Program Directory for

IBM Language Environment for MVS & VM

With US English and Japanese

National Language Support

Release 05.00

Program Number 5688-198

for Use with
VM/ESA

Document Date: September 5th 1995

xxxx-yyyy-zz

Note!

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

This program directory, dated September 5th 1995, applies to IBM Language Environment for MVS & VM Release 05.00 (Language Environment), Program Number 5688-198 for the following:

COMPIDs	Feature Numbers	System Name
568819801	5890	VM/ESA
568819802	5891	
568819803	5892	
568819805	5894	
	6200	
	6201	
	6202	
	6203	
	7547	
	7562	

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.

Contents

Figures

Notices

Trademarks and Service Marks

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 8 describes the IBM support available for Language Environment.
- 4.0, “Program and Service Level Information” on page 9 lists the APARs (program level) and PTFs (service level) incorporated into Language Environment.
- 5.0, “Installation Requirements and Considerations” on page 12 identifies the resources and considerations for installing and using Language Environment.
- 6.0, “Installation Instructions” on page 16 provides detailed installation instructions for Language Environment.
- 7.0, “Service Instructions” on page 31 provides detailed servicing instructions for Language Environment.
- 8.0, “Selecting/Installing National Languages” on page 39 provides instructions on how to install NLS (KANJI) for Language Environment.
- 9.0, “Customizing Language Environment” on page 41 provides instructions on how to customize Language Environment.
- 10.0, “Define and Build The Language Environment Saved Segments” on page 46 provides instructions on how to install Language Environment in Saved Segments.
- Appendix A, “Overriding the VMSYS File Pool Name” on page 56 describes how to change the default file pool name.
- Appendix B, “Contents of COBPACKs (IGZCPAC/IGZCPCO)” on page 59 lists and describes routines eligible for inclusion in COBPACKs.
- Appendix C, “Segment Build Lists (CEEBLSGA/CEEBLSGB)” on page 64 describes build lists for Saved Segments for the full Language Environment product.
- Appendix D, “Segment Build Lists (CEEBLSPA/CEEBLSPB) - POSIX” on page 67 describes build lists for Saved Segments for the ′Language Environment POSIX environment.
- Appendix E, “Language Environment Run-time Options” on page 70 describes Run-time options for Language Environment
- Appendix F, “Language Environment National Language Support Country Codes” on page 149 describes Language Environment National Language Support Country Codes.

Before installing Language Environment, read 3.1, “Preventive Service Planning” on page 8. This section tells you how to find any updates to the information and procedures in this program directory.

2.0 Program Materials

An IBM program is identified by a program number. The program number for Language Environment is 5688-198.

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

The distribution medium for this program is 9-track magnetic tape (written at 6250 BPI), a 3480 tape cartridge or 1/4″ cartridge. The tape or cartridge contains all the programs and data needed for installation. See 6.0, “Installation Instructions” on page 16 for more information about how to install the program. Figure 1 describes the physical tape or cartridge. Figure 3 describes the file content of the program tape or cartridge.

Figure 1. Basic Material: Program Tape US English National Language Feature

Feature Number	Medium	Physical Volume	Tape Content	External Tape Label
5891	6250 Tape	1 of 1	Language Environment	Lang Environ VM
5892	3480 Cart.	1 of 1	Language Environment	Lang Environ VM
5894	1/4″ Cart. QIC120 VB	1 of 1	Language Environment	Lang Environ VM
7547	4MM-DAT Cart.	1 of 1	Language Environment	Lang Environ VM

Figure 2. Basic Material: Program Tape Japanese National Language Feature

Feature Number	Medium	Physical Volume	Tape Content	External Tape Label
6201	6250 Tape	1 of 1	Language Environment	Lang Environ VM
6202	3480 Cart.	1 of 1	Language Environment	Lang Environ VM
6203	1/4″ Cart. QIC120 VB	1 of 1	Language Environment	Lang Environ VM
7562	4MM-DAT Cart	1 of 1	Language Environment	Lang Environ VM

Figure 3 (Page 1 of 2). Program Tape: File Content

Tape File	Content
1	Tape/Product Header
2	Tape/Product Header

Figure 3 (Page 2 of 2). Program Tape: File Content

Tape File	Content
3	Tape/Product Header
4	Product Memo
5	Service Apply Lists (If applicable)
6	PTFPARTs (If applicable)
7	Language Environment Service (If applicable)
8	Language Environment Service (If applicable)
9	Language Environment Base Code - (CEL - C)
10	Language Environment Base Code - (COBOL - PL/I)
11	Language Environment Softcopy Samples - (CEL - C)
12	Language Environment Softcopy Samples - (COBOL - PL/I)
13	Language Environment Executable Code - (CEL - C)
14	Language Environment Executable Code - (COBOL - PL/I)

2.2 Program Publications

The following sections identify the basic and optional publications for Language Environment.

2.2.1 Basic Unlicensed Publications

One copy of the following publications is included when you order the basic materials for Language Environment. For additional copies, contact your IBM representative.

Figure 4. Basic Material: Unlicensed Publications US English NLF

Publication Title	Form Number
IBM Language Environment for MVS & VM Debugging Guide and Run-time Messages	SC26-4829
IBM Language Environment for MVS & VM Licensed Program Specifications	GC26-4774

Figure 5. Basic Material: Unlicensed Publications - Japanese NLF

Publication Title	Form Number
IBM Language Environment for MVS & VM Debugging Guide and Run-time Messages	N:SC26-4829
IBM Language Environment for MVS & VM Licensed Program Specifications	N:GC26-4774

2.2.2 Optional Program Publications

Figure 6 identify the optional unlicensed publications for Language Environment You can request one free copy of manuals with a 7xxx feature number by specifying the feature number.

Figure 6. Optional Material: Program Publications - US English NLF

Publication Title	Form Number	Feature Number
IBM Language Environment for MVS & VM Product Library	SK2T-2389	7576
IBM Language Environment for MVS & VM Programming Guide	SC26-4818	7576
IBM Language Environment for MVS & VM Programming Reference	SC26-3312	7576
IBM Language Environment for MVS & VM Specification Sheet	GC26-4785	7576
IBM Language Environment for MVS & VM Concepts Guide	GC26-4786	7576
IBM Language Environment for MVS & VM Run-Time Migration Guide	SC26-8232	7576
IBM Language Environment for MVS & VM Writing Interlanguage Communication Applications	SC26-8351	7576
AD/CYCLE C/370 Migration Guide	SC09-1359	7576
COBOL/370 and COBOL for MVS & VM Compiler and Run-Time Migration Guide	GC26-4764	7576
PLI for MVS & VM Compiler and Run-Time Migration Guide	GC26-3118	7576

Figure 7. Optional Material: Program Publications - Japanese NLF

Publication Title	Form Number	Feature Number
IBM Language Environment for MVS & VM Programming Guide	SC26-4818	7576
IBM Language Environment for MVS & VM Programming Reference	SC26-3312	7576
IBM Language Environment for MVS & VM Specification Sheet	GC26-4785	7576
IBM Language Environment for MVS & VM Concepts Guide	GC26-4786	7576
IBM Language Environment for MVS & VM Run-Time Migration Guide	SC26-8232	7576
IBM Language Environment for MVS & VM Writing Interlanguage Communication Applications	SC26-8351	7576
COBOL/370 and COBOL for MVS & VM Compiler and Run-Time Migration Guide	GC26-4764	7576
PL/I for MVS & VM Compiler and Run-Time Migration Guide	SC26-3118	7576
AD/CYCLE C/370 Migration Guide	SC09-1359	7576

The following manuals are available as on-line books on the *Language Environment for MVS & VM Online Product Library*:

- IBM Language Environment for MVS & VM Concepts Guide
- IBM Language Environment for MVS & VM Programming Reference
- IBM Language Environment for MVS & VM Debugging Guide and Run-time Messages
- IBM Language Environment for MVS & VM Writing Interlanguage Communication Applications
- IBM Language Environment for MVS & VM Run-Time Migration Guide

2.3 Microfiche Support

There is no microfiche for Language Environment.

2.4 Publications Useful During Installation

The publications listed in Figure 9 may be useful during the installation of Language Environment. To order copies, contact your IBM representative.

Figure 8. Publications Useful During Installation / Service on VM/ESA Version 1

Publication Title	**Form Number**
VMSES/E Introduction and Reference	SC24-5444
VM/ESA CP Planning and Administration	SC24-5521
VM/ESA Service Guide	SC24-5527
VM/ESA CMS Command Reference	SC24-5461
VM/ESA SFS and CRR Planning, Administration, and Operation	SC24-5649
VM/ESA System Messages and Codes	SC24-5529
VMSES/E Introduction and Reference Release 1.5 370 Feature	SC24-5680

Figure 9. Publications Useful During Installation / Service on VM/ESA Version 2

Publication Title	**Form Number**
VMSES/E Introduction and Reference	SC24-5747
VM/ESA Service Guide	SC24-5749
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	SC24-5784

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 (Service Link) 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 10. PSP UPGRADE and SUBSET ID

UPGRADE	SUBSET
LEMVSVMR150	See Figure 11 below

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

Figure 11. Component IDs

Retain			
COMPID	**Release**	**Component Name**	**PSP SUBSET**
568819801	51N	Language Environment Base	VM/51N
568819801	51G	Language Environment Base - Mixed Case English	VM/51G
568819801	51J	Language Environment Base - Japanese	VM/51J
568819802	52N	Language Environment COBOL - Base	VM/52N
568819802	52G	Language Environment COBOL - Mixed Case English	VM/52G
568819802	52J	Language Environment COBOL - Japanese	VM/52J
568819803	53N	Language Environment PL/I - Base	VM/53N
568819803	53G	Language Environment PL/I - Mixed Case English	VM/53G
568819803	53J	Language Environment PL/I - Japanese	VM/53J
568819805	55N	Language Environment ″C″ - Base	VM/55N
568819805	55G	Language Environment ″C″ - Mixed Case English	VM/55G
568819805	55J	Language Environment ″C″ - Japanese	VM/55J
568819805	55C	Language Environment ″C″ - Japanese Msgs	VM/55C

4.0 Program and Service Level Information

This section identifies the program and any relevant service levels of IBM Language Environment for MVS & 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 Updates to the PL/I Component of IBM Language Environment for MVS & VM

The release of this product incorporates the following APARs and PTFs against the "PL/I" component of IBM Language Environment for MVS & VM in Figure 12 below.

Figure 12. APARs and Related PTFs against the PL/I Component of IBM Language Environment for MVS & VM

APAR	PTF(s) for VM
PN70410	UN76635
PN70413	UN76677
PN70423	UN76032
PN70424	UN76110
PN70425	UN76107
PN70544	UN76399
PN70859	UN76287
PN71569	UN78581
PN71760	UN77813
PN71097	UN79865
PN74410	UN80464

4.2 Service Updates to the COBOL Component of IBM Language Environment for MVS & VM

The release of this product incorporates the following APARs and PTFs against the “COBOL” component of IBM Language Environment for MVS & VM

Figure 13. APARs and Related PTFs against the COBOL Component of IBM Language Environment for MVS & VM

APAR	**PTF(s) for VM**
PN60433	UN73662 UN73663 UN79788
PN60476	UN70187 UN73663 UN79788
PN64656	UN78423
PN66485	UN76639
PN68190	UN75985 UN78719
PN68703	UN76333
PN70085	UN78113
PN70475	UN77490
PN70713	UN79788 UN79789
PN71379	UN80345
PN71570	UN78404
PN72497	UN78719
PN72981	UN79387
PN72982	UN79387
PN72983	UN79496
PN73472	UN79788 UN79789

4.3 Service Updates to the Base component of IBM Language Environment for MVS & VM

The release of this product incorporates the following APARs and PTFs against the Base component of IBM Language Environment for MVS & VM

Figure 14. APARs and Related PTFs against the Base Component of IBM Language Environment for MVS & VM

APAR	PTF(s) for VM
PN60910Z	UN77243
PN64276Z	UN77489
PN65018Z	UN76343
PN66097Z	UN78154
PN66497Z	
PN67457Z	UN76150
PN68449Z	UN76015
PN69352Z	UN76156
PN69674Z	UN76700
PN70209Z	
PN70575Z	
PN71735Z	UN77466
PN72252Z	UN78607

Note: Check the LE370R150 PSP bucket for further PTF information.

4.4 Cumulative Service Tape

There is no cumulative service tape for Language Environment. Cumulative service for this product is available through a monthly corrective service tape, Expanded Service Option (ESO).

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 Virtual Storage/Loader Table Considerations

During initial install, a minimum of 10M of virtual storage and 12 LDRTBLS is recommended. If customization is to be done, the minimum storage must be 22M as VMSES/E loads ″all″ of the required info into storage to rebuild.

Note: When loading Saved Segments via the SES panels, the Loader Tables are automatically set back to 4 after you ipl 190. Care should be taken that they are reset back to 12 prior to issuing the VMFBLD command to build the saved segments.

5.3 Program Considerations

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

5.3.1 Operating System Requirements

Language Environment operates under the currently-supported releases of any one of the following system environments (or subsequent releases):

- VM/ESA(R) Version 2 (5654-030)
- VM/ESA(R) Version 1 Release 1 Modification 5 (370 Feature)
- VM/ESA(R) Version 1 Release 2 Modification 1 (or higher)
- APAR **VM59732** MUST be applied to VM/ESA 2.1, 1.1.5, and 2.2 prior to installing Language Environment. Failure to do so will result in the failing of the install/service process of this product.
- APAR **VM59640** MUST be applied to VM/ESA 2.1 only as this fix is already in the base of 2.2 and follow on releases. Failure to do so could result in problems during the rebuilding of the saved segments after applying service.
- **RSU9405 service level** or above must be applied to VMSES/E on 2.1 and VM/ESA 1.5 370 Feature prior to installing Language Environment

- **RSU9403 service level** or above must be applied to VMSES/E on VM/ESA 2.2 prior to installing Language Environment

5.3.2 Other Program Product Requirements

Language Environment requires the following:

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

5.3.3 Program Installation / Service Considerations

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

- 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 “P688198E” 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.

  Note: It might be easier to make the changes during the installation procedure 6.2, “Plan Your Installation For Language Environment” step 6 on page 18, after you have installed this product.

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 Language Environment library contains library routines having names identical to those of the other C, COBOL or PL/I library routines.

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

5.4 DASD Storage and USERID Requirements

Figure 15 on page 15 lists the USERID and minidisks that are used to install and service Language Environment.

Important Installation Notes:

- User ID(s) and minidisks are defined in 6.2, “Plan Your Installation For Language Environment” on page 17 and are listed here so that you can get an idea of the resources that you need prior to allocating them.
- If you are installing on a VM/ESA 2.1.0 system, then the userid and minidisks are already defined. Use Figure 15 on page 15 to make sure the existing minidisks match the sizes shown. Your MAINT′s 19E disk may also have to be increased by 10 cylinders in order to handle the full product.
- P688198E 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. This can be done in 6.2, “Plan Your Installation For Language Environment” step 6 on page 18.
- If you choose to install Language Environment on a common User ID the default minidisk addresses for Language Environment might already be defined. If any of the default minidisks required by Language Environment are already in use you will have to create an override to change the default minidisks for Language Environment so they are unique.
- If you plan on installing the KANJI feature or customizing the “C” Run time options, the “191” disk size must be increased by a factor of 4 (e.g 6 Cylinders of 3380 must now be 24 cylinders of 3380). This is due to these two options rebuilding the SCEERUN LOADLIB during the updates and VM using the “A” disk as an interm work disk while building the loadlib.

Figure 15. DASD Storage Requirements for Target Minidisks

Minidisk owner (User ID)	**Default Address**	**Storage in Cylinders**		**FB-512 Blocks**	**SFS 4K Blocks**	**Usage**
		DASD	**CYLS**			
P688198E	2B2	9345 3390 3380 3375 3350	80 65 80 125 100	79200	12000	Contains all base code shipped with Language Environment SFS Name: **VMSYS:P688198E.LE370.BASE**
P688198E	2C2	9345 3390 3380 3375 3350	6 5 6 10 8	7200	900	Contains sample files and user local modifications for Language Environment SFS Name: **VMSYS:P688198E.LE370.LOCAL**
P688198E	2D2	9345 3390 3380 3375 3350	25 20 25 40 25	3600	3750	Contains serviced files SFS Name: **VMSYS:P688198E.LE370.DELTA**
P688198E	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 SFS Name: **VMSYS:P688198E.LE370.ALTAPPLY**
P688198E	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 SFS Name: **VMSYS:P688198E.LE370.PRODAPPLY**
P688198E	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.
P688198E	191	9345 3390 3380 3375 3350	45 35 45 65 55	7200	5250	P688198E User ID′s 191 minidisk **NOTE** See Section 5.4 “DASD Storage and User ID Requirements” if installing KANJI or customizing “C” runtime options. SFS Name: **VMSYS:P688198E.**

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 install and activate Language Environment.

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 2.2 system. If you′re installing Language Environment on a different VM/ESA system, the results obtained for some commands might differ from those depicted here.

6.1 VMSES/E Installation Process Overview

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

- Plan Your Installation

  The VMFINS command is used to load several VMSES/E files from the product tape and to obtain Language Environment resource requirements.

- Allocate Resources

  The information obtained from the previous step is used to allocate the appropriate minidisks (or SFS directories) and User IDs needed to install and use Language Environment.

- Install the Language Environment Product

  The VMFINS command is used to load the Language Environment product files from tape to the test BUILD and BASE minidisks/directories. VMFINS is then used to update the VM SYSBLDS file used by VMSES/E for software inventory management.

- 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.

- Performing Post installation Tasks

  Information about file tailoring is presented in 6.6, “Post-Installation Considerations” on page 30.

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

- *VMSES/E Introduction and Reference*

6.2 Plan Your Installation For Language Environment

The VMFINS command is used to plan the installation. This section has 2 main steps that will:

- load the first tape file, containing installation files
- generate a ′PLANINFO′ file listing
  - all User ID/mdisks requirements
  - required products

To obtain planning information for your environment:

1 Log on as Language Environment installation planner.

This User ID can be any ID that has read access to MAINT′s 5E5 minidisk and write access to the MAINT′s 51D minidisk.

2 Mount the Language Environment installation tape and attach it to the User ID at virtual address 181. The VMFINS EXEC requires the tape drive at virtual address 181.

3 Establish read access to the VMSES/E code.

link MAINT 5e5 5e5 rr
access 5e5 B

The 5E5 disk contains the VMSES/E code.

4 Establish write access to the Software Inventory disk.

link MAINT 51d 51d mr
access 51d D

The MAINT 51D disk is where the VMSES/E system-level Software Inventory and other dependent files reside.

Note: If another user already has the MAINT 51D mindisk linked in write mode (R/W), you′ll only obtain read access (R/O) to this minidisk. If this occurs, you′ll need to have that user relink the 51D in read-only mode (RR), and then reissue the above LINK and ACCESS commands. Do not continue with these procedures until a R/W link is established to the 51D minidisk.

5 Load the Language Environment product control files to the 51D minidisk.

vmfins install info (nomemo

The NOMEMO option loads the memos from the tape but does not issue a prompt to send them to the system printer. Specify the MEMO option if you want to be prompted for printing the memo.

This command performs the following:

- load Memo-to-Users
- load various product control files, including the Product Parameter File (PPF) and the PRODPART files
- create VMFINS PRODLIST on your A-disk. The VMFINS PRODLIST contains a list of products on the installation tape.

```
VMFINS2760I VMFINS processing started
VMFINS1909I VMFINS PRODLIST created on your A-disk
VMFINS2760I VMFINS processing completed successfully
Ready;
```

Figure 16. Sample console output from step 5

6 Obtain resource planning information for Language Environment.

Notes:

a. The product will **not** be loaded by the VMFINS command at this time.

vmfins install PPF 5688198E {LE370|LE370SFS} (nomemo plan

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

The PLAN option indicates that VMFINS will perform requisite checking, plan system resources, and provide an opportunity to override the defaults in the product parameter file.

You can override any of the following:

- the name of the product parameter file
- the default User IDs
- minidisk/directory definitions

Notes:

a. If you change the PPF name, a default User ID, or other parameters via a PPF override, you′ll need to use your changed values instead of those indicated (when appropriate), throughout the rest of the installation instructions, as well as those provided for servicing Language Environment. For example, you′ll need to specify your PPF override file name instead of 5688198E for certain VMSES/E commands.

b. If you′re not familiar with creating PPF overrides using VMFINS, you should review the ′Using the Make Override Panel′ section in Chapter 3 of the *VMSES/E Introduction and Reference* before you continue.

c. For more information about changing the VMSYS file pool name see Appendix A, "Overriding the VMSYS File Pool Name" on page 56 except if you are running VM/ESA Release 2.2 or VM/ESA Version 2 then refer to Chapter 3 in the *VMSES/E Introduction and Reference*.

```
vmfins install ppf 5688198e le370 (nomemo plan
VMFINS2767I Reading VMFINS DEFAULTS B for additional options
VMFINS2760I VMFINS processing started
VMFINS2601R Do you want to create an override for :PPF 5688198E LE370 :PRODID
            5688198E%LE370?
            Enter 0 (No), 1 (Yes) or 2 (Exit)
0
VMFINS2603I Processing product :PPF 5688198E LE370 :PRODID 5688198E%LE370
VMFREQ2805I Product :PPF 5688198E LE370 :PRODID 5688198E%LE370 has passed
            requisite checking
VMFINT2603I Planning for the installation of product :PPF 5688198E LE370 :PRODID
            5688198E%LE370
VMFRMT2760I VMFRMT processing started
VMFRMT2760I VMFRMT processing completed successfully
VMFINS2760I VMFINS processing completed successfully
Ready; T=7.54/8.08 07:40:20
```

Figure 17. Sample console output from step 6

7 Review the install message log ($VMFINS $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 online HELP.

vmfview install

6.3 Allocate Resources for Installing Language Environment

Note

If you are running under VM/ESA Version 2, the directory and minidisks have already been defined. In this case, the following steps may not be applicable. You should however, check figure 15 for the correct sizes and increase these sizes as appropriate.

Care should be taken to confirm that MAINT′s 19E is also large enough to handle the complete product should the test system disk be copied over to it.

Use the planning information in the 5688198E PLANINFO file, created in the **PLAN** step, to:

- Create the P688198E user directory for minidisk install

  OR

- Create the P688198E user directory for SFS install

6.3.1 Preparing to install Language Environment on Minidisk

1 Obtain the user directory from the 5688198E PLANINFO file.

Note: The user directory entry is located at the bottom of the PLANINFO file of the resource section. These entries contain all of the links and privilege classes necessary for the P688198E User ID Use the directory entry found in PLANINFO as a model as input to your system directory.

2 Add the MDISK statements to the directory entry for P688198E. Use Figure 15 on page 15 to obtain the minidisk requirements.

3 If you′re installing Language Environment on a VM/ESA 1.5 370 Feature system, the following directory entry changes must be made:

- For the P688198E directory entry:
  - Change the user ID storage from 24M to 16M.

4 Add the P688198E directory to the system directory. Change the passwords for P688198E from xxxxx to a valid password, in accordance with your security guidelines.

5 Place the new directories on-line using VM/Directory Maintenance (DIRMAINT) or an equivalent CP directory maintenance method.

Note

All minidisks for the P688198E User ID must be formatted before installing Language Environment.

6.3.2 Preparing to install Language Environment in SFS Directories

1 Obtain the user directory from the 5688198E PLANINFO file.

Note: The user directory entry is located at the bottom of the PLANINFO file of the resource section. These entries contain all of the links and privilege classes necessary for the P688198E User ID. Use the directory entry found in PLANINFO as a model as input to your system directory.

2 Add the MDISK statement for the 29E test build disk to the directory entry for P688198E Use Figure 15 on page 15 to obtain the minidisk requirements.

3 If you′re installing Language Environment on a VM/ESA 1.5 370 Feature system, the following directory entry changes must be made:

- For the P688198E directory entry:
  - Change the user ID storage from 24M to 16M.

4 Add the P688198E directory to the system directory. Change the passwords for P688198E from xxxxx to a valid password, in accordance with your security guidelines.

5 Place the new directories online using VM/Directory Maintenance (DIRMAINT) or an equivalent CP directory maintenance method.

6 An SFS install also requires the following steps:

a 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 15 on page 15

This information is used when enrolling the P688198E to the VMSYS filepool.

b Enroll user P688198E in the VMSYS filepool using the ENROLL USER command:

ENROLL USER P688198E VMSYS (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 VMSYS: filepool.

c Determine if there are enough blocks available in the filepool 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 filepool 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 filepool. See *VM/ESA SFS and CRR Planning, Administration, and Operation* manual for information on adding space to a filepool.

d Create the necessary subdirectories listed in the 5688198E PLANINFO file using the CREATE DIRECTORY command.

set filepool vmsys:
create directory *dirid*

dirid is the name of the SFS directory you′re creating, such as:

```
create directory vmsys:P688198E.LE370
create directory vmsys:P688198E.LE370.object
   :
```

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

A complete list of default LE370 SFS directories is provided in Figure 15 on page 15.

e If you intend to use an SFS directory as the work space for the P688198E used ID, include the following IPL control statement in the P688198E directory entry:

```
IPL CMS PARM FILEPOOL VMSYS
```

This will cause CMS to automatically access the P688198E′s top directory as file mode A.

6.4 Install Language Environment

The *ppfname* used throughout these installation instructions is **5688198E**, 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 **5688198E**. The *ppfname* you use should be used **throughout** the rest of this procedure.

The *compname* used throughout these installation instructions is either **LE370** or **LE370SFS**, which assumes you are using the component name within the 5688198E PPF file. If you specify your own *ppfname*, you should use the *compname* from that file instead of **LE370** or **LE370SFS**. The *compname* you use should be used **throughout** the rest of this procedure.

1 Logon to the installation User ID **P688198E**.

2 Create a PROFILE EXEC that will contain the ACCESS commands for MAINT 5E5 and 51D minidisks.

```
xedit profile exec a
===> input /**/
===> input ′access 5e5 B′
===> input ′access 51d D′
===> file
```

If either 5E5 or 51D is in shared file (SFS) then substitute your SFS directory name in the access command.

3 Execute the profile to access MAINT′s minidisks.

```
profile
```

4 Establish write access to the Software Inventory disk, if it is not already linked R/W.

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 R/W access to it.

```
link MAINT 51d 51d mr
access 51d D
```

5 Have the Language Environment installation tape mounted and attached to P688198E at virtual address 181. The VMFINS EXEC requires the tape drive at virtual address 181.

6 Install Language Environment.

Notes:

If you′ve already created a PPF override file, you should specify your override file name after the **PPF** keyword for the following VMFINS command. The **PROD** keyword should *not* be used.

You might be prompted for additional information during VMFINS INSTALL processing depending on your installation environment. If you′re unsure how to respond to a prompt, refer to the ′Installing Products with VMFINS′ and ″Install Scenarios′ chapters in *VMSES/E Introduction and Reference* to decide how to proceed.

```
vmfins install PPF 5688198E {LE370 | LE370SFS} (nomemo nolink
```

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

The NOLINK option indicates that you don′t want VMFINS to link to the appropriate minidisks, only access them if not accessed.

```
vmfins install ppf 5688198e le370 (nomemo nolink
VMFINS2767I Reading VMFINS DEFAULTS B for additional options
VMFINS2760I VMFINS processing started
VMFINS2601R Do you want to create an override for PPF 5688198E LE370 PRODID
            5688198E%LE370?
            Enter 0 (No), 1 (Yes) or 2 (Exit)
0
VMFINS2603I Processing product PPF 5688198E LE370 PRODID 5688198E%LE370
VMFREQ2805I Product PPF 5688198E LE370 PRODID 5688198E%LE370 has passed
            requisite checking
VMFINT2603I Installing product PPF 5688198E LE370 PRODID 5688198E%LE370
VMFSET2760I VMFSETUP processing started for 5688198E LE370
VMFUTL2205I Minidisk|Directory Assignments:
            String    Mode  Stat  Vdev  Label/Directory
VMFUTL2205I LOCALSAM  E      R/W  2C2   TLMODS
VMFUTL2205I APPLY     F      R/W  2A6   TAPPLZ
VMFUTL2205I           G      R/W  2A2   TAPPLY
VMFUTL2205I DELTA     H      R/W  2D2   TDELTA
VMFUTL2205I BUILD0    I      R/W  29E   TBUILD
VMFUTL2205I BASE1     J      R/W  2B2   TBASE
VMFUTL2205I --------  A      R/W  222   ATEMP
VMFUTL2205I --------  B      R/O  5E5   MNT5E6
VMFUTL2205I --------  D      R/W  51D   TMP51D
VMFUTL2205I --------  S      R/O  190   MNT490
VMFUTL2205I --------  Y/S    R/O  19E   ESA19E
ST:VMFSET2760I VMFSETUP processing completed successfully
ST:VMFREC2760I VMFREC processing started
ST:VMFREC1852I Volume 1 of 1 of INS TAPE 9400
ST:VMFREC1851I (1 of 8) VMFRCAXL processing AXLIST
ST:VMFRCX2159I Loading 0 part(s) to DELTA 2D2 (H)
ST:VMFREC1851I (2 of 8) VMFRCPTF processing PARTLST
ST:VMFRCP2159I Loading 0 part(s) to DELTA 2D2 (H)
ST:VMFREC1851I (3 of 8) VMFRCCOM processing DELTA
ST:VMFRCC2159I Loading 0 part(s) to DELTA 2D2 (H)
ST:VMFREC1851I (4 of 8) VMFRCALL processing APPLY
ST:VMFRCA2159I Loading part(s) to APPLY 2A6 (F)
ST:VMFRCA2159I Loaded 1 part(s) to APPLY 2A6 (F)
ST:VMFREC1851I (5 of 8) VMFRCALL processing BASEB
ST:VMFRCA2159I Loading part(s) to BASE1 2B2 (J)
ST:VMFRCA2159I Loaded 1346 part(s) to BASE1 2B2 (J)
ST:VMFREC1851I (6 of 8) VMFRCALL processing BASEP
ST:VMFRCA2159I Loading part(s) to BASE1 2B2 (J)
ST:VMFRCA2159I Loaded 2933 part(s) to BASE1 2B2 (J)
ST:VMFREC1851I (7 of 8) VMFRCALL processing BUILDP
ST:VMFRCA2159I Loading part(s) to BUILD0 29E (I)
ST:VMFRCA2159I Loaded 561 part(s) to BUILD0 29E (I)
ST:VMFREC1851I (8 of 8) VMFRCALL processing BUILDB
ST:VMFRCA2159I Loading part(s) to BUILD0 29E (I)
ST:VMFRCA2159I Loaded 125 part(s) to BUILD0 29E (I)
ST:VMFREC2760I VMFREC processing completed successfully
ST:VMFINT2603I Product installed
ST:VMFINS2760I VMFINS processing completed successfully
Ready; T=211.44/240.98 08:45:40
```

Figure 18. Sample console output from step 6

7 Review the install message log ($VMFINS $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 online HELP.

vmfview install

6.4.1 Update Build Status Table for Language Environment

1 If you are running on VM/ESA Version 2, you need to rename all occurrences of the **SRVBLDS** tables (found on the APPLY disks, 2A6 and 2A2). This is needed so that any service that may have already been applied (POSIX Environment) will get re-applied.

2 Update the VM SYSBLDS software inventory file for Language Environment.

vmfins build ppf 5688198E {LE370|LE370SFS} (serviced nolink

Use **LE370** if installing using minidisks or **LE370SFS** if installing using SFS

The **serviced** option will build any parts that are flagged as serviced on the install tape. These parts might consist of saved segments or other parts which cannot be shipped on a VMSES/E formatted install tape.

```
VMFINS BUILD PPF 5688198E LE370 (SERVICED NOLINK

VMFINS2767I Reading VMFINS DEFAULTS B for additional options
VMFINS2760I VMFINS processing started
VMFINS2603I Building product 5688198E LE370 5688198E%LE370.
            5688198E LE370 5688198E%LE370 is a VMSES product
VMFREQ2805I Product 5688198E LE370 5688198E%LE370 has passed requisite
            checking
VMFSET2760I VMFSETUP processing started
VMFUTL2205I Minidisk|Directory Assignments:
            String    Mode  Stat  Vdev  Label/Directory
VMFUTL2205I LOCALSAM  E      R/W  2C2   TLMODS
VMFUTL2205I APPLY     F      R/W  2A6   TAPPLX
VMFUTL2205I           G      R/W  2A2   TAPPLY
VMFUTL2205I DELTA     H      R/W  2D2   TDELTA
VMFUTL2205I BUILD0    I      R/W  29E   TBUILD
VMFUTL2205I BASE1     J      R/W  2B2   TBASE
VMFUTL2205I --------  A      R/W  222   ATEMP
VMFUTL2205I --------  B      R/W  5E5   VM5E5
VMFUTL2205I --------  D      R/W  51D   VM51D
VMFUTL2205I --------  S      R/O  190   MT2190
VMFUTL2205I --------  Y/S    R/O  19E   ESA19E
VMFSET2760I VMFSETUP processing completed successfully
VMFBLD2760I VMFBLD processing started
VMFBLD1851I Reading build lists
VMFBLD2182I Identifying new build requirements
VMFBLD2182I No new build requirements identified
VMFBLD2179I There are no build requirements matching your request at this time.
            No objects will be built
VMFBLD2180I There are 0 build requirements remaining
VMFBLD2760I VMFBLD processing completed successfully
```

Figure 19 (Part 1 of 3). Sample Install/Verification console

```
VMFINB2173I Executing verification exec V5688198
==========================================================
===  Start of BASE component verification
==========================================================

DMSLIO740I Execution begins...

 Today is FRIDAY, 11 AUGUST 1995.
 Today is FRI, AUG 11, 1995 6:28 AM
 Today is FRIDAY, 08/11/95 06:28:38.23
 Today is day 223 of 1995
 Program Complete.
 ==========================================================
 ===  Verification of BASE component is complete.
 ==========================================================

Press ENTER key to continue
==========================================================
===  Start of C/370 Library component verification
==========================================================

*** IBM C run-time component of Language Environment for MVS & VM
    Version 1 Release 5 Modification 0 ***
=== C/370 Library Verification Successful.
Hello SPC !    *** Test successful ***
=== C/370 SPC Verification Successful.
*** IBM C run-time component of Language Environment for MVS & VM
    Version 1 Release 5 Modification 0 ***
=== C/370 Prelinker Verification Successful.
 ==========================================================
 ===  Verification of C/370 Library component is complete.
 ==========================================================

Press ENTER key to continue
==========================================================
===  Start of COBOL component verification
==========================================================

DMSLIO740I Execution begins...
***** START OF CALLIVP1 *****
***** CALLIVP1 SUCCESSFUL *****
 ==========================================================
 ===  Verification of COBOL component is complete.
 ==========================================================

Press ENTER key to continue
```

Figure 19 (Part 2 of 3). Sample Install/Verification console

```
═══════════════════════════════════════════════════
══  Start of PL/I component verification
═══════════════════════════════════════════════════

DMSLIO740I Execution begins...

 ***********************
 *** Word-use Report ***
 ***********************
 -count-   --word--
     3      BEGIN
     1      CLOSE
    13      DCL
    24      DECLARE
     2      DISPLAY
    14      DO
    13      ELSE
    23      END
     1      GO
    13      IF
------------The previous value should have been    14
     7      LIST
     4      ON
     1      OPEN
     2      PROC
     3      PROCEDURE
     2      READ
     4      RETURN
     1      SELECT
     2      STOP
    13      THEN
     2      WHEN

There were            148 references to    36 words.

There was a discrepancy in at least one of the word-counts.

 ══════════════════════════════════════════════════════
 ==  A discrepancy is expected in this test.          ══
 ==                                                   ══
 ==  Verification of PL/I component is complete.      ══
 ══════════════════════════════════════════════════════
Ready; T=1.24/1.78 06:28:53
```

Figure 19 (Part 3 of 3). Sample Install/Verification console

6.5 Place Language Environment Into Production

6.5.1 Copy Language Environment 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 P688198E 29e 29e rr
access 29e e
access 19e f
vmfcopy * * e = = f2 (prodid 5688198E%LE370 olddate replace
```

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

Language Environment is now installed and built on your system.

6.6 Post-Installation Considerations

Upon successfull installation, the following items can now be implemented:

1. Installation of NLS feature (KANJI)
2. Customization
3. Installation into Saved Segments

See Chapter 8.0, “Selecting/Installing National Languages” on page 39 for a full description on how to install the NLS feature and Chapter 9.0, “Customizing Language Environment” on page 41 for a full description on how to customize and install in saved segments.

7.0 Service Instructions

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*

These manuals 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

7.2.1 Prepare to Receive Service

The *ppfname* used throughout these servicing instructions is **5688198E**, 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 **5688198E**. The *ppfname* you use should be used **throughout** the rest of this procedure, unless otherwise stated differently.

The *compname* used throughout these servicing instructions is either **LE370** or **LE370SFS**, which assumes you are using the component name within the 5688198E PPF file. If you specify your own *ppfname*, you should use the *compname* from that file instead of **LE370** or **LE370SFS**. The *compname* you use should be used **throughout** the rest of this procedure.

Note

If you are running on VM/ESA Version 2 and have **NOT** installed the full Language Environment product, then the *compname* you should use in all of the VMSES/E service commands **MUST** be **POSIX**.

1 Log onto Language Environment service User ID **P688198E**

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 ***P688198E***.

4 Establish the correct minidisk access order.

vmfsetup 5688198E {LE370|LE370SFS}

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

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 Service Link (electronic service) see Appendix A, ′Receiving Service for VMSES Envelopes′, section ″Receive the Service″ in the *VM/ESA Service Guide*. When complete, return back to this step.

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.

vmfmrdsk 5688198E {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 Service Link (electronic service) see Appendix A, ′Receiving Service for VMSES Envelopes′, in *VM/ESA Service Guide*

1 Receive the service.

```
vmfrec ppf 5688198E {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 5688198E {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 **5688198E**
- **esa** should be **5688198E**
- *compname* should be **LE370** or **LE370SFS** (minidisk or SFS)
- *appid* should be **5688198E**
- *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 5688198E {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 5688198E $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 5688198E $PPF
on 191 (A) from level $PFnnnnn
```

vmfppf 5688198E *{LE370 | LE370SFS}*

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

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

copyfile 5688198E $PPF a = = d (olddate replace
erase 5688198E $PPF a

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

vmfbld ppf 5688198E *{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 5688198E {LE370|LE370SFS} (serviced

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

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 Language Environment on a VM/ESA 1.5 370 Feature system, refer to section 10.1, “Defining Saved Segments on VM/ESA 1.5 370 Feature” on page 47, step 7. Upon completion of that step, return back to section 7.3.2.

2 If Language Environment is installed on a system with VMSES/E 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

3 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 P688198E 29e 29e rr
access 29e e
access 19e f
vmfcopy * * e = = f2 (prodid 5688198E%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 Selecting/Installing National Languages

Note

Due to the size and having to rebuild the SCEERUN LOADLIB for this install option, your “A” disk, which VM uses as an interm work disk during the rebuild, must be at least 20 cylinders of 3390 or equivalent.

When installing Language Environment you can choose the national language you want used for things such as system and Language Environment text, run-time messages, Language Environment reports, and output of such Language Environment 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. See Appendix E, “Language Environment Run-time Options” on page 70 for a description of the NATLANG runtime option.

Language Environment′s KANJI NLS feature will be installed through VMSES/E support using the User ID, P688198E and the following commands:

1 Install KANJI NLS Message modules.

```
VMFBLD PPF 5688198E LE370KANJI EDCBLHPJ (ALL
```

2 Rebuild the SCEERUN LOADLIB with updated Language Function.

```
VMFBLD PPF 5688198E LE370KANJI EDCBLRUN CEEEV003 (ALL
```

3 Should you wish to return to Mixed/Upper Case English Messages simply rebuild the required language function modules by issuing the following commands:

```
VMFBLD PPF 5688198E LE370 EDCBLHPE (ALL
VMFBLD PPF 5688198E LE370 EDCBLRUN CEEEV003 (ALL
VMFBLD PPF 5688198E 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 EDCPMSGE.MSGS
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

VMFBLD PPF 5688198E LE370KANJI EDCBLRUN CEEEV003 (ALL

VMFBLD2760I VMFBLD processing started
VMFBLD1851I Reading build lists
VMFBLD2182I Identifying new build requirements
VMFBLD2182I New build requirements identified
VMFBLD1851I (1 of 1) VMFBDLLB processing EDCBLRUN EXEC
VMFLLB2219I Processing object CEEEV003
VMFBLD1851I (1 of 1) VMFBDLLB completed with return code 0
VMFBLD2180I There are 0 build requirements remaining
VMFBLD2760I VMFBLD processing completed successfully
Ready; T=147.84/153.11 11:07:12
```

Figure 20. Sample KANJI install console.

9.0 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;
   - Language Environment Runtime Options
   - Language Environment User Exit
   - COBOL COBPACKs
   - ′C′ Component Locale Time Information
   - Saved Segements Components
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 5 Mod 0

      1)  Run Time Options
      2)  User Exits
      3)  †C† Locale Time Information
      4)  Saved Segments
      5)  COBOL COBPACKs

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

Figure 21. Customization EXEC - Panel 1

9.1 Updating Run-Time Options

Language Environment 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, “Language Environment Run-time Options” on page 70 for a complete description of the run-time options.

9.2 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."

9.3 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." Modules which will be rebuilt are IGZCPAC and IGZCPCO which are in Build List "IGZBLPAC."

9.4 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 interm work disk during the rebuild, must be at least 20 cylinders of 3390 or equivalent.

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.

9.5 Updating Saved Segments

After successfully installing IBM Language Environment for MVS & 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 Language Environment are four build lists (CEEBLSGA and CEEBLSGB for the full Language Environment environment and CEEBLSPA and CEEBLSPB for POSIX environment) plus the necessary LSEG files required to install specific routines of Language Environment into segments. By selecting option 5 in the Customization Exec, these individual build lists can be tailored to load only specific routines of the Language Environment 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 5 Modification 0                 *
*                                                                   *
*              Licensed Materials -- Property of IBM                *
*          5688-198  (C) Copyright IBM Corporation 1995             *
*                       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 IGZCPCO  )
: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 C, "Segment Build Lists (CEEBLSGA/CEEBLSGB)" on page 64 and Appendix D, "Segment Build Lists (CEEBLSPA/CEEBLSPB) - POSIX" on page 67 for a full description of the segment build lists.

For more information on the defining/loading of the Saved segments for Language Environment, see Chapter 10.0, "Define and Build The Language Environment Saved Segments" on page 46.

9.6 Installing in Saved Segments

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

```
     SCEE (CEEBLSGB)                 SCEEX  (CEEBLSGA)
       (Below Line)                    (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
   ** COBOL Components **             CEEMUEN3   7420
      IGZCPAC   1AF18                 CEEMUEN4   0460
      IGZCPCO   BA90                  CEEMUEN5   3598
                                      CEEMENU0   2890
   **  PL/I Components **             CEEMENU2   2DB8
                                      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 22. Component Module Size

10.0 Define and Build The Language Environment Saved Segments

It is recommended that segments be built for 5688198E. First the segments are defined to the system using the segment mapping tool VMFSGMAP. Once the segments are defined VMFBLD is used to build them.

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.

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

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

Note: The defining 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.

Note

Installing saved segments under the VM/ESA Release 1.5 370 Feature, is described in Section 10.1 while installing saved segments under VMSES/E is described in Section 10.2

10.1 Defining Saved Segments on VM/ESA 1.5 370 Feature

VMSES/E does not support the build function of saved segments on VM/ESA Release 1.5 370 Feature.

Notes:

You must

1. Have Class E privileges to install a saved segment.
2. Have a virtual storage size at least 0.5 MB greater than the address of the end of the segment.
3. Ensure that the shared segment does not overlap any other shared segment or saved system. For details, see the *VM/ESA CP Planning and Administration for 370* manual.

The process to create a saved segment under VM/ESA Release 1.5 370 Feature is as follows:

- Define saved segment by means of an OVERRIDE file
- Load the saved segment by the SEGGEN command
- Update the CMS system disk

10.1.1 Saved Segment Build for VM/ESA 370 Feature

1 Logon to installation userid **P688198E**.

2 Establish read access to VMSES/E code.

link MAINT 5e5 5e5 rr
access 5e5 b

The 5E5 disk contains the VMSES/E code.

3 Establish write access to the Software Inventory disk.

link MAINT 51d 51d mr
access 51d d

4 Establish disk access to Language Environment.

vmfsetup 5688198E {LE370|LE370SFS}

5688198E is the PPF shipped with Language Environment. If you have your own PPF override, substitute your PPF name for 5688198E.

Use **LE370** if using minidisk or **LE370SFS** if using Shared File System directories.

5 Add following entry to OVERRIDE file.

The recommended method to generate the Library shared segment is to make an entry for the segment in the SNT OVERRIDE file and activate it with the OVERRIDE command.

Notes:

a. Please be aware this is only a suggested DEFSEF TAG, as it will vary depending on other products.

b. *segname* can be anything. The default Language Environment segment can be either *SCEE|SCEEX*

```
:DEFSEG.segname      /* segment name             */
Volume=cccccc        /* volume serial number     */
SaveLoc=(mm,nn)      /* starting location on Vol.*/
Size=320K            /* required but ignored     */
PageCount=80         /* number of pages          */
PageList=(2432-2511) /* page numbers, from-to    */
SegList=(152-156)    /* segment numbers, from-to */
IPLAddr=IGNORE       /* must be IGNORE for a SS  */
```

6 Issue OVERRIDE command.

override fn ft fm (immediate	**fn ft fm** should be the name of the override file containing the *segname* DEFSEG entry.

7 Invoke SEGGEN to build and save physical saved segment.

SEGGEN *segname*

NOTE

A new file, SYSTEM SEGID, now appears on your A-disk (or other READ/WRITE disk, depending on the location of the MODULEs). You must copy this file to the S-disk to replace the SYSTEM SEGID that is currently there. Remember to re-SAVE CMS to avoid the `Shared S-STAT not available` message.

For VM/ESA 2.0 and above, the SYSTEM SEGID is updated on MAINT 51D. You must copy this file to the S-disk to stay in sync with the system's SEGID.

The *segname* segment is now generated. Any users who log on to the system after this time receive the benefits of this. All MODULEs stated in SEQASEG are now loaded from Saved segments instead of from the MODULE on the disk.

10.2 Define and Build Saved Segments Using VMSES/E

1 Logon to the installation User ID **P688198E**.

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

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

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

```
vmfsgmap segbld esasegs segblist
```

This command displays a panel for making segment updates. See Figure 23 on page 50 for an example of the Segment Map panel that will be displayed.

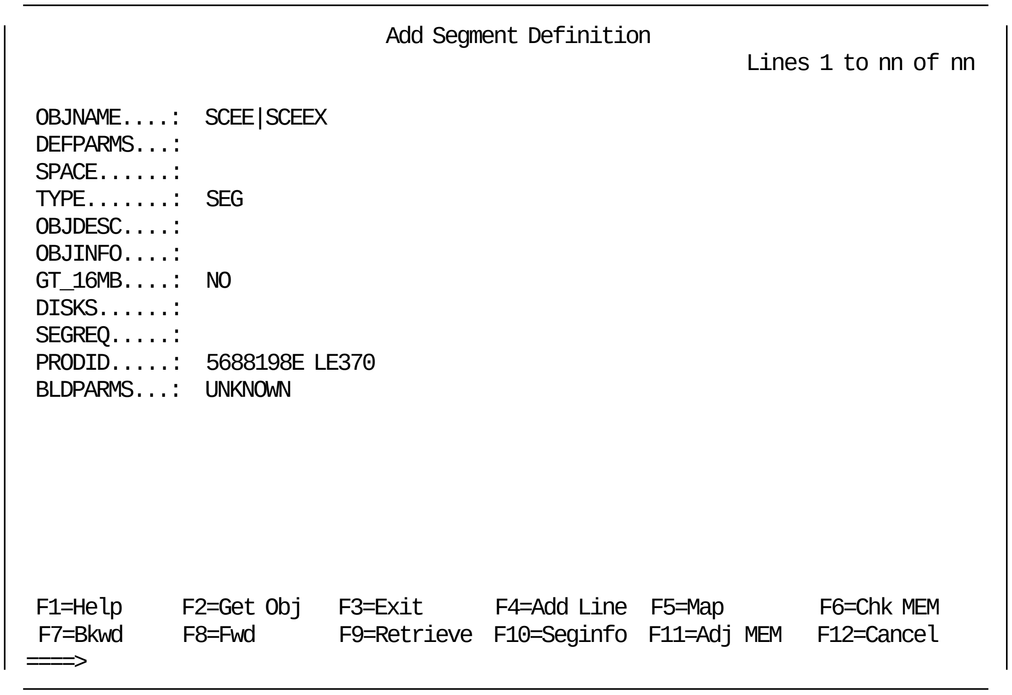
```
                           VMFSGMAP - Segment Map                 More: +
                                                         Lines 1 to nn of nn

              000-MB          001-MB          002-MB          003-MB
  Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
M CMS      SYS W-W-------------1...............2...............3...............
M GCS      SYS W---------------1...............2...............3...............

              004-MB          005-MB          006-MB          007-MB
  Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
  CMSPIPES DCS 4...............5...............6...............RRRR------------
M GCS      SYS RRRRRRNNNNNNNNNNNNNNNNNNNNNNNNNN6...............7...............
M HLASM    DCS 4...............5...............RRRRRRRRRRRRRRRR7...............

              008-MB          009-MB          00A-MB          00B-MB
  Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
  DOSBAM   SPA 8...............9...............A...............====------------
  CMSBAM   MEM 8...............9...............A...............BRRR............
  CMSDOS   MEM 8...............9...............A...............R...............
  CMSVMLIB DCS RRRRRRRRRRRRRRRR9...............A...............B...............
  DOSINST  DCS 8...............R---------------A...............B...............

              00C-MB          00D-MB          00E-MB          00F-MB
  Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
  HELPINST DCS RRRRRRRRRRRRRRRRD...............E...............F...............
M CMS      SYS C...............D...............RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR>

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

              010-MB          011-MB          012-MB          013-MB
 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 23. Segment Map panel example.

4 Go to Add Segment Definition panel by pressing PF10.

F10

F10 will take you from the Segment Map panel to the Add Segment Definition panel. See Figure 24 on page 51 to see the Add Segment Definition panel that will be displayed.



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

 OBJNAME....:  SCEE|SCEEX
 DEFPARMS...:
 SPACE......:
 TYPE.......:  SEG
 OBJDESC....:
 OBJINFO....:
 GT_16MB....:  NO
 DISKS......:
 SEGREQ.....:
 PRODID.....:  5688198E LE370
 BLDPARMS...:  UNKNOWN







 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 24. Initial "Add Segment Definition" panel example.

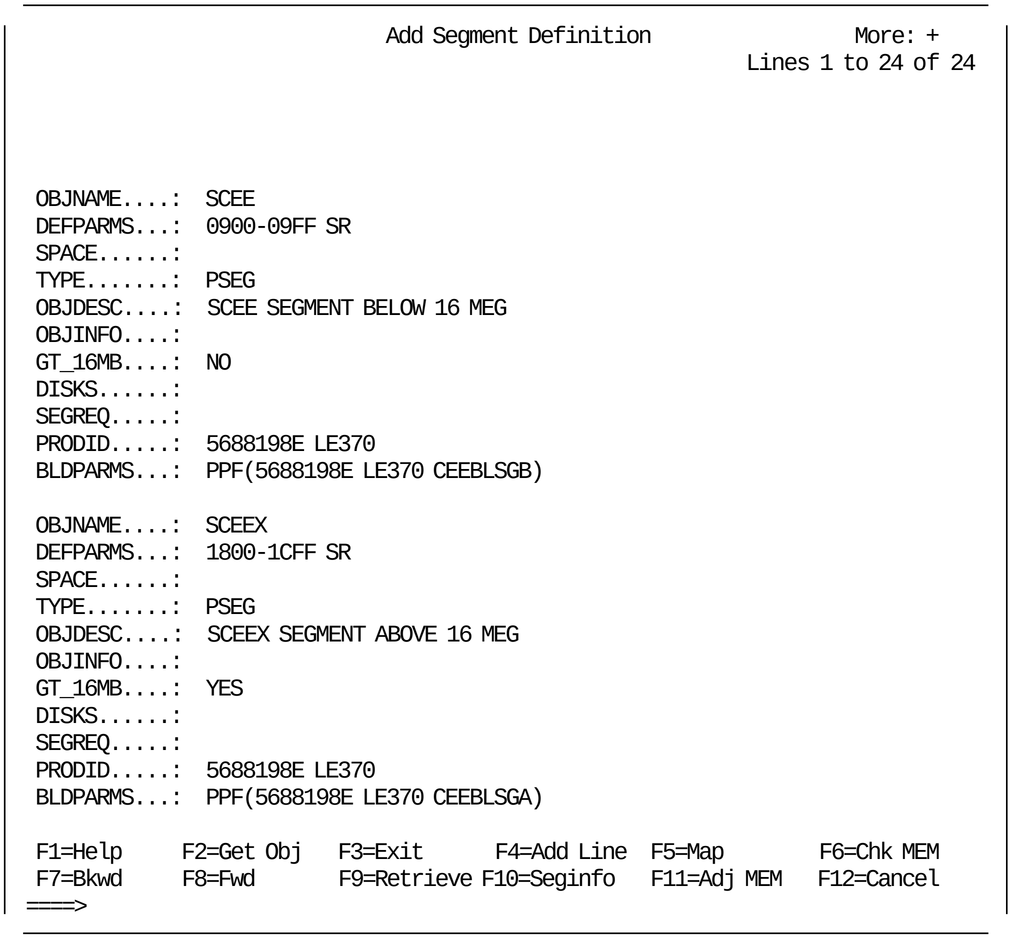
5 Obtain the LE370 segment definitions from the prodpart file..

OBJNAME....: **{SCEE|SCEEX}**
PRODID.....: **5688198E {LE370|POSIX}**
	Use **SCEE** for building below the 16M line or **SCEEX** for building above the 16M line.
	
	Use **LE370** component name if running with full Language Environment product installed or **POSIX** component name if running with VM/ESA 2.1.0 component only.

F10	**F10** will obtain the Language Environment segment information from the 5688198E PRODPART file. See Figure 25 on page 52 for the refreshed Add Segment definition panel that will be displayed.

Notes:

Not entering an *OBJNAME* parm, will cause both segments to be displayed at the same time as shown in Figure 25 on page 52. Entering an *OBJNAME* parm will result in seeing only the segment requested.



```
                     Add Segment Definition           More: +
                                                Lines 1 to 24 of 24



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

 OBJNAME....:  SCEEX
 DEFPARMS...:  1800-1CFF SR
 SPACE......:
 TYPE.......:  PSEG
 OBJDESC....:  SCEEX SEGMENT ABOVE 16 MEG
 OBJINFO....:
 GT_16MB....:  YES
 DISKS......:
 SEGREQ.....:
 PRODID.....:  5688198E LE370
 BLDPARMS...:  PPF(5688198E 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 25. Segment Definition panel showing SCEE Segment information

F5	Confirm information shown on panels is correct and press **F5** to return to the Segment Map panel. See Figure 26 on page 53 for the refreshed Segment Map panel that will be displayed.

```
                             VMFSGMAP - Segment Map               More: +
                                                          Lines 1 to 25 of 32


Meg             008-MB          009-MB          00A-MB          00B-MB
St Name    Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
  SCEE     DCS 8...............RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRB................
M SEQA     DCS 8...............RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRB................
M GCS2     SYS 8...............RRRRRRNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
M CMS11A   SYS C...............D...............RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR>

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

Meg             018-MB          019-MB          01A-MB          01B-MB
St Name    Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
M QMF310E  MEM RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR................
  SCEEX    DCS RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR>

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

Meg             030-MB          031-MB          032-MB          033-MB
St Name    Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
M VSF260D  DCS R---------------1...............2...............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 26. Segment Map panel with SCEE/SCEEX 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.

7 Prepare to build the segments.

a IPL CMS to clear the virtual storage

ipl 190 clear parm nosprof instseg no ***** DO NOT press ENTER at the VMREAD!*****	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).
access (noprof	Bypass the execution of the PROFILE EXEC.

b Access the VMSES/E code

access 5E5 B

c Establish write access the Software Inventory Disk

link maint 51D 51D mr
access 51d D

d Reset Loader Tables back to 12.

Set LDRTBLS 12

8 Issue VMFBLD command to build the Language Environment segments.

vmfbld ppf segbld esasegs segblist {SCEE|SCEEX} (all

```
Ready; T=1.39/1.53 05:56:57
vmfbld ppf segbld esasegs segblist scee (all
VMFBLD2760I VMFBLD processing started
VMFBLD1851I Reading build lists
VMFBLD2182I Identifying new build requirements
VMFBLD2182I New build requirements identified
VMFBLD1851I (1 of 1) VMFBDSEG processing SEGBLIST EXC00000, 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) VMFBDSEG completed with return code 4
VMFBLD2180I There are 9 build requirements remaining
VMFBLD2760I VMFBLD processing completed with warnings
Ready(00004); T=22.05/23.26 05:57:59
```

Figure 27. Sample Console ouput 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) VMFBDSEG processing SEGBLIST EXC00000, 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
GLOBAL TXTLIB EDCNSS01 SCEELKED
GLOBAL TXTLIB EDCNSS02 EDCNSS03 SCEELKED
GLOBAL TXTLIB EDCNSS02 EDCNSS03 SCEELKED
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) VMFBDSEG completed with return code 4
VMFBLD2180I There are 8 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 28. Sample Console ouput for SCEEX Segment Load

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

vmfview build

10 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.

Appendix A. Overriding the VMSYS File Pool Name

This section provides information to help you change the name of the file pool where Language Environment files will reside when Language Environment is installed using SFS directories, if you are running on a VM/ESA system at a lower level than VM/ESA 1.2.2. If you are on VM/ESA 1.2.2 then this support has been added into VMSES/E so refer to the *VMSES/E Introduction and Reference* instead of this appendix.

During the VMFINS installation process, you are presented with an opportunity to override the default installation parameters defined in the 5688198E $PPF file. If you choose to do this, the ′Make Override Panel′ will be displayed, from which you can then change various installation parameters, including the SFS directory names used to organize the Language Environment files. However, this panel does not support changing the name of the file pool with which these directories are associated—VMSYS.

VMSYS is the IBM default name for a file pool that′s intended to be used for system data and programs that are to be shared among users. See *VM/ESA Planning and Administration* for more information about the VMSYS file pool and its characteristics.

If you intend to change **only** the VMSYS file pool name, you′ll need to manually create a PPF override for the **:DCL.** section of the 5688198E $PPF file **before** you install Language Environment, as described in 6.4, "Install Language Environment" on page 22.

If you intend to change the VMSYS file pool name in addition to other installation parameters, you should first create a PPF override file during the installation process to change those parameters, then modify the resulting $PPF override file to account for the VMSYS-related changes.

Note: Do **not** modify the product supplied 5688198E $PPF or 5688198E PPF files to change the VMSYS file pool name or any other installation parameters. If the 5688198E $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 the override file created via the ′Make Override Panel′ function.

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. This override will be used for the installation of Language Environment.

```
:OVERLST. LE370SFS
*
*==========================================================*
:LE370SFS. LE370
*==========================================================*
:DCL. UPDATE
 &191    DIR MYPOOL1:P688198E.                  * A DISK
 &SAMPZ  DIR MYPOOL1:P688198E.LE370.LOCAL       * SAMPLE/LOCAL FILES
 &DELTZ  DIR MYPOOL1:P688198E.LE370.DELTA       * PRODUCT SERVICE
 &APPLY  DIR MYPOOL1:P688198E.LE370.ALTAPPLY  * AUX/INVENTORY FILES
 &APPLZ  DIR MYPOOL1:P688198E.LE370.PRODAPPLY * PROD. APPLY DISK
 &BLD0Z  LINK  P688198E  39E  39E  MR           * TEST USABLE FORMS
 &BAS1Z  DIR MYPOOL1:P688198E.LE370.OBJECT    * BASE DISK
 &LE3ID1 USER  P688198E
:EDCL.
:END.
*==========================================================*
*
```

This override will replace the ***:DCL.*** section of the LE370SFS override area of the 5688198E $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 5688198E as the PPF name to be used for those VMSES/E commands that require a PPF name.

Appendix B. Contents of COBPACKs (IGZCPAC/IGZCPCO)

B.1 Contents of General COBPACK - IGZCPAC

Figure 29 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 29 (Page 1 of 3). 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
IGZCINS	INSPECT	Both	31	ANY
IGZCIN1	INSPECT library	Both	31	ANY

Figure 29 (Page 2 of 3). Routines Eligible for Inclusion in General COBPACK (IGZCPAC)

Name	Description	OS/ CICS*	Link- Edited AMODE	Link- Edited RMODE
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
IGZCMSG	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
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

Figure 29 (Page 3 of 3). Routines Eligible for Inclusion in General COBPACK (IGZCPAC)

Name	Description	OS/ CICS*	Link-Edited AMODE	Link-Edited RMODE
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
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.

B.2 Contents of the Environment-Specific COBPACK (IGZCPCO)

Figure 30 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 30 (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
CEEBLLST[5]	Language list CSECT	31	ANY
CEEBPIRA[5]	Common initialization	31	ANY
CEEBTRM[5]	Common termination	31	ANY
IGZCSG5	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
IGZEVAM[6]	VSAM-to-IDCAMS interface	31	24

Figure 30 (Page 2 of 2). Routines Eligible for Inclusion in the Environment-Specific COBPACK (IGZCPCO)

Name	**Description**	**Link-Edited AMODE**	**Link-Edited RMODE**
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 C. Segment Build Lists (CEEBLSGA/CEEBLSGB)

C.1 CEEBLSGB

```
************************************************************
*       IBM Language Environment for MVS & VM              *
*         Version 1 Release 5 Modification 0               *
*                                                          *
*       Licensed Materials -- Property of IBM              *
*     5688-198  (C) Copyright IBM Corporation 1995         *
*               All Rights Reserved                        *
************************************************************
*Buildlist CEEBLSGB for ¢SCEE PSEG¢ Saved Segment (Below 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 IGZCPCO  )
:EOBJNAME.
*
```

Figure 31. Contents of CEEBLSGB Build List

C.2 CEEBLSGA

```
*************************************************************
*     IBM Language Environment for MVS & VM                 *
*       Version 1 Release 5 Modification 0                  *
*                                                           *
*      Licensed Materials -- Property of IBM                *
*   5688-198 (C) Copyright IBM Corporation  1995            *
*              All Rights Reserved                          *
*************************************************************
*Build list CEEBLSGA for ¢SCEEX¢ Saved Segment (Above Line)    *
*************************************************************
*
:FORMAT. 2
*
:OBJNAME. SCEEX.SEGMENT
:BLDREQ.  CEEBLMOD.CEECOPP.MODULE
          CEEBLMOD.CEEMUEN0.MODULE
          CEEBLMOD.CEEMUEN2.MODULE
          CEEBLMOD.CEEMUEN3.MODULE
          CEEBLMOD.CEEMUEN4.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
          EDCBLNS1
          EDCBLNS2
          EDCBLNS3
```

Figure 32 (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 EDCNSS01 PROFILE NSS01)
          LOADFUNC ( LSEG EDCNSS02 PROFILE NSS02)
          LOADFUNC ( LSEG EDCNSS03 PROFILE NSS03)
          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 32 (Part 2 of 2). Contents of CEEBLSGA Build List

Appendix D. Segment Build Lists (CEEBLSPA/CEEBLSPB) - POSIX

D.1 CEEBLSPB

```
*************************************************************
*        IBM Language Environment for MVS & VM              *
*          Version 1 Release 5 Modification 0               *
*                                                           *
*        Licensed Materials -- Property of IBM              *
*     5688-198  (C) Copyright IBM Corporation 1995          *
*               All Rights Reserved                         *
*************************************************************
*Buildlist CEEBLSPB for ¢SCEE PSEG¢ Saved Segment (Below line)*
*************************************************************
*
:FORMAT. 2
*
:OBJNAME. SCEE.SEGMENT
:BLDREQ.  CEEBLMOD.CEEBINIT.MODULE
          CEEBLMOD.CEEBLIIA.MODULE
          CEEBLMOD.CEEPIPI.MODULE
          CEEBLMOD.CEEBPICI.MODULE
          EDCBLSP2
:GLOBAL. TXTLIB SCEESPC
:OPTIONS. LOADFUNC ( LSEG CEEBINIT )
          LOADFUNC ( LSEG CEEBLIIA )
          LOADFUNC ( LSEG CEEPIPI  )
          LOADFUNC ( LSEG CEEBPICI )
:EOBJNAME.
*
```

Figure 33. Contents of CEEBLSPB Build List

D.2 CEEBLSPA

```
*****************************************************************
*     IBM Language Environment for MVS & VM                     *
*       Version 1 Release 5 Modification 0                      *
*                                                               *
*      Licensed Materials -- Property of IBM                    *
*   5688-198 (C) Copyright IBM Corporation  1995                *
*              All Rights Reserved                              *
*****************************************************************
*Buildlist CEEBLSPA for ¢SCEEX¢ Saved Segments - (Above line)  *
*****************************************************************
*
:FORMAT. 2
*
:OBJNAME. SCEEX.SEGMENT
:BLDREQ.  CEEBLMOD.CEECOPP.MODULE
          CEEBLMOD.CEEMUEN0.MODULE
          CEEBLMOD.CEEMUEN2.MODULE
          CEEBLMOD.CEEMUEN3.MODULE
          CEEBLMOD.CEEMUEN4.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
          EDCBLNS1
          EDCBLNS2
          EDCBLNS3
```

Figure 34 (Part 1 of 2). Contents of CEEBLSPA 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 EDCNSS01 PROFILE NSS01)
          LOADFUNC ( LSEG EDCNSS02 PROFILE NSS02)
          LOADFUNC ( LSEG EDCNSS03 PROFILE NSS03)
          LOADFUNC ( LSEG EDCZUMSG )
          LOADFUNC ( LSEG CEEEV010 )
          LOADFUNC ( LSEG CEEMENU0 )
          LOADFUNC ( LSEG CEEMENU2 )
          LOADFUNC ( LSEG CEEMENU3 )
          LOADFUNC ( LSEG EDCZEMSG )
          LOADFUNC ( LSEG CEEMJPN0 )
          LOADFUNC ( LSEG CEEMJPN2 )
          LOADFUNC ( LSEG CEEMJPN3 )
          LOADFUNC ( LSEG CEEMJPN4 )
          LOADFUNC ( LSEG CEEMJPN5 )
          LOADFUNC ( LSEG EDCZJMSG )
:EOBJNAME.
*
```

Figure 34 (Part 2 of 2). Contents of CEEBLSPA Build List

Appendix E. Language Environment 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 35 (Page 1 of 5). Run-Time Options Quick Reference

Run-Time Options	Function	Page
NONE OVR ABPERC = ((abcode) , NONOVR)	Percolates a specified abend.	81
RETCODE OVR ABTERMENC = ((ABEND) , NONOVR)	Sets the enclave termination behavior for an enclave ending with an unhandled condition of severity 2 or greater.	82
OFF OVR AIXBLD = ((ON) , 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.	84
OFF OVR ALL31 = ((ON) , NONOVR)	Indicates whether an application does or does not run entirely in AMODE(31).	85
ANYWHERE ANYHEAP = ((init_size , incr_size , ANY , BELOW FREE OVR KEEP) , NONOVR)	Controls allocation of library heap storage not restricted to below the 16M line.	86
OVR NOAUTOTASK (NONOVR) OVR NOAUTOTASK ((loadmod , numtasks) , NONOVR)	Specifies whether Fortran Multitasking Facility is to be used by your program and the number of tasks that are allowed to be active.	88
FREE BELOWHEAP = ((init_size , incr_size , KEEP) , OVR NONOVR)	Controls allocation of library heap storage below the 16M line.	88

Figure 35 (Page 2 of 5). Run-Time Options Quick Reference

Run-Time Options	Function	Page
CBLOPTS = ((ON OFF) , OVR NONOVR)	Specifies the format of the argument string on application invocation when the main program is COBOL.	90
CBLPSHPOP = ((ON OFF) , OVR NONOVR)	Controls whether CICS PUSH HANDLE and CICS POP HANDLE commands are issued when a COBOL subprogram is called.	91
CBLQDA = ((ON OFF) , OVR NONOVR)	Controls COBOL QSAM dynamic allocation.	92
CHECK = ((ON OFF) , OVR NONOVR)	Indicates whether “checking errors” within an application should be detected.	92
COUNTRY = ((country_code) , OVR NONOVR)	Specifies the default formats for date, time, currency symbol, decimal separator, and the thousands separator based on a country.	93
DEBUG = ((ON OFF) , OVR NONOVR)	Activates the COBOL batch debugging features specified by the “debugging lines” or the USE FOR DEBUGGING declarative.	94
DEPTHCONDLMT = ((limit) , OVR NONOVR)	Limits the extent to which conditions can be nested.	95
ENVAR = ((, string) , OVR NONOVR)	Sets the initial values for the environment variables specified in *string*.	97
ERRCOUNT = ((number) , OVR NONOVR)	Specifies how many conditions of severity 2, 3, and 4 can occur per thread before an enclave terminates abnormally.	98
ERRUNIT = ((number) , OVR NONOVR)	Identifies the unit number to which run-time error information is to be directed.	99
FILEHIST = ((ON OFF) , OVR NONOVR)	FILEHIST specifies whether to allow the file definition of a file referred to by a ddname to be changed during run time.	100
HEAP = ((init_size , incr_size , ANYWHERE ANY BELOW , KEEP FREE , initsz24 , incrsz24) , OVR NONOVR)	Controls allocation of the heaps.	101

Figure 35 (Page 3 of 5). Run-Time Options Quick Reference

Run-Time Options	Function	Page
INQPCOPN = ((ON OFF) , OVR NONOVR)	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.	103
INTERRUPT = ((OFF ON) , OVR NONOVR)	Causes attentions recognized by the host operating system to be recognized by Language Environment.	104
LIBSTACK = ((init_size , incr_size , FREE KEEP) , OVR NONOVR)	Controls the allocation of the thread's library stack storage.	105
MSGFILE = ((ddname , recfm , lrecl , blksize) , OVR NONOVR)	Specifies the *ddname* of the run-time diagnostics file.	107
MSGQ = ((number) , OVR NONOVR)	Specifies the number of ISI blocks allocated on a per-thread basis during execution.	110
NATLANG = ((ENU UEN JPN) , OVR NONOVR)	Specifies the national language to use for the run-time environment.	111
NONONIPTSTACK NONIPTSTACK = ((init_size , incr_size , BELOW ANYWHERE ANY , KEEP FREE) , OVR NONOVR)	Controls stack allocation for each thread, except the initial thread, in a multithread environment.	112
OCSTATUS = ((ON OFF) , OVR NONOVR)	Controls whether the OPEN and CLOSE status specifiers are verified.	114
NOPCF PCF	Specifies that Fortran static common blocks are not shared among load modules.	115
PLITASKCOUNT = ((tasks) , OVR NONOVR)	Controls the maximum number of tasks active at one time while you are running PL/I MTF applications.	116
POSIX = ((OFF ON) , OVR NONOVR)	Specifies whether the enclave can run with the POSIX semantics.	116
PRTUNIT = ((number) , OVR NONOVR)	Identifies the unit number used for PRINT and WRITE statements that do not specify a unit number.	118
PUNUNIT = ((number) , OVR NONOVR)	Identifies the unit number used for PUNCH statements that do not specify a unit number.	118

Figure 35 (Page 4 of 5). Run-Time Options Quick Reference

Run-Time Options	Function	Page
RDRUNIT = ((number) , OVR NONOVR)	Identifies the unit number used for READ statements that do not specify a unit number.	119
RECPAD = ((OFF ON NONE ALL VAR) , OVR NONOVR)	Specifies whether a formatted input record is padded with blanks.	119
RPTOPTS = ((OFF ON) , OVR NONOVR)	Specifies that a report of the run-time options in use by the application be generated.	120
RPTSTG = ((OFF ON) , OVR NONOVR)	Specifies that a report of the storage used by the application be generated at the end of execution.	123
RTEREUS = ((OFF ON) , OVR NONOVR)	Initializes the run-time environment to be reusable when the first COBOL program is invoked.	126
SIMVRD = ((OFF ON) , OVR NONOVR)	Specifies whether your COBOL programs use a VSAM KSDS to simulate variable length relative organization data sets.	128
STACK = ((init_size , incr_size , BELOW ANYWHERE ANY , KEEP FREE) , OVR NONOVR)	Controls the allocation and management of thread-level heap storage.	128
STORAGE = ((heap_alloc_value , heap_free_value , dsa_alloc_value , reserve_size) , OVR NONOVR)	Controls the value of storage that is allocated and freed.	131
TERMTHDACT = ((TRACE QUIET MSG DUMP UADUMP) , OVR NONOVR)	Sets the level of information produced due to an unhandled error of severity 2 or greater.	134
THREADHEAP = ((init_size , incr_size , ANYWHERE ANY BELOW , KEEP FREE) , OVR NONOVR)	Controls the allocation and management of thread-level heap storage.	138
NOTEST TEST = ((ALL ERROR NONE , commands_file * , PROMPT NOPROMPT * ; command , preference_file *) , OVR NONOVR)	Specifies that a debug tool is to be given control according to the suboptions specified.	136

Figure 35 (Page 5 of 5). Run-Time Options Quick Reference

```
Run-Time Options                                                   Function                          Page

                  OFF                  DUMP        LE=0           Determines whether Language        140
   TRACE  =  (  (   ON    ,  table_size  ,  NODUMP    ,   LE=1    )   Environment run-time library
                                                   LE=2           tracing is active.
                                                   LE=3
     OVR
  ,  NONOVR   )

                 ON           OVR                                 Specifies how Language             142
   TRAP  =  (  (   OFF    )  ,   NONOVR    )                      Environment routines handle
                                                                  abends and program interrupts.

                                OVR                               Sets the eight UPSI switches on    144
   UPSI  =  (  (  nnnnnnnn  )  ,   NONOVR    )                    or off.  Affects only COBOL
                                                                  programs.

     NOUSRHDLR                       OVR                          USRHDLR registers a user           145
     USRHDLR     =  (  (  lmname  )  ,   NONOVR    )              condition handler at stack frame
                                                                  0.

                   OFF          OVR                               Specifies whether any language     146
   VCTRSAVE  =  (  (   ON     )  ,   NONOVR    )                  in an application uses the vector
                                                                  facility when user-written
                                                                  condition handlers are called.

                 AUTO          OVR                                Specifies whether an exponent      147
   XUFLOW  =  (  (   ON      )  ,   NONOVR    )                   underflow causes a program
                 OFF                                              interrupt.
```

E.2 Language Run-Time Option Mapping

Figure 36 (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.
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.

Figure 36 (Page 2 of 2). C and Language Environment Options

C Option	Language Environment Equivalent	Notes
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 37 (Page 1 of 2). 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.
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.

Figure 37 (Page 2 of 2). COBOL and Language Environment Options

COBOL Option	Language Environment Equivalent	Notes
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 *Language Environment for MVS & VM Programming Guide* and *Language Environment for MVS & VM Installation and Customization on MVS*. 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 *COBOL/370 and COBOL for MVS & VM Compiler and Run-Time Migration Guide*. 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 *COBOL/370 and COBOL for MVS & VM Compiler and Run-Time Migration Guide*.
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.
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 *COBOL/370 and COBOL for MVS & VM Compiler and Run-Time Migration Guide*.

Figure 38 (Page 1 of 2). 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.
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.

Figure 38 (Page 2 of 2). Fortran and Language Environment Options

Fortran Option	**Language Environment Equivalent**	**Notes**
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.
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 39 (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.

Figure 39 (Page 2 of 2). PL/I and Language Environment Options

PL/I Option	Language Environment Equivalent	Notes
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.
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:

```
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)

Syntax

```
                    NONE               OVR
ABPERC  =  (  (     abcode     )  ,    NONOVR     )
```

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:

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

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

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

You can identify only one 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(0C*x*). 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.

E.3.1.2 For More Information

- For more information about the assembler user exit (CEEBXITA), 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

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

Syntax

```
                    OFF          OVR
  AIXBLD  =  (  (   ON     )  ,  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 107 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)

Syntax

```
                     OFF          OVR
    ALL31  =  (  (   ON     ) ,   NONOVR    )
```

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.

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 128 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)

Syntax

```
                                          ANYWHERE      FREE
   ANYHEAP = ( ( init_size , incr_size ,  ANY       ,   KEEP   ) ,
                                          BELOW

    OVR
    NONOVR    )
```

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

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

ANYWHERE|ANY
Specifies that heap storage can be allocated anywhere in storage. On systems that support bimodal addressing, storage can be allocated either above or below the 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 123 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*.

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)

Syntax

```
                      OVR
NOAUTOTASK    (    NONOVR    )
                                                    OVR
NOAUTOTASK    (  (  loadmod  ,  numtasks  )  ,    NONOVR    )
```

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)

Syntax

```
                                                 FREE          OVR
     BELOWHEAP  =  (  (  init_size  ,  incr_size  ,   KEEP    )  ,   NONOVR    )
```

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

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

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

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

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

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 123 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*.

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)

Syntax

```
                        ON           OVR
    CBLOPTS  =  (  (    OFF    ) ,   NONOVR    )
```

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)

Syntax

```
                        ON           OVR
   CBLPSHPOP  =  (  (   OFF    )  ,  NONOVR    )
```

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*.

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

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

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

```
                                   OVR
COUNTRY = ( ( country_code ) ,     NONOVR    )
```

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.

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 149 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)

Syntax

```
                    ON            OVR
  DEBUG  =  (  (    OFF    )  ,   NONOVR    )
```

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 40 on page 96 illustrates the effect of DEPTHCONDLMT(3) on condition handling. The initial condition and two nested conditions are handled in this example. The third nested condition is not handled.

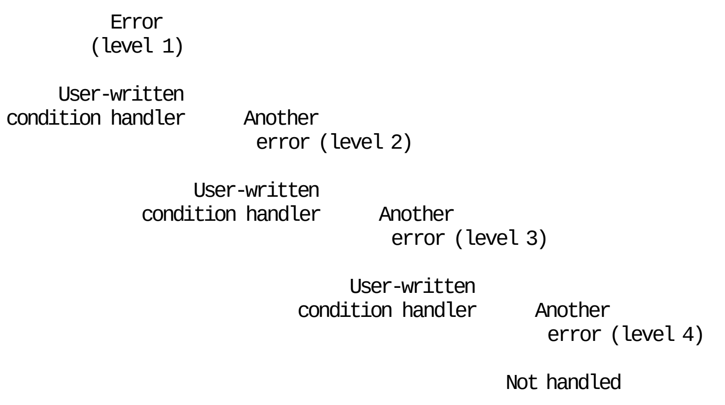


Figure 40. Effect of DEPTHCONDLMT(3) on Condition Handling

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

Syntax

```
                                          OVR
DEPTHCONDLMT  =  (  (  limit  )  ,    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)

Syntax

```
                 ,                 OVR
ENVAR  =  (  (     string     )  ,    NONOVR    )
```

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

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

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

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

OVR
Specifies that the option can be overridden.

NONOVR
Specifies that the option cannot be overridden.

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 120.

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

```
                              OVR
ERRCOUNT = ( ( number ) ,     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

```
                                   OVR
ERRUNIT  =  (  (  number  )  ,     NONOVR     )
```

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

```
                        ON           OVR
    FILEHIST  =  (  (   OFF    )  ,   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)

Syntax

```
                                         ANYWHERE       KEEP
   HEAP  =  (  (  init_size  ,  incr_size  ,    ANY        ,    FREE    ,  initsz24  ,
                                         BELOW

                  OVR
  incrsz24  )  ,    NONOVR    )
```

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

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

ANYWHERE|ANY
Specifies that you can allocate heap storage anywhere in storage. On systems that support bimodal addressing, you can allocate storage either above or below the 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.

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*, *n*K, or *n*M number of bytes. The amount of storage is rounded up to the nearest multiple of 8 bytes.

initsz24 applies to 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*, *n*K, or *n*M number of bytes. The amount of storage is rounded up to the nearest multiple of 8 bytes.

incrsz24 applies to 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 123 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)

Syntax

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

<u>**ON**</u>
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.

<u>**OVR**</u>
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

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

<u>**OFF**</u>
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.

<u>**OVR**</u>
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 136 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 116.

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

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

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

Specify *init_size* as *n*, *n*K, or *n*M bytes of storage. *init_size* can be preceded by a minus sign. 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.

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

If you do not specify *incr_size*, 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 123 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

```
                                                      OVR
MSGFILE = ( ( ddname , recfm , lrecl , blksize ) ,    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.

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* × *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* × *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 120 and E.3.36, “RPTSTG” on page 123.
- 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*.

- 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. CEECMI allocates storage for the ISI, if necessary.

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

Syntax

```
                                OVR
  MSGQ  =  (  (  number  )  ,   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 CEECMI 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)

Syntax

```
                     ENU          OVR
   NATLANG  =  (  (  UEN    )  ,  NONOVR    )
                     JPN
```

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 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 110 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)

Syntax

```
  NONONIPTSTACK                                      BELOW          KEEP
  NONIPTSTACK     = (  (  init_size  ,  incr_size  ,   ANYWHERE    ,    FREE    )
                                                     ANY

    OVR
 ,  NONOVR    )
```

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 specified with NONONIPTSTACK remain in effect. If you respecify NONONIPTSTACK with different suboptions, they override the defaults.
- PL/I MTF consideration—NONIPTSTACK(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 128.
- For more information about the ALL31 run-time option, see E.3.4, “ALL31” on page 85.

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

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

Syntax

```
                OFF          OVR
   PC  =  (  (  ON     )  ,  NONOVR    )
```

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)

Syntax

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

Syntax

```
                  OFF          OVR
  POSIX  =  (  (   ON     )  ,  NONOVR    )
```

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 kernal dependency, it does not take effect.
  - If you invoke a POSIX function that has an OpenEdition kernal 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 kernal 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 104.

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)

Syntax

```
                                        OVR
PRTUNIT  =  (  (  number  )  ,    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

```
                                        OVR
PUNUNIT  =  (  (  number  )  ,    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

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

Syntax

```
                    OFF          OVR
RECPAD  =  (  (     ON      ) ,  NONOVR    )
                    NONE
                    ALL
                    VAR
```

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.

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 41 on page 122 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)

Syntax

```
                  OFF          OVR
RPTOPTS  =  (  (  ON     ) ,   NONOVR    )
```

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.

```
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      CBLPSHPOP(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      NONONIPTSTACK(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†,†INSPPREF†)
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 41. 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 107 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 93 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 42 on page 125 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)

Syntax

```
                    OFF          OVR
RPTSTG  =  (  (     ON     )  ,  NONOVR    )
```

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.

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

```
    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 42 (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 107.
- For more information about the CEE3RPH callable service, see *Language Environment for MVS & VM Programming Reference*.
- See E.3.12, "COUNTRY" on page 93 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 86.
- For more information about the BELOWHEAP run-time option, see E.3.7, "BELOWHEAP" on page 88.
- For more information about the HEAP run-time option, see E.3.19, "HEAP" on page 101.
- For more information about the LIBSTACK run-time option, see E.3.22, "LIBSTACK" on page 105.
- For more information about the STACK run-time option, see E.3.39, "STACK" on page 128.
- 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)

Syntax

```
                    OFF            OVR
  RTEREUS  =  ( (   ON      ) ,    NONOVR     )
```

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).
- 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.

Language Environment also offers preinitialization support in addition to RTEREUS.

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.

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

```
                  OFF         OVR
   SIMVRD  =  (  (   ON     )  ,   NONOVR    )
```

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)

Syntax

```
                                                BELOW          KEEP
  STACK  =  (  (  init_size  ,  incr_size  ,    ANYWHERE   ,   FREE    )  ,
                                                ANY

   OVR
   NONOVR    )
```

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

init_size can be preceded by a minus sign. 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*, *n*K, or *n*M bytes of storage. The actual amount of allocated storage is the larger of two values— *incr_size* or the requested size—rounded up to the nearest multiple of 8 bytes

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

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.

<u>KEEP</u>
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.

<u>OVR</u>
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 112.
- 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 85, for more information about the ALL31 run-time option.
- See E.3.36, "RPTSTG" on page 123, 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)

Syntax

```
 STORAGE = ( ( heap_alloc_value , heap_free_value , dsa_alloc_value ,

                         OVR
reserve_size ) ,         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*, *n*K, or *n*M 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).

  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)

Syntax

```
                      TRACE            OVR
  TERMTHDACT  =  (  (   QUIET      )  ,    NONOVR     )
                      MSG
                      DUMP
                      UADUMP
```

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.

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 140, 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 &dtool. supplied with &lecodel.) 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,INSPPREF),OVR)

Syntax

```
  NOTEST              ALL                                 PROMPT
  TEST     =  (  (    ERROR    ,    commands_file    ,    NOPROMPT    ,
                      NONE          *                     *
                                                          ;
                                                          command

                           OVR
 preference_file    )  ,   NONOVR    )
 *
```

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.

<u>**PROMPT**</u>
Specifies that the debug tool is invoked at Language Environment initialization.

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 INSPPREF.

***** (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 &dtool. 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 *&dtool. Reference Guide* for details and for examples of the TEST run-time option as it relates to &dtool..

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)

Syntax

```
                                              ANYWHERE        KEEP
THREADHEAP  =  (  (  init_size  ,  incr_size  ,  ANY       ,    FREE    )  ,
                                              BELOW

 OVR
 NONOVR    )
```

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.

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 subatasks 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)

Syntax

```
                  OFF                        DUMP        LE=0           OVR
 TRACE  =  (  (   ON     ,  table_size  ,    NODUMP  ,   LE=1    )  ,   NONOVR
                                                         LE=2
                                                         LE=3
 )
```

OFF
Indicates that the tracing facility is inactive.

ON
Indicates that the tracing facility is active.

table_size
Determines the size of the tracing table as specified in bytes (*n*K or *n*M). The upper limit is 16M.

DUMP
Requests that 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).

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 134.
- 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 43. 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.
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)

Syntax

```
                ON          OVR
   TRAP  =  (  (   OFF   ) ,   NONOVR   )
```

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.
- 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 82 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)

Syntax

```
                                OVR
  UPSI  =  (  (  nnnnnnnn  )  ,   NONOVR    )
```

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*.

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)

Syntax

```
NOUSRHDLR                         OVR
USRHDLR     = (  (  lmname  )  ,  NONOVR   )
```

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.

lmname
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 *lmname* 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 *lmname* 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)

Syntax

```
                       OFF          OVR
  VCTRSAVE  =  (  (    ON     )  ,    NONOVR    )
```

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.

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)

Syntax

```
                  AUTO          OVR
  XUFLOW  =  (  (  ON      )  ,  NONOVR    )
                  OFF
```

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 44 (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 44 (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
SK	Slovakia	SL	Sierra Leone

Figure 44 (Page 3 of 3). Country Codes

Code	**Country**	**Code**	**Country**
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 for MVS & VM Release 05.00

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                                    very         not
satisfied  <===================>  dissatisfied  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 for MVS & VM Development group):

IBM Corporation, Department J58
P.O. Box 49023
San Jose, California 95161-9023
USA

Program Number:	5688-198	5890
		5891
		5892
		5894
		6200
		6201
		6202
		6203
		7547

Printed in U.S.A.

XXXX-YYYY-ZZ