.LE370.DELTA
create directory userfilepool:P688198H.LE370.ALTAPPLY
create directory userfilepool:P688198H.LE370.PRODAPPLY
```

If necessary, see *VM/ESA CMS Command Reference* for more information about the CREATE DIRECTORY command.

- 5 If you intend to use an SFS directory as the work space for the P688198H used ID, include the following IPL control statement in the P688198H directory entry:

```
IPL CMS PARM FILEPOOL VMSYS
```

This will cause CMS to automatically access P688198H's top directory as file mode A.

B.2 Copy Minidisk (or SFS) Files to SFS Directories

- 1 Copy the files from the Language Environment minidisks (or from the VMSYS file pool, if Language Environment is already installed there) to your new user-defined SFS file pool and directories using the VMFCOPY command.

Note: Repeat the ACCESS and VMFCOPY commands, that follow, for each disk or SFS directory you need to copy. Refer to Figure 6 on page 13 for the default minidisk and SFS directory names for Language Environment.

```
access vdev e
access dirid f
vmfcopy * * e = f (prodid 5688198H%LE370 olddate replace
```

Where

- *vdev* is the minidisk you are copying files from. If you are copying from SFS then substitute in the SFS directory name in place of *vdev*.
- *dirid* is the name of the (target) SFS directory that you are copying files to.
- 5688198H%LE370 is the PRODID defined within the 5688198H PPF file, for the minidisk and SFS components of Language Environment.

The VMFCOPY command will update the VMSES PARTCAT file on the target directory.

B.3 Create Product Parameter File (PPF) Override

This section provides information to help you create a product parameter file (PPF) override. The example used in this section is how to change the shared file system (SFS) file pool where Language Environment files reside. Refer to the *VMSES/E Introduction and Reference* for more information on PPF overrides.

Note: Do **not** modify the product supplied 5688198H \$PPF or 5688198H PPF files to change the VMSYS file pool name or any other installation parameters. If the 5688198H \$PPF file is serviced, the existing \$PPF file will be replaced, and any changes to that file will be lost. By creating your own \$PPF override, your updates will be preserved.

The following process describes changing the default file pool name, VMSYS to MYPOOL1:

- 1 Create a new \$PPF override file or edit an existing override file.

xedit *overname* \$PPF *fm*2

overname is the PPF override file name (such as "myLE370") that you want to use.

fm is an appropriate file mode. If you create this file yourself, specify a file mode of A.

If you modify an existing override file, specify a file mode of A or D, based on where the file currently resides (A being the file mode of a R/W 191 minidisk, or equivalent; D, that of the MAINT 51D minidisk).

- 2 Create (or modify as required) the Variable Declarations (:DCL.) section for the LE370SFS override area so that it resembles the :DCL. section as shown below.

```

:OVERLST. LE370SFS
*
*=====
:LE370SFS. LE370 5688198H
*=====
:DCL. UPDATE
&191  DIR MYPPOOL1:P688198H.          * A DISK
&SAMPZ DIR MYPPOOL1:P688198H.LE370.LOCAL * SAMPLE/LOCAL FILES
&DELTZ DIR MYPPOOL1:P688198H.LE370.DELTA * PRODUCT SERVICE
&APPLY DIR MYPPOOL1:P688198H.LE370.ALTAPPLY * AUX/INVENTORY FILES
&APPLZ DIR MYPPOOL1:P688198H.LE370.PRODAPPLY * PROD. APPLY DISK
&BLD0Z LINK P688198H 29E 29E MR          * TEST USABLE FORMS
&BAS1Z DIR MYPPOOL1:P688198H.LE370.OBJECT * BASE DISK
&LE3ID1 USER P688198H
:EDCL.
:END.
*=====
*

```

This override will replace the :DCL. section of the LE370SFS override area of the 5688198H \$PPF file.

- 3 If your \$PPF override file was created at file mode A, copy it to file mode D—the Software Inventory minidisk (MAINT 51D).

file

copyfile *overname* \$PPF *fm* = = d (olddate)

- 4 Compile your changes to create the usable *overname* PPF file.

vmfppf *overname* LE370SFS

where *overname* is the file name of your \$PPF override file.

Now that the *overname* PPF file has been created, you should specify *overname* instead of 5688198H as the PPF name to be used for any VMSES/E commands that require a PPF name.

Appendix C. Contents of COBPACKs (IGZCPAC/IGZPCO)

C.1 Contents of General COBPACK - IGZCPAC

Figure 16 lists routines you can include in the general IGZCPAC COBPACK and briefly describes each to help you determine which to include in your tailored COBPACK.

Figure 16 (Page 1 of 4). Routines Eligible for Inclusion in General COBPACK (IGZCPAC)

Name	Description	OS/ CICS*	Link- Edited AMODE	Link- Edited RMODE
IGZCACP	ACCEPT and STOP literal	OS	31	ANY
IGZCACS	Alternate collating sequence comparison	Both	31	ANY
IGZCANE	Alphanumeric editing	Both	31	ANY
IGZCANF	Format with figurative constant	Both	31	ANY
IGZCBID	Binary to internal decimal	Both	31	ANY
IGZCBUG ⁶	Used for debugging	Both	31	24
IGZCCLS	Class test	Both	31	ANY
IGZCCTL ⁴	Batch/interactive debug control	Both	31	ANY
IGZCCVB	Numeric conversion	Both	31	ANY
IGZCDSP	DISPLAY	OS	31	ANY
IGZCFCC ⁶	Linkage manager for COBOL for MVS & VM (dynamic call and cancel)	OS	31	24
IGZCFDP ⁵	Formatted FDUMP	Both	31	ANY
IGZCFDW	TRUNC floating point to binary conversion	Both	31	ANY
IGZCFPW	Exponentiates double precision floating-point numbers	Both	31	ANY
IGZCGDR	Segment refresh	Both	31	ANY
IGZCHCM	Condition management events handler	Both	31	ANY
IGZCIDB	Internal decimal to binary	Both	31	ANY

Figure 16 (Page 2 of 4). Routines Eligible for Inclusion in General COBPACK (IGZCPAC)

Name	Description	OS/ CICS*	Link- Edited AMODE	Link- Edited RMODE
IGZCINS	INSPECT	Both	31	ANY
IGZCIN1	INSPECT library	Both	31	ANY
IGZCIN2	INSPECT library	Both	31	ANY
IGZCIPS	Initialization for internal program setup	Both	31	ANY
IGZCIVL	Comparison with figurative constant	Both	31	ANY
IGZCKCL	Kanji class test	Both	31	ANY
IGZCLDL	Load/delete subroutines	Both	31	ANY
IGZCLDR ¹	Partition loader (COBLDR)	Both	31	ANY
IGZCLLM ²	Load list manager	Both	31	ANY
IGZCLNC ⁶	Linkage manager for OS/VS COBOL, DEBUG, and IGZBRDGE (dynamic call and cancel)	Both	31	24
IGZCLNK ⁶	Linkage manager for VS COBOL II and COBOL/370 (dynamic call and cancel)	Both	31	24
IGZCMED	Median function processor	Both	31	ANY
IGZCMLT ⁵	Message table	Both	31	ANY
IGZCMMSG	Message process control routine	Both	31	ANY
IGZCNMV	NUMVAL/NUMVAL-C function processor	Both	31	ANY
IGZCONV	Conversion routine for floating point	Both	31	ANY
IGZCPPL ²	Linkage manager for procedure-pointers	Both	31	24
IGZCPRC ²	Program cleanup	Both	31	ANY
IGZCPRS ²	Program setup	Both	31	ANY
IGZCRCL ³	Run unit cleanup	Both	31	ANY
IGZCREV	Reverse function processor	Both	31	ANY
IGZCRSU ²	Run unit setup	Both	31	ANY
IGZCSCH	Binary search of table	Both	31	ANY

Figure 16 (Page 3 of 4). Routines Eligible for Inclusion in General COBPACK (IGZCPAC)

Name	Description	OS/ CICS*	Link- Edited AMODE	Link- Edited RMODE
IGZCSMV	Move right-justified	Both	31	ANY
IGZCSPA	Printer spacing	OS	31	ANY
IGZCSPC	Call by content	Both	31	ANY
IGZCSPM	Space manager	Both	31	ANY
IGZCSSN	Separate sign numeric	Both	31	ANY
IGZCSSR	SSRANGE compile-time option	Both	31	ANY
IGZCSTA	Statistical routine function processor	Both	31	ANY
IGZCSTG	STRING	Both	31	ANY
IGZCULE ⁶	User I/O logic error handler	OS	31	24
IGZCUPL	Upper and lowercase function	Both	31	ANY
IGZCUST	UNSTRING	Both	31	ANY
IGZCVDP ⁵	Variable dump routine 1	Both	31	ANY
IGZCVIN	VSAM initialization	OS	31	ANY
IGZCVLD ²	Verify loader	Both	31	ANY
IGZCVMO	Variable length move	Both	31	ANY
IGZCXDI	Double precision division	Both	31	ANY
IGZCXFR ⁶	I/O declarative transfer	OS	31	24
IGZCXMU	Double precision multiplication	Both	31	ANY
IGZCXPR	Decimal fixed-point exponentiation	Both	31	ANY
IGZIBPC ⁴	Build program control tables	Both	31	ANY
IGZICAL ⁴	Call intercept routine	Both	31	ANY

Figure 16 (Page 4 of 4). Routines Eligible for Inclusion in General COBPACK (IGZCPAC)

Name	Description	OS/ CICS*	Link- Edited AMODE	Link- Edited RMODE
IGZICUD ⁴	Describe CU	Both	31	ANY

Notes to Routines Eligible for inclusion in General COBPACK (IGZCPAC):

- 1 Highly recommended for a partially loaded COBPACK.
 - 2 Highly recommended for inclusion in the general COBPACK, regardless of whether the location is above or below the 16M address line.
 - 3 Highly recommended for inclusion in the general COBPACK if it is located below the 16M address line.
 - 4 If IGZCCTL is included in the COBPACK, you should also include modules IGZIBPC, IGZICAL, and IGZICUD.
 - 5 If IGZCFDP is included in the COBPACK, you should also include modules IGZCMLT and IGZCVDP.
 - 6 This routine is not included in the IBM supplied COBPACK IGZCPAC so that the COBPACK is RMODE(ANY) and will be loaded above the 16M line.
-

C.2 Contents of the Environment-Specific COBPack (IGZCPCO)

Figure 17 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.

Figure 17 (Page 1 of 2). Routines Eligible for Inclusion in the Environment-Specific COBPack (IGZCPCO)

Name	Description	Link- Edited AMODE	Link- Edited RMODE
CEEARLU ⁵	Anchor lookup	31	ANY
CEEELLST ⁵	Language list CSECT	31	ANY
CEEBPIRA ⁵	Common initialization	31	ANY
CEEBTRM ⁵	Common termination	31	ANY
IGZCSG ⁵	COBOL signature	31	ANY
IGZCBET ⁵	Common table CSECT	31	ANY
IGZECKP	Checkpoint	31	ANY
IGZECMS ⁴	CMS command handler	31	ANY
IGZEDMR ⁶	Reusable environment deactivation	31	24
IGZEDTE	Date, day, and time of day	31	ANY
IGZEINI ^{2,3,6}	Environment initialization	31	24
IGZEINP ⁶	Accept input reader	31	24
IGZEMSG	Object-time message writer	31	ANY
IGZENRT	NORES termination	31	ANY
IGZEOPN ⁶	OPENS SYSIN and SYSPUNCH in the initial Program Thread (IPT)	31	24
IGZEOUT ⁶	Display output writer	31	24
IGZEPTV	Printer overflow	31	ANY
IGZEQBL ⁶	QSAM initialization transmission verbs, error exits	31	24
IGZEQOC ⁶	QSAM OPEN/ CLOSE	31	24
IGZESCD ⁶	SORT-CONTROL I/O handling routine	31	24
IGZESMG ⁶	Sort/Merge interface	31	24
IGZETCL ¹	Thread cleanup	31	ANY
IGZETRM ⁶	Environment termination	31	24
IGZETSU ¹	Thread setup	31	ANY

Figure 17 (Page 2 of 2). Routines Eligible for Inclusion in the Environment-Specific COBPACK (IGZCPCO)

Name	Description	Link-Edited AMODE	Link-Edited RMODE
IGZEVAM ⁶	VSAM-to-IDCAMS interface	31	24
IGZEVEX ⁶	VSAM exit module for SYNAD and LERAD	31	24
IGZEVIO	VSAM input/output	31	ANY
IGZEVOC	VSAM OPEN/CLOSE	31	ANY
IGZEVOP	VSAM OPEN interface for variable length records	31	ANY
IGZEVSV	VSAM I/O for simulated relative record data sets with variable length records	31	ANY

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 16M address line.
- 2 Must exist outside the OS ESM COBPACK, even if it also exists in it.
- 3 Highly recommended for inclusion in a COBPACK if it is located below the 16M address line.
- 4 IGZECMS is applicable under CMS only and must be available at link-time if the load module is to run under CMS.
- 5 If IGZEINI is included in the COBPACK, the following routines must also be included: CEEARLU, CEEBLLST, CEEBPIRA, CEEBTRM, and IGZCBET.
- 6 This routine is not included in the IBM supplied COBPACK IGZCPCO so that the COBPACK is RMODE(ANY) and will be loaded above the 16M line.

Appendix D. Segment Build Lists (CEEBLSGA/CEEBLSGB)

D.1 CEEBLSGB

```
-----
*****
*           IBM Language Environment for MVS & VM           *
*           Version 1 Release 8 Modification 0               *
*                                                         *
*           Licensed Materials -- Property of IBM           *
*           5688-198 (C) Copyright IBM Corporation 1997    *
*           All Rights Reserved                             *
*****
*           Build List for 'SCEE PSEG' Saved Segment (Below line) *
*           "LE/370" Environment                             *
*****
*
:FORMAT. 2
*
:OBJNAME. SCEE.SEGMENT
:BLDREQ.  CEEBLMOD.CEEBINIT.MODULE
          CEEBLMOD.CEEBLIIA.MODULE
          CEEBLMOD.CEEPIPI.MODULE
          CEEBLMOD.CEEBPICI.MODULE
          EDCBLSP2
          IBMBLMOD.IBMRCOMP.MODULE
          IBMBLMOD.IBMRLIB1.MODULE
          IBMBLMOD.IBMRPTLA.MODULE
:GLOBAL. TXTLIB SCEESPC
:OPTIONS. LOADFUNC ( LSEG CEEBINIT )
          LOADFUNC ( LSEG CEEBLIIA )
          LOADFUNC ( LSEG CEEPIPI )
          LOADFUNC ( LSEG CEEBPICI )
          LOADFUNC ( LSEG IBMRLIB1 )
          LOADFUNC ( LSEG IBMRCOMP )
          LOADFUNC ( LSEG IGZCPAC )
          LOADFUNC ( LSEG IGZPCPCO )
:EOBJNAME.
*
```

Figure 18. Contents of CEEBLSGB Build List

D.2 CEEBLSGA

```
-----
*****
*           IBM Language Environment for MVS & VM           *
*           Version 1 Release 8 Modification 0             *
*                                                         *
*           Licensed Materials -- Property of IBM         *
*           5688-198 (C) Copyright IBM Corporation 1997   *
*           All Rights Reserved                           *
*****
*           Build List for SCEEX PSEG (Above Line only)   *
*****
*
:FORMAT. 2
*
:OBJNAME. SCEEX.SEGMENT
:BLDREQ.  CEEBLMOD.CEECOPP.MODULE
          CEEBLMOD.CEEMUEN0.MODULE
          CEEBLMOD.CEEMUEN2.MODULE
          CEEBLMOD.CEEMUEN3.MODULE
          CEEBLMOD.CEEMUEN4.MODULE
          CEEBLMOD.CEEMUEN5.MODULE
          CEEBLMOD.CEEPLPKA.MODULE
          CEEBLMOD.CEEQMATH.MODULE
          CEEBLNLS.CEEMENU0.MODULE
          CEEBLNLS.CEEMENU2.MODULE
          CEEBLNLS.CEEMENU3.MODULE
          CEEBLNLS.CEEMENU4.MODULE
          CEEBLNLS.CEEMENU5.MODULE
          CEEBLNLS.CEEMJPN0.MODULE
          CEEBLNLS.CEEMJPN2.MODULE
          CEEBLNLS.CEEMJPN3.MODULE
          CEEBLNLS.CEEMJPN4.MODULE
          CEEBLNLS.CEEMJPN5.MODULE
          IGZBLMOD.CEEEV005.MODULE
          IGZBLMOD.IGZINSH.MODULE
          IGZBLMOD.IIGZMSGT.MODULE
          IGZBLNLS.IGZCMGEN.MODULE
          IGZBLNLS.IGZCMGJA.MODULE
          IBMBLMOD.CEEEV010.MODULE
          EDCBLMOD.CEEEV003.MODULE
          EDCBLMOD.EDCZ24.MODULE
          EDCBLMOD.EDCNINSP.MODULE
```

Figure 19 (Part 1 of 2). Contents of CEEBLSGA Build List

```
:OPTIONS. LOADFUNC ( LSEG CEECOPP )
LOADFUNC ( LSEG CEEPLPKA )
LOADFUNC ( LSEG CEEQMATH )
LOADFUNC ( LSEG CEEMUEN0 )
LOADFUNC ( LSEG CEEMUEN2 )
LOADFUNC ( LSEG CEEMUEN3 )
LOADFUNC ( LSEG CEEEV005 )
LOADFUNC ( LSEG IIGZMSGT )
LOADFUNC ( LSEG IGZINSH )
LOADFUNC ( LSEG CEEEV003 )
LOADFUNC ( LSEG EDCZ24 )
LOADFUNC ( LSEG EDCNINSP )
LOADFUNC ( LSEG EDCZUMSG )
LOADFUNC ( LSEG CEEEV010 )
LOADFUNC ( LSEG CEEMENU0 )
LOADFUNC ( LSEG CEEMENU2 )
LOADFUNC ( LSEG CEEMENU3 )
LOADFUNC ( LSEG IGZCMGEN )
LOADFUNC ( LSEG EDCZEMSG )
LOADFUNC ( LSEG CEEMJPN0 )
LOADFUNC ( LSEG CEEMJPN2 )
LOADFUNC ( LSEG CEEMJPN3 )
LOADFUNC ( LSEG CEEMJPN4 )
LOADFUNC ( LSEG CEEMJPN5 )
LOADFUNC ( LSEG IGZCMGJA )
LOADFUNC ( LSEG EDCZJMSG )
:EOBJNAME.
*
```

Figure 19 (Part 2 of 2). Contents of CEEBLSGA Build List

Appendix E. IBM Language Environment VM Run-time Options

This chapter includes descriptions of the Language Environment run-time options. Where noted, some of the run-time options might be used only by a COBOL routine. A quick reference table is provided for convenience. In addition, there is a table that maps Language Environment run-time options to HLL run-time options to help you plan your customization.

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.

E.1 Quick Reference Table of Language Environment Run-Time Options

Figure 20 (Page 1 of 5). Run-Time Options Quick Reference

Run-Time Options	Function	Page
▶▶ ABPERC = ((NONE abcode) , (OVR NONOVR))	Percolates a specified abend.	72
▶▶ ABTERMENC = ((RETCODE ABEND) , (OVR NONOVR))	Sets the enclave termination behavior for an enclave ending with an unhandled condition of severity 2 or greater.	74
▶▶ AIXBLD = ((OFF ON) , (OVR NONOVR))	Invokes the access method services (AMS) for VSAM indexed and relative data sets to complete the file and index definition procedures for COBOL routines.	75
▶▶ ALL31 = ((OFF ON) , (OVR NONOVR))	Indicates whether an application does or does not run entirely in AMODE(31).	76
▶▶ ANYHEAP = ((init_size , incr_size , (ANYWHERE ANY BELOW)) , (OVR NONOVR))	Controls allocation of library heap storage not restricted to below the 16M line.	78
▶▶ (FREE KEEP) , (OVR NONOVR)		
▶▶ NOAUTOTASK = ((OVR NONOVR))		80
▶▶ NOAUTOTASK = ((loadmod , numtasks) , (OVR NONOVR))	Specifies whether Fortran Multitasking Facility is to be used by your program and the number of tasks that are allowed to be active.	

Figure 20 (Page 2 of 5). Run-Time Options Quick Reference

Run-Time Options	Function	Page
▶▶ BELOWHEAP = (((<u>init_size</u> , <u>incr_size</u> , <u>FREE</u> / <u>KEEP</u>) , <u>NONOVR</u>)	Controls allocation of library heap storage below the 16M line.	80
▶▶ CBLOPTS = (((<u>ON</u> / <u>OFF</u>) , <u>NONOVR</u>)	Specifies the format of the argument string on application invocation when the main program is COBOL.	82
▶▶ CBLPSHPOP = (((<u>ON</u> / <u>OFF</u>) , <u>NONOVR</u>)	Controls whether CICS PUSH HANDLE and CICS POP HANDLE commands are issued when a COBOL subprogram is called.	83
▶▶ CBLQDA = (((<u>ON</u> / <u>OFF</u>) , <u>NONOVR</u>)	Controls COBOL QSAM dynamic allocation.	84
▶▶ CHECK = (((<u>ON</u> / <u>OFF</u>) , <u>NONOVR</u>)	Indicates whether "checking errors" within an application should be detected.	84
▶▶ COUNTRY = (((<u>country_code</u>) , <u>NONOVR</u>)	Specifies the default formats for date, time, currency symbol, decimal separator, and the thousands separator based on a country.	85
▶▶ DEBUG = (((<u>ON</u> / <u>OFF</u>) , <u>NONOVR</u>)	Activates the COBOL batch debugging features specified by the "debugging lines" or the USE FOR DEBUGGING declarative.	86
▶▶ DEPTHCONDLMT = (((<u>limit</u>) , <u>NONOVR</u>)	Limits the extent to which conditions can be nested.	87
▶▶ ENVAR = (((<u>string</u>) , <u>NONOVR</u>)	Sets the initial values for the environment variables specified in <i>string</i> .	89
▶▶ ERRCOUNT = (((<u>number</u>) , <u>NONOVR</u>)	Specifies how many conditions of severity 2, 3, and 4 can occur per thread before an enclave terminates abnormally.	90
▶▶ ERRUNIT = (((<u>number</u>) , <u>NONOVR</u>)	Identifies the unit number to which run-time error information is to be directed.	91
▶▶ FILEHIST = (((<u>ON</u> / <u>OFF</u>) , <u>NONOVR</u>)	FILEHIST specifies whether to allow the file definition of a file referred to by a ddname to be changed during run time.	92
▶▶ HEAP = (((<u>init_size</u> , <u>incr_size</u> , <u>ANYWHERE</u> / <u>ANY</u> / <u>BELOW</u>) , <u>KEEP</u> / <u>FREE</u> , <u>initsz24</u> , <u>incrsz24</u>) , <u>NONOVR</u>)	Controls allocation of the heaps.	93

Figure 20 (Page 3 of 5). Run-Time Options Quick Reference

Run-Time Options	Function	Page
<p>▶▶ INQPCOPN = ((<input type="checkbox"/> ON <input type="checkbox"/> OFF) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>INQPCOPN controls whether the OPENED specifier on an INQUIRE by unit statement can be used to determine whether a preconnected unit has had any I/O statements directed to it.</p>	95
<p>▶▶ INTERRUPT = ((<input type="checkbox"/> OFF <input type="checkbox"/> ON) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>Causes attentions recognized by the host operating system to be recognized by Language Environment.</p>	96
<p>▶▶ LIBSTACK = ((<input type="checkbox"/> init_size , <input type="checkbox"/> incr_size , (<input type="checkbox"/> FREE <input type="checkbox"/> KEEP) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>Controls the allocation of the thread's library stack storage.</p>	97
<p>▶▶ MSGFILE = ((<input type="checkbox"/> ddname , <input type="checkbox"/> recfm , <input type="checkbox"/> lrecl , <input type="checkbox"/> blksize) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>Specifies the ddname of the run-time diagnostics file.</p>	99
<p>▶▶ MSGQ = ((<input type="checkbox"/> number) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>Specifies the number of ISI blocks allocated on a per-thread basis during execution.</p>	102
<p>▶▶ NATLANG = ((<input type="checkbox"/> ENU <input type="checkbox"/> JEN <input type="checkbox"/> JPN) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>Specifies the national language to use for the run-time environment.</p>	102
<p>▶▶ <input type="checkbox"/> NONONIPTSTACK <input type="checkbox"/> NONIPTSTACK = ((<input type="checkbox"/> init_size , <input type="checkbox"/> incr_size , (<input type="checkbox"/> BELOW <input type="checkbox"/> ANYWHERE <input type="checkbox"/> ANY) , (<input type="checkbox"/> KEEP <input type="checkbox"/> FREE) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>Controls stack allocation for each thread, except the initial thread, in a multithread environment.</p>	104
<p>▶▶ OCSTATUS = ((<input type="checkbox"/> ON <input type="checkbox"/> OFF) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>Controls whether the OPEN and CLOSE status specifiers are verified.</p>	106
<p>▶▶ <input type="checkbox"/> NOPCF <input type="checkbox"/> PCF</p>	<p>Specifies that Fortran static common blocks are not shared among load modules.</p>	107
<p>▶▶ PLITASKCOUNT = ((<input type="checkbox"/> tasks) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>Controls the maximum number of tasks active at one time while you are running PL/I MTF applications.</p>	107
<p>▶▶ POSIX = ((<input type="checkbox"/> OFF <input type="checkbox"/> ON) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>Specifies whether the enclave can run with the POSIX semantics.</p>	108
<p>▶▶ PRTUNIT = ((<input type="checkbox"/> number) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>Identifies the unit number used for PRINT and WRITE statements that do not specify a unit number.</p>	109
<p>▶▶ PUNUNIT = ((<input type="checkbox"/> number) , (<input type="checkbox"/> OVR <input type="checkbox"/> NONOVR))</p>	<p>Identifies the unit number used for PUNCH statements that do not specify a unit number.</p>	110

Figure 20 (Page 4 of 5). Run-Time Options Quick Reference

Run-Time Options	Function	Page
▶▶ RDRUNIT == (((<i>number</i>)) , (OVR / NONOVR))	Identifies the unit number used for READ statements that do not specify a unit number.	110
▶▶ RECPAD == (((OFF / ON / NONE / ALL / VAR)) , (OVR / NONOVR))	Specifies whether a formatted input record is padded with blanks.	111
▶▶ RPTOPTS == (((OFF / ON)) , (OVR / NONOVR))	Specifies that a report of the run-time options in use by the application be generated.	112
▶▶ RPTSTG == (((OFF / ON)) , (OVR / NONOVR))	Specifies that a report of the storage used by the application be generated at the end of execution.	115
▶▶ RTEREUS == (((OFF / ON)) , (OVR / NONOVR))	Initializes the run-time environment to be reusable when the first COBOL program is invoked.	118
▶▶ SIMVRD == (((OFF / ON)) , (OVR / NONOVR))	Specifies whether your COBOL programs use a VSAM KSDS to simulate variable length relative organization data sets.	120
▶▶ STACK == (((<i>init_size</i> , <i>incr_size</i> , (BELOW / ANYWHERE / ANY)) , (OVR / NONOVR))	Controls the allocation and management of thread-level heap storage.	121
▶▶ (KEEP / FREE) , (OVR / NONOVR)		
▶▶ STORAGE == (((<i>heap_alloc_value</i> , <i>heap_free_value</i> , <i>dsa_alloc_value</i> , <i>reserve_size</i>)) , (OVR / NONOVR))	Controls the value of storage that is allocated and freed.	123
▶▶ TERMTHDACT == (((TRACE / QUIET / MSG / DUMP / UADUMP)) , (OVR / NONOVR))	Sets the level of information produced due to an unhandled error of severity 2 or greater.	126
▶▶ THREADHEAP == (((<i>init_size</i> , <i>incr_size</i> , (ANYWHERE / ANY / BELOW)) , (OVR / NONOVR))	Controls the allocation and management of thread-level heap storage.	131
▶▶ (KEEP / FREE) , (OVR / NONOVR)		
▶▶ (NOTEST / TEST) == (((ALL / ERROR / NONE) , <i>commands_file</i>) , (OVR / NONOVR))	Specifies that a debug tool is to be given control according to the suboptions specified.	128
▶▶ (PROMPT / NOPROMPT / * / ; / <i>command</i>) , (<i>preference_file</i>) , (OVR / NONOVR)		

Figure 20 (Page 5 of 5). Run-Time Options Quick Reference

Run-Time Options	Function	Page
▶▶ TRACE = (((OFF) , table_size , (DUMP) NODUMP) , (LE=0) (LE=1) (LE=2) (LE=3))) , ((OVR) NONOVR))	Determines whether Language Environment run-time library tracing is active.	133
▶▶ TRAP = (((ON) (OFF)) , (OVR) NONOVR))	Specifies how Language Environment routines handle abends and program interrupts.	134
▶▶ UPSI = (((nnnnnnnn) , (OVR) NONOVR))	Sets the eight UPSI switches on or off. Affects only COBOL programs.	137
▶▶ (NOUSRHDLR) (USRHDLR) = (((lmmame) , (OVR) NONOVR))	USRHDLR registers a user condition handler at stack frame 0.	138
▶▶ VCTRSAVE = (((OFF) (ON)) , (OVR) NONOVR))	Specifies whether any language in an application uses the vector facility when user-written condition handlers are called.	139
▶▶ XUFLOW = (((AUTO) (ON) (OFF)) , (OVR) NONOVR))	Specifies whether an exponent underflow causes a program interrupt.	140

E.2 Language Run-Time Option Mapping

Figure 21 (Page 1 of 2). C and Language Environment Options

C Option	Language Environment Equivalent	Notes
ISAINC	STACK	If you don't change the C/370 run-time option ISAINC, you will receive a warning message during execution.
ISASIZE	STACK	If you don't change the C/370 run-time option ISASIZE, you will receive a warning message during execution.
LANGUAGE	NATLANG	Mixed-case and uppercase U.S. English and Japanese are supported. If you don't change the C/370 run-time option LANGUAGE, you will receive a warning message during execution.

Figure 21 (Page 2 of 2). C and Language Environment Options

C Option	Language Environment Equivalent	Notes
REPORT NOREPORT	RPTSTG(ON OFF), RPTOPT(ON OFF)	RPTSTG(ON OFF) and RPTOPT(ON OFF) provide behavior compatible with REPORT NOREPORT, and affects all languages in an enclave. If you don't change the C/370 run-time option REPORT NOREPORT, you will receive a warning message during execution.
SPIE NOSPIE	TRAP(ON OFF)	If SPIE NOSPIE is specified in input, then TRAP is set according to the option: TRAP(ON) for SPIE, and TRAP(OFF) for NOSPIE. If both SPIE NOSPIE and STAE NOSTAE are specified together in input, then TRAP is set according to both options: TRAP(OFF) when both options are negative, and TRAP(ON) otherwise. TRAP(ON) must be in effect for applications to run successfully.
STAE NOSTAE	TRAP(ON OFF)	If STAE NOSTAE is specified in input, then TRAP is set according to the option: TRAP(ON) for STAE, and TRAP(OFF) for NOSTAE. If both SPIE NOSPIE and STAE NOSTAE are specified together in input, then TRAP is set according to both options: TRAP(OFF) when both options are negative, and TRAP(ON) otherwise. TRAP(ON) must be in effect for applications to run successfully.

Figure 22 (Page 1 of 3). COBOL and Language Environment Options

COBOL Option	Language Environment Equivalent	Notes
AIXBLD NOAIXBLD	AIXBLD NOAIXBLD	Access Method Services (AMS) messages are directed to the ddname specified in the Language Environment run-time option MSGFILE when running under MVS. Under CMS, the messages are erased, which is the same behavior as VS COBOL II. AIXBLD NOAIXBLD is not applicable under CICS.
DEBUG NODEBUG	DEBUG NODEBUG	DEBUG NODEBUG provides behavior compatible with VS COBOL II.
FLOW NOFLOW	FLOW NOFLOW	FLOW NOFLOW provides behavior compatible with VS COBOL II.

Figure 22 (Page 2 of 3). COBOL and Language Environment Options

COBOL Option	Language Environment Equivalent	Notes
LANGUAGE	NATLANG	NATLANG replaces LANGUAGE, which is a VS COBOL II installation option. You can select a national language at run time or installation time by using the NATLANG option.
LIBKEEP NOLIBKEEP	Not applicable	LIBKEEP NOLIBKEEP is not supported under Language Environment. To obtain similar performance function, use the Library Routine Retention (LRR) feature described in <i>Language Environment for MVS & VM Programming Guide</i> and <i>Language Environment for MVS & VM Installation and Customization on MVS</i> . The LIBKEEP NOLIBKEEP option is not applicable under CICS.
MIXRES NOMIXRES	Not applicable	MIXRES NOMIXRES is not supported under Language Environment. MIXRES applications supported by Language Environment always exhibit RES behavior. For more information, see <i>COBOL/370 and COBOL for MVS & VM Compiler and Run-Time Migration Guide</i> . MIXRES NOMIXRES is not applicable under CICS.
RTEREUS NORTEREUS	RTEREUS NORTEREUS	RTEREUS NORTEREUS provides similar behavior to the VS COBOL II RTEREUS option, but it will not work if you are using more than one language. RTEREUS is not recommended. RTEREUS NORTEREUS is not applicable under CICS.
SIMVRD NOSIMVRD	SIMVRD NOSIMVRD	SIMVRD NOSIMVRD provides behavior compatible with VS COBOL II.
SPOUT NOSPOUT	RPTOPTS(ON OFF), RPTSTG(ON OFF)	Storage reports are directed to the ddname specified in the Language Environment option MSGFILE. For information about report formats and tuning programs, see <i>COBOL/370 and COBOL for MVS & VM Compiler and Run-Time Migration Guide</i> .
SSRANGE NOSSRANGE	CHECK(ON OFF)	CHECK(ON OFF) provides behavior compatible with SSRANGE NOSSRANGE.
STAE NOSTAE	TRAP(ON OFF)	If STAE NOSTAE is specified in input, then TRAP is set according to the option: TRAP(ON) for STAE, and TRAP(OFF) for NOSTAE. TRAP(ON) must be in effect for applications to run successfully.

Figure 22 (Page 3 of 3). COBOL and Language Environment Options

COBOL Option	Language Environment Equivalent	Notes
UPSI	UPSI	UPSI provides behavior compatible with VS COBOL II.
WSCLEAR NOWSCLEAR	STORAGE(00)	For behavior similar to WSCLEAR NOWSCLEAR, use the Language Environment STORAGE(00) option. For more information, see the <i>COBOL/370 and COBOL for MVS & VM Compiler and Run-Time Migration Guide</i> .

Figure 23 (Page 1 of 3). Fortran and Language Environment Options

Fortran Option	Language Environment Equivalent	Notes
ABSDUMP NOABSDUMP	TERMTHDACT	TERMTHDACT(DUMP) replaces ABSDUMP to produce a Language Environment dump at termination. TERMTHDACT with suboptions TRACE, QUIET, or MSG replaces NOABSDUMP to avoid getting a Language Environment dump at termination.
AUTOTASK NOAUTOTASK	AUTOTASK NOAUTOTASK	AUTOTASK NOAUTOTASK provides behavior compatible with VS FORTRAN Version 2.
CNVIOERR NOCNVIOERR	Not applicable	There is no Language Environment equivalent for CNVIOERR NOCNVIOERR. Fortran semantics will behave as if CNVIOERR is in effect.
DEBUG NODEBUG	Not applicable	The Debug Tool does not support Fortran.
DEBUNIT	Not applicable	There is no Language Environment equivalent for DEBUNIT. If specified, you will receive an informational message during execution.
ECPACK NOECPACK	Not applicable	There is no Language Environment equivalent for ECPACK NOECPACK. You cannot run programs with Language Environment that use access registers or that were compiled with the EC or EMODE compiler options.
ERRUNIT	ERRUNIT	ERRUNIT provides behavior compatible with VS FORTRAN Version 2.
FAIL	ABTERMENC	ABTERMENC replaces FAIL. ABTERMENC must be specified to control whether a condition of severity 2 or greater is terminated with a return code or an abend.

Figure 23 (Page 2 of 3). Fortran and Language Environment Options

Fortran Option	Language Environment Equivalent	Notes
FILEHIST NOFILEHIST	FILEHIST NOFILEHIST	FILEHIST NOFILEHIST provides behavior compatible with VS FORTRAN Version 2.
INQPCOPN NOINQPCOPN	INQPCOPN NOINQPCOPN	INQPCOPN NOINQPCOPN provides behavior compatible with VS FORTRAN Version 2.
IOINIT NOIOINIT	Not applicable	There is no Language Environment equivalent for IOINIT NOIOINIT. The message file is opened only when the first record is written to it. If no allocation for the ddname has been made for the message file, it is dynamically allocated to the terminal (under TSO) or to SYSOUT=* (under MVS batch).
OCSTATUS NOOCSTATUS	OCSTATUS NOOCSTATUS	OCSTATUS NOOCSTATUS provides behavior compatible with VS FORTRAN Version 2.
PARALLEL NOPARALLEL	Not applicable	There is no Language Environment equivalent for PARALLEL NOPARALLEL. Parallel programs cannot be run with Language Environment. If specified, you will receive an informational message during execution.
PRTUNIT	PRTUNIT	PRTUNIT provides behavior compatible with VS FORTRAN Version 2.
PTRACE NOPTRACE	Not applicable	There is no Language Environment equivalent for PTRACE NOPTRACE. Parallel programs cannot be run with Language Environment. If specified, you will receive an informational message during execution.
PUNUNIT	PUNUNIT	PUNUNIT provides behavior compatible with VS FORTRAN Version 2.
RDRUNIT	RDRUNIT	RDRUNIT provides behavior compatible with VS FORTRAN Version 2.
RECPAD NORECPAD RECPAD(VAR)	RECPAD(OFF NONE VAR ALL ON)	NORECPAD automatically maps to RECPAD(OFF). RECPAD(VAR) provides behavior compatible with VS FORTRAN Version 2. RECPAD must be changed to RECPAD(ON).
SPIE NOSPIE, STAE NOSTAE	TRAP(ON OFF)	If either SPIE or STAE is specified in input, TRAP is set to TRAP(ON). If both NOSPIE and NOSTAE are specified, TRAP is set to TRAP(OFF). TRAP(ON) must be in effect for applications to run successfully.

Figure 23 (Page 3 of 3). Fortran and Language Environment Options

Fortran Option	Language Environment Equivalent	Notes
XUFLOW NOXUFLOW	XUFLOW(ON AUTO) XUFLOW(OFF)	There is no automatic mapping of XUFLOW to the Language Environment XUFLOW. NOXUFLOW maps to the Language Environment XUFLOW(OFF), which provides compatible behavior.

Figure 24 (Page 1 of 2). PL/I and Language Environment Options

PL/I Option	Language Environment Equivalent	Notes
COUNT NOCOUNT	Not applicable	There is no Language Environment equivalent for COUNT NOCOUNT. It is not processed but produces an informational message.
FLOW NOFLOW	Not applicable	There is no Language Environment equivalent for FLOW NOFLOW. Language Environment honors this option only as a COBOL option.
ISAINC	STACK, NONIPTSTACK, or PLITASKCOUNT	ISAINC maps to three Language Environment options, STACK, NONIPTSTACK, and PLITASKCOUNT, which provide compatible behavior.
ISASIZE	STACK, NONIPTSTACK, or PLITASKCOUNT	ISASIZE maps to three Language Environment options, STACK, NONIPTSTACK, and PLITASKCOUNT, which provide compatible behavior.
LANGUAGE	NATLANG	Mixed-case and uppercase U.S. English and Japanese are supported.
REPORT NOREPORT	RPTSTG(ON OFF), RPTOPTS(ON OFF)	RPTSTG(ON OFF) and RPTOPTS(ON OFF) provide behavior compatible with REPORT NOREPORT.
SPIE NOSPIE	TRAP(ON OFF)	If SPIE NOSPIE is specified in input, then TRAP is set according to the option: TRAP(ON) for SPIE, and TRAP(OFF) for NOSPIE. If both SPIE NOSPIE and STAE NOSTAE are specified together in input, then TRAP is set according to both options: TRAP(OFF) when both options are negative, and TRAP(ON) otherwise. TRAP(ON) must be in effect for applications to run successfully.

Figure 24 (Page 2 of 2). PL/I and Language Environment Options

PL/I Option	Language Environment Equivalent	Notes
STAE NOSTAE	TRAP(ON OFF)	If STAE NOSTAE is specified in input, then TRAP is set according to the option: TRAP(ON) for STAE, and TRAP(OFF) for NOSTAE. If both SPIE NOSPIE and STAE NOSTAE are specified together in input, then TRAP is set according to both options: TRAP(OFF) when both options are negative, and TRAP(ON) otherwise. TRAP(ON) must be in effect for applications to run successfully.
TASKHEAP	THREADHEAP	THREADHEAP provides behavior compatible with TASKHEAP.

E.2.1 COBOL Compatibility

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

For example:

Under MVS:

CBLOPTS=OFF:

```
//GO EXEC PGM=PROGRAM1,PARM='AIXBLD/'
```

CBLOPTS=ON:

```
//GO EXEC PGM=PROGRAM1,PARM=' /AIXBLD'
```

Under VM:

CBLOPTS=OFF:

```
LOAD
START * AIXBLD/
```

CBLOPTS=ON:

ABPERC

```
LOAD  
START * /AIXBLD
```

E.3 Language Environment 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 Language Environment run-time options, see *Language Environment for MVS & VM 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** A warning 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 Language Environment callable service, it is always signaled to the condition manager instead of being returned synchronously to the caller.

E.3.1 ABPERC

ABPERC percolates an abend whose code you specify. TRAP(ON) must be in effect for ABPERC to have an effect.

The ABPERC option is a debug tool that specifies the application can run with the TRAP run-time option set to ON. This provides Language Environment semantics for everything except one abend, whose code you specify.

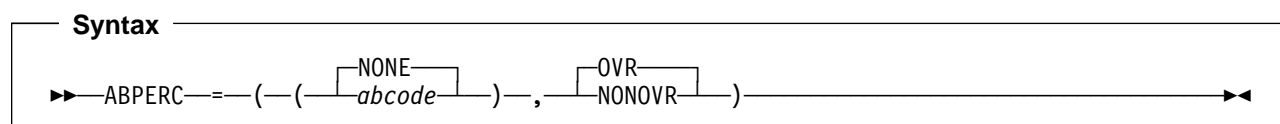
When you run with ABPERC and encounter the specified abend:

- User condition handlers are not enabled.
- In OpenEdition MVS, POSIX signal handling semantics are not enabled for the abend.
- No storage report or run-time options report is generated.

- No Language Environment messages or Language Environment 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 Language Environment, so records might be lost.
- Resources acquired by Language Environment are not freed.
- The debug tool is not notified of the error.

You can also use the CEEBXITA assembler user exit to specify a list of abend codes for Language Environment to percolate.

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



NONE

Specifies that all abends are handled according to Language Environment condition handling semantics.

abcode

Specifies the code number of the abend to percolate.

abcode can be specified as:

Shhh A system abend code where *hhh* is the hex system abend code

Udddd 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 abend code with this option. However, an abend U0000 is interpreted in the same way as S000.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.1.1 Usage Notes

- Language Environment ignores ABPERC(0Cx). In this instance, no abend is percolated, and Language Environment condition handling semantics are in effect.
- CICS consideration—ABPERC is ignored under CICS.
- OpenEdition consideration—ABPERC percolates an abend regardless of the thread in which it occurs.

ABTERMENC

E.3.1.2 For More Information

- For more information about the assembler user exit (CEEEXITA), see *Language Environment for MVS & VM Programming Guide*.

E.3.2 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.

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

Syntax

```
▶▶ ABTERMENC = ( ( RETCODE  
ABEND ) , ( OVR  
NONOVR ) ) ▶▶
```

RETCODE

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

However, the 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, Language Environment 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, Language Environment issues an abend to terminate the enclave. Language Environment 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, Language Environment 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.

ABEND

Specifies that Language Environment issues an abend to end the enclave regardless of the setting of the CEEAUE_ABND flag by the assembler user exit. However, the setting of the CEEAUE_ABND flag affects the abend processing, as follows:

When CEEAUE_ABND is set to OFF, the following occurs:

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

- Abend task/step attribute (on MVS): An abend is issued to terminate the task.

When CEEAUE_ABND is set to ON, Language Environment 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: Language Environment requests a system dump only if the assembler user exit sets CEEAUE_DUMP to ON. The system abend dump goes to the system abend ddname with the filename you define in your JCL (for MVS) or in your FILEDEF (for VM). The filename is the name defined in the DD card.
- Abend task/step attribute (on MVS): If the assembler user exit sets CEEAUE_STEPS to ON, Language Environment issues an abend to terminate the step. Otherwise, Language Environment issues an abend to terminate the task.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.2.1 Usage Notes

- COBOL considerations—ABEND is the recommended setting for COBOL customers who use MVS. Your system administrator can change the default value of ABTERMENC to ABEND.
- CICS consideration—The default under CICS is ABTERMENC(ABEND,OVR).

E.3.2.2 For More Information

- For information about return code calculation CEEAUE_RETURN, CEEAUE_ABND, and assembler user exit CEEBXTA processing, see *Language Environment for MVS & VM Programming Guide*.
- For more information about abend codes, see *Language Environment for MVS & VM Programming Guide*.
- For a list of abend code values, see *Language Environment for MVS & VM Programming Guide*.

E.3.3 AIXBLD (COBOL Only)

AIXBLD invokes the access method services (AMS) for VSAM indexed and relative data sets (KSDS and RRDS) to complete the file and index definition procedures for COBOL programs.

AIXBLD conforms to the ANSI 1985 COBOL standard.

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

ALL31

Syntax

```
▶ AIXBLD=( (OFF), (ON) ), (OVR, NONOVR) ▶▶
```

OFF

Does not invoke the access method services for VSAM indexed and relative data sets.

ON

Invokes the access method services for VSAM indexed and relative data sets. AIXBLD can be abbreviated AIX.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.3.1 Usage Notes

- The only valid abbreviations for AIXBLD and NOAIXBLD are AIX and NOAIX, respectively.
- When specifying this option in CEEDOPT or CEEUOPT, use the syntax AIXBLD(ON) or AIXBLD(OFF). Use AIXBLD and NOAIXBLD only on the command line.
- CICS consideration—This option is ignored under CICS.
- MVS consideration—If you also specify the MSGFILE run-time option, the access method services messages are directed to the MSGFILE *ddname* or to the default SYSOUT.

E.3.3.2 Performance Considerations

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

E.3.3.3 For More Information

- See *COBOL/370 Programming Guide* or *COBOL for MVS & VM Programming Guide* for more details.
- See E.3.23, “MSGFILE” on page 99 for information about the MSGFILE run-time option.

E.3.4 ALL31

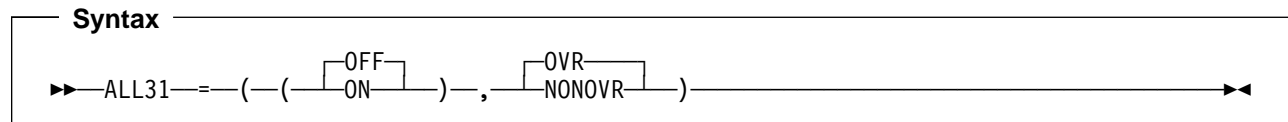
ALL31 specifies whether an application can run entirely in AMODE 31 or whether the application has one or more AMODE 24 routines.

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 Language Environment does not support the invocation of a nested enclave requiring ALL31(OFF) from an enclave running with ALL31(ON).

In a multithread environment, Language Environment invokes all start routines, which are specified in a C pthread_create() function call, in AMODE 31. However, for PL/I MTF applications, Language Environment provides AMODE switching. Thus, the first routine of a task can be in AMODE 24.

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



OFF

Indicates that one or more routines of a Language Environment application are AMODE 24.

With ALL31(OFF) specified:

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

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

ON

Indicates that no user routines of a Language Environment application are AMODE 24.

With ALL31(ON) specified:

- AMODE switching across calls to Language Environment common run-time routines is minimized. For example, no AMODE switching is performed on calls to Language Environment 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.

E.3.4.1 Usage Notes

- CICS consideration—The default under CICS is ALL31=((ON),OVR).
- OpenEdition consideration—The ALL31 option applies to the enclave.

ANYHEAP

E.3.4.2 Performance Consideration

If your application consists entirely of AMODE 31 routines, it might run faster and use less below-the-line storage with ALL31(ON) than with ALL31(OFF), since mode switching code is not required.

E.3.4.3 For More Information

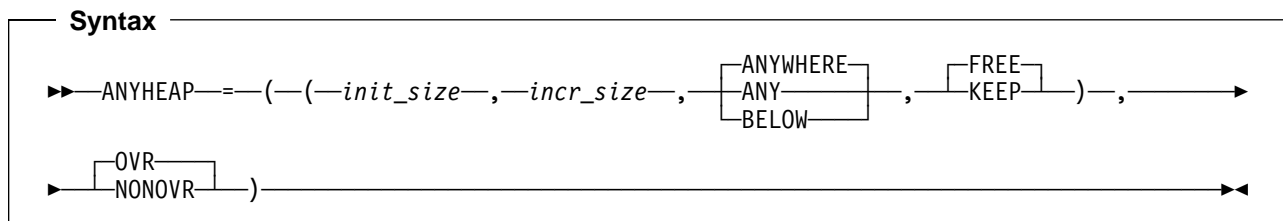
- See E.3.39, "STACK" on page 121 for information about the STACK run-time option.

E.3.5 ANYHEAP

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

The ANYHEAP option is always in effect. If you do not specify ANYHEAP or if you specify ANYHEAP(0), Language Environment allocates the value of 16K when a call is made to get heap storage.

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



init_size

Determines the minimum initial size of the anywhere heap storage. This value can be specified as *n*, *nK*, or *nM* 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*, *nK*, or *nM* 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 16M line. If there is no storage available above the line, storage is acquired below the line. On systems that do not support bimodal addressing (for example, when VM/ESA* is initially loaded in 370 mode), this option is ignored and heap storage is placed below 16M.

The only valid abbreviation for ANYWHERE is ANY.

BELOW

Specifies that heap storage must be allocated below the 16M 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.

E.3.5.1 Usage Notes

- CICS consideration—Under CICS, ANYHEAP assumes the defaults ANYHEAP=((4K,4K,ANYWHERE,FREE),OVR). Both the initial size and the increment size are rounded up to the nearest multiple of 8 bytes. The minimum is 4K. If you specify ANYHEAP or ANYHEAP(0), Language Environment assumes the default value of 4K. The maximum initial and increment size for ANYHEAP under CICS is 1 gigabyte (1024M).
- OpenEdition consideration—The ANYHEAP option applies to the enclave.

E.3.5.2 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 the report numbers to help determine what values to specify.

E.3.5.3 For More Information

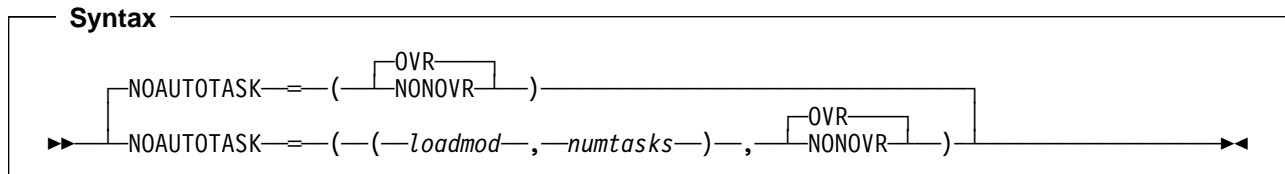
- See *Language Environment for MVS & VM Programming Guide* for more information about Language Environment heap storage.
- See E.3.36, “RPTSTG” on page 115 for more information about the RPTSTG run-time option.
- For more information about heap storage tuning with storage report numbers, see *Language Environment for MVS & VM Programming Guide*.

BELOWHEAP

E.3.6 AUTOTASK | NOAUTOTASK (Fortran Only)

AUTOTASK specifies whether Fortran Multitasking Facility is to be used by your program and the number of tasks that are allowed to be active.

IBM-Supplied Default: NOAUTOTASK=(OVR)



NOAUTOTASK

Disables the MTF and nullifies the effects of previous specifications of AUTOTASK parameters.

loadmod

The name of the load module that contains the concurrent subroutines that run in the subtasks created by MTF.

numtasks

The number of subtasks created by MTF. This value can range from 1 through 99.

OVR

Specifies that the option can be overridden.

NONOVR

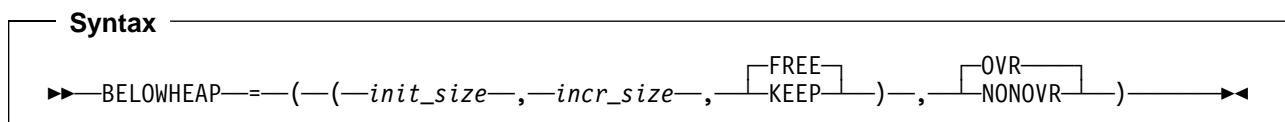
Specifies that the option cannot be overridden.

E.3.7 BELOWHEAP

BELOWHEAP controls the allocation of library heap storage that must be located below the 16M 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 value of 8K is allocated when a call is made to get heap storage.

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



init_size

Determines the minimum initial size of the below heap storage. This value can be specified as *n*, *nK*, or *nM* 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 16M line, and is specified in *n*, *nK*, or *nM* 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.

E.3.7.1 Usage Notes

- CICS considerations—Under CICS, BELOWHEAP assumes the defaults BELOWHEAP=((4K,4K,FREE),OVR).

Both the initial size and the increment size are rounded to the nearest multiple of 8 bytes. The minimum is 4K. If you specify BELOWHEAP(0), both *init_size* and *incr_size* assume the IBM-supplied default of 4K.

- OpenEdition consideration—The BELOWHEAP option applies to the enclave.

E.3.7.2 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 its numbers to help determine what values to specify.

E.3.7.3 For More Information

- See *Language Environment for MVS & VM Programming Guide* for more information about Language Environment heap storage.
- See E.3.36, “RPTSTG” on page 115 for more information about the RPTSTG run-time option.
- For more information about tuning your application with storage report numbers, see *Language Environment for MVS & VM Programming Guide*.

CBLOPTS

E.3.8 CBLOPTS (COBOL Only)

CBLOPTS specifies the format of the parameter string on application invocation when the main program 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.

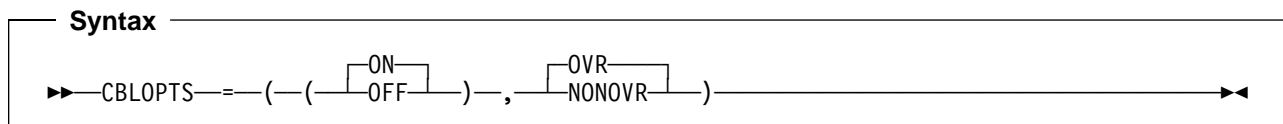
When you specify the ON suboption of CBLOPTS in CEEUOPT or CEEDOPT, the run-time options and program arguments specified in the JCL or on the command line are honored in the following order:

program arguments/run-time options

This order is the reverse of that normally honored by Language Environment.

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)



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.

E.3.8.1 For More Information

- For more information about CEEUOPT or CEEDOPT, see *Language Environment for MVS & VM Installation and Customization on MVS*.

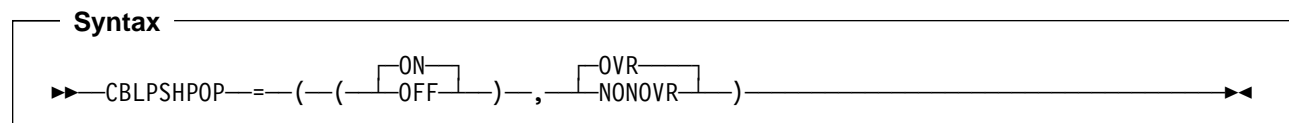
E.3.9 CBLPSHPOP (COBOL Only)

CBLPSHPOP controls whether CICS PUSH HANDLE and CICS POP HANDLE commands are issued when a COBOL (VS COBOL II, COBOL/370, or COBOL for MVS & VM) subroutine is called.

Specify CBLPSHPOP(ON) to avoid compatibility problems when calling VS COBOL II, COBOL/370, or COBOL for MVS & VM 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: CBLPSHPOP=((ON),OVR)



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.

E.3.9.1 Performance Consideration

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

E.3.9.2 For More Information

- For more information about CEEUOPT, see *Language Environment for MVS & VM Programming Guide*.

CHECK

E.3.10 CBLQDA (COBOL Only)

CBLQDA controls COBOL QSAM dynamic allocation on an OPEN statement.

CBLQDA does not affect dynamic storage allocation for the message file specified in MSGFILE or the dump file.

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

Syntax

```
► CBLQDA = ( ( ON  
OFF ) , ( OVR  
NONOVR ) )
```

ON

Specifies that COBOL QSAM dynamic allocation is permitted. ON conforms to the 1985 COBOL Standard.

OFF

Specifies that COBOL QSAM dynamic allocation is not permitted.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.10.1 Usage Note

- CICS consideration—This option is ignored under CICS.
- MVS consideration—You should use CBLQDA(OFF) under MVS, because this prevents a temporary data set from being created in case there is a misspelling in your JCL. If you specify CBLQDA(ON) and have a misspelling in your JCL, Language Environment creates a temporary file, writes to it, and then MVS deletes it. You receive a return code of 0 but no output.

E.3.11 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=((ON),OVR)

Syntax

```
► CHECK = ( ( ON  
OFF ) , ( OVR  
NONOVR ) )
```

ON

Specifies that run-time checking is performed.

OFF

Specifies that run-time checking is not performed.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.11.1 Usage Note

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

E.3.11.2 Performance Consideration

If your COBOL program was compiled with SSRANGE, and you are not testing or debugging an application, performance improves when you specify CHECK(OFF).

E.3.12 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 Language Environment NLS services, not the Language Environment locale callable services.

You can set the country value using the run-time option COUNTRY or the callable service CEE3CTY.

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) with US signifying the United States.

Syntax

```
►► COUNTRY=--(--country_code--), --OVRNONOVR--►►
```

country_code

A 2-character code that indicates to Language Environment the country on which to base the default settings.

OVR

Specifies that the option can be overridden.

DEBUG

NONOVR

Specifies that the option cannot be overridden.

E.3.12.1 Usage Notes

- If you specify a *country_code* that is not supported by Language Environment, Language Environment accepts the value and issues an informational message. When you specify an unavailable country code, you must provide a message template for that code.

CEEUOPT and CEEDOPT permit the specification of an unavailable country code, but give a return code of 4 and a warning message.

- C/C++ consideration—Language Environment provides locales used in C and 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 Language Environment NLS and date and time services. `setlocale()` and CEESETL affect only C/C++ locale-sensitive functions and Language Environment locale callable services.

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

- OpenEdition consideration—The COUNTRY option sets the initial value for the enclave.

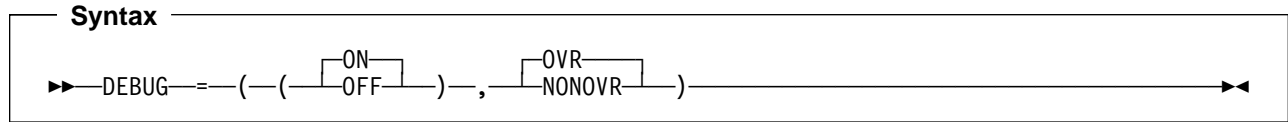
E.3.12.2 For More Information

- For more information about the CEE3CTY callable service, see *Language Environment for MVS & VM Programming Reference*.
- See Appendix F, “Language Environment National Language Support Country Codes” on page 142 for a list of countries and their codes.
- For more information about the CEESETL callable service, see *Language Environment for MVS & VM Programming Reference*.
- For more information on `setlocale()`, see *AD/Cycle C/370 Programming Guide*, *C/MVS Programming Guide*, or *C++/MVS Programming Guide*.

E.3.13 DEBUG (COBOL Only)

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

IBM-Supplied Default: `DEBUG=((ON),OVR)`

**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.

OFF

Suppresses the COBOL batch debugging features.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.13.1 Usage Note

- When specifying this option in CEEDOPT or CEEUOPT, use the syntax DEBUG(ON) or DEBUG(OFF). Use DEBUG and NODEBUG only on the command line.

E.3.13.2 Performance Consideration

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

E.3.13.3 For More Information

- See *COBOL/370 Programming Guide* or *COBOL for MVS & VM Programming Guide* for more details on the USE FOR DEBUGGING declarative.

E.3.14 DEPTHCONDLMT

DEPTHCONDLMT specifies the extent to which conditions can be nested. Figure 25 on page 88 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.

DEPTHCONDLMT

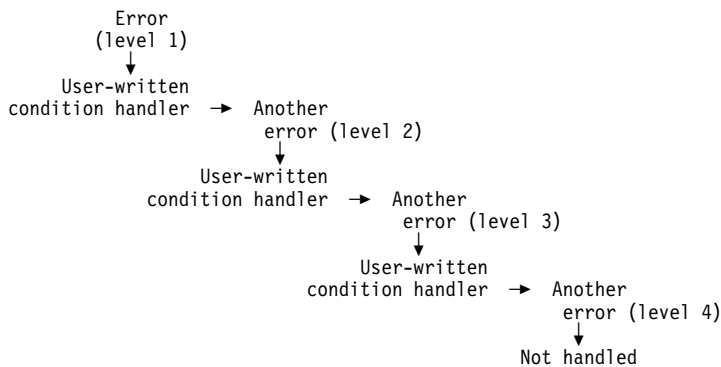


Figure 25. Effect of DEPTHCONDLMT(3) on Condition Handling

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

Syntax

```
▶▶ DEPTHCONDLMT= ( ( (limit) ) , ( OVR / NONOVR ) ) ▶▶
```

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 1 value specifies handling of the initial condition, but does not allow handling of nested conditions that occur while handling a condition. With a 5 value, 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.

E.3.14.1 Usage Notes

- PL/I consideration—DEPTHCONDLMT(0) provides PL/I compatibility.
- PL/I MTF consideration—In a PL/I MTF application, DEPTHCONDLMT sets the limit for how many nested synchronous conditions are allowed for a PL/I task. If the number of nested conditions exceeds the limit, the application terminates abnormally.
- OpenEdition consideration—The DEPTHCONDLMT option sets the limit for how many nested

synchronous conditions are allowed for a thread. Asynchronous signals do not affect DEPTHCONDLMT.

E.3.14.2 For More Information

- For more information on nested conditions, see *Language Environment for MVS & VM Programming Guide*.

E.3.15 ENVAR

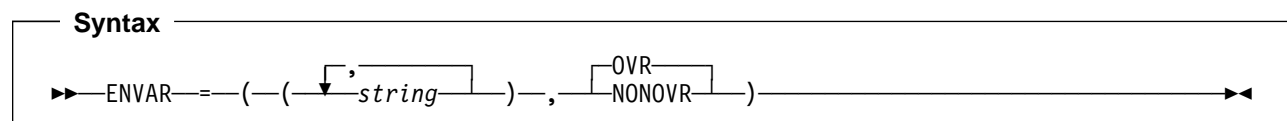
ENVAR sets the initial values for the environment variables specified in *string*. With ENVAR, you can pass into the application switches or tagged information that can then be accessed using the C functions `getenv`, `setenv`, and `clearenv`.

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)`



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.

ERRCOUNT

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.15.1 Usage Notes

- The ENVAR option functions independently of the POSIX run-time option setting.
- C consideration—An application can access the environment variables using C function `getenv` or the POSIX variable `environ`, which is defined as:

```
extern char **environ;
```

Access through `getenv` is recommended, especially in a multithread environment.

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. Environment variables that are propagated across the EXEC override those established by the ENVAR option. `getenv` serializes access to the environment variables.

- 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.
- OpenEdition consideration—The environment variables apply to the enclave.

E.3.15.2 For More Information

- For more information about the RPTOPTS run-time option, see E.3.35, “RPTOPTS” on page 112.

E.3.16 ERRCOUNT

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

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

Syntax

```
▶▶ ERRCOUNT = ( ( number ) , [ OVR / NONOVR ] ) ▶▶
```

number

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.16.1 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—Language Environment counts severity 1 messages with the facility ID IGZ. When the limit is reached, additional severity 1 messages are suppressed.
- PL/I consideration—Use the default setting of ERRCOUNT(0) if you are using PL/I.
- PL/I MTF consideration—In a PL/I MTF application, ERRCOUNT sets the threshold for the total number severity 2, 3, and 4 synchronous conditions that can occur for each task. If the number of conditions exceeds the threshold, the application terminates normally.
- OpenEdition consideration—Synchronous signals that are associated with a condition of severity 2, 3, and 4 do not affect ERRCOUNT. Asynchronous signals do not affect ERRCOUNT.
- C++ consideration—The ERRCOUNT option sets the threshold for the total number of severity 2, 3, and 4 synchronous conditions that can occur. Note that each thrown object is considered a severity 3 condition. However, this condition does not affect ERRCOUNT.

E.3.16.2 For More Information

- For more information about the CEEDHLR callable service, see *Language Environment for MVS & VM Programming Reference*.
- For more information about the CEESGL callable service, see *Language Environment for MVS & VM Programming Reference*.
- See *Language Environment for MVS & VM Programming Guide* for more information about the facility ID part of messages.

E.3.17 ERRUNIT (Fortran Only)

ERRUNIT identifies the unit number to which run-time error information is to be directed.

IBM-Supplied Default: ERRUNIT=((6),OVR)

Syntax

►► ERRUNIT= ((*number*) , (OVR / NONOVR)) ◀◀

FILEHIST

number

A valid unit number in the range 0-99. You can establish your own default number at installation time. The Language Environment message file and the file connected to the Fortran error message unit are the same.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.18 FILEHIST (Fortran Only)

FILEHIST specifies whether to allow the file definition of a file referred to by a ddname to be changed during run time. This option is intended for use with applications called by Fortran that reallocate Fortran's supplied DD names.

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

Syntax

```
▶ FILEHIST=( ( ON / OFF ) , ( OVR / NONOVR ) ) ▶
```

ON

Causes the history of a file to be used in determining its existence. It checks to see whether:

- The file was ever internally opened (in which case it exists)
- The file was deleted by a CLOSE statement (in which case it does not exist).

OFF

Causes the history of a file to be disregarded in determining its existence.

If you specify FILEHIST(OFF), you should consider:

- **If you change file definitions during run time**, the file is treated as if it were being opened for the first time. Before the file definition can be changed, the existing file must be closed.
- **If you do not change file definitions during run time**, you must use STATUS='NEW' to re-open an empty file that has been closed with STATUS='KEEP', because the file does not appear to exist to Fortran.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.19 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.

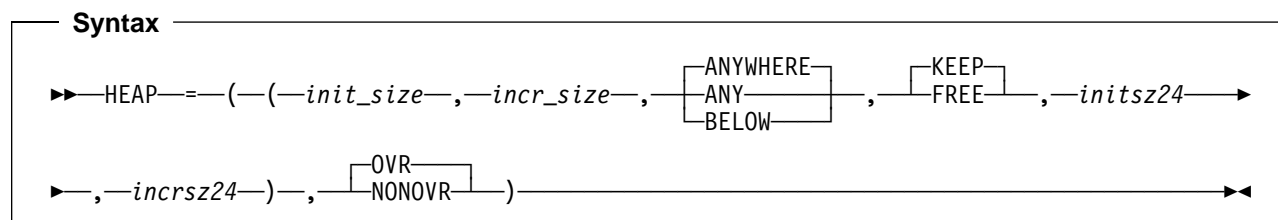
Heaps are storage areas where you allocate memory for user-controlled dynamically allocated variables such as:

- 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

The HEAP option is always in effect. If you do not specify HEAP, Language Environment allocates the default value of heap storage when a call is made to get heap storage.

Language Environment does not allocate heap storage until the first call to get heap storage is made. You can get heap storage by using language constructs or by making a call to CEEGTST.

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



init_size

Determines the minimum initial allocation of heap storage. Specify this value as *n*, *nK*, or *nM* 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*, *nK*, or *nM* 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 16M line. If there is no available storage above the line, storage is acquired below the line. On systems that do not support bimodal addressing (for example, when VM/ESA is initially loaded in 370 mode), Language Environment ignores this option and places the heap storage below 16M.

The only valid abbreviation of ANYWHERE is ANY.

HEAP

BELOW

Specifies that you must allocate heap storage below the 16M 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 16M line for applications running with ALL31(OFF) when these applications specify ANYWHERE in the HEAP run-time option. Specify *initsz24* as *n*, *nK*, or *nM* number of bytes. The amount of storage is rounded up to the nearest multiple of 8 bytes.

initsz24 applies to the initial heap and other heaps created with the CEECRHP callable service that are not allocated strictly below the 16M line.

incrsz24

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

incrsz24 applies to the initial heap and other heaps created with the CEECRHP callable service that are not allocated strictly below the 16M line.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.19.1 Usage Notes

- Applications running in AMODE 24 that request heap storage get the storage below the 16M line regardless of the setting of ANYWHERE | BELOW.
- COBOL consideration—You can use the HEAP option to provide function similar to the VS COBOL II space management tuning table.
- C/C++ consideration—If your C application runs below the 16M (AMODE 24) line, you must specify HEAP(,BELOW,,) as an installation default for the HEAP run-time option, on the command line when invoking the program, or at compile time as a #pragma runopts.
- PL/I consideration—The ANYWHERE | BELOW and KEEP | FREE suboptions are positional. ANYWHERE | BELOW must be in the third position, and KEEP | FREE must be in the fourth position. If you want to omit *init_size* and *incr_size*, you must specify: HEAP(,,ANY,KEEP).

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(,ANY) is in effect.
- The main routine is AMODE 31.

In pre-Language Environment-conforming PL/I, the ANYWHERE | BELOW and KEEP | FREE suboptions were not positional. They could be in any order respective to each other. If *init_size* and/or *incr_size* was not specified, the suboptions could be in the first or second position as well.

- CICS consideration—If HEAP is not specified or if HEAP(0) is specified, Language Environment uses the IBM-supplied default of HEAP=((4K,4K,ANYWHERE,KEEP,4K,4K),OVR). Both the initial HEAP allocation and HEAP increments are rounded to the next higher multiple of 8 bytes (not 4K bytes). The minimum is 4K bytes.

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.

- PL/I MTF consideration—In a PL/I MTF application, HEAP specifies the heap storage allocation and management for a PL/I main task.
- OpenEdition considerations—The HEAP option applies to the enclave.

Under OpenEdition, heap storage is managed at the thread level using `pthread_key_create`, `pthread_setspecific`, and `pthread_getspecific`.

E.3.19.2 Performance Considerations

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

E.3.19.3 For More Information

- See *Language Environment for MVS & VM Programming Guide* for more information about Language Environment heap storage or about specifying run-time options at application invocation.
- For more information about the CEECRHP callable service, see *Language Environment for MVS & VM Programming Reference*.
- For more information about the CEEGTST callable service, see *Language Environment for MVS & VM Programming Reference*.
- See E.3.36, “RPTSTG” on page 115 for more information about the RPTSTG run-time option.

E.3.20 INQPCOPN (Fortran Only)

INQPCOPN controls whether the OPENED specifier on an INQUIRE by unit statement can be used to determine whether a preconnected unit has had any I/O statements directed to it.

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

INTERRUPT

Syntax

```
▶ INQPCOPN = ( ( ON  
OFF ) , ( OVR  
NONOVR ) ) ▶
```

ON

Causes the running of an INQUIRE by unit statement to provide the value *true* in the variable or array element given in the OPENED specifier if the unit is connected to a file. This includes the case of a preconnected unit, which can be used in an I/O statement without running an OPEN statement, even if no I/O statements have been run for that unit.

OFF

Causes the running of an INQUIRE by unit statement to provide the value *false* for the case of a preconnected unit for which no I/O statements other than INQUIRE have been run.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.21 INTERRUPT

INTERRUPT causes attention interrupts recognized by the host system to be recognized by Language Environment after the Language Environment environment has been initialized. The way you request an attention interrupt varies from operating system to operating system. When you request the interrupt, you can give control to your application or to a debug tool.

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

Syntax

```
▶ INTERRUPT = ( ( OFF  
ON ) , ( OVR  
NONOVR ) ) ▶
```

OFF

Specifies that Language Environment does not recognize attention interrupts.

ON

Specifies that Language Environment recognizes attention interrupts. In addition, if you have specified the TEST(ERROR) or TEST(ALL) run-time option, the interrupt causes the debug tool to gain control.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.21.1 Usage Notes

- PL/I consideration—Language Environment supports the PL/I method of polling code. Note that the PL/I routine must be compiled with the INTERRUPT compiler option in order for the INTERRUPT run-time option to have an effect.
- PL/I MTF consideration—To receive the attention interrupt, the PL/I routine must be compiled with the INTERRUPT compiler option, and the INTERRUPT run-time option must be in effect.
- CICS consideration—INTERRUPT is ignored under CICS.
- PL/I MTF consideration—The INTERRUPT option applies to the enclave. However, only one thread in the enclave is affected for a particular attention interrupt.
- OpenEdition consideration—The INTERRUPT option applies to the enclave. However, only one thread in the enclave is affected for a particular attention interrupt.

E.3.21.2 For More Information

- See E.3.42, “TEST | NOTEST” on page 128 for more information about the TEST run-time option.
- For more information about the POSIX run-time option, see E.3.30, “POSIX” on page 108.

E.3.22 LIBSTACK

LIBSTACK controls the allocation of the thread's library stack storage. This stack is used by Language Environment and HLL library routines that require save areas below the 16M line.

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

Syntax

```
►► LIBSTACK= ( ( ( init_size , incr_size , FREE  
KEEP ) , OVR  
NONOVR ) ) ►►
```

init_size

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

Specify *init_size* as *n*, *nK*, or *nM* bytes of storage. *init_size* can be preceded by a minus sign. On systems other than CICS, if you specify a negative number, all available storage minus the amount specified is used for the initial stack segment.

In all supported systems except CICS, an *init_size* of 0 or -0 requests half of the largest block of contiguous storage below the 16M line.

At initialization, Language Environment allocates the storage rounded up to the nearest multiple of 8 bytes.

LIBSTACK

incr_size

Determines the minimum size of any subsequent increment to the library stack area. Specify this value as *n*, *nK*, or *nM* 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*, Language Environment uses the IBM-supplied default setting of 4K. If *incr_size=0*, Language Environment gets 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, then Language Environment gets a 9000 byte stack increment from the operating system to satisfy the request. If the requested size is smaller than 8K, Language Environment gets an 8K stack increment from the operating system.

FREE

Specifies that Language Environment 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 Language Environment 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.

E.3.22.1 Usage Notes

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

The IBM-supplied default setting for LIBSTACK under CICS is LIBSTACK=((4K,4K,FREE),OVR).

- OpenEdition consideration—The LIBSTACK option sets the library stack characteristics on each thread.

The recommended setting for LIBSTACK under OpenEdition is LIBSTACK=((12K,12K,FREE),OVR).

E.3.22.2 Performance Considerations

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

E.3.22.3 For More Information

- See E.3.36, “RPTSTG” on page 115 for more information about the RPTSTG run-time option.
- For more information about using the storage reports generated by the RPTSTG run-time option to tune the stacks, see *Language Environment for MVS & VM Programming Guide*.

E.3.23 MSGFILE

MSGFILE specifies the *ddname* 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 *ddname* for CEEMSG and CEEMOUT callable services.

IBM-Supplied Default: MSGFILE=((SYSOUT,FBA,121,0),OVR)

Syntax

```

▶▶MSGFILE=—(—(—ddname—,—recfm—,—lrecl—,—blksize—)—,—OVR  
NONOVR—)▶▶

```

ddname

The *ddname* of the run-time diagnostics file.

recfm

The default record format (RECFM) value for the message file. *recfm* is used when this information is not available either in a file definition or in the label of an existing file. The following values are acceptable: F, FA, FB, FBA, FBS, FBSA, U, UA, V, VA, VB, and VBA.

lrecl

The default record length (LRECL) value for the message file. *lrecl* is used when this information is not available either in a file definition or in the label of an existing file. *lrecl* is expressed as bytes of storage.

The *lrecl* value (whether from MSGFILE or from another source) cannot exceed the *blksize* value, whose maximum value is 32760. For variable-length record formats, the *lrecl* value is limited to the *blksize* value minus 4.

blksize

The default block size (BLKSIZE) value for the message file. *blksize* is used when this information is not available either in a file definition or in the label of an existing file. *blksize* is expressed as bytes of storage.

blksize (whether from MSGFILE or from another source) cannot exceed 32760.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

MSGFILE

E.3.23.1 Usage Notes

- CICS considerations—The MSGFILE option is ignored under CICS. Run-time output under CICS is directed instead to a transient data queue named CESE.
- HLL compiler options, such as the COBOL OUTDD compiler option, can affect whether your run-time output goes to MSGFILE *ddname*.
- Use commas to separate suboptions of the MSGFILE run-time option. If you do not specify a suboption but do specify a subsequent one, you must still code the comma to indicate its omission. However, trailing commas are not required.

If you do not specify any suboptions, either of the following is valid: MSGFILE or MSGFILE().

- If one of the suboptions of the MSGFILE run-time option is not present in any source, including CEEDOPT, then an IBM-supplied default value is used. The default values for *ddname*, *recfm*, *lrecl*, and *blksize* are SYSOUT, FBA, 121, and 0, respectively.
- If there is no block size in the MSGFILE run-time option, in a file definition, or in the label of an existing file, block size is determined as follows:
 - For a *recfm* value that specifies unblocked fixed-length format records (F or FA) or undefined-format records (U or UA), the *blksize* value is the same as the *lrecl* value.
 - For a *recfm* value that specifies unblocked variable-length format records (V or VA), the *blksize* value is the *lrecl* value plus 4.
 - For a DASD device on MVS and a *recfm* value that specifies blocked records (FB, FBA, FBS, FBSA, VB, or VBA), the *blksize* value is left at 0 by Language Environment so that the system can determine the optimum *blksize* value.
 - For a terminal and a *recfm* value that specifies blocked fixed-length format records (FB, FBA, FBS, or FBSA), the *blksize* value is the same as the *lrecl* value.
 - For a terminal and a *recfm* value that specifies blocked variable-length format records (VB or VBA), the *blksize* value is the *lrecl* value plus 4.
 - For all other cases, *blksize* has a value which gives 100 records per block if the *blksize* value wouldn't exceed 32760, otherwise, a value giving the largest number of records per block such that the *blksize* value that doesn't exceed 32760.

Or, to put it another way:

- For a *recfm* value that specifies blocked fixed-length format records (FB, FBA, FBS, or FBSA), the *blksize* value is $lrecl \times bfact$ where *bfact* is the largest integer not exceeding 100 such that the *blksize* value does not exceed 32760.
 - For a *recfm* value that specifies blocked variable-length format records (VB or VBA), the *blksize* value is $(lrecl \times bfact)$ plus 4 where *bfact* is the largest integer not exceeding 100 such that the *blksize* value does not exceed 32760.
- Language Environment detects certain invalid values for the MSGFILE suboptions, namely an invalid value for *recfm* and a value of *lrecl* or *blksize* that exceeds 32760. A message is printed, and any incorrect values are ignored.

- Invalid combinations of *recfm*, *lrecl*, and *blksize* values are not diagnosed by Language Environment but can cause an error condition to be detected by the system on the first attempt to write to the message file.
- Language Environment does not check the validity of the MSGFILE *ddname*. An invalid *ddname* generates an error condition on the first attempt to issue a message.
- C/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 MSGFILE, instead of to the PL/I SYSPRINT STREAM PRINT file.

User-specified output is still directed to the PL/I SYSPRINT STREAM PRINT file. To direct this output to the Language Environment MSGFILE file, specify MSGFILE(SYSPRINT).

- OpenEdition MVS considerations—The MSGFILE option specifies the *ddname* of the diagnostic file for the enclave. When multiple threads write to the message file, the output is interwoven by line. To group lines of output, serialize MSGFILE access (by using a mutex, for example).

When OpenEdition MVS is available and the MSGFILE option specifies a *ddname* nominating a POSIX file, Language Environment uses POSIX services to write the message file. A *ddname* nominates a POSIX file using the keyword *PATH=*.

OpenEdition MVS must be available on the underlying operating system for the MSGFILE option to write to a POSIX file. If the *ddname* nominates a POSIX file and OpenEdition is not present, then Language Environment tries to dynamically allocate an MVS file to be used as the message file.

If the message file is allocated (whether POSIX or MVS), Language Environment directs the output to this file. If the current message file is not allocated, and the application carries out a `fork()/exec`, `spawn()`, or `spawnp()`, Language Environment checks whether File Descriptor 2 exists. If it does exist, then Language Environment uses it; otherwise, Language Environment dynamically allocates the message file to the POSIX file system and attempts to open the file SYSOUT in the current working directory; or, if there is no current directory, then in the directory `/tmp`.

- OpenEdition for VM/ESA considerations—If your application is running under the OpenEdition shell or any environment that has file descriptor 2 (FD2) open, MSGFILE output is directed to whatever FD2 points to. Under the shell, this is typically your terminal. If FD2 is closed when your application is invoked (via `spawn()` or `exec()`), no message file is created.

E.3.23.2 For More Information

- For more information about the RPTOPTS and RPTSTG run-time options, see E.3.35, “RPTOPTS” on page 112 and E.3.36, “RPTSTG” on page 115.
- For more information about the CEEMSG and CEEMOUT callable services, see *Language Environment for MVS & VM Programming Reference*.
- For details on how HLL compiler options affect messages, see information on HLL I/O statements and message handling in *Language Environment for MVS & VM Programming Guide*.
- For more information about `perror()` and `stderr` see C message output information in *Language Environment for MVS & VM Programming Guide*.

NATLANG

- For more information about the CESE transient data queue, see *Language Environment for MVS & VM Programming Guide*.

E.3.24 MSGQ

MSGQ specifies the number of ISI blocks that Language Environment allocates on a per thread basis for use by the application. The ISI contains information for Language Environment to use when identifying and reacting to conditions, providing access to q_data tokens, and assigning space for message inserts used with user-created messages. When an ISI is needed and one is not available, Language Environment uses the least recently used ISI. CEEECMI allocates storage for the ISI, if necessary.

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

Syntax

```
▶ MSGQ = ( ( number ) , ( OVR | NONOVR ) ) ▶
```

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.

E.3.24.1 Usage Notes

- PL/I MTF consideration—In a PL/I MTF application, MSGQ sets the number of message queues allowed for each task.

E.3.24.2 For More Information

- For more information about the CEEECMI callable service, see *Language Environment for MVS & VM Programming Reference*.
- For more information about the ISI, see *Language Environment for MVS & VM Programming Guide*.

E.3.25 NATLANG

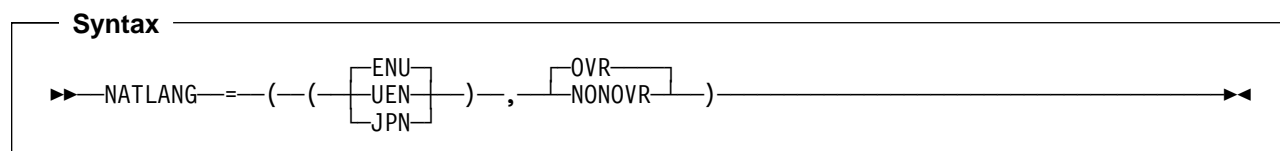
NATLANG specifies the initial national language to be used for the run-time environment, including error messages, month names, and day of the week names. Message translations are provided for Japanese and for uppercase and mixed-case U.S. English. NATLANG also determines how the message facility formats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

NATLANG affects only the Language Environment NLS and date and time services, not the Language Environment locale callable services.

You can set the national language by using the NATLANG run-time option or the SET function of the CEE3LNG callable service. Language Environment maintains one current language at the enclave level. The 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 CEE3LNG callable service, ENU becomes the current national language.

Language Environment writes storage and options reports and dump output only in mixed-case U.S. English.

IBM-Supplied Default: **NATLANG=((ENU),OVR)**



ENU

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

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

UEN

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

Message text consists of SBCS characters and includes only uppercase letters.

JPN

A 3-character ID specifying Japanese.

Message text can contain a mixture of SBCS and DBCS characters.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.25.1 Usage Notes

- If you specify a national language that is not available on your system, Language Environment uses the IBM-supplied default ENU (mixed-case 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/C++ consideration—Language Environment provides locales used in C and C++ to establish default formats for the locale-sensitive functions and locale callable services, such as date and time

NONIPTSTACK | NONONIPTSTACK

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 Language Environment NLS and date and time services. `setlocale()` and CEESETL affect only C/C++ locale-sensitive functions and Language Environment locale callable services.

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

- PL/I MTF consideration—NATLANG affects every task in the application. The SET function of CEE3LNG is supported for the relinked OS PL/I or PL/I for MVS & VM MTF applications only.
- OpenEdition consideration—The NATLANG option specifies the initial value for the enclave.

E.3.25.2 For More Information

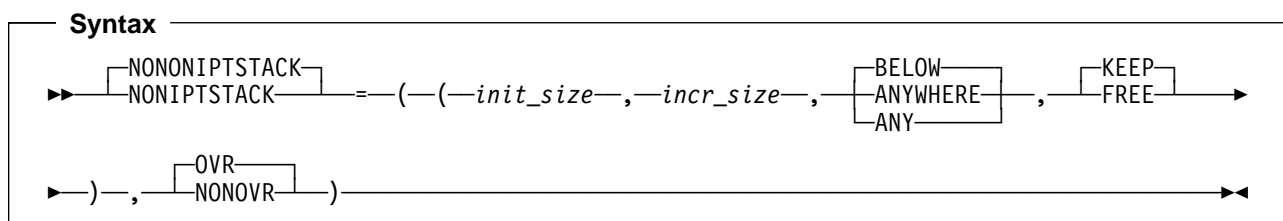
- For more information about the CEE3LNG callable service, see *Language Environment for MVS & VM Programming Reference*.
- See E.3.24, “MSGQ” on page 102 for more information about the MSGQ run-time option.
- For more information on `setlocale()`, see *AD/Cycle C/370 Programming Guide, C/MVS Programming Guide*, or *C++/MVS Programming Guide*.

E.3.26 NONIPTSTACK | NONONIPTSTACK

NONIPTSTACK controls stack allocation for each thread, except the initial thread, in a multithread environment. If the thread attribute object does not provide an explicit stack size, then the allocation values can be inherited from the STACK option or specified explicitly on the NONIPTSTACK option. NONONIPTSTACK causes the values specified in the STACK option to be used.

In PL/I MTF applications, NONIPTSTACK specifies stack storage for every subtask. If you use the IBM-supplied default NONONIPTSTACK, the STACK option specifies stack storage for both the main task and subtasks.

IBM-Supplied Default: NONONIPTSTACK=((4K,4K,BELOW,KEEP),OVR)



NONONIPTSTACK

Indicates that the allocation options of the STACK option are used for thread stack allocation. Any suboption specified with NONONIPTSTACK is ignored.

NONIPTSTACK

Controls the stack allocation for each thread, except the initial thread, in a multithread environment.

init_size

The length of each noninitial thread initial stack storage area. This is an unsigned integer, n, nK, or nM. The actual amount of allocated storage is rounded up to the nearest multiple of 8 bytes.

A value of zero (0) causes an allocation of 4K.

incr_size

The minimum amount by which the stack storage for any noninitial thread is incremented, and is specified in n, nK, or nM. 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 8 bytes) is obtained.

BELOW

Specifies that the stack storage must be allocated below the 16M line. Applications running with ALL31(OFF) must specify NONIPTSTACK(,BELOW) to ensure that stack storage is addressable by the application.

ANYWHERE|ANY

Specifies that the stack storage can be allocated anywhere in storage either above or below the 16M line.

The only valid abbreviation of ANYWHERE is ANY.

KEEP

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

FREE

Specifies that storage allocated to NONIPTSTACK increments is released when the last of the storage in the thread stack increment is freed.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.26.1 Usage Notes

- All storage allocated to NONIPTSTACK segments is freed when the thread terminates.
- The initial stack segment of the thread is never released until the thread terminates, regardless of the KEEP/FREE state.
- You can specify sub-options with NONONIPTSTACK, but they are ignored. If you override the NONONIPTSTACK option with NONIPTSTACK and you omit suboptions, then the suboptions you

OCSTATUS

specified with NONONIPTSTACK remain in effect. If you respecify NONONIPTSTACK with different suboptions, they override the defaults.

- PL/I MTF consideration—NONONIPTSTACK(4K, 4K, BELOW, KEEP) provides PL/I compatibility for stack storage allocation and management for each subtask in the application.
- CICS consideration—This option is ignored under CICS.

E.3.26.2 For More Information

- For more information about the STACK run-time option, see E.3.39, “STACK” on page 121.
- For more information about the ALL31 run-time option, see E.3.4, “ALL31” on page 76.

E.3.27 OCSTATUS (Fortran Only)

OCSTATUS controls the verification of file existence and whether a file is actually deleted based on the STATUS specifier on the OPEN and CLOSE statement, respectively.

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

Syntax

```
▶▶ OCSTATUS= ( ( ON OFF ) , ( OVR NONOVR ) ) ▶▶
```

ON

Specifies that file existence is checked with each OPEN statement to verify that the status of the file is consistent with STATUS='OLD' and STATUS='NEW'. It also specifies that file deletion occurs with each CLOSE statement with STATUS='DELETE' for those devices which support file deletion. Preconnected files are included in these verifications. OCSTATUS consistency checking applies to DASD files, PDS members, VSAM files, MVS labeled tape files, and dummy files only. For dummy files, the consistency checking occurs only if the file was previously opened successfully in the current program.

In addition, when a preconnected file is disconnected by a CLOSE statement, an OPEN statement is required to reconnect the file under OCSTATUS. Following the CLOSE statement, the INQUIRE statement parameter OPENED indicates that the unit is disconnected.

OFF

Bypasses file existence checking with each OPEN statement and bypasses file deletion with each CLOSE statement.

If STATUS='NEW', a new file is created; if STATUS='OLD', the existing file is connected.

If STATUS='UNKNOWN' or 'SCRATCH', and the file exists, it is connected; if the file does not exist, a new file is created.

In addition, when a preconnected file is disconnected by a CLOSE statement, an OPEN statement is *not* required to reestablish the connection under OCSTATUS(OFF). A sequential READ, WRITE,

BACKSPACE, REWIND, or ENDFILE will reconnect the file to a unit. Before the file is reconnected, the INQUIRE statement parameter OPENED will indicate that the unit is disconnected; after the connection is reestablished, the INQUIRE statement parameter OPENED will indicate that the unit is connected.

OVR

Specifies that the option can be overridden.

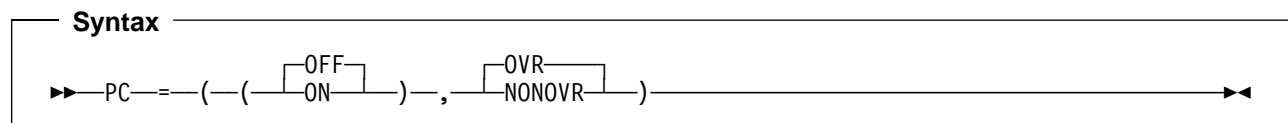
NONOVR

Specifies that the option cannot be overridden.

E.3.28 PC (Fortran Only)

PC controls whether Fortran status common blocks are shared among load modules.

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



OFF

Specifies that Fortran static common blocks with the same name but in different load modules all refer to the same storage. PC(OFF) applies only to static common blocks referenced by compiled code produced by any of the following compilers and that were **not** compiled with the PC compiler option:

- VS FORTRAN Version 2 Release 5
- VS FORTRAN Version 2 Release 6

ON Specifies that Fortran static common blocks with the same name but in different load modules do not refer to the same storage.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.29 PLITASKCOUNT (PL/I Only)

PLITASKCOUNT controls the maximum number of tasks active at one time while you are running PL/I MTF applications.

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

POSIX

Syntax

```
▶▶ PLITASKCOUNT= ( ( tasks ) , OVR  
NONOVR ) ▶▶
```

tasks

A decimal integer that is the maximum number of tasks allowed in a PL/I MTF application at any one time during execution. The total tasks include the main task and subtasks created directly or indirectly from the main task.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.29.1 Usage Notes

- A value of zero (0) assumes the IBM-supplied default of 20.
- PL/I MTF consideration—If *tasks* or the IBM-supplied default of 20 exceeds the OpenEdition MVS installation default of the maximum number of threads, Language Environment assumes the OpenEdition MVS installation default.
- If a request to create a task would take the number of currently active tasks over the allowable limit, condition IBM0566S is signalled and the task is not created.

E.3.30 POSIX

POSIX specifies whether the enclave can run with the POSIX semantics.

POSIX is an application characteristic that is maintained at the enclave level. After you have established the characteristic during enclave initialization, you cannot change it.

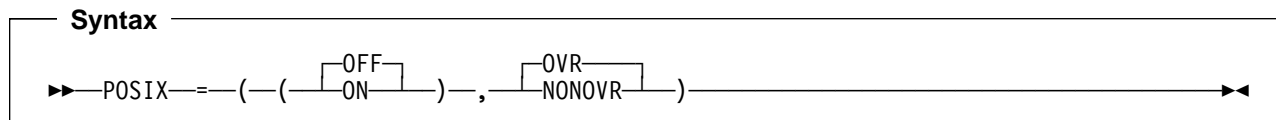
When you set POSIX to ON, you can use functions that are unique to POSIX, such as `pthread_create()`.

One of the effects of POSIX(ON) is the enablement of POSIX signal handling semantics, which interact closely with the Language Environment condition handling semantics.

ANSI C routines can access the OpenEdition MVS Hierarchical File System (HFS) on MVS independent of the POSIX setting. They can also access the OpenEdition for VM/ESA Byte File System (BFS) on VM independent of the POSIX setting. Where ambiguities exist between ANSI and POSIX semantics, the POSIX run-time option setting indicates the POSIX semantics to follow.

If you set POSIX to ON and you run non-thread-safe languages such as COBOL, PL/I, and C++ in a thread other than the initial thread, the behavior is undefined.

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

**OFF**

Indicates that the application is not POSIX-enabled.

ON

Indicates that the application is POSIX-enabled.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.30.1 Usage Notes

- If you set POSIX to ON when OpenEdition is not active, the following events occur:
 - The message file receives a warning, but the application continues to run.
 - If you invoke a POSIX function that has an OpenEdition kernel dependency, it does not take effect.
 - If you invoke a POSIX function that has an OpenEdition kernel dependency and has no provision for failure, for example, `alarm`, a severity 3 condition is raised.
- POSIX(ON) applies to MVS/ESA and VM/ESA, but explicitly excludes CICS. If you set POSIX to ON while an application is running under CICS, you receive a warning message and the application continues to run. You can specify POSIX(ON) for both DB2* and IMS applications.
- Within nested enclaves, only one enclave can have the POSIX option set to ON. All other nested enclaves must have the POSIX option set to OFF.

E.3.30.2 For More Information

- For more information on POSIX functions that have an OpenEdition kernel dependency, see C/C++ for MVS/ESA Library Reference.
- For more information about the INTERRUPT run-time option, see E.3.21, "INTERRUPT" on page 96.

E.3.31 PRTUNIT (Fortran Only)

PRTUNIT identifies the unit number used for PRINT and WRITE statements that do not specify a unit number.

IBM-Supplied Default: PRTUNIT=((6),OVR)

RDRUNIT

Syntax

```
▶▶ PRTUNIT= ( ( number ) , OVR  
NONOVR ) ▶▶
```

number

A valid unit number in the range 0-99. You can establish your own default number at installation time.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.32 PUNUNIT (Fortran Only)

PUNUNIT identifies the unit number used for PUNCH statements that do not specify a unit number.

IBM-Supplied Default: PUNUNIT=((7),OVR)

Syntax

```
▶▶ PUNUNIT= ( ( number ) , OVR  
NONOVR ) ▶▶
```

number

A valid unit number in the range 0-99. You can establish your own default number at installation time.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.33 RDRUNIT (Fortran Only)

RDRUNIT identifies the unit number used for READ statements that do not specify a unit number.

IBM-Supplied Default: RDRUNIT=((5),OVR)

Syntax

```
▶▶ RDRUNIT= ( ( number ) , OVR  
NONOVR ) ▶▶
```

number

A valid unit number in the range 0-99. You can establish your own default number at installation time.

OVR

Specifies that the option can be overridden.

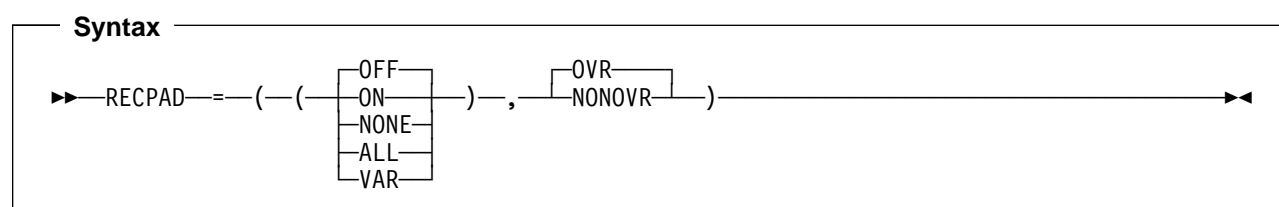
NONOVR

Specifies that the option cannot be overridden.

E.3.34 RECPAD (Fortran Only)

RECPAD specifies whether a formatted input record is padded with blanks.

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

**OFF|NONE**

Specifies that no blank padding be applied when an input list and format specification requires more data from an input record than the record contains. If more data is required, the error described by condition FOR1002 is detected.

ON|ALL

Specifies that a formatted input record within an internal file, or a varying or undefined length record (RECFM=U or V) external file, be padded with blanks when an input list and format specification require more data from the record than the record contains. Blanks added for padding are interpreted as though the input record actually contains blanks in those fields.

VAR

Applies blank padding to any of the following types of files:

- An external, non-VSAM file with a record format (the RECFM value) that allows the lengths of records to differ within the file. Such record formats are variable(V), variable blocked (VB), undefined (U), variable spanned (VS), and variable blocked spanned (VBS); this excludes fixed (F) and fixed blocked (FB).
- An external, VSAM entry-sequenced data set (ESDS) or key-sequenced data set (KSDS).
- An internal file.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

RPTOPTS

E.3.34.1 Usage Notes

- NORECPAD has the same effect as RECPAD(OFF) and RECPAD(NONE). RECPAD has the same effect as RECPAD(ON) and RECPAD(ALL).
- The PAD specifier of the OPEN statement can be used to indicate padding for individual files.

E.3.35 RPTOPTS

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

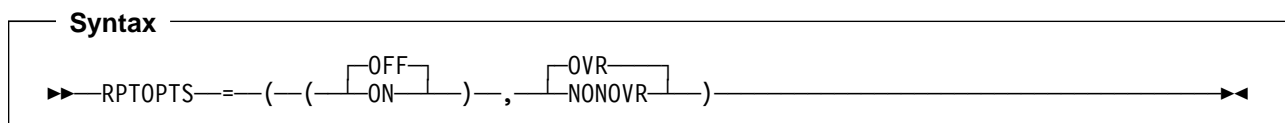
Language Environment directs the report to the *ddname* specified in the MSGFILE run-time option.

RPTOPTS does not generate the options report if Language Environment abends but does generate a report in all other cases.

Figure 26 on page 114 shows the sample output when RPTOPTS is set to ON. RPTOPTS(ON) lists the declared run-time options in alphabetical 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 CEE3RPH. The date and time formats are affected by the country code set by the COUNTRY run-time option or the CEE3CTY 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)



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.

E.3.35.1 Usage Note

- OpenEdition consideration—The RPTOPTS option reports run-time options for the enclave.

E.3.35.2 Performance Considerations

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

RPTOPTS

Options Report for Enclave ABC 08/07/95 1:12:20 PM

LAST WHERE SET	OPTION
Programmer default	ABPERC(NONE)
Installation default	ABTERMENC(RETCODE)
Installation default	NOAIXBLD
Programmer default	ALL31(OFF)
Assembler user exit	ANYHEAP(32768,16384,ANYWHERE,FREE)
Installation default	NOAUTOTASK
Assembler user exit	BELOWHEAP(8192,8192,FREE)
Installation default	CBLOPTS(ON)
Installation default	CBLP SHPOP(ON)
Installation default	CBLQDA(ON)
Installation default	CHECK(ON)
Installation default	COUNTRY(US)
Installation default	DEBUG
Programmer default	DEPTHCONDLMT(20)
Installation default	ENVAR("")
Programmer default	ERRCOUNT(7)
Installation default	ERRUNIT(6)
Installation default	FILEHIST
Default setting	NOFLOW
Assembler user exit	HEAP(32768,32768,ANYWHERE,KEEP,8192,4096)
Installation default	INQPCOPN
Installation default	INTERRUPT(OFF)
Invocation command	LIBSTACK(8192,4096,FREE)
Installation default	MSGFILE(SYSOUT,FBA,121,0)
Installation default	MSGQ(15)
Installation default	NATLANG(ENU)
Invocation command	NONONIPSTACK(4096,4096,BELOW,KEEP)
Installation default	OCSTATUS
Installation default	NOPC
Installation default	PLITASKCOUNT(20)
Installation default	POSIX(OFF)
Installation default	PRTUNIT(6)
Programmer default	PUNUNIT(7)
Installation default	RDRUNIT(5)
Installation default	RECPAD(OFF)
Invocation command	RPTOPTS(ON)
Installation default	RPTSTG(OFF)
Installation default	NORTEREUS
Installation default	NOSIMVRD
Invocation command	STACK(65536,65536,BELOW,KEEP)
Assembler user exit	STORAGE(NONE,NONE,NONE,131072)
Programmer default	TERMTHDACT(TRACE)
Installation default	NOTEST(ALL,"*","PROMPT","INSPREF")
Installation default	THREADHEAP(4096,4096,ANYWHERE,KEEP)
Installation default	TRACE(OFF,4096,DUMP,LE=0)
Installation default	TRAP(ON)
Installation default	UPSI(00000000)
Installation default	NOUSRHDLR()
Installation default	VCTRSAVE(OFF)
Programmer default	XUFLOW(AUTO)

Figure 26. Options Report Produced by Language Environment Run-Time Option RPTOPTS(ON)

E.3.35.3 For More Information

- See E.3.23, “MSGFILE” on page 99 for more information about the MSGFILE run-time option.
- For more information about the CEE3RPH callable service, see *Language Environment for MVS & VM Programming Reference*.
- See E.3.12, “COUNTRY” on page 85 for more information about the COUNTRY run-time option.
- For more information about the CEE3CTY callable service, see *Language Environment for MVS & VM Programming Reference*.

E.3.36 RPTSTG

RPTSTG generates, after an application has run, a report of the storage the application used. The report is directed to the *ddname* specified in the MSGFILE run-time option.

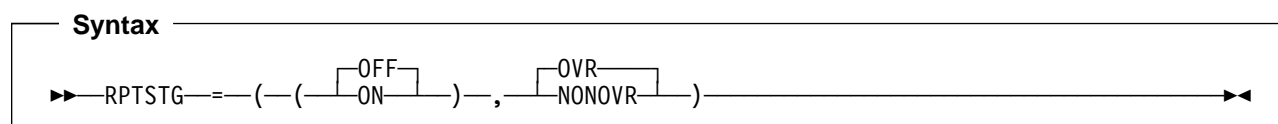
Figure 27 on page 117 shows a sample report created with the RPTSTG option set to ON.

The storage report heading is set by CEE3RPH. 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 CEE3CTY callable service.

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

Language Environment writes storage reports only in mixed-case U.S. English.

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



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.

RPTSTG

E.3.36.1 Usage Notes

- 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:
 - On VM/ESA, the number of CMSSTOR OBTAIN and CMSSTOR RELEASE requests, respectively.
 - On CICS, the number of EXEC CICS GETMAIN and EXEC CICS FREEMAIN requests, respectively.
- RPTSTG includes PL/I task-level information on stack and heap utilization.
- OpenEdition consideration—The RPTSTG option applies to storage utilization for the enclave, including thread-level information on stack utilization, and stack storage used by multiple threads.

E.3.36.2 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 storage report numbers generated by the RPTSTG option as an aid in setting the initial and increment size for STACK and HEAP. This reduces the number of times that the Language Environment storage manager makes requests to acquire storage. For example, you can use the storage report numbers to set appropriate values in the HEAP and STACK *init_size* and *incr_size* fields for allocating storage.

Storage Report for Enclave main 08/07/95 12:59:59 PM

```
STACK statistics:
  Initial size:                131072
  Increment size:              131072
  Maximum used by all concurrent threads: 12600
  Largest used by any thread:  12600
  Number of segments allocated: 1
  Number of segments freed:    0
NONIPTSTACK statistics:
  Initial size:                32768
  Increment size:              32768
  Maximum used by all concurrent threads: 6552
  Largest used by any thread:  2232
  Number of segments allocated: 4
  Number of segments freed:    0
LIBSTACK statistics:
  Initial size:                8192
  Increment size:              4096
  Maximum used by all concurrent threads: 784
  Largest used by any thread:  784
  Number of segments allocated: 1
  Number of segments freed:    0
THREADHEAP statistics:
  Initial size:                4096
  Increment size:              4096
  Maximum used by all concurrent threads: 0
  Largest used by any thread:  0
  Successful Get Heap requests: 0
  Successful Free Heap requests: 0
  Number of segments allocated: 0
  Number of segments freed:    0
HEAP statistics:
  Initial size:                32768
  Increment size:              32768
  Total heap storage used (sugg. initial size): 20312
  Successful Get Heap requests: 25
  Successful Free Heap requests: 3
  Number of segments allocated: 1
  Number of segments freed:    0
ANYHEAP statistics:
  Initial size:                16384
  Increment size:              8192
  Total heap storage used (sugg. initial size): 105256
  Successful Get Heap requests: 412
  Successful Free Heap requests: 391
  Number of segments allocated: 2
  Number of segments freed:    0
```

Figure 27 (Part 1 of 2). Storage Report Produced by Language Environment Run-Time Option RPTSTG(ON)

RTEREUS

```
BELOWHEAP statistics:
  Initial size:                8192
  Increment size:             4096
  Total heap storage used (sugg. initial size): 240304
  Successful Get Heap requests: 41
  Successful Free Heap requests: 32
  Number of segments allocated: 5
  Number of segments freed:    4
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
Largest number of threads concurrently active: 2
End of Storage Report
```

Figure 27 (Part 2 of 2). Storage Report Produced by Language Environment Run-Time Option RPTSTG(ON)

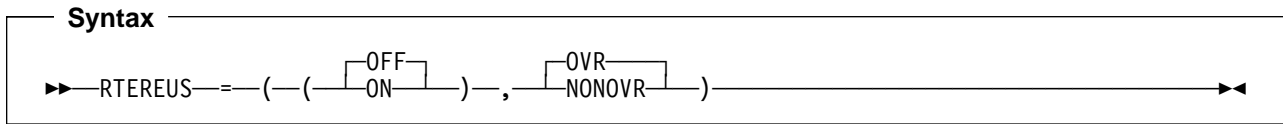
E.3.36.3 For More Information

- For more information about the MSGFILE run-time option, see E.3.23, “MSGFILE” on page 99.
- For more information about the CEE3RPH callable service, see *Language Environment for MVS & VM Programming Reference*.
- See E.3.12, “COUNTRY” on page 85 for more information about the COUNTRY run-time option.
- For more information about the CEE3CTY callable service, see *Language Environment for MVS & VM Programming Reference*.
- For more information about the ANYHEAP run-time option, see E.3.5, “ANYHEAP” on page 78.
- For more information about the BELOWHEAP run-time option, see E.3.7, “BELOWHEAP” on page 80.
- For more information about the HEAP run-time option, see E.3.19, “HEAP” on page 93.
- For more information about the LIBSTACK run-time option, see E.3.22, “LIBSTACK” on page 97.
- For more information about the STACK run-time option, see E.3.39, “STACK” on page 121.
- For more information about tuning your application with storage numbers, see *Language Environment for MVS & VM Programming Guide*.

E.3.37 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)



OFF

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

ON

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.

E.3.37.1 Usage Notes

- Avoid using RTEREUS(ON) as an installation default, because doing so can cause problems for other HLLs such as C/C++ and PL/I.
- When you specify RTEREUS in CEEDOPT or CEEUOPT, the only accepted syntax is RTEREUS(ON) or RTEREUS(OFF).
- The IGZERREO CSECT affects the handling of program checks in the non-Language Environment conforming driver that repeatedly invokes COBOL programs.
- CICS consideration—This option is ignored under CICS.
- IMS consideration—RTEREUS is not recommended for use under IMS.

E.3.37.2 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 and use STOP RUN, Language Environment 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.

The IGZERREO CSECT affects the performance of running with RTEREUS.

Language Environment also offers preinitialization support in addition to RTEREUS.

SIMVRD

E.3.37.3 For More Information

- For more information about CEEUOPT or CEEDOPT, see *Language Environment for MVS & VM Installation and Customization on MVS*.
- See *Language Environment for MVS & VM Programming Guide* for more information about preinitialization.
- For more information about IFZERREO, see “Updating COBOL Component Reusable Environment” on page 42.

E.3.38 SIMVRD (COBOL Only)

SIMVRD specifies whether your COBOL routines use a VSAM KSDS to simulate variable-length relative organization data sets.

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

Syntax

```
▶ SIMVRD = ( ( OFF ) , ( OVR ) )
```

OFF

Do not use a VSAM KSDS to simulate variable-length relative organization.

ON

Use a VSAM KSDS to simulate variable-length relative organization.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.38.1 Usage Notes

- When you specify SIMVRD in CEEDOPT or CEEUOPT, the only accepted syntax is SIMVRD(ON) or SIMVRD(OFF).
- CICS consideration—This option is ignored under CICS.

E.3.38.2 For More Information

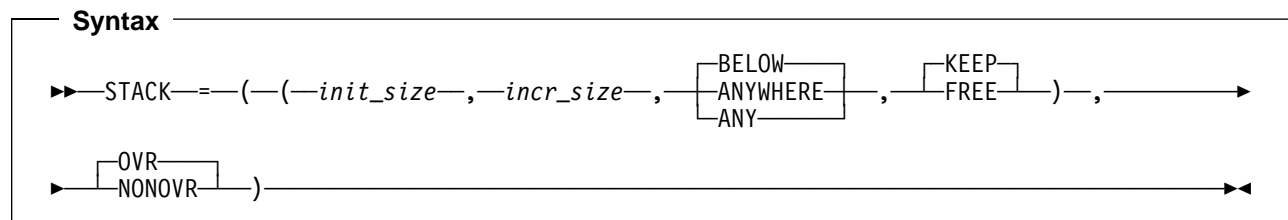
- See *COBOL/370 Programming Guide* or *COBOL for MVS & VM Programming Guide* for more details.
- See *Language Environment for MVS & VM Installation and Customization on MVS* for more information on CEEDOPT and CEEUOPT.

E.3.39 STACK

STACK controls the allocation of the thread's stack storage. Typical items residing in the stack are C or PL/I automatic variables, COBOL LOCAL-STORAGE data items, and 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: `STACK=((128K,128K,BELOW,KEEP),OVR)`



init_size

Determines the size of the initial stack segment. The storage is contiguous. You specify the *init_size* value as *n*, *nK*, or *nM* 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. On systems other than CICS, if you specify a negative number Language Environment 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 in the region below the 16M 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*, *nK*, or *nM* 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 multiple of 8 bytes, 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, Language Environment gets a 9000 byte stack increment from the operating system to satisfy the request. If the requested size is smaller than 8K, Language Environment gets an 8K stack increment from the operating system.

BELOW

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

STACK

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 16M line. If there is no storage available above the line, Language Environment acquires storage below the line. On systems that do not support bimodal addressing (for example, when VM/ESA is initial program loaded in 370 mode) Language Environment ignores this option and places the stack storage below 16M.

The only valid abbreviation for ANYWHERE is ANY.

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.

E.3.39.1 Usage Notes

- Applications running with ALL31(OFF) must specify STACK(,BELOW) to ensure that stack storage is addressable by the application.
- CICS consideration—The IBM-supplied default setting for STACK under CICS is STACK=((4K,4K,ANYWHERE,KEEP),OVR). However, when you define your CICS transaction, if the value of the TASKDATALOC suboption is set to or defaults to BELOW, it overrides the setting STACK(,ANYWHERE) and forces GETMAINs to obtain stack storage below the line.

The maximum initial and increment size for CICS above 16M is 1 gigabyte (1204M). 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 and the increment size minimum is 4K.

If you do not specify STACK, Language Environment assumes the default value of 4K. Under CICS, STACK(0), STACK (-0), and STACK (-n) are all interpreted as STACK(4K).

- PL/I consideration—PL/I automatic storage above the 16M line is supported under control of the Language Environment STACK option. When the Language Environment 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 16M. Stack frame extensions are also constrained to 16M. Therefore, the size of an automatic aggregate, temporary variable, or dummy argument cannot exceed 16M. Violation of this constraint might have unpredictable results.

If an OS PL/I application does not contain any edited stream I/O and if it is running with AMODE 31, you can relink it with Language Environment to use STACK(,ANY). Doing so is particularly useful under CICS to help relieve below-the-line storage constraints.

- PL/I MTF consideration—The STACK option allocates and manages stack storage for the PL/I main task only. For information about stack storage management in the subtasks, see E.3.26, “NONIPTSTACK | NONONIPTSTACK” on page 104.
- OpenEdition consideration—The STACK option specifies the characteristics of the user stack for the initial thread. In particular, it gets the initial size of the user stack for the initial thread.

The characteristics that indicate *incr_size*, ANYWHERE, and KEEP | FREE apply to any thread created using `pthread_create`. Language Environment gets the initial stack size from the threads attribute object specified in the `pthread_create` function. The default size to be set in the thread's attribute object is obtained from the STACK run-time option's initial size.

The recommended default setting for STACK under OpenEdition is `STACK=((12K,12K,ANYWHERE,KEEP),OVR)`.

E.3.39.2 Performance Considerations

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

E.3.39.3 For More Information

- See E.3.4, “ALL31” on page 76, for more information about the ALL31 run-time option.
- See E.3.36, “RPTSTG” on page 115, for more information about the RPTSTG run-time option.
- For more information about using the storage reports generated by the RPTSTG run-time option to tune the stacks, see *Language Environment for MVS & VM Programming Guide*.

E.3.40 STORAGE

STORAGE controls the initial content of storage when allocated and freed. It also controls 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 that parameter is initialized to the specified 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: `STORAGE=((NONE,NONE,NONE,8K),OVR)`

STORAGE

Syntax

```
STORAGE=(--(---heap_alloc_value---,---heap_free_value---,---dsa_alloc_value---,  
reserve_size---),---OVR  
NONOVR---))
```

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 NONE, the allocated heap storage is not initialized.

heap_free_value

The value of 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 NONE, the freed heap storage is not initialized.

dsa_alloc_value

The initialized value of stack frames from the Language Environment 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 Language Environment stack storage, including automatic variable storage, is initialized to *dsa_alloc_value*. Stack frames allocated outside the Language Environment 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 NONE, the stack storage is not initialized.

reserve_size

The amount of storage for the Language Environment storage manager to reserve in the event of an out-of-storage condition. You can specify the *reserve_size* value as *n*, *nK*, or *nM* bytes of storage. The amount of storage is rounded 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 on a non-CICS system, Language Environment does not immediately abend when your application runs out of storage. Instead, when the stack overflows, Language Environment attempts to get another stack segment and add it to the stack.

If unsuccessful, Language Environment temporarily adds the reserve stack segment to the overflowing stack, and signals the out-of-storage condition. This causes a user-written condition handler to gain control and release storage. If the reserve stack segment overflows while this is happening, Language Environment 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.

E.3.40.1 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 IBM-supplied default setting for STORAGE under CICS is STORAGE=((NONE,NONE,NONE,0K),OVR).

TERMTHDACT

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

- OpenEdition consideration—A reserve stack of the size specified by the *reserve_size* suboption of STORAGE is allocated for each thread.

E.3.40.2 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(NONE,NONE,NONE) when you are not debugging.

E.3.41 TERMTHDACT

TERMTHDACT sets the level of information that is produced when Language Environment percolates a condition of severity 2 or greater beyond the first routine's stack frame.

The Language Environment service CEE3DMP is called for the TRACE and DUMP suboptions of TERMTHDACT.

The following CEE3DMP options are passed for TRACE:

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

The following options are passed for DUMP and UADUMP:

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

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:

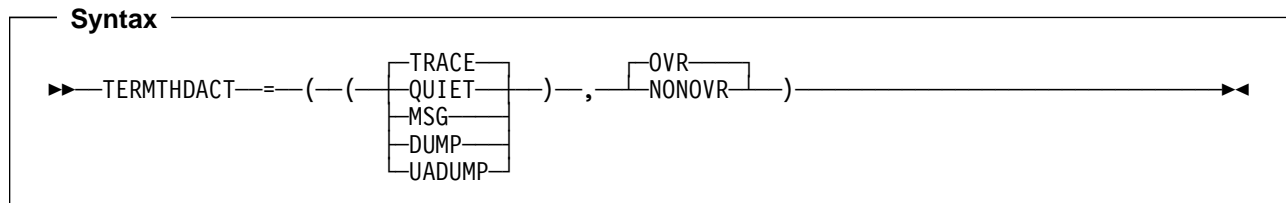
- TERMTHDACT(TRACE)—generates a dump containing the trace table and the traceback
- TERMTHDACT(QUIET)—generates a dump containing the trace table only
- TERMTHDACT(MSG)—generates a dump containing the trace table only
- TERMTHDACT(DUMP)—generates a dump containing thread/enclave/process storage and control blocks (the trace table is included as an enclave control block)

- TERMTHDACT(UADUMP)—generates a system dump of the user address space.

Under normal termination, the following dump contents are generated:

- Independent of the TERMTHDACT setting, Language Environment generates a dump containing the trace table only.

IBM-Supplied Default: TERMTHDACT=((TRACE),OVR)



TRACE

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

QUIET

Specifies that Language Environment 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, Language Environment 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, Language Environment generates a message indicating the cause of the termination, a trace of the active routines on the activation stack, and a Language Environment dump.

UADUMP

Specifies that when a thread terminates due to an unhandled condition of severity 2 or greater, Language Environment generates a message indicating the cause of the termination, a trace of the active routines on the activation stack, a Language Environment dump, and, if the appropriate DD statement is specified in the GO step of your JCL, a system dump of the user address space.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

TEST | NOTEST

E.3.41.1 Usage Notes

- PL/I considerations—After a normal return from a PL/I ERROR ON-unit or from a PL/I FINISH ON-unit, Language Environment 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.
- PL/I MTF considerations—TERMTHDACT applies to a task when the task terminates abnormally due to an unhandled condition of severity 2 or higher that is percolated beyond the initial routine's stack frame. All active subtasks created from the incurring task also terminate abnormally, but the enclave can continue to run.
- CICS consideration—All TERMTHDACT output is written to a transient data queue named CESE.
- OpenEdition consideration—The TERMTHDACT option applies when a thread terminates abnormally. Abnormal termination of a single thread causes termination of the entire enclave. If an unhandled condition of severity 2 or higher percolates beyond the first routine's stack frame, the enclave terminates abnormally.

If an enclave terminates due to a POSIX default signal action, TERMTHDACT applies only to conditions that result from program checks or abends.

E.3.41.2 For More Information

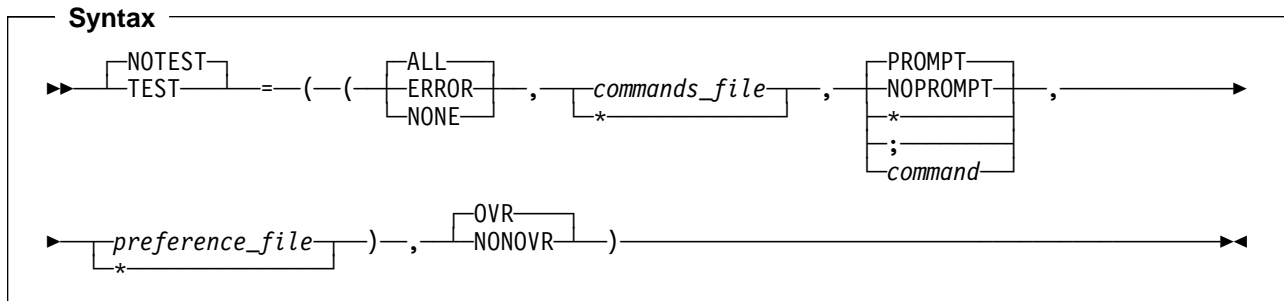
- See E.3.44, "TRACE" on page 133, for more information about the TRACE run-time option.
- For more information about the CEE3DMP service and its parameters, see *Language Environment for MVS & VM Programming Reference*.
- See *Language Environment for MVS & VM Programming Guide* for more information about the TERMTHDACT run-time option and condition message.
- For More Information about CESE, see *Language Environment for MVS & VM Programming Guide*.

E.3.42 TEST | NOTEST

TEST specifies the conditions under which a debug tool (such as the Debug Tool supplied with CODE/370) 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,INSPREF),OVR)

**ALL**

Specifies that any of the following causes the debug tool to gain control even without a defined AT OCCURRENCE for a particular condition or AT TERMINATION:

- The ATTENTION function
- Any Language Environment condition of severity 1 or above
- Application termination

ERROR

Specifies that only one of the following causes the debug tool to gain control without a defined AT OCCURRENCE for a particular condition or AT TERMINATION:

- The ATTENTION function
- Any Language Environment-defined error condition of severity 2 or higher
- Application termination

NONE

Specifies that no condition causes the debug tool to gain control without a defined AT OCCURRENCE for a particular condition or AT TERMINATION.

commands_file

A valid *ddname*, data set name (MVS), or file name (CMS), specifying the primary commands file for this run. If you do not specify this parameter all requests for commands go to the user terminal.

You can enclose *commands_file* in single or double quotes to distinguish it from the rest of the TEST | NOTEST suboption list. It can have a maximum length of 80 characters. If the data set name provided could be interpreted as a *ddname*, it must be preceded by a slash (/). The slash and data set name must be enclosed in quotes.

A primary commands file is required when running in a batch environment.

* (asterisk—in place of *commands_file*)

Specifies that no *commands_file* is supplied. The terminal, if available, is used as the source of the debug tool commands.

PROMPT

Specifies that the debug tool is invoked at Language Environment initialization.

TEST | NOTEST

NOPROMPT

Specifies that the debug tool is not invoked at Language Environment initialization.

* (asterisk—in place of PROMPT/NOPROMPT)

Specifies that the debug tool is not invoked at Language Environment initialization; equivalent to NOPROMPT.

;(semicolon—in place of PROMPT/NOPROMPT)

Specifies that the debug tool is invoked at Language Environment initialization; equivalent to PROMPT.

command

A character string that specifies a valid debug tool command. The command list 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 list contains embedded blanks, commas, semicolons, or parentheses. The list can have a maximum of 250 characters.

preference_file

A valid *ddname*, data set name (MVS), or file name (CMS), specifying 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. It is analogous to creating a profile for a text editor, or initializing an S/370 terminal session.

You can enclose *preference_file* in single or double quotes to distinguish it from the rest of the TEST parameter list. It can have a maximum of 80 characters.

If a specified data set name could be interpreted as a *ddname*, it must be preceded by a slash (/). The slash and data set name must be enclosed in quotes.

The IBM-supplied default setting for *preference_file* is INSPREF.

* (asterisk—in place of *preference_file*)

Specifies that no *preference_file* is supplied.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.42.1 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 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.
- OpenEdition consideration—Language Environment honors the initial command string before the main routine runs on the initial thread.

The test level (ALL, ERROR, NONE) applies to the enclave.

Language Environment honors the preference file when the debug tool is initialized, regardless of which thread first requests the debug tool services.

E.3.42.2 Performance Consideration

To improve performance, use this option only while debugging.

E.3.42.3 For More Information

- See *Debug Tool Reference Guide* for details and for examples of the TEST run-time option as it relates to Debug Tool.

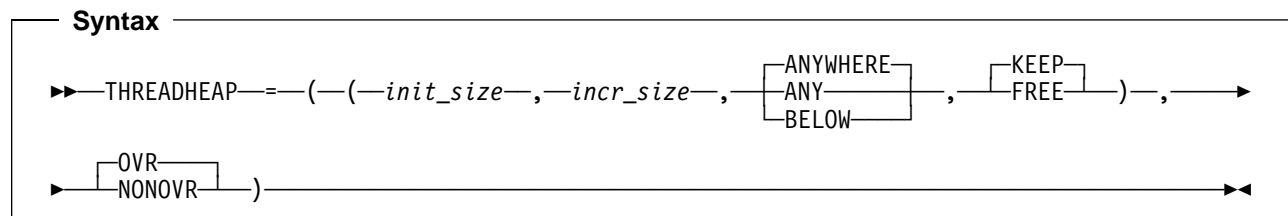
E.3.43 THREADHEAP

THREADHEAP controls the allocation and management of thread-level heap storage. Separate heap segments are allocated and freed for each thread based on the THREADHEAP specification.

For PL/I MTF applications, controlled and based variables declared in a subtask are allocated from heap storage specified by THREADHEAP. Variables in the main task are allocated from heap storage specified by HEAP.

Library use of heap storage in a substack is allocated from the enclave-level heap storage specified by the ANYHEAP and BELOWHEAP options.

IBM-Supplied Default: THREADHEAP=((4K,4K,ANY,KEEP),OVR)



init_size

The minimum initial size of thread heap storage, and is specified in n, nK, or nM. Storage is acquired in multiples of 8 bytes.

A value of zero (0) causes an allocation of 4K.

incr_size

The minimum size of any subsequent increment to the noninitial heap storage is specified in n, nK, or nM. 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 8 bytes) is obtained.

THREADHEAP

ANYWHERE|ANY

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

The only valid abbreviation of ANYWHERE is ANY.

BELOW

Specifies that the heap storage must be allocated below the 16M line.

KEEP

Specifies that storage allocated to THREADHEAP increments is not released when the last of the storage in the thread heap increment is freed.

FREE

Specifies that storage allocated to THREADHEAP increments is released when the last of the storage in the thread heap increment is freed.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.43.1 Usage Notes

- If the requesting routine is running in 24-bit addressing mode and THREADHEAP(,,ANY) is in effect, THREADHEAP storage is allocated below the 16M line based upon the HEAP(,,initsz24,incrsz24) settings.
- PL/I MTF considerations—The thread-level heap is allocated only in applications that use the PL/I MTF. For PL/I MTF applications, controlled and based variables specified in subtasks are located in the thread-level heap.

If the main program is AMODE 24 and THREADHEAP(,,ANY) is in effect, heap storage is allocated below the 16M line. The only case in which storage is allocated above the line is when all of the following conditions exist:

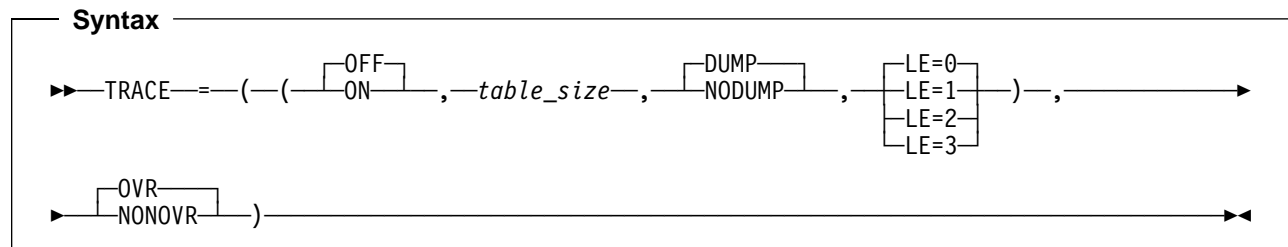
- The user routine requesting the storage is running in 31-bit addressing mode.
 - HEAP(,,ANY) is in effect.
 - The main routine is AMODE 31.
- When running PL/I with POSIX(ON) in effect, THREADHEAP is used for allocating heap storage for PL/I base variables declared in non-IPTs. Storage allocated to all THREADHEAP segments is freed when the thread terminates.
 - The initial thread heap segment is never released until the thread terminates.
 - THREADHEAP has no effect on C/C++ applications.
 - CICS consideration—This option is ignored under CICS.

E.3.44 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 Language Environment 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)



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 (*nK* or *nM*). The upper limit is 16M.

DUMP

Requests that a Language Environment-formatted dump (containing the trace table) be taken at program termination regardless of the setting of the TERMTHDACT run-time option.

NODUMP

Requests that a Language Environment-formatted dump not be taken at program termination.

LE=0

Specifies that no trace events be recorded.

LE=1

Specifies that entry to and exit from Language Environment member libraries be recorded (such as, in the case of C, entry and exit of the `printf()` library function).

TRAP

LE=2

Specifies that mutex init/destroy and locks/unlocks from Language Environment member libraries be recorded.

LE=3

Activates both the entry/exit trace and the mutex trace.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.44.1 Usage Note

- PL/I MTF consideration—The TRACE(ON,,,LE=2) setting provides the following trace table entries for PL/I MTF support:
 - Trace entry 100 occurs when a task is created.
 - Trace entry 101 occurs when a task that contains the tasking CALL statements is terminated.
 - Trace entry 102 occurs when a task that does not contain the tasking CALL statements is terminated.
- When running PL/I with POSIX(ON) in effect, no PL/I-specific trace information is provided.

E.3.44.2 For More Information

- For more information about the dump contents, see E.3.41, “TERMTHDACT” on page 126.
- For more information about using the tracing facility, see *Language Environment for MVS & VM Debugging Guide and Run-Time Messages*.

E.3.45 TRAP

TRAP specifies how Language Environment routines handle abends and program interrupts.

TRAP(ON) must be in effect in order for applications to run successfully.

TRAP(ON) must be in effect for the ABTERMENC run-time option to have effect.

This option is similar to the STAE | NOSTAE run-time option currently offered by COBOL, C, and PL/I, and the SPIE | NOSPIE option offered by C and PL/I:

Figure 28 (Page 1 of 2). TRAP Run-Time Option Settings

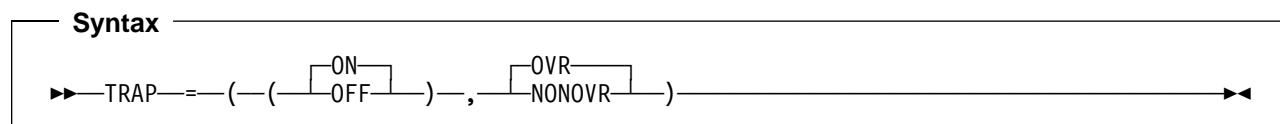
if...	then...
a single option is specified in input,	TRAP is set according to that option, TRAP(OFF) for NOSTAE or NOSPIE, TRAP(ON)for STAE or SPIE.

Figure 28 (Page 2 of 2). TRAP Run-Time Option Settings

If...	then...
both options are specified in input,	TRAP is set ON, unless both options are negative, then TRAP is set OFF.
STAE is specified in one #pragma runopts statement, and NOSPIE in another,	the option in the last #pragma runopts determines the setting of TRAP.
multiple instances of STAE NOSTAE are specified,	TRAP is set according to the last instance only. All others are ignored.
multiple instances of SPIE NOSPIE are specified,	TRAP is set according to the last instance only. All others are ignored.
an options string has TRAP(ON) or TRAP(OFF) together with SPIE NOSPIE, and/or STAE NOSTAE,	the TRAP setting takes preference over all others.

CEESGL is unaffected by this option.

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



ON

Fully enables the Language Environment condition handler.

OFF

Prevents language condition handlers or handlers registered by CEEHDLR from being notified of abends or program checks; prevents application of POSIX signal handling semantics for abends and program checks.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.45.1 Usage Notes

- Use TRAP(OFF) only when you need to analyze a program exception before Language Environment handles it.
- When you specify TRAP(OFF) in a non-CICS environment, neither ESPIE nor ESTAE is issued. Language Environment does not handle conditions raised by program interrupts or abends initiated by SVC 13 as Language Environment conditions, and does not print messages for such conditions.

TRAP

- Running with TRAP(OFF) (for exception diagnosis purposes) can cause many side effects, because Language Environment uses condition handling internally and requires TRAP(ON). When you run with TRAP(OFF), you can get side effects even if you do not encounter a software-raised condition, program check, or abend. If you do encounter a program check or an abend with TRAP(OFF) in effect, the following side effects can occur:
 - The ABTERMENC run-time option has no effect.
 - The ABPERC run-time option has no effect.
 - Resources acquired by Language Environment are not freed.
 - Files opened by HLLs are not closed by Language Environment, so records might 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 debugger is not notified of the error.
 - No storage report or run-time options report is generated.
 - No Language Environment messages or Language Environment dump output is generated.
 - In OpenEdition, POSIX signal handling semantics are not enabled for the abend.

The enclave terminates abnormally if such conditions are raised.

- TRAP(ON) must be in effect when you use the CEEBXITA assembler user exit for enclave initialization to specify a list of abend codes that Language Environment percolates.
- C++ consideration—TRAP(ON) must be in effect in order for the C++/MVS try/throw/catch condition handling mechanisms to work.
- When TRAP(ON) is in effect, and the abend code is in the CEEAUE_CODES list in CEEBXITA, Language Environment percolates the abend. Normal Language Environment condition handling is never invoked to handle these abends. This feature is useful when you do not want Language Environment condition handling to intervene for certain abends or when you want to prevent invocation of the abnormal termination exit for certain abends, such as when IMS issues a user ABEND code 777.

When TRAP(OFF) is specified and there is a program interrupt, the user exit for termination is not driven.

- If your application uses extended-precision arithmetic and runs on a 370-mode machine, you must specify TRAP(ON) and add the CMSLIB TXTLIB with the GLOBAL TXTLIB command.
- CICS consideration—When you specify TRAP(OFF) in a CICS environment, the standard CICS system action occurs. Language Environment does not print messages for conditions raised by program interruptions or transaction abends.
- OpenEdition consideration—The TRAP option applies to the entire enclave and all threads within.

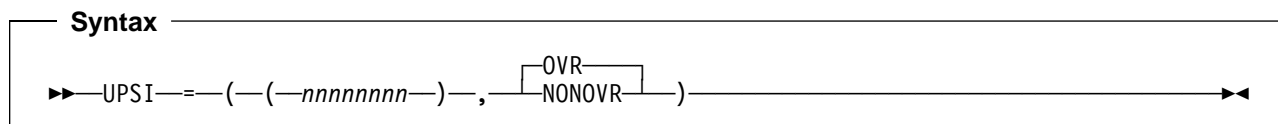
E.3.45.2 For More Information

- See E.3.2, “ABTERMENC” on page 74 for more information about the ABTERMENC run-time option.
- See *Language Environment for MVS & VM Programming Reference* for more information about the CEESGL callable service.
- For more information about the CEEHDLR callable service, see *Language Environment for MVS & VM Programming Reference*.
- See *Language Environment for MVS & VM Programming Guide* for more information about the CEEBXITA assembler user exit.

E.3.46 UPSI (COBOL Only)

UPSI sets the eight UPSI switches on or off for applications that use COBOL programs.

IBM-Supplied Default: UPSI=((00000000),OVR)



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).

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.46.1 Usage Note

- When you specify this option in CEEDOPT or CEEUOPT, specify UPSI with a string of eight binary-valued flags; for example, UPSI(00000000). Use UPSI, not followed by a string, only on the command line.

E.3.46.2 For More Information

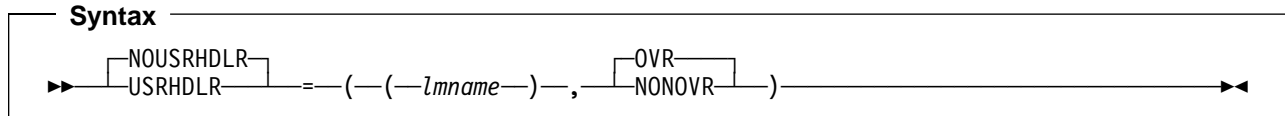
- For more information on how COBOL routines access the UPSI switches, see *COBOL/370 Programming Guide* or *COBOL for MVS & VM Programming Guide*.

USRHDLR | NOUSRHDLR

E.3.47 USRHDLR | NOUSRHDLR

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)



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.

lname

The name of a load module (or an alias name of a load module) 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.

E.3.47.1 Usage Notes

- The user condition handler specified by the USRHDLR run-time option must be in a separate load module rather than be link-edited with the rest of the application.
- The user condition handler *lname* 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 *lname* occurs only when that user condition handler needs to be invoked the first time.

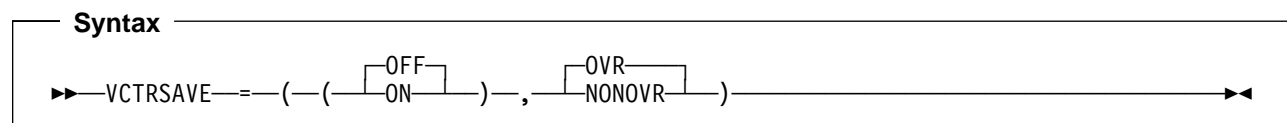
E.3.47.2 For More Information

- For information on registering a user condition handler, see the CEEHDLR callable service in *Language Environment for MVS & VM Programming Reference*.

E.3.48 VCTRSAVE

VCTRSAVE specifies whether any language in the application uses the vector facility when user-written condition handlers are called.

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



OFF

No language in the application uses the vector facility when user-provided condition handlers are called.

ON

A language in the application uses the vector facility when user-provided condition handlers are called.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

E.3.48.1 Usage Note

- OpenEdition consideration—The VCTRSAVE option applies to the entire enclave and all threads within.

E.3.48.2 Performance Considerations

When a condition handler plans to use the vector facility (that is, run any vector instructions), the entire vector environment has to be saved on every condition and restored on return to the application code. You can avoid this extra work by specifying VCTRSAVE(OFF) when you are not running an application under vector hardware.

XUFLOW

E.3.49 XUFLOW

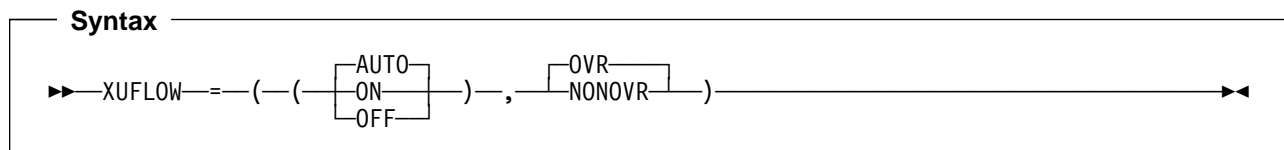
XUFLOW specifies whether an exponent underflow causes a program interrupt. An exponent underflow occurs when a floating point number becomes too small to be represented.

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.

Language Environment preserves the language semantics for C/C++ and COBOL regardless of the XUFLOW setting. Language Environment preserves the language semantics for PL/I only when XUFLOW is set to AUTO or ON. Language Environment does not preserve the language semantics for PL/I when XUFLOW is set to OFF.

An exponent underflow caused by a C/C++ or COBOL routine does not cause a condition to be raised.

IBM-Supplied Default: XUFLOW=((AUTO),OVR)



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.

E.3.49.1 Usage Notes

- PL/I consideration—When setting XUFLOW to OFF, be aware that the semantics of PL/I require the underflow to be signaled.
- OpenEdition consideration—The XUFLOW option applies to the entire enclave and all threads within.

Appendix F. Language Environment National Language Support Country Codes

The following table contains valid country identifiers along with their respective countries:

Figure 29 (Page 1 of 3). Country Codes

Code	Country	Code	Country
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

Figure 29 (Page 2 of 3). Country Codes

Code	Country	Code	Country
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
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

Figure 29 (Page 3 of 3). Country Codes

Code	Country	Code	Country
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	Republic of China
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		

Reader's Comments

IBM Language Environment VM Release 08.00

You may use this form to comment about this document, its organization, or subject matter with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

For each of the topics below please indicate your satisfaction level by circling your choice from the rating scale. If a statement does not apply, please circle N.

RATING SCALE					
very satisfied	<=====	=====	>	very dissatisfied	not applicable
1	2	3	4	5	N

	Satisfaction					
Ease of product installation	1	2	3	4	5	N
Contents of program directory	1	2	3	4	5	N
Installation Verification Programs	1	2	3	4	5	N
Time to install the product	1	2	3	4	5	N
Readability and organization of program directory tasks	1	2	3	4	5	N
Necessity of all installation tasks	1	2	3	4	5	N
Accuracy of the definition of the installation tasks	1	2	3	4	5	N
Technical level of the installation tasks	1	2	3	4	5	N
Ease of getting the system into production after installation	1	2	3	4	5	N

Did you order this product as an independent product or as part of a package?

- Independent
- Package

If this product was ordered as part of a package, what type of package was ordered?

- CustomPac*
- FunctionPac*
- SystemPac*
- System Delivery Offering (SDO)
- Other - Please specify type:

Is this the first time your organization has installed this product?

- Yes
- No

Were the people who did the installation experienced with the installation of VM products?

___ Yes
___ No

If yes, how many years? ___

If you have any comments to make about your ratings above, or any other aspect of the product installation, please list them below:

Please provide the following contact information:

Name and Job Title

Organization

Address

Telephone

Thank you for your participation.

Please send the completed form to (or give to your IBM representative who will forward it to the IBM Language Environment VM Development group):

IBM Corporation, Department J58
P.O. Box 49023
San Jose, California 95161-9023
USA



Program Number: 5654-030

Printed in U.S.A.

G110-4676-00

