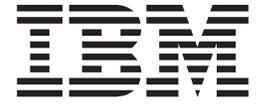
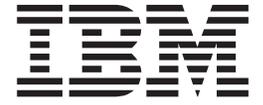


z/OS



Language Environment Customization

z/OS



Language Environment Customization

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 227.

Fourth Edition, September 2002

This is a major revision of SA22-7564-02.

This edition applies to Language Environment[®] in Version 1 Release 4 of z/OS[™] (5694-A01), Version 1 Release 4 of z/OS.e[™] (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

Order documents through your IBM[®] representative or the IBM branch office serving your locality. Documents are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405

FAX (Other Countries):

Your International Access Code +1+845+432-9405

IBMLink[™] (United States customers only): IBMUSM10(MHVRCFS)

Internet e-mail: mhvrdfs@us.ibm.com

World Wide Web: <http://www.ibm.com/servers/eserver/zseries/zos/webqs.html>

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this document
- Page number or topic related to your comment

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

© **Copyright International Business Machines Corporation 1991, 2002. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	vii
Tables	ix
About this document	xi
Using your documentation	xii
How to read syntax diagrams	xiii
Symbols	xiii
Syntax items.	xiii
Syntax examples	xiv
Where to find more information	xv
Accessing z/OS licensed documents on the Internet	xv
Using LookAt to look up message explanations	xvi
Summary of Changes	xvii
Chapter 1. Customization Overview	1
Deciding Whether and What to Customize	1
Chapter 2. Description of Language Environment target libraries	3
History of Changes	5
Chapter 3. Choosing your Language Environment run-time library access	7
Methods of Language Environment data set access.	7
LNKLST	7
STEPLIB	7
Run-time library services (RTLs)	7
Setting up run-time library services (RTLs)	8
Saving your environment.	8
Setting up RTLs in PARMLIB	9
Setting Run-Time Options	11
Controlling Other Data Sets with RTLs	13
Performance Considerations	13
Restrictions	13
Chapter 4. Customizing Language Environment Run-Time Options	19
Setting Default Options with the CEEXOPT Macro	21
Changing Installation-Wide Run-Time Options Defaults (Non-CICS)	25
Changing Installation-Wide Run-Time Options Defaults (CICS)	26
Creating a Region-Specific Run-Time Options Load Module	26
Creating an Application-Specific Run-Time Options Module	27
Chapter 5. Customizing User Exits	29
An Example	30
Changing the Assembler Language User Exit.	30
Changing the Installation-Wide Assembler Language User Exit (Non-CICS)	31
Changing the Installation-Wide Assembler Language User Exit (CICS)	31
Creating an Application-Specific Assembler Language User Exit	32
Changing the High-Level Language User Exit	32
Customizing Language Environment Abnormal Termination Exits	33
Creating a Language Environment Abnormal Termination Exit.	33
CEEEXTAN Abnormal Termination Exit CSECT	33
Identifying the Abnormal Termination Exit (Non-CICS)	35

Identifying the Abnormal Termination Exit (CICS)	35
Creating a Load Notification User Exit	36
Identifying the Load Notification User Exit	36
CEEBLNUE CSECT	36
CEEBLNUE Sample	37
Creating a Storage Tuning User Exit	38
Chapter 6. Customizing the Cataloged Procedures	39
Making the cataloged procedure library available to your jobs.	39
How to Do It.	40
Tailoring the Cataloged Procedures and CLISTs to Your Site	41
Chapter 7. Placing Language Environment Modules in Link Pack and LIBPACK	43
Tailoring the Fortran LIBPACKs	44
Choices to Make Now	44
Some Examples	45
Listing the contents of Fortran LIBPACKs	45
Deleting Routines from Fortran LIBPACKs	45
Adding Routines to Fortran LIBPACKs	46
Where to Place the Tailored Fortran LIBPACKs	48
Chapter 8. Using Language Environment under CICS	49
Add program resource definitions for CICS	49
Add destination control table (DCT) entries	50
Add Language Environment-CICS Data Sets to the CICS Startup Job Stream	53
Language Environment automatic storage tuning for CICS	54
Enclaves eligible for automatic storage tuning	54
Automatic storage tuning behavior.	55
Altering the automatic storage tuning behavior	56
Chapter 9. Using Language Environment under IMS	57
Initializing Library Routine Retention	57
Terminating Library Routine Retention	57
Chapter 10. Customizing Language-Specific Features	59
Choices to Make Now	59
Modifying the OS/VS COBOL compatability library routines	59
OS/VS COBOL Considerations	60
Modifying the COBOL Parameter List Exit	61
Modifying the COBOL Reusable Environment	62
Changing the C/C++ locale time information	63
Steps for modifying the JCL for EDCLLOCL	64
Chapter 11. Customizing for Fortran applications	65
Tailoring the Language Environment Fortran Unit Attribute Table.	65
Tailoring the VS FORTRAN Compatibility Unit Attribute Table	65
Tailoring VS FORTRAN Compatibility Run-Time Options.	65
Tailoring the VS FORTRAN Compatibility Error Option Table	66
Chapter 12. Language Environment Run-Time Options	67
COBOL Compatibility	67
Run-time options	67
ABPERC	68
ABTERMENC	69
AIXBLD (COBOL Only)	71

ALL31	72
ANYHEAP	73
AUTOTASK NOAUTOTASK (Fortran Only)	75
BELOWHEAP	76
CBLOPTS (COBOL Only)	77
CBLPSHPOP (COBOL Only)	78
CBLQDA (COBOL Only)	79
CHECK (COBOL Only)	80
COUNTRY	80
DEBUG (COBOL Only)	81
DEPTHCONDLMT	82
ENVAR	84
ERRCOUNT	85
ERRUNIT (Fortran Only)	86
FILEHIST (Fortran Only)	87
FILETAG (C/C++ only)	88
HEAP	90
HEAPCHK	92
HEAPOOLS (C/C++ only)	94
INFMSGFILTER	95
INQPCOPN (Fortran Only)	97
INTERRUPT	97
LIBRARY	98
LIBSTACK	99
MSGFILE	101
MSGQ	104
NATLANG	105
OCSTATUS (Fortran Only)	107
PC (Fortran Only)	108
PLITASKCOUNT (PL/I Only)	109
POSIX	109
PROFILE	111
PRTUNIT (Fortran Only)	111
PUNUNIT (Fortran Only)	112
RDRUNIT (Fortran Only)	113
RECPAD (Fortran Only)	113
RPTOPTS	114
RPTSTG	117
RTEREUS (COBOL Only)	127
RTLS	128
SIMVRD (COBOL Only)	129
STACK	130
STORAGE	133
TERMTHDACT	136
TEST NOTEST	142
THREADHEAP	145
THREADSTACK	146
TRACE	149
TRAP	151
UPSI (COBOL Only)	153
USRHDLR	154
VCTRSAVE	155
VERSION	156
XUFLOW	157

Appendix A. Customizing Language Environment run-time options using z/OS msys for Setup	159
Who should use msys for Setup?	159
What is the Language Environment customization task?	159
Recommendations when using msys for Setup for Language Environment customization	160
Restrictions when using msys for Setup for Language Environment customization	160
Where to find information about msys for Setup	161
Appendix B. Using Fortran with Language Environment	163
Customizing for Fortran applications link-edited with Language Environment	163
Changing the unit attribute table default values	163
Customizing for Fortran Applications Link-Edited with VS FORTRAN.	169
Changing the Unit Attribute Table Default Values	170
Changing VS FORTRAN Run-Time Option Defaults	175
Changing the Error Option Table Defaults	180
Customizing Fortran LIBPACKs	184
Contents of the Fortran LIBPACK AFHPRNAG	185
Contents of the Fortran LIBPACK AFHPRNBG	189
Contents of the Fortran LIBPACK AFH5RENA	190
Contents of the Fortran LIBPACK AFH5RENB	192
Appendix C. Using IBM C/C++ with Language Environment	195
Planning to Customize Locale Time Information	195
Customizing the Locale Time Information.	195
Time Information Options Reference	196
System Programming Facilities	197
Appendix D. Modules eligible for the link pack area	199
Language Environment base modules	199
Language Environment C/C++ component modules	200
Language Environment COBOL component modules	201
Language Environment Fortran component modules	202
Language Environment PL/I Component Modules	216
Appendix E. Modifying the JCL for Japanese National Language Support	221
Appendix F. Language Environment National Language Support Country Codes.	223
Appendix G. Accessibility	225
Using assistive technologies	225
Keyboard navigation of the user interface.	225
Notices	227
Programming Interface Information	229
Trademarks.	229
Bibliography	231
Language Products Publications	231
Related Publications	232
Softcopy Publications	233
Index	235

Figures

1. An Example of a CSVRTLS PARMLIB Member for OS/390 Version 2 Release 4	10
2. Sample Invocation of CEEXOPT within the CEEDOPT Member	22
3. Sample Invocation of CEEXOPT within the CEECOPT Member	23
4. Default CEEEXTAN	35
5. Updated CEEEXTAN	35
6. Sample of CEEBLNUE Load Notification User Exit CSECT	37
7. Format of an Output Transient Data Queue	51
8. Example of DFHDCT Macro.	53
9. Effect of DEPTHCONDLMT(3) on Condition Handling	83
10. Options Report Example Produced by Run-Time Option RPTOPTS(ON)	116
11. Storage Report Produced by Run-Time Option RPTSTG(ON)	119
12. Storage Report Produced by RPTSTG(ON) with XPLINK	123
13. IBM-Supplied Macro Instructions	168
14. Modified IBM-Supplied Macro Instructions	169
15. IBM-Supplied Macro Instructions	174
16. Modified IBM-Supplied Macro Instructions	175
17. Modified IBM-Supplied Macro Instructions	175
18. Example of Time Zone and Daylight Savings Time Information in Module EDCLOCTZ	196

Tables

1.	How to Use z/OS Language Environment Publications	xii
2.	Syntax examples.	xiv
3.	Description of data set target libraries for Language Environment	3
4.	Versions of SYSCEE Logical Library with z/OS	9
5.	RTLS Restrictions	14
6.	Worksheet: Planning to Customize Language Environment Run-Time Options	19
7.	Sample jobs to change run-time options defaults	21
8.	Sample Customization Jobs for the User Exits	29
9.	Sample Assembler User Exits for Language Environment	30
10.	Language Environment Invocation Procedures in CEE.SCEEPROC	39
11.	Deciding How to Make Cataloged Procedures Available to Your Jobs	40
12.	Cataloged Procedures and CLISTs Information	42
13.	Language Environment Sample IEALPAnn or PROGxx Members in CEE.SCEESAMP	43
14.	Making the trade-off: Performance time versus storage use	44
15.	SMP/E Sample Jobs for Deleting Routines from Fortran LIBPACKs	46
16.	SMP/E Sample Jobs for Adding Routines to Fortran LIBPACKs.	47
17.	Excluding Programming Language Support under CICS	50
18.	Customizing Programming Languages with Sample Customization Jobs	59
19.	Using the USERMODs in the IGZWZAP Job to Modify the COBOL Compatibility Library	59
20.	Condition Handling of 0Cx ABENDS	140
21.	Handling of software raised conditions	140
22.	TRAP Run-Time Option Settings	151
23.	Fortran LIBPACKs	184
24.	Routines Eligible for Inclusion in the Fortran LIBPACK AFHPRNAG	185
25.	Routines Eligible for Inclusion in the Fortran LIBPACK AFHPRNBG	189
26.	Routines Eligible for Inclusion in the Fortran LIBPACK AFH5RENA	190
27.	Routines Eligible for Inclusion in the Fortran LIBPACK AFH5RENB	192
28.	Language Environment Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area	199
29.	C/C++ Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area	200
30.	COBOL Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area	201
31.	Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area	202
32.	PL/I Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area	216
33.	Japanese National Language Support (NLS) JCL Modifications	221
34.	Country Codes	223

About this document

This document supports z/OS (5694–A01) and z/OS.e (5655–G52).

IBM z/OS Language Environment (also called Language Environment) provides common services and language-specific routines in a single run-time environment for C, C++, COBOL, Fortran (z/OS only; no support for z/OS UNIX System Services or CICS®), PL/I, and assembler applications. It offers consistent and predictable results for language applications, independent of the language in which they are written.

Language Environment is the prerequisite run-time environment for applications generated with the following IBM compiler products:

- z/OS C/C++
- OS/390® C/C++
- C/C++ Compiler for MVS/ESA™
- AD/Cycle® C/370™ Compiler
- VisualAge for Java, Enterprise Edition for OS/390
- Enterprise COBOL for z/OS and OS/390
- COBOL for OS/390 & VM
- COBOL for MVS & VM (formerly COBOL/370)
- Enterprise PL/I for z/OS and OS/390
- VisualAge PL/I for OS/390
- PL/I for MVS & VM (formerly PL/I MVS™ & VM)
- VS FORTRAN and FORTRAN IV (in compatibility mode)

Restrictions: The following restrictions apply to z/OS.e:

- The following compilers are not licensed for use on z/OS.e:
 - COBOL
 - PL/I
 - FORTRAN
- The following subsystems are not licensed for use on z/OS.e:
 - CICS
 - IMS™
- Execution of applications written in the following languages is not functionally supported on z/OS.e:
 - COBOL (except for precompiled COBOL DB2® stored procedures and other precompiled COBOL applications using the Language Environment preinitialization interface)
 - FORTRAN
- The following are not functional and/or not licensed for use on z/OS.e:
 - Language Environment Library Routine Retention (LRR)
 - Language Environment compatibility preinitialization for C and PL/I
 - Run-time library services (RTLS)
- Customers are not permitted to use lower levels of Language Environment on z/OS.e.

Language Environment supports, but is not required for, an interactive debug tool for debugging applications in your native z/OS environment. The IBM interactive Debug Tool is available with z/OS, or with the latest releases of the C/C++, COBOL, PL/I and VisualAge for Java compiler products.

Language Environment supports, but is not required for, VS Fortran Version 2 compiled code (z/OS only).

Language Environment consists of the common execution library (CEL) and the run-time libraries for C/C++, COBOL, Fortran, and PL/I.

For more information on VisualAge for Java, Enterprise Edition for OS/390, program number 5655-JAV, see the product documentation.

Using your documentation

The publications provided with Language Environment are designed to help you:

- Manage the run-time environment for applications generated with a Language Environment-conforming compiler.
- Write applications that use the Language Environment callable services.
- Develop interlanguage communication applications.
- Customize Language Environment.
- Debug problems in applications that run with Language Environment.
- Migrate your high-level language applications to Language Environment.

Language programming information is provided in the supported high-level language programming manuals, which provide language definition, library function syntax and semantics, and programming guidance information.

Each publication helps you perform different tasks, some of which are listed in Table 1. All books are available in printed and softcopy formats. For a complete list of publications that you may need, see “Bibliography” on page 231.

Table 1. How to Use z/OS Language Environment Publications

To ...	Use ...
Evaluate Language Environment	<i>z/OS Language Environment Concepts Guide</i>
Plan for Language Environment	<i>z/OS Language Environment Concepts Guide</i> <i>z/OS Language Environment Run-Time Migration Guide</i>
Install Language Environment	<i>z/OS Program Directory</i>
Customize Language Environment	<i>z/OS Language Environment Customization</i>
Understand Language Environment program models and concepts	<i>z/OS Language Environment Concepts Guide</i> <i>z/OS Language Environment Programming Guide</i>
Find syntax for Language Environment run-time options and callable services	<i>z/OS Language Environment Programming Reference</i>
Develop applications that run with Language Environment	<i>z/OS Language Environment Programming Guide</i> and your language programming guide
Debug applications that run with Language Environment, diagnose problems with Language Environment	<i>z/OS Language Environment Debugging Guide</i>
Get details on run-time messages	<i>z/OS Language Environment Run-Time Messages</i>
Develop interlanguage communication (ILC) applications	<i>z/OS Language Environment Writing Interlanguage Communication Applications</i> and your language programming guide

Table 1. How to Use z/OS Language Environment Publications (continued)

To ...	Use ...
Migrate applications to Language Environment	<i>z/OS Language Environment Run-Time Migration Guide</i> and the migration guide for each Language Environment-enabled language

This book is intended for system programmers and system administrators who plan to customize Language Environment on the z/OS platform.

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

The first usage of every term listed in the glossary is indicated by *italics*. You can find the definitions for these terms in *z/OS Language Environment Concepts Guide*.

How to read syntax diagrams

This section describes how to read syntax diagrams. It defines syntax diagram symbols, items that may be contained within the diagrams (keywords, variables, delimiters, operators, fragment references, operands) and provides syntax examples that contain these items.

Syntax diagrams pictorially display the order and parts (options and arguments) that comprise a command statement. They are read from left to right and from top to bottom, following the main path of the horizontal line.

Symbols

The following symbols may be displayed in syntax diagrams:

Symbol	Definition
▶—	Indicates the beginning of the syntax diagram.
—▶	Indicates that the syntax diagram is continued to the next line.
▶—	Indicates that the syntax is continued from the previous line.
—▶◀	Indicates the end of the syntax diagram.

Syntax items

Syntax diagrams contain many different items. Syntax items include:

- Keywords - a command name or any other literal information.
- Variables - variables are italicized, appear in lowercase and represent the name of values you can supply.
- Delimiters - delimiters indicate the start or end of keywords, variables, or operators. For example, a left parenthesis is a delimiter.
- Operators - operators include add (+), subtract (-), multiply (*), divide (/), equal (=), and other mathematical operations that may need to be performed.
- Fragment references - a part of a syntax diagram, separated from the diagram to show greater detail.
- Separators - a separator separates keywords, variables or operators. For example, a comma (,) is a separator.

Keywords, variables, and operators may be displayed as required, optional, or default. Fragments, separators, and delimiters may be displayed as required or optional.

Item type	Definition
Required	Required items are displayed on the main path of the horizontal line.
Optional	Optional items are displayed below the main path of the horizontal line.
Default	Default items are displayed above the main path of the horizontal line.

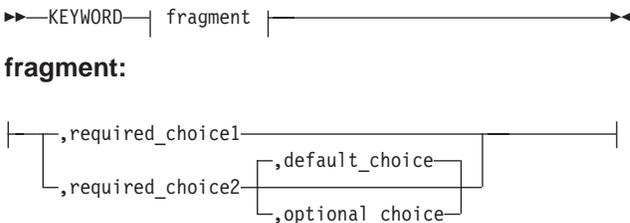
Syntax examples

The following table provides syntax examples.

Table 2. Syntax examples

Item	Syntax example
Required item.	
Required items appear on the main path of the horizontal line. You must specify these items.	<p>Diagram: A horizontal line with a double arrowhead on the left and a double arrowhead on the right. The text 'KEYWORD' is positioned at the start of the line, and 'required_item' is positioned further along the line.</p>
Required choice.	
A required choice (two or more items) appears in a vertical stack on the main path of the horizontal line. You must choose one of the items in the stack.	<p>Diagram: A horizontal line with a double arrowhead on the left and a double arrowhead on the right. The text 'KEYWORD' is positioned at the start of the line. Below the line, a vertical stack of two items is shown: 'required_choice1' on top and 'required_choice2' on the bottom, connected by a bracket on the right side.</p>
Optional item.	
Optional items appear below the main path of the horizontal line.	<p>Diagram: A horizontal line with a double arrowhead on the left and a double arrowhead on the right. The text 'KEYWORD' is positioned at the start of the line. Below the line, the text 'optional_item' is positioned.</p>
Optional choice.	
An optional choice (two or more items) appears in a vertical stack below the main path of the horizontal line. You may choose one of the items in the stack.	<p>Diagram: A horizontal line with a double arrowhead on the left and a double arrowhead on the right. The text 'KEYWORD' is positioned at the start of the line. Below the line, a vertical stack of two items is shown: 'optional_choice1' on top and 'optional_choice2' on the bottom, connected by a bracket on the right side.</p>
Default.	
Default items appear above the main path of the horizontal line. The remaining items (required or optional) appear on (required) or below (optional) the main path of the horizontal line. The following example displays a default with optional items.	<p>Diagram: A horizontal line with a double arrowhead on the left and a double arrowhead on the right. The text 'KEYWORD' is positioned at the start of the line. Above the line, the text 'default_choice1' is positioned. Below the line, a vertical stack of two items is shown: 'optional_choice2' on top and 'optional_choice3' on the bottom, connected by a bracket on the right side.</p>

Table 2. Syntax examples (continued)

Item	Syntax example
Variable.	
Variables appear in lowercase italics. They represent names or values.	
Repeatable item.	
An arrow returning to the left above the main path of the horizontal line indicates an item that can be repeated.	
A character within the arrow means you must separate repeated items with that character.	
An arrow returning to the left above a group of repeatable items indicates that one of the items can be selected, or a single item can be repeated.	
Fragment.	
The <code> fragment </code> symbol indicates that a labelled group is described below the main syntax diagram. Syntax is occasionally broken into fragments if the inclusion of the fragment would overly complicate the main syntax diagram.	 <p data-bbox="829 779 954 804">fragment:</p>

Where to find more information

Please see *z/OS Information Roadmap* for an overview of the documentation associated with z/OS, including the documentation available for z/OS Language Environment.

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

<http://www.ibm.com/servers/resourceLink>

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (G110-0671), that includes this key code. ¹

To obtain your IBM Resource Link user ID and password, log on to:

<http://www.ibm.com/servers/resourceLink>

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

1. z/OS.e customers received a Memo to Licensees, (G110-0684) that includes this key code.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for most messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

<http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/>

or from anywhere in z/OS where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX System Services running OMVS). You can also download code from the *z/OS Collection* (SK3T-4269) and the LookAt Web site that will allow you to access LookAt from a handheld computer (Palm Pilot VIIx suggested).

To use LookAt as a TSO/E command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO/E from a disk on your *z/OS Collection* (SK3T-4269) or from the **News** section on the LookAt Web site.

Some messages have information in more than one document. For those messages, LookAt displays a list of documents in which the message appears.

Summary of Changes

Summary of Changes for SA22-7564-03 z/OS Version 1 Release 4

This book contains information previously presented in SA22-7564-02, which supports z/OS Version 1 Release 3.

New Information

- Information has been added to indicate that this book supports z/OS.e.

Changed Information

- Invocation procedures in “Making the cataloged procedure library available to your jobs” on page 39 have been updated. CEEXR and CEEXLR replace CEEXG and CEEXLG.
- Chapter 8, “Using Language Environment under CICS” on page 49 includes updated program resource definition information, including new examples.
- The options report in “RPTOPTS” on page 114 has been updated.
- The storage reports in “RPTSTG” on page 117 have been updated.

Moved Information

- Chapter 12, “Language Environment Run-Time Options” on page 67 has been moved from the appendix to the body of the book.
- The NONIPTSTACK | NONONIPTSTACK run-time option has been removed because it is no longer supported. It was replaced by the THREADSTACK option.

This book contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Starting with z/OS V1R2, you may notice changes in the style and structure of some content in this book, for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our books.

Summary of Changes for SA22-7564-02 z/OS Version 1 Release 3

This book contains information previously presented in SA22-7564-01, which supports z/OS Version 1 Release 2.

New Information

- A new appendix on Managed System Infrastructure for Setup (msys for Setup) has been added.
- An appendix with z/OS product accessibility information has been added.

Changed Information

- Information on TERMTHDACT was updated. See “TERMTHDACT” on page 136.

- The appendix on Language Environment User Exits was moved to *z/OS Language Environment Programming Guide*.

**Summary of Changes
for SA22-7564-01
z/OS Version 1 Release 2**

This book contains information previously presented in SA22-7564-00, which supports z/OS Version 1 Release 1.

New Information

- Added a new section about automatic storage tuning for CICS. This section was moved to *z/OS Language Environment Programming Guide* in V1R3.
- Added a new chapter, Chapter 9, “Using Language Environment under IMS” on page 57

Changed Information

- The following run-time options or defaults were changed:
 - ALL31
 - FILETAG
 - HEAPCHK
 - STACK
 - TERMTHDACT
 - THREADSTACK

See Figure 2 on page 22 or Figure 3 on page 23.

Deleted Information

- The COBOL COBPACKs can no longer be modified. Information related to modifying the COBOL COBPACKs has been removed. For example, the appendix “Using COBOL with Language Environment” has been deleted, and several pages pertaining to COBPACK information have been deleted from Chapter 7.

This book contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Chapter 1. Customization Overview

You can customize Language Environment by either tailoring and installing IBM-supplied usermods, or by tailoring and running specific jobs.

To tailor and install usermods:

1. Get the list of usermods that suit the programmer needs at your site. This chapter will help you create this list.
2. Copy the customization jobs from the SCEESAMP data set into one of your private data sets so you will have unmodified copies of the jobs for your later reference and use.
3. Apply the usermods to the target libraries, **but do not accept them, and do not modify the distribution libraries.**
4. Use SMP/E RESTORE to remove a usermod if necessary (for example, if programming needs at your site change) or before you apply service to the modules it changes.
5. Reapply the usermod after successful installation of service.

To modify the JCL for customization jobs:

1. Copy the customization jobs from the SCEESAMP data set into one of your private data sets so you will have unmodified copies of the jobs for your later reference and use.
2. Add a job card appropriate for your site.
3. Add a JES Route card if your site requires one.
4. Modify the job according to the comments in the JCL or the instructions in this book.
5. Save and submit the job.
6. Most jobs will run with a condition code of 0. Check the description of each job to find out what condition code to expect. If the job did not run with the condition code you expected:
 - Check for an error message on the system console or the list output to find the cause of the problem.
 - Correct the problem.
 - Rerun the job.
 - Recheck the condition code.

Deciding Whether and What to Customize

You should consider whether the IBM-supplied values for the run-time options provided with Language Environment suit the needs of your site. These values control such features as:

- The national language in which messages appear
- How a debug tool is invoked
- When condition handling is invoked
- How storage is allocated to the heap and stack
- How much storage is allocated above and below the 16 MB line
- The format of the program invocation character parameter
- Creation of a storage and/or run-time options report
- Shared storage allocations
- When Run-Time Library Services (RTLS) is used

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

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

You need to make decisions about customizing:

- Run-time library access method (see page 7)
- Run-time options (see page 19)
- Assembler user exits (see page 29)
- Cataloged procedures (see page 39)

You also need to decide:

- Whether to install some routines in the link pack area (see page 43)
- Whether to make Language Environment available under CICS (see page 49)
- Whether to customize any programming language-specific features (see page 55)

Chapter 2. Description of Language Environment target libraries

Table 3 provides a description of the Language Environment target libraries and when they are used. In most cases, the DDDEF entry for the data set is the same as the low-level qualifier. For the cases where this is not true, the appropriate DDDEF entry is listed. The high-level qualifier of these data sets may differ from customer to customer, but the default value is CEE.

The data sets in Table 3 have a legend associated with them in the rightmost columns of the table. The following descriptions explain the use and placing of these data sets.

- **AD** (Application Development) — These data sets are used during the assembly, compilation or link-edit phases of application development. This does not include the procedures and CLISTs that can be used by application developers.
- **Ex** (Execution) — These data sets are used during execution of an application and must be placed in the program search order or be accessed directly through DD statements.
- **O** (Other) — These data sets contain sample jobs, source code, procedures or CLISTs that are not used when assembling, compiling, link-editing or executing programs.

Table 3. Description of data set target libraries for Language Environment

DDDEF Entry	Data Set Name	Description	AD	Ex	O
	SAFHFORT	The Fortran-specific link-edit library used to resolve certain Fortran intrinsic function names. In link-edit steps, this library must precede SCEELKED if Fortran functions are needed.	X		
	SCEEBIND	Contains all Language Environment resident routines for XPLINK applications. This one library replaces the four libraries of resident routines for non-XPLINK applications (SCEELKED, SCEELKEX, SCEEOBJ and SCEEPP). It must be used only when link-editing a program which includes XPLINK-compiled object modules. This data set will be eliminated in the near future and is being replaced with SCEEBND2. Customers should use the SCEEBND2 data set instead of SCEEBIND during XPLINK application development.	X		
	SCEEBND2	Contains all Language Environment resident routines for XPLINK applications. This one library replaces the four libraries of resident routines for non-XPLINK applications (SCEELKED, SCEELKEX, SCEEOBJ and SCEEPP). It must be used only when link-editing a program which includes XPLINK-compiled object modules. The only difference between this data set and SCEEBIND is the record format. SCEEBND2 has a fixed blocked record format.	X		
	SCEECICS	Contains the COBOL-specific CICS run-time modules. Will only be used in the DHFRPL DD concatenation.		X	
	SCEECLST	Provides TSO/E CLISTs that C/C++ application developers can use.			X
	SCEECMAP	Contains the source for charmap files.			X

Table 3. Description of data set target libraries for Language Environment (continued)

DDDEF Entry	Data Set Name	Description	AD	Ex	O
	SCEECPP	Contains Language Environment resident definitions that non-XPLINK C++ programs might need. This data set must be used whenever link-editing a non-XPLINK program which includes any C++ object.	X		
	SCEEGXLT	Contains the GENXLT source for the code set converters.			X
SCEEH	SCEEH	Contains ANSI C++ language headers used when compiling C++ programs.	X		
SCEEHARP	SCEEH.ARPA.H	Contains C-language headers used when compiling C programs.	X		
SCEEHH	SCEEH.H	Contains C-language headers used when compiling C programs.	X		
SCEEHNET	SCEEH.NET.H	Contains C-language headers used when compiling C programs.	X		
SCEEHNEI	SCEEH.NETINET.H	Contains C-language headers used when compiling C programs.	X		
SCEEHSYS	SCEEH.SYS.H	Contains C-language headers used when compiling C programs.	X		
SCEEHT	SCEEH.T	Contains ANSI C++ template files used when compiling C++ programs.	X		
	SCEELIB	Contains side-decks for DLLs provided by Language Environment. Many of the language-specific callable services available to XPLINK-compiled applications appear externally as DLL functions. To resolve these references from XPLINK applications, definition side-decks are required.	X		
	SCEELKED	Contains the link-edit stubs for non-XPLINK C/C++, PL/I, COBOL and Fortran languages and Language Environment-provided routines.	X		
	SCEELKEX	Contains non-XPLINK C/C++ stubs that are not uppercased, truncated or mapped to another symbol. In link-edit steps this library must precede SCEELKED if unmapped names are used.	X		
	SCEELOCL	Provides the locale source files (pre-XPG4).			X
	SCEELOCX	Provides the locale source files as defined by the XPG4 standard.			X
	SCEELPA	Contains a subset of the SCEERUN modules that are reentrant and reside above the 16 MB line. This data set should be added to LPALSTxx for performance benefits.		X	
	SCEEMAC	Provides Assembler macros to be used when writing assembler language code and using Language Environment services.	X		
	SCEEMSGP	Contains the message file to be used by the C pre-linker.	X		
	SCEEOBJ	Contains Language Environment resident definitions which may be required for non-XPLINK OS/390 UNIX System Services programs. This data set must be used whenever link-editing a non-XPLINK UNIX program.	X		
	SCEEPROC	Provides procedures used to link-edit and run Language Environment-conforming applications.			X

Table 3. Description of data set target libraries for Language Environment (continued)

DDDEF Entry	Data Set Name	Description	AD	Ex	O
	SCEERTLS	Provides run-time routines needed when using the RTLS (Run-Time Library Services) function.		X	
	SCEERUN	Contains the run-time library routines needed during execution of applications written in C/C++, PL/I, COBOL and FORTRAN.		X	
	SCEERUN2	Contains the run-time library routines needed during execution of applications, and those that require to reside in a PDSE.		X	
	SCEESAMP	Provides sample jobs, usermods, parmlib samples, some C headers and some assembler macros.	X		
	SCEESPC	Provides the System Programmer C (SPC) routines to build free standing C applications. In link-edit steps, this library must precede SCEELKED in the SYSLIB DD concatenation.	X		
	SCEESPCO	Provides the object decks for the SPC routines for the SCEESPC data set.			X
	SCEEUMAP	Provides the Universal 2-octet Coded Character Set (UCS-2) converter source.			X
	SCEEUTBL	Provides the Universal 2-octet Coded Character Set (UCS-2) converter binaries.		X	
	SIBMCALL	Provides the support for OS PL/I PLICALLA and PLICALLB entry points. In link-edit steps, this library must precede SCEELKED if PL/I for MVS and VM applications use OS PL/I PLICALLA or PLICALLB as entry points.	X		
	SIBMCAL2	Provides the support for OS PL/I PLICALLA and PLICALLB entry points. In link-edit steps, this library must precede SCEELKED if VisualAge PL/I for OS/390 applications use OS PL/I PLICALLA or PLICALLB as entry points.	X		
	SIBMMATH	Contains the stubs for old PL/I Version 2 Release 3 math library routines. In link-edit steps, this library must precede SCEELKED if PL/I for MVS and VM applications use OS PL/I PLICALLA or PLICALLB as entry points.	X		
	SIBMTASK	Provides the PL/I Multi-Tasking Facility. In link-edit steps, this library must precede SCEELKED if PL/I MTF is to be used.	X		

History of Changes

Some target libraries have been added, deleted or renamed in the following OS/390 and z/OS releases:

OS/390 Version 2 Release 6 added one data set and renamed two others:
 SCEELPA was added
 SCEEUCS2.UCMAP was renamed to SCEEUMAP
 SCEEUCS2.UCONVTBL was renamed to SCEEUTBL

OS/390 Version 2 Release 9 added one new data set:
 SIBMCAL2

OS/390 Version 2 Release 10 added three new data sets:
SCEERUN2
SCEEBIND
SCEELIB

z/OS Version 1 Release 2 added three new data sets:
SCEEBND2
SCEEH
SCEEH.T

Chapter 3. Choosing your Language Environment run-time library access

Applications that require the run-time library provided by Language Environment, can access the SCEERUN and SCEERUN2 data sets using

- LNKLST
- STEPLIB

or any combination of the above two methods.

The Language Environment run-time library can also be accessed using a third method, Run-time Library Services (RTLS). The RTLS function allows you to access different levels of the Language Environment run-time libraries, controlled by run-time options. These run-time options allow you to control an application or your entire application environment. In addition, run-time library access using STEPLIB can still be used.

Methods of Language Environment data set access

LNKLST

The Language Environment run-time libraries, SCEERUN and SCEERUN2, can be placed in LNKLST. In addition, heavily-used modules can be placed in LPA. For further information see Appendix D, “Modules eligible for the link pack area” on page 199.

STEPLIB

If the SCEERUN and SCEERUN2 data sets cannot be placed in LNKLST, you can STEPLIB the data sets for each application that requires them. One reason why the Language Environment run-time libraries are not to be placed in LNKLST might be that the pre-Language Environment runtime libraries (VS COBOL II, OS PL/I) are placed in LNKLST and your site has not completed the migration to Language Environment. See *z/OS Language Environment Run-Time Migration Guide* for details.

Applications that currently STEPLIB to the SCEERUN data set to gain access to the run-time library provided by Language Environment, do not need to add the SCEERUN2 data set as part of their STEPLIB concatenation. In fact, since SCEERUN2 contains module names that do not intersect with any pre-Language Environment run-time library or any existing library, IBM recommends that SCEERUN2 be added to the LNKLST. This will not result in any adverse effects.

Run-time library services (RTLS)

Run-Time Library Services (RTLS) allows users to eliminate the use of STEPLIBs in order to tie the application with a certain level of the run-time libraries. RTLS-specific run-time options and the use of z/OS system services allow users to tag applications with a specific Language Environment run-time library level.

Setting up run-time library services (RTLS)

If RTLS is the method that your site has chosen to access the Language Environment run-time libraries, you must ensure that the Language Environment run-time library SCEERUN data set is not part of the LNKLST or LPALST concatenation.

If you plan on using RTLS to access multiple levels of the Language Environment run-time library you should maintain the older levels of Language Environment in separate SMP/E zones.

For a list of restrictions that apply when using RTLS, see “Restrictions” on page 13.

Saving your environment

The following procedures should be used to separate Language Environment from the SMP/E environment only when Language Environment is not already in a separate zone:

1. SMP/E ACCEPT or RESTORE all service and USERMODs for Language Environment.
 2. Use the SMP/E BUILD MCS command to create an SMP/E installable image of your existing Language Environment.

The BUILD MCS command builds an SMP/E installable image from information in the target and distribution zones and the code in the distribution libraries. Therefore all code must be accepted or restored (and then re-applied).
 3. Define new SMP/E zones for Language Environment.
 4. Allocate separate target and distribution libraries for Language Environment.
 - A sample allocation job called CEEWALOC, found in the SCEESAMP data set, can be used to allocate the new libraries. Consider using meaningful and identifiable data set names.

Recommendation: You should not use the CEEISALC job to allocate the new libraries. CEEISALC is intended to be used during the initial install of the z/OS CBPDO (Custom-Built Product Delivery Option) software delivery package and not to save levels of Language Environment.
 - Define SMP/E DDDEFs for the new target and distribution libraries for Language Environment.
- A sample DDDEF job called CEEWDDDF, found in the SCEESAMP data set, can be used to define the new libraries.

Recommendation: You should not use the CEEISDDD job to define the new libraries. CEEISDDD is intended to be used during the initial install of the z/OS CBPDO (Custom-Built Product Delivery Option) software delivery package and not to save levels of Language Environment.

SMP/E RECEIVE, APPLY, and ACCEPT the SMP/E installable image built by the BUILD MCS command into the new separate zones.
 - You do not need to install the Language Environment HFS FMID if you don't plan on doing any C/C++ application development in the z/OS UNIX environment. However, if you do plan on using the C headers at your existing level, then you will need to create the target libraries. The target libraries in this case will be HFS directories. A sample job called CEEWMKD is provided in the SCEESAMP data set to create the necessary directories. Before you run this job, remember to add a high-level directory to every PATH to prevent installing an older level of the headers under /usr/include.

- Define the SYSLIB macro concatenation for new zones.
Ensure that the Language Environment MACLIB or SCEEMAC data set is included in the SYSLIB macro concatenation.
- If you chose to do the RESTORE in procedure 1 on page 8, re-APPLY any service or USERMODs that were RESTOREd.

Setting up RTLS in PARMLIB

This section contains an example of a CSVRTLxx member of PARMLIB. It defines four versions of the SYSCEE logical library, with OS/390 Version 2 Release 4 Language Environment the default version. See Table 4.

Table 4. Versions of SYSCEE Logical Library with z/OS

LIBRARY	VERSION	SCEERUN PDS (CATALOGUED)
SYSCEE	V1R5M0	CEE.V1R5M0.SCEERUN
SYSCEE	R2	CEE.OS390R2.SCEERUN
OS390R3	R3	CEE.OS390R3.SCEERUN
LATEST	0	CEE.SCEERUN (DEFAULT)

Each physical library is allowed to consume up to 8 MB above the line and 160 KB below the line for cached Language Environment modules. The combined maximum allowed is 20 MB above the line and 400 KB below.

```

MAXBELOW (400K)
MAXABOVE (20M)
PHYSICAL (
    LIBRARY(CEEL150) ADD
    DSLIST(CEE.V1R5M0.SCEERUN)
    MODULES(
        CEEBINSS
        ,CEEPLPKA
    )
    MAXBELOWP(160K)
    MAXABOVEP(8M)
)
PHYSICAL (
    LIBRARY(CEE0120) ADD
    DSLIST(CEE.OS390R2.SCEERUN)
    MODULES(
        CEEBINSS
        ,CEEPLPKA
    )
    MAXBELOWP(160K)
    MAXABOVEP(8M)
)
PHYSICAL (
    LIBRARY(CEE0130) ADD
    DSLIST(CEE.OS390R3.SCEERUN)
    MODULES(
        CEEBINSS
        ,CEEPLPKA
    )
    MAXBELOWP(160K)
    MAXABOVEP(8M)
)
PHYSICAL (
    LIBRARY(CEE0240) ADD
    DSLIST(CEE.SCEERUN)
    MODULES(
        CEEBINSS
        ,CEEPLPKA
    )
    MAXBELOWP(160K)
    MAXABOVEP(8M)
)
LOGICAL (
    LIBRARY(SYSCEE) VERSION(V1R5M0) ADD
    PHYSICAL(CEEL150)
)
LOGICAL (
    LIBRARY(SYSCEE) VERSION(R2) ADD
    PHYSICAL(CEE0120)
)
LOGICAL (
    LIBRARY(OS390R3) VERSION(R3) ADD
    PHYSICAL(CEE0130)
)
LOGICAL (
    LIBRARY(LATEST) VERSION(0) ADD
    PHYSICAL(CEE0240)
    DEFAULT
)

```

Figure 1. An Example of a CSVRTLS PARMLIB Member for OS/390 Version 2 Release 4

Setting Run-Time Options

The RTLS, LIBRARY, and VERSION Language Environment run-time options control the use of RTLS when Language Environment applications are run.

When RTLS(ON) is specified, the LIBRARY(lib) and VERSION(ver) options are used to select which RTLS logical library and version are used. This logical library and version must be defined in the CSVRTLxx PARMLIB member, and RTLS must currently be managing it when the Language Environment application is started. CEEGINIT must be available in the z/OS program search order, which means that the SCEERTLS data set must be part of TASKLIB, STEPLIB/JOBLIB, LPA/LNKLST. The SCEERTLS data set cannot be controlled by RTLS.

When RTLS(OFF) is in effect, the LIBRARY and VERSION options are ignored. Whenever RTLS(ON) is specified, and the SCEERTLS data set is not in the program search order, RTLS(ON) is ignored, and no RTLS logical library will be used.

The RTLS, LIBRARY, and VERSION run-time options can be specified in the following places:

- CEEDOPT

The RTLS, LIBRARY, and VERSION options can be specified in the CEEDOPT CSECT link-edited with Language Environment. Note that when using multiple levels of SCEERUN with one level of SCEERTLS (always the current level), the settings for RTLS, LIBRARY, and VERSION in the CEEDOPT CSECT linked with CEEGINIT (in SCEERTLS) should match those linked into the other Language Environment modules (in the various copies of SCEERUN).

For more information on creating the CEEDOPT CSECT, see Chapter 4, “Customizing Language Environment Run-Time Options” on page 19.

- CEEUOPT

The RTLS, LIBRARY, and VERSION options can be specified in the CEEUOPT CSECT which is linked into the main program of the application.

For more information on creating the CEEUOPT CSECT, see Chapter 4, “Customizing Language Environment Run-Time Options” on page 19.

- #pragma runopts

For C/C++ programs, the RTLS, LIBRARY, and VERSION options can be coded on the #pragma runopts statement, along with other Language Environment run-time options. For more information on specifying Language Environment run-time options with #pragma runopts, see *z/OS Language Environment Programming Guide* and *C/C++ Language Reference*.

- CEEROPT

CEEROPT can be used in the same manner as CEEDOPT or CEECOPT to specify run-time options. This module will be its own load module, not linked with any other load modules. CEEROPT can be used to set run-time option defaults for environments in a CICS region, or an IMS or batch region using Library Routine Retention (LRR). The fact that this module resides in its own load module, not linked with any other load module, avoids the maintenance problems associated with linking it into a load module containing executable code. The CEEROPT will be loaded and merged with the installation run-time option defaults during region initialization. CEEROPT is optional. During environment initialization, an attempt to locate CEEROPT is performed. If it is found, the run-time options specified within will be merged with the installation defaults specified in CEEDOPT or CEECOPT. The CEEROPT load module can be created by running the CEEWROPT job found in the SCEESAMP data set. For

more information on specifying run-time options with CEEROPT, see *z/OS Language Environment Programming Guide*.

- PLIXOPT external variable

For PL/I main programs, the RTLS, LIBRARY, and VERSION options can be coded in the PLIXOPT external variable, along with other Language Environment run-time options. For more information on specifying Language Environment run-time options with PLIXOPT, see *z/OS Language Environment Programming Guide*.

- Command line

If the PLIST and EXECOPS settings allow Language Environment run-time options on the command line, RTLS, LIBRARY, and VERSION can be specified along with other run-time options. For more information on specifying Language Environment run-time options on the command line, see *z/OS Language Environment Programming Guide*.

When starting `/usr/sbin/init`, `BPX BATCH`, and the TSO/E OMVS command, Language Environment command line options can also be specified in the `BPXPRMxx PARMLIB` member. For more information on modifying the `BPXPRMxx PARMLIB` member, see *z/OS MVS Initialization and Tuning Reference*.

For the TSO/E OMVS command, the RUNOPTS operand can be used to specify Language Environment run-time options, including RTLS, LIBRARY, and VERSION. If specified, options specified in RUNOPTS override any options present in the `BPXPRMxx PARMLIB` member. Any Language Environment run-time options used by OMVS (from RUNOPTS, or `BPXPRMxx`) will be put into the `_CEE_RUNOPTS` environment variable as the login programs and the z/OS UNIX shells are invoked. If the `_CEE_RUNOPTS` environment variable is not reset, the RTLS, LIBRARY, and VERSION options from the OMVS command will be used for all z/OS UNIX shell commands run from the OMVS session. For more information on the TSO/E OMVS command syntax, see *z/OS UNIX System Services Command Reference*.

- `_CEE_RUNOPTS` environment variable

For programs invoked with one of the exec family functions (like z/OS UNIX shell commands), the RTLS, LIBRARY, and VERSION options can be specified in the `_CEE_RUNOPTS` environment variable.

For more information on setting the `_CEE_RUNOPTS` environment variable, see *z/OS Language Environment Programming Guide*.

- CEEBXITA

The Assembler user exit CEEBXITA cannot alter settings for the RTLS, LIBRARY, and VERSION run-time options. By the time that CEEBXITA is invoked, it is too late to change the RTLS logical library, since certain Language Environment modules have already been loaded.

When these options are specified in more than one place, the usual Language Environment options merge takes place except for the output of CEEBXITA, which is ignored. However, for nested enclaves, the settings for RTLS, LIBRARY, and VERSION are always inherited from the main enclave. All enclaves must use the same RTLS logical library and version.

Controlling Other Data Sets with RTLS

Customers who use the Debug Tool and who control SCEERUN using RTLS may also want to control the SEQAMOD data set with RTLS. This will result in better performance when using the Debug Tool. To do this, remove the SEQAMOD data set from any STEPLIB or LNKLST, and define an RTLS physical library containing the SEQAMOD data set. Then define an RTLS logical library that concatenates the SCEERUN physical library with the SEQAMOD physical library. When using the Debug Tool, ensure the necessary run-time options are specified.

You may also want to control IBM-supplied or user DLLs with RTLS. To do this, define physical libraries containing one or more of the DLL data sets. Then define RTLS logical libraries that concatenate the SCEERUN physical library with the DLL physical libraries.

You can also control data sets containing user load modules that are fetched or dynamically called. Similarly, define RTLS physical libraries containing these data sets and logical libraries as described above.

Performance Considerations

The use of RTLS may result in varying performance. If modules are cached using the MAXBELOW, MAXABOVE and other CSVRTL PARMLIB options, performance of your application will be noticeably better than using STEPLIB or LNKLST. If modules are not cached, the performance of your application will be comparable to STEPLIB.

See *z/OS MVS Initialization and Tuning Reference* for these PARMLIB options.

To avoid unnecessary performance degradation:

1. The SCEERTLS data set should not be in the program search order when RTLS(OFF) is in effect. In this case, if SCEERTLS is specified, an extra load for CEEGINIT is done, followed by the normal LOAD for CEEBINIT (CEEBINSS). This load is done once, during Language Environment initialization. Also, an extra DELETE of CEEGINIT is done during Language Environment termination.
2. When RTLS(ON) is in effect, the logical library should contain all frequently loaded user modules. LOADs of Language Environment modules will come from the RTLS logical library, and will cause no performance degradation (especially if they are already cached). However, if the application does frequent dynamic calls, C fetch()s, CANCELs, etc., there will be unnecessary overhead. For each user module fetched, the RTLS logical library is searched first, and this search will fail. Then, the module is fetched using the normal z/OS LOAD service, which will succeed.

Eliminate the extra searching of the RTLS logical library by creating a second physical library containing the user load modules. Define a logical library which concatenates the physical library containing the user modules with the one containing SCEERUN. This way, there is no longer a failing RTLS load request for each user module fetched.

Restrictions

This section lists certain restrictions that must be observed when using RTLS.

All z/OS elements and features that require Language Environment must use the current (latest) version. Use of earlier levels of the SCEERUN data set is not supported for these programs.

The following table contains RTLS restrictions when running in different environments.

Table 5. RTLS Restrictions

Conditions not supported	Notes
applications running under IMS	Unpredictable problems may occur if this restriction is violated.
applications that use System Programmer C (SPC)	
XPLINK applications	
C and PL/I preinitialization support	RTLS(ON) will be ignored, although it may appear in the options report.
Language Environment preinitialization facility (CEEPIPI)	RTLS(ON) will be ignored, although it may appear in the options report.
OS/VS COBOL applications	
reusable COBOL environments	The reusable COBOL environments are initiated by using the RTEREUS(ON) run-time option or by using the ILBOSTP0 or IGZERRE run-time routines.
LRR (Library Routine Retention)	Unpredictable problems may occur if this restriction is violated.
PL/I checkpoint/restart support	Unpredictable problems may occur if this restriction is violated.
COBOL checkpoint/restart support	Unpredictable problems may occur if this restriction is violated.
PL/I Multitasking applications compiled with pre-Language Environment PL/I compilers unless they have been relink-edited with Language Environment stubs.	Unpredictable problems may occur if this restriction is violated.
mixed PL/I modules that need both the Language Environment and shared PL/I libraries concatenated (or just the PL/I shared libraries)	Unpredictable results will occur if this restriction is violated.
RTLS(ON) does not affect the fetching of modules or DLLs from an HFS, even if the sticky bit is on (meaning that the module is really fetched from a data set or LPA).	Any Language Environment locale-oriented modules fetched from the HFS are not version-controlled by RTLS.

Other restrictions include:

- When the COBOL SORT or PL/I PLISRTx facilities are used, with RTLS(ON), the SORT module must be present in TASKLIB, STEPLIB/JOBLIB, or LPA/LNKLST, as usual. The SORT module is not fetched from an RTLS-controlled library.
- When using RTLS(ON) with the C Multitasking facility (MTF), the parallel load module (the user-written module containing EDCMTFS, whose name is passed to `tinit()`) cannot be RTLS-controlled. This module must be present in the usual data set, as if RTLS(ON) was not being used.

Note: If CEEDOPT has not been used to turn RTLSON globally, the user will need to link-edit a customized version of EDCMTFS into the parallel module. A `#pragma runopts(RTLSON)` statement must be added to the customized EDCMTFS source code. See *z/OS C/C++ Programming Guide* for a printout of the EDCMTFS source code.

- When using RTLSON with the FORTRAN Multitasking Facility (MTF), the module containing VFEIS# (called the parallel load module) cannot be RTLSON-controlled.

When running FORTRAN MTF, the AUTOTASK DD must still point to the data set containing the module with VFEIS#. This data set cannot be RTLSON-controlled.

- Split-RMODE modules (and any other multi-extent modules) cannot be fetched from RTLSON-controlled libraries.
- Programs with overlay segments will not work when they have been fetched from RTLSON-controlled libraries.

Note that programs with overlay segments can still run with RTLSON and can fetch other modules (such as those in the SCEERUN data set) from RTLSON-controlled libraries.

- Only the current (latest) version of the SCEERTLS data set for a given level of z/OS can be used.

The current version of the SCEERTLS data set is supported with all levels of SCEERUN that are supported by RTLSON on the current version of z/OS.

- When running VS COBOL II applications under Language Environment and RTLSON is used to control the level of SCEERUN, only the latest (current) level of SCEERUN can be used.
- When running pre-Language Environment C/370 2.1 and 2.2 applications under Language Environment and RTLSON is used to control the level of SCEERUN, only the latest (current) level of the SCEERUN data set can be used.

Unpredictable results will occur if this restriction is violated.

- When running pre-Language Environment PL/I 2.3 applications under Language Environment and RTLSON is used to control the level of SCEERUN, only the latest (current) level of the SCEERUN data set can be used.

For example, running a PL/I 2.3 application with RTLSON set up to use the Language Environment 1.5, OS/390 V1R2, or OS/390 V1R3 levels of SCEERUN is not allowed on OS/390 V2R4. Only the OS/390 V2R4 level of SCEERUN can be used with RTLSON.

Unpredictable results will occur if this restriction is violated.

- RTLSON can not be used to access load modules that reside in the SCEERUN2 data set.

- In nested enclaves, the RTLSON(OFF), LIBRARY and VERSION options inherited from the main enclave will always be in effect.

If different values for these options are specified for the nested enclave, these specified values will be syntax-checked and will show up in the options report for the nested enclave. However, the RTLSON logical library (if any) from the main enclave will be used in the nested enclave, so the options report for the nested enclave will be inaccurate.

- The settings of the RTLSON(OFF), LIBRARY(), and VERSION() options in the CEEDOPT CSECT linked with CEEGINIT should be the same as those in the CEEDOPT CSECT link-edited with the rest of Language Environment.

If this restriction is violated, the options report may be inaccurate.

- The assembler user exit routines (CEEBOXITA, IBMBXITA, IBMFXITA, etc.) cannot alter the RTLS options already in effect from CEEDOPT, CEEUOPT, CEEROPT, or the command line.
- If the SCEERTLS data set is present in the program search order when COBOL programs are invoked with non-MVS-EXEC-style parameter lists, certain problems may occur:
 - If a customized IGZEPSX user module has been installed, this module can turn on or off command line option parsing. Whenever command line parsing is turned off, Language Environment may erroneously look for the RTLS, LIBRARY, and VERSION options in the first passed-in parameter as if it contained Language Environment run-time options.
 - Without a customized IGZEPSX module, problems may occur for COBOL programs that are not invoked using ATTACH. COBOL programs that are not directly ATTACHED often do not have any passed-in Language Environment run-time options (under ISPF, for example). However, when looking for the RTLS, LIBRARY, and VERSION options, Language Environment assumes that the first parameter is a length-prefixed character string containing Language Environment run-time options. Scanning this parameter may cause ABENDs, or erroneous RTLS(ON/OFF) settings.
- SCEERUN must be available in the normal order of search when running the AD/CYCLE 1.2 C compiler (set to use Language Environment), unless the CEEDOPT CSECT link-edited into Language Environment specifies RTLS(ON). This compiler specifies NOEXECOPS, so the RTLS(ON) runtime option is not accepted on the command line.
- RTLS(ON) cannot be specified when running the Language Environment-enabled PL/I compiler (since Language Environment run-time options are not used). The SCEERUN data set must still be present in TASKLIB, STEPLIB/JOBLIB, or LPA/LNKLST (if it was required before).
- Utility programs are shipped in the SCEERUN data set. Whenever these are used, SCEERUN must still be present in the program search order. RTLS(ON) should not be used, since these IBM-provided utilities always need the current version of Language Environment that is provided in SCEERUN.
Examples of these utilities are: EDCPRLK, EDCICONV, EDCGNXLT, and EDCALIAS.
- Debuggers that do SVC screening of MVS LOADs and DELETEs (SVCs 8/122 and 9) or issue their own MVS LOADs and DELETEs may not work properly with RTLS(ON).
When modules are fetched from an RTLS-controlled LIBRARY, CSVRTLS is used rather than LOAD or DELETE.
- The C compiler does not accept certain valid values for the VERSION keyword if it is specified in the *#pragma runopts* directive. If a suboption to *#pragma runopts* is not a valid C/C++ token, you must surround the suboptions to *#pragma runopts* in double quotes.
An example is *#pragma runopts(RTLS(ON) VERSION(1.8.0))*. If you specify *#pragma runopts("RTLS(ON) VERSION(1.8.0)")*, then it will work.
CEEUOPT, CEEDOPT, CEEROPT, or the command line may be used to specify the VERSION in these cases.
The C++ compiler does not have this restriction.
- Certain problems may occur whenever multiple versions of a module are loaded into the same address space. An example of this type of problem is:
 1. There are two levels of user modules in use. LEVEL1.LOAD (in LPA or the LNKLST) contains production versions of these user-written modules, and is

normally used. LEVEL2.LOAD contains experimental new versions of these same modules. When these are needed, RTLS is used to fetch them. There is an RTLS logical library, LEVEL2, that points to CEE.SCEERUN and LEVEL2.LOAD.

2. A TSO user enters an z/OS UNIX shell, using the TSO/E OMVS command. `_BPX_SHAREAS=ON` is set, so that many z/OS UNIX shell commands will execute in the TSO/E address space.
3. The user then starts a long-running (user-written) z/OS UNIX shell command that employs `RTLS(ON) LIBRARY(LEVEL2)`, to access the experimental library level. This application then issues `fetch()` for `MODULE1`, which is fetched in from `LEVEL2.LOAD`.
4. While the long-running application is executing, PF6 is pressed to escape to TSO. The user then starts another application, which is a TSO command (not using Language Environment at all). This application issues `ATTACH` (or `LINK` or `XCTL`) for `MODULE1`, and expects to use the copy from `LEVEL1.LOAD` in the `LNKLST`. However, there is already a copy of `MODULE1` (from `LEVEL2.LOAD`) loaded for this address space, and the `ATTACH` may use that copy, rather than fetch a new copy. If so, the TSO command will use the new experimental version of `MODULE1` rather than the production version.

This type of problem may occur whenever `LINK`, `XCTL`, `ATTACH`, `CSVQUERY`, or `CSVINFO` is used and multiple versions of a module are available in an address space. Use of `LOAD` and `DELETE` does not cause problems when multiple versions of RTLS-controlled modules are present in the address space.

Some of the cases where Language Environment fetches modules using `ATTACH` or `LINK` are:

1. With the C Multitasking Facility (the module with `EDCMTF`)
2. With the FORTRAN Multitasking Facility (the module with `VFEIS#`)
3. Non-POSIX system()
4. When doing COBOL SORT or PL/I PLISRTx (SORT module)

Chapter 4. Customizing Language Environment Run-Time Options

The run-time option values IBM supplies with Language Environment may not suit the application programmers' needs at your site. Resetting the defaults will save the programmers' time because they will not need to override the run-time option defaults as often.

Refer to Chapter 12, "Language Environment Run-Time Options" on page 67 for detailed information about the run-time options, default values, and syntax. **You may not need to change most default values.** You can fill in the blanks in Table 6 with the changes you plan to make in the defaults for both z/OS batch and CICS processing.

Table 6. Worksheet: Planning to Customize Language Environment Run-Time Options

Run-time option	z/OS Default Value	New z/OS Default	CICS Default Value	New CICS Default	Page
ABPERC	(NONE)		N/A	N/A	68
ABTERMENC	(ABEND)		(ABEND)		69
AIXBLD	(OFF)		N/A	N/A	71
ALL31	(ON)		(ON)		72
ANYHEAP	(16K,8K, ANYWHERE, FREE)		(4K,4080, ANYWHERE, FREE)		73
AUTOTASK	NOAUTOTASK		N/A	N/A	75
BELOWHEAP	(8K,4K,FREE)		(4K,4080, FREE)		76
CBLOPTS	(ON)		N/A	N/A	77
CBLPSHPOP	N/A	N/A	(ON)		78
CBLQDA	(OFF)		N/A	N/A	79
CHECK	(ON)		(ON)		80
COUNTRY	(US)		(US)		80
DEBUG	(OFF)		(OFF)		81
DEPTHCONDLMT	(10)		(10)		82
ENVAR	(")		(")		84
ERRCOUNT	(0)		(0)		85
ERRUNIT	(6)		N/A	N/A	86
FILEHIST	(ON)		N/A	N/A	87
FILETAG	(NOAUTOCVT,NOAUTOTAG)		N/A	N/A	88
HEAP	(32K,32K, ANYWHERE, KEEP,8K, 4K)		(4K,4080, ANYWHERE, KEEP,4K, 4080)		90
HEAPCHK	(OFF,1,0,0)		(OFF,1,0,0)		92
HEAPOOLS	(OFF,8,10,32,10, 128,10,256,10, 1024,10,2048,10)		(OFF,8,10,32,10, 128,10,256,10, 1024,10,2048,10)	N/A	94
INFOMSGFILTER	(OFF,,,,)		(OFF,,,,)	N/A	95
INQPCOPN	(ON)		N/A	N/A	97
INTERRUPT	(OFF)		N/A	N/A	97

Table 6. Worksheet: Planning to Customize Language Environment Run-Time Options (continued)

Run-time option	z/OS Default Value	New z/OS Default	CICS Default Value	New CICS Default	Page
LIBRARY	(SYSCEE)		N/A	N/A	98
LIBSTACK	(4K,4K,FREE)		(32,4080,FREE)		99
MSGFILE	(SYSOUT,FBA, 121,0,NOENQ)		N/A	N/A	101
MSGQ	(15)		N/A	N/A	104
NATLANG	(ENU)		(ENU)		105
NONIPTSTACK	Replaced by THREADSTACK				
OCSTATUS	(ON)		N/A	N/A	107
PC	(OFF)		N/A	N/A	108
PLITASKCOUNT	(20)		N/A	N/A	109
POSIX	(OFF)		N/A	N/A	109
PROFILE	(OFF, '')		(OFF, '')	(OFF, '')	111
PRTUNIT	(6)		N/A	N/A	111
PUNUNIT	(7)		N/A	N/A	112
RDRUNIT	(5)		N/A	N/A	113
RECPAD	(OFF)		N/A	N/A	113
RPTOPTS	(OFF)		(OFF)		114
RPTSTG	(OFF)		(OFF)		117
RTEREUS	(OFF)		N/A	N/A	127
RTLS	(OFF)		N/A	N/A	128
SIMVRD	(OFF)		N/A	N/A	129
STACK	(128K,128K,ANYWHERE,KEEP,512K,128K)		(4K,4080,ANYWHERE,KEEP,4K,4080))		130
STORAGE	(NONE,NONE,NONE,0K)		(NONE,NONE,NONE,0K)		133
TERMTHDACT	(TRACE,,96)		(TRACE,CESE,96)		136
TEST	NOTEST(ALL, *,PROMPT, INSPREF)		NOTEST(ALL, *,PROMPT, INSPREF)		142
THREADHEAP	(4K,4K,ANYWHERE, KEEP)		N/A	N/A	145
THREADSTACK	(OFF,4K,4K, ANYWHERE, KEEP,128K,128K)		N/A	N/A	146
TRACE	(OFF,4K,DUMP,LE=0)		(OFF,4K,DUMP,LE=0))		149
TRAP	(ON,SPIE)		(ON,SPIE)		151
UPSI	(00000000)		(00000000)		153
USRHDLR	NOUSRHDLR()		NOUSRHDLR()		154
VCTRSAVE	(OFF)		N/A	N/A	155
VERSION	('')		N/A	N/A	156
XUFLOW	(AUTO)		(AUTO)		157

Note: The abbreviation N/A is used for not applicable.

You also need to choose which sample customization jobs you need to modify and run. Table 7 lists the sample jobs that are members of Language Environment sample library SCEESAMP. These sample jobs require other members of the SCEESAMP data set.

Table 7. Sample jobs to change run-time options defaults

Set Defaults For	Sample Job	Required Member
Installation-wide z/OS batch	CEEWD OPT	CEEDOPT
Installation-wide CICS	CEEWCOPT	CEECO PT
Region-specific CICS	CEEWROPT	CEECO PT
Region-specific IMS	CEEWROPT	CEEDOPT
Application-specific	CEEUOPT	CEEUOPT

Setting Default Options with the CEEXOPT Macro

When you run the sample jobs, they create the CEEDOPT CSECT, an options control block which establishes the defaults for the options. The jobs invoke CEEXOPT during the assembly of the CEEDOPT module. When you modify the CEEXOPT macro invocation to change installation-wide defaults, you must specify each run-time option as either OVR or NONOVR.

Guideline: To invoke CEEXOPT, adhere to the syntax of the IBM-supplied template for CEEDOPT as shown in Figure 2 for z/OS and Figure 3 for CICS. These are samples and should be compared to the actual code before you attempt to use them. Xs are in column 72.

Figure 2 on page 22 shows the IBM-supplied version of the CEEDOPT CSECT contained within the CEEDOPT member, with the default suboption values for each of the options.

Figure 3 on page 23 shows the IBM-supplied version of the CEEDOPT CSECT contained within the CEECOPT member, with the default suboption values for each of the options. The CEECOPT member:

- is the CICS-specific version of the CEEDOPT member.
- contains CEEDOPT CSECT, whose contents differ from those of CEEDOPT CSECT in the CEEDOPT member.

```

CEEDOPT CSECT
CEEDOPT AMODE ANY
CEEDOPT RMODE ANY
CEEOPT ABPERC=((NONE),OVR), X
      ABTERMENC=((ABEND),OVR), X
      AIXBLD=((OFF),OVR), X
      ALL31=((ON),OVR), X
      ANYHEAP=((16K,8K,ANYWHERE,FREE),OVR), X
      BELOWHEAP=((8K,4K,FREE),OVR), X
      CBLPTS=((ON),OVR), X
      CBLPSHPOP=((ON),OVR), X
      CBLQDA=((OFF),OVR), X
      CHECK=((ON),OVR), X
      COUNTRY=((US),OVR), X
      DEBUG=((OFF),OVR), X
      DEPTHCONDLMT=((10),OVR), X
      ENVAR=('',OVR), X
      ERRCOUNT=((0),OVR), X
      ERRUNIT=((6),OVR), X
      FILEHIST=((ON),OVR), X
      FILETAG=((NOAUTOCVT,NOAUTOTAG),OVR), X
      HEAP=((32K,32K,ANYWHERE,KEEP,8K,4K),OVR), X
      HEAPCHK=((OFF,1,0,0),OVR), X
      HEAPPOOLS=((OFF,8,10,32,10,128,10,256,10,1024,10,2048, X
      10),OVR), X
      INFMSGFILTER=((OFF,,,),OVR), X
      INQPCOPN=((ON),OVR), X
      INTERRUPT=((OFF),OVR), X
      LIBRARY=((SYSCEE),OVR), X
      LIBSTACK=((4K,4K,FREE),OVR), X
      MSGFILE=((SYSOUT,FBA,121,0,NOENQ),OVR), X
      MSGQ=((15),OVR), X
      NATLANG=((ENU),OVR), X
      NOAUTOTASK=(OVR), X
      NOTEST=((ALL,*,PROMPT,INSPREF),OVR), X
      NOUSRHDLR=('',OVR), X
      OCSTATUS=((ON),OVR), X
      PC=((OFF),OVR), X
      PLITASKCOUNT=((20),OVR), X
      POSIX=((OFF),OVR), X
      PROFILE=((OFF,''),OVR), X
      PRTUNIT=((6),OVR), X
      PUNUNIT=((7),OVR), X
      RDRUNIT=((5),OVR), X
      RECPAD=((OFF),OVR), X
      RPTOPTS=((OFF),OVR), X
      RPTSTG=((OFF),OVR), X
      RTEREUS=((OFF),OVR), X
      RTLS=((OFF),OVR), X
      SIMVRD=((OFF),OVR), X
      STACK=((128K,128K,ANYWHERE,KEEP,128K,128K),OVR), X
      STORAGE=((NONE,NONE,NONE,0K),OVR), X
      TERMTHDACT=((TRACE,,96),OVR), X
      THREADHEAP=((4K,4K,ANYWHERE,KEEP),OVR), X
      THREADSTACK=((OFF,4K,4K,ANYWHERE,KEEP,128K,128K),OVR), X
      TRACE=((OFF,4K,DUMP,LE=0),OVR), X
      TRAP=((ON,SPIE),OVR), X
      UPSI=((00000000),OVR), X
      VCTRSAVE=((OFF),OVR), X
      VERSION=('',OVR), X
      XUFLOW=((AUTO),OVR)
END

```

Figure 2. Sample Invocation of CEEEOPT within the CEEDOPT Member

```

CEEDOPT CSECT
CEEDOPT AMODE ANY
CEEDOPT RMODE ANY
CEEXOPT ABPERC=((NONE),OVR), X
        ABTERMENC=((ABEND),OVR), X
        AIXBLD=((OFF),OVR), X
        ALL31=((ON),OVR), X
        ANYHEAP=((4K,4080,ANYWHERE,FREE),OVR), X
        BELOWHEAP=((4K,4080,FREE),OVR), X
        CBLPTS=((ON),OVR), X
        CBLPSHPOP=((ON),OVR), X
        CBLQDA=((OFF),OVR), X
        CHECK=((ON),OVR), X
        COUNTRY=((US),OVR), X
        DEBUG=((OFF),OVR), X
        DEPTHCONDLMT=((10),OVR), X
        ENVAR=('',OVR), X
        ERRCOUNT=((0),OVR), X
        ERRUNIT=((6),OVR), X
        FILEHIST=((ON),OVR), X
        FILETAG=((NOAUTOCVT,NOAUTOTAG),OVR), X
        HEAP=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR), X
        HEAPCHK=((OFF,1,0,0),OVR), X
        HEAPPOOLS=((OFF,8,10,32,10,128,10,256,10,1024,10,2048, X
        10),OVR), X
        INFOMSGFILTER=((OFF,,,),OVR), X
        INQPCOPN=((ON),OVR), X
        INTERRUPT=((OFF),OVR), X
        LIBRARY=((SYSCEE),OVR), X
        LIBSTACK=((32,4080,FREE),OVR), X
        MSGFILE=((SYSOUT,FBA,121,0,NOENQ),OVR), X
        MSGQ=((15),OVR), X
        NATLANG=((ENU),OVR), X
        NOAUTOTASK=(OVR), X
        NOTEST=((ALL,*,PROMPT,INSPREF),OVR), X
        NOUSRHDLR=(),OVR), X
        OCSTATUS=((ON),OVR), X
        PC=((OFF),OVR), X
        PLITASKCOUNT=((20),OVR), X
        POSIX=((OFF),OVR), X
        PROFILE=((OFF,''),OVR), X
        PRTUNIT=((6),OVR), X
        PUNUNIT=((7),OVR), X
        RDRUNIT=((5),OVR), X
        RECPAD=((OFF),OVR), X
        RPTOPTS=(OFF,OVR), X
        RPTSTG=((OFF),OVR), X
        RTEREUS=((OFF),OVR), X
        RTLS=((OFF),OVR), X
        SIMVRD=((OFF),OVR), X
        STACK=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR), X
        STORAGE=((NONE,NONE,NONE,0K),OVR), X
        TERMTHDACT=((TRACE,CESE,96),OVR), X
        THREADHEAP=((4K,4080,ANYWHERE,KEEP),OVR), X
        THREADSTACK=((OFF,4K,4080,ANYWHERE,KEEP,4K,4080),OVR), X
        TRACE=((OFF,4K,DUMP,LE=0),OVR), X
        TRAP=((ON,SPIE),OVR), X
        UPSI=((00000000),OVR), X
        VCTRSAVE=((OFF),OVR), X
        VERSION=('',OVR), X
        XUFLOW=((AUTO),OVR)
END

```

Figure 3. Sample Invocation of CEEXOPT within the CEECOPT Member

Requirements for coding the CEEXOPT macro

- **Continuing lines of code**

A continuation character (X in the source) must be in column 72 on each line of the CEEXOPT invocation except the last line. The continuation line must start in column 16. You can break the coding after any comma.

- **Case sensitivity**

Options and suboptions must be in uppercase. Only suboptions that are strings can be specified in mixed-case or lowercase. For example, both MSGFILE=(SYSOUT) and MSGFILE=(sysout) are acceptable.

- **Comma**

A comma must end each option except for the final option. If the comma is omitted, everything following the option is treated as a comment.

- **Special characters**

If one of the string suboptions contains a special character (for example, an embedded blank or unmatched right or left parenthesis), the string must be enclosed in single quotation marks ('), not in double quotation marks ("). (You can specify a null string with either contiguous single or double quotation marks.)

To obtain a single quotation mark (') or a single ampersand (&) within a string, you must specify two contiguous instances of the character. The pair is counted as only one character in determining whether the maximum allowable string length has been exceeded and in setting the effective length of the string.

- **Maximum length**

Macro instruction operands cannot be longer than 255 characters. Therefore, it is not possible for each suboption of the NOTEST/TEST option to attain the maximum allowable length normally permitted by Language Environment. For example, the command suboption of NOTEST/TEST permits 250 characters while the preference suboption of NOTEST/TEST allows 80 characters. The total number of characters for these two suboptions exceeds that allowed by the CEEXOPT macro.

If the number of characters to the right of the equal sign is greater than 255 for any keyword parameter in the CEEXOPT invocation in CEEDOPT, CEECOPT, CEEROPT, or CEEUOPT, a return code of 12 is produced for the assembly, and the options are not parsed properly.

- **Apostrophes**

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

```
IEV03 *** ERROR *** NO ENDING APOSTROPHE
```

Such a message bears no particular spatial relationship to the offending suboption, and the options are not parsed properly if this error is detected.

- **Omissions permitted in CEEUOPT and CEEROPT**

You can completely omit the specification of any option in CEEUOPT and CEEROPT. Default values are then supplied for each of the missing suboptions in the options control block that is generated, and these values are ignored at the time Language Environment merges the options.

There are two recommended ways of omitting an option. The HEAP run-time option is used below to demonstrate:

- Specify the option with only a comma following the equal sign:
HEAP=, X
- Specify the option with empty parentheses and comma following the equal sign:

HEAP=(), X

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

In CEEUOPT, IBM recommends that you omit any options you do not wish to change. The options you omit from the macro will default to the installation-wide defaults you set in CEEDOPT, CEEROPT, or CEECOPT.

- **Omission of suboptions in CEEUOPT**

In CEEUOPT, you can use commas to indicate the omission of one or more suboptions for options having more than one suboption. For example, if you wish to specify only the second suboption of the STORAGE option, the omission of the 1st, 3rd, and 4th suboptions can be indicated in any of the following ways:

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

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

- **Options that permit only one suboption**

Options that permit only one suboption do not need to enclose that suboption in parentheses. For example, you can specify the COUNTRY option in CEEUOPT in either of the following ways:

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

- **No omissions permitted in CEEDOPT and CEECOPT**

Each option in the CEEDOPT or CEECOPT template must be present, and each of its suboptions must be specified with one of the legal suboption values, except for the suboptions of the AUTOTASK | NOAUTOTASK and USRHDLR | NOUSRHDLR options. The final suboption for each CEEDOPT or CEECOPT option must be OVR or NONOVR. OVR means that the option can be overridden at run-time. NONOVR means that the option cannot be overridden at run-time.

Changing Installation-Wide Run-Time Options Defaults (Non-CICS)

Use the CEEWDOPT sample job to change the installation-wide defaults for the Language Environment run-time options. Use the worksheet in Table 6 on page 19 to select your default values and use the information in Chapter 12, “Language Environment Run-Time Options” on page 67 for more detail about the options and their syntax.

These defaults apply to applications running with the Language Environment library. This includes the C/C++ Compiler, Prelinker, and Object Library Utility.

Modifying the JCL for CEEWDOPT

1. Copy the CEEDOPT member from CEE.SCEESAMP into CEEWDOPT in place of the comment lines following the ++ SRC statement.
2. Change the parameters on the CEEXOPT macro statement in CEEDOPT to reflect the values you have chosen for your installation-wide default run-time options.
3. Change #GLOBALCSI to the data set name of your global CSI data set.
4. Change #TZONE to the name of your target zone.
5. Add necessary SMP/E PRE information for PTFs you have installed on your system which contain the same part.

CEEWDOPT should run with a condition code of 0.

Note: Alternatively, you can use ++ SRCUPD to change a subset of the run-time options in this sample job.

Changing Installation-Wide Run-Time Options Defaults (CICS)

Use the CEEWCOPT sample job to change the installation-wide defaults for the Language Environment run-time options under CICS. Use the worksheet in Table 6 on page 19 to select your default values and use the information in Chapter 12, “Language Environment Run-Time Options” on page 67 for more detail about the options and their syntax.

Modifying the JCL for CEEWCOPT

1. Copy member CEECOPT from CEE.SCEESAMP into CEEWCOPT in place of the comment lines following the ++ SRC statement.
2. Change the parameters on the CEEXOPT macro statement in CEECOPT to reflect the values you have chosen for your installation-wide default run-time options.
3. Change #GLOBALCSI to the data set name of your global CSI data set.
4. Change #TZONE to the name of your target zone.
5. Add necessary SMP/E PRE information for PTFs you have installed on your system which contain the same part.

CEEWCOPT should run with a condition code of 0.

Note: Alternatively, you can use ++ SRCUPD to change a subset of the run-time options in this sample job.

Creating a Region-Specific Run-Time Options Load Module

The system programmers at your site might need to set different run-time options on a CICS or IMS region basis. For example, one CICS region (Region A) can be designated to run only AMODE 31 programs, while another region (Region B) will run both AMODE 24 and AMODE 31 programs. This would require Region B to have the ALL31(OFF) option setting while Region A could perform better with the ALL31(ON) option setting.

A run-time options load module, CEEROPT, is available just for this purpose. CEEROPT is a stand-alone load module that can be used to set Language Environment run-time options that are different than the installation defaults with:

1. CICS
2. IMS with Library Routine Retention (LRR)
3. Any other users of LRR

If a CEEROPT load module is present in a program search order, Language Environment will load and merge the options in CEEROPT with the installation default run-time options.

The CEEROPT load module can be created by running the CEEWROPT job found in the SCEESAMP data set. Language Environment does not ship a default CEEROPT load module. Each run of the CEEWROPT sample job will create a new CEEROPT options load module in a user-specified library. The system programmer can then include this specific library that contains the CEEROPT load module as part of the STEPLIB concatenation for IMS startup job streams or the DFHRPL

concatenation for CICS. Run-time options in CEEROPT override the default options in CEEDOPT or CEECOPT, unless NONOVR was specified for the option when CEEDOPT or CEECOPT was created.

CEEWROPT does not use SMP/E to create the options load module, so it can be run several times to create several different CEEROPT load modules, each in its own specific library. Any run-time options omitted from CEEROPT will be picked up from the installation defaults. Therefore, the system programmer does not need to rebuild CEEROPT when upgrading the z/OS release. Use the information in Chapter 12, “Language Environment Run-Time Options” on page 67 to select the values for your application-specific run-time options load modules.

Modifying the JCL for CEEWROPT

1. Copy member CEEDOPT or CEECOPT from CEE.SCEESAMP into CEEWROPT in place of the comment lines following the SYSIN DD statement.
2. Change the CSECT name and labels from CEEDOPT to CEEROPT.
3. Change the parameters on the CEEXOPT macro statement to reflect the values you have chosen for this region-specific run-time options load module.
4. Code just the options you want to change. Options you omit from CEEROPT will remain the same as the installation defaults.
5. Change DSNAME=YOURLIB in the SYSLMOD DD statement to the name of the partitioned data set into which you want your CEEROPT load module to be link-edited. This data set does not need to be APF authorized.

Note: If you have a CEEROPT load module in your current data set, it will be replaced by the new version.

6. Check the SYSLIB DD statement to ensure the data set names are correct.

CEEWROPT should run with a condition code of 0.

Creating an Application-Specific Run-Time Options Module

The application programmers at your site might need more than one set of run-time options. You can accommodate this need by establishing one set of options as the installation-wide default by using the CEEWDOPT or CEEWCOPT sample job described above, then you can create application-specific run-time options modules using the CEEWUOPT sample job.

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

CEEWUOPT does not use SMP/E to create the options module, so it can be run several times to create several different CEEUOPT modules, each in its own user-specified library. Use the information in Chapter 12, “Language Environment Run-Time Options” on page 67 to select the values for your application-specific run-time options modules.

Modifying the JCL for CEEWUOPT

1. Copy member CEEUOPT from CEE.SCEESAMP into CEEWUOPT in place of the comment lines following the SYSIN DD statement.

2. Change the parameters on the CEEXOPT macro statement in CEEUOPT to reflect the values you have chosen for this application-specific run-time options module.
3. Code just the options you want to change. Options you omit from CEEUOPT will remain the same as the installation defaults.
4. Change DSNAME=YOURLIB in the SYSLMOD DD statement to the name of the partitioned data set into which you want your CEEUOPT module to be link-edited.
Note: If you have a CEEUOPT module in your current data set, it will be replaced by the new version.
5. Check the SYSLIB DD statement to ensure the data set names are correct.

CEEWUOPT should run with a condition code of 0.

Chapter 5. Customizing User Exits

Language Environment provides support for the following user exits:

- **Assembler user exit**

The assembler user exit can be used to perform functions for enclave initialization, normal and abnormal enclave termination, and process termination. See “Changing the Assembler Language User Exit” on page 30.

- **High-level language (HLL) user exit**

The HLL user exit can be used to perform functions for enclave initialization. See “Changing the High-Level Language User Exit” on page 32.

- **Abnormal termination user exit**

The abnormal termination user exit can be used to collect problem determination data when Language Environment is terminating an enclave due to an unhandled condition. See “Customizing Language Environment Abnormal Termination Exits” on page 33.

- **Load notification user exit**

The load notification user exit can be used to improve performance by preventing frequently used modules from being loaded and deleted with each use. See “Creating a Load Notification User Exit” on page 36.

The load notification user exit is only available when Library Routine Retention (LRR) is used.

- **Storage tuning user exit**

The storage tuning user exit provides a programming interface that allows you to collect Language Environment storage tuning information and to set the Language Environment run-time option values for STACK, LIBSTACK, HEAP, ANYHEAP and BELOWHEAP. See “Creating a Storage Tuning User Exit” on page 38.

The storage tuning user exit is available for CICS, and for non-CICS environments when LRR is used.

Refer to *z/OS Language Environment Programming Guide* for detailed information about the features of the exits, default values, and syntax.

Choose which sample customization jobs to modify and run. Table 8 lists the sample jobs that are members of Language Environment sample library SCEESAMP.

Table 8. Sample Customization Jobs for the User Exits

Use This Sample Job	To
CEEWDXIT	Change installation-wide assembler language user exit.
CEEWCXIT	Change installation-wide CICS assembler language user exit.
CEEWUXIT	Create an application-specific assembler language user exit.
CEEWHLLX	Change high-level language user exit.
CEEWDEXT	Identify an abnormal termination exit (non-CICS).
CEEWCEXT	Identify an abnormal termination exit (CICS).
CEEWLNUE	Identify a load notification user exit.

An Example

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

Examples of conditions that are severity 2 or greater include:

- Program interrupts
- System abends
- Conditions detected by Language Environment: for example, a program load failure

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

Changing the Assembler Language User Exit

Three sample jobs are installed in the CEE.SCEESAMP target data set to help you modify the assembler language user exit. Two of the jobs use SMP/E USERMODs to replace the IBM-supplied installation-wide assembler user exits. The third sample job creates an application-specific assembler user exit that can be link-edited with applications that need its functions. You can create several different application-specific user exits, each in a different partitioned data set, to satisfy the needs of different application programs. Source code for the sample assembler user exits is installed as members in the CEE.SCEESAMP data set.

Table 9. Sample Assembler User Exits for Language Environment

Example User Exit	Operating System	Language (if Language-Specific)
CEEBXITA	z/OS (default)	
CEEBXITC	TSO/E	
CEECXITA	CICS (default)	
CEEBX05A	z/OS	VS COBOL II compatibility

Notes:

1. CEEBXITA and CEECXITA are the defaults on your system for z/OS and CICS, if Language Environment is installed at your installation without modification.
2. The source code for CEEBXITA, CEEBXITC, CEECXITA, and CEEBX05A can be found in the SCEESAMP sample library.

Use the information in *z/OS Language Environment Programming Guide* to assist you in modifying the IBM-supplied user exits or in creating your own.

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

The assembler user exit CEEBXITA cannot alter settings for the RTLS, LIBRARY, and VERSION run-time options. By the time that CEEBXITA is invoked, it is too late to change the RTLS logical library, since certain Language Environment modules have already been loaded.

CEEBXITA performs functions for enclave initialization, normal and abnormal enclave termination, and process termination. CEEBXITA must be written in assembler language, because an HLL environment might not be established when the exit is invoked.

You can set up user exits for tasks such as:

- Installation accounting and charge back
- Installation audit controls
- Programming standard enforcement
- Common application run-time support

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

Use the CEEWDXIT sample job to change the installation-wide assembler language user exit. You must replace the comment in CEEWDXIT with your source for CEEBXITA. You can copy the source for the IBM-supplied default installation-wide assembler language user exit from CEEBXITA in CEE.SCEESAMP and modify it to suit your needs, or you can create your own source for CEEBXITA. Use the information in *z/OS Language Environment Programming Guide* to guide you in coding your changes.

Modifying the JCL for CEEWDXIT

1. Replace the comment lines following the ++ SRC statement in the job with your source program for the installation-wide assembler language user exit.
2. Change #GLOBALCSI to the data set name of your global CSI data set.
3. Change #TZONE to the name of your target zone.
4. Add necessary SMP/E PRE information for PTFs you have installed on your system which contain the same part.

CEEWDXIT should run with a condition code of 0.

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

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

Note the difference between the IBM-supplied CEEBXITA and the IBM-supplied CEECXITA. You can retain some or all of these differences in your user exit. Use the information in *z/OS Language Environment Programming Guide* to guide you in coding your changes.

Modifying the JCL for CEEWCXIT

1. Replace the comment lines following the ++ SRC statement in the job with your source program for the installation-wide CICS assembler language user exit.
2. Change #GLOBALCSI to the data set name of your global CSI data set.
3. Change #TZONE to the name of your target zone.
4. Add necessary SMP/E PRE information for PTFs you have installed on your system which contain the same part.

CEEWCXIT should run with a condition code of 0.

Creating an Application-Specific Assembler Language User Exit

Use the CEEWUXIT sample job to create as many application-specific assembler language user exits as your site requires. You must replace the comment in CEEWUXIT with your source. You can copy the source for the IBM-supplied default installation-wide assembler language user exit from CEEBXITA or CEEBXITC in CEE.SCEESAMP and modify it to suit your needs, or you can create your own source.

CEEWUXIT does not use SMP/E to create the assembler language user exit module, so it can be run several times to create several different CEEBXITA modules, each in its own user-specified library. Use the information in *z/OS Language Environment Programming Guide* to guide you in coding your changes.

Modifying the JCL for CEEWUXIT

1. Replace the comment lines following the //SYSIN statement in the job with your source program for the application-specific assembler language user exit.
2. Change DSNAME=YOURLIB in the SYSLMOD DD statement to the name of the partitioned data set you want your CEEBXITA module link-edited into.
Note: A CEEBXITA module currently in the chosen data set is replaced by the new version.
3. Check the SYSLIB DD statement to ensure that the data set names are correct.

CEEWUXIT should run with a condition code of 0.

Changing the High-Level Language User Exit

The CEEWHLLX sample job contains an SMP/E USERMOD that replaces the IBM-supplied high-level language user exit with your high-level language user exit. The USERMOD contains the object program for the user exit, not the high-level language source.

SMP/E is not able to compile a source language other than assembler language, so you must compile your user exit and place the object program produced by the compiler into the USERMOD in CEEWHLLX. Refer to *z/OS Language Environment Programming Guide* for a description of the high-level language user exit interface.

If you write your high-level language user exit in C/C++ use the #pragma csect statement to name the CSECT CEEBINT. This lets SMP/E maintain the CSECT properly. You can also write high-level language user exits in PL/I and Language Environment-conforming assembler.

If you use any of the C/C++ library functions, the CEEWHLLX job might generate the following message.

```
IEW2454W nnnn SYMBOL xxxxxxxx UNRESOLVED.  
NO AUTOCALL (NCAL) SPECIFIED.
```

Although you might receive a condition code of 04, this does not indicate an error.

Modifying the JCL for CEEWHLLX

1. Replace the comment lines following the ++ MOD statement in CEEWHLLX with the object program obtained by compiling your high-level language user exit.
2. Change #GLOBALCSI to the data set name of your global CSI data set.
3. Change #TZONE to the name of your target zone.

4. Add necessary SMP/E PRE information for PTFs you have installed on your system which contain the same part.

CEEWHLLX should run with a condition code of 0.

Exception: If your exit is written in C/C++, you could get a condition code of 4 if your job runs correctly.

Customizing Language Environment Abnormal Termination Exits

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

The sample abnormal termination exits CEEBDATX and CEECDATX are no longer provided with Language Environment. You can use the TERMTHDACT(UADUMP) run-time option to generate a system dump of the user address space.

The CEEEXTAN (non-CICS) and CEECXTAN (CICS) CSECTs, which are installed in the CEE.SCEESAMP target data set, contain the instructions for defining which abnormal termination exits, if any, will be called when a routine terminates abnormally. Use the CEEWDEXT (non-CICS) and CEEWCEXT (CICS) sample jobs to replace the existing CSECTs with your updated CSECTs in your run-time library. For the syntax and other considerations for abnormal termination exits, see *z/OS Language Environment Programming Guide*.

Creating a Language Environment Abnormal Termination Exit

To create an abnormal termination exit:

1. Create an assembler language routine that conforms to the syntax described in *z/OS Language Environment Programming Guide*.
2. Assemble and link-edit your exit into a library that Language Environment can access at run-time, such as SCEERUN.
3. Code a CEEEXTAN CSECT that contains a CEEXART macro identifying your exit. The macro specifies your routine as an abnormal termination exit routine. The CEEEXTAN CSECT can be found in source file CEECXTAN (for CICS) or CEEEXTAN (for non-CICS). See “CEEEXTAN Abnormal Termination Exit CSECT” for more information.
4. Replace the existing CEEEXTAN CSECT with the updated CEEEXTAN as described in the sections below.

CEEEXTAN Abnormal Termination Exit CSECT

CEEEXTAN is a CSECT explicitly linked with the Language Environment condition handling routines, and it is the CSECT that you create by coding the CEEXAHD, CEEXART, and CEEXAST macros. Specifically, CEEEXTAN is linked with the CEEPLPKA and CEECCICS load modules. CEEEXTAN CSECT is created through the use of the following Language Environment-provided assembler macros:

CEEXAHD

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

CEEXART

Identifies the name of the abnormal termination exit to be invoked. It

generates one entry for an abnormal termination exit. It has only one keyword parameter, `TERMXIT=`, which is the load name for the abnormal termination exit. There is a limit of 8 characters for the load name, and no validation of the name is performed by the macro.

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

CEEXAST

Identifies the end of the list of abnormal termination exits. It generates the trailer for the `CEEEXTAN` CSECT. It has no parameters.

Language Environment validates the format of the abnormal termination exit CSECT and issues a load of the names as identified in the table. The `LOAD` is attempted only for terminations due to unhandled conditions of severity 2 or greater. If the `LOAD` is successful, an abnormal termination exit is invoked according to the interface described below. If the `LOAD` fails (the routine cannot be found, or there is not enough storage for the routine, for example), no error indication is delivered and either the next name in `CEEEXTAN` is chosen, or termination continues (if the names have been exhausted). This allows a `STEPLIB` to either contain or omit the load names, depending on whether you want the exit to be used for this job.

Note: When `RTLS(ON)` is in effect, the `RTLS` logical library is searched for the load names ahead of the `TASKLIBs`, `STEPLIB`/`JOBLIB`, `LPA`/`LNKLST`, etc.

Jobs to Generate and Modify CEEEXTAN CSECT

You can use two source files to generate `CEEEXTAN` CSECT, one for `CICS` and one for non-`CICS`. The following source files are provided in the `SCEESAMP` data set:

CEEEXTAN

Source to generate `CEEEXTAN` CSECT for `CICS`

CEEEXTAN

Source to generate `CEEEXTAN` CSECT for non-`CICS`

You can use the following two jobs to replace `CEEEXTAN` CSECT:

CEEWCEXT

Replaces `CEEEXTAN` CSECT for `CICS`

CEEWDEXT

Replaces `CEEEXTAN` CSECT for non-`CICS`

Figure 4 on page 35 contains the source for the IBM-supplied `CEEEXTAN`:

```

                TITLE 'LE/370 Abnormal Termination User exit CSECT'
                CEEXAHD          ,User exit header
*
*****
* To specify an abnormal termination exit, change the line
* where CEEXART is specified:
* - change the XXXXXXXX to the name of the abnormal termination exit
* - change the '*' in column 1 to a blank
*****
*          CEEXART  TERMxit=XXXXXXXX
*
                CEEXAST          ,Terminate the list

```

Figure 4. Default CEEEXTAN

If you want to add your own abnormal termination exit called WHODIDIT, then the code should look like the following:

```

                TITLE 'LE/370 Abnormal Termination User exit CSECT'
                CEEXAHD          ,User exit header
*
*****
* To specify an abnormal termination exit, change the line
* where CEEXART is specified:
* - change the XXXXXXXX to the name of the abnormal termination exit
* - change the '*' in column 1 to a blank
*****
                CEEXART  TERMxit=WHODIDIT
*
                CEEXAST          ,Terminate the list

```

Figure 5. Updated CEEEXTAN

Identifying the Abnormal Termination Exit (Non-CICS)

Use the CEEWDEXT sample job to specify your own abnormal termination exit in a non-CICS environment.

Modifying the JCL for CEEWDEXT

1. Replace the comment lines following the ++ SRC statement in CEEWDEXT with your updated CEEEXTAN CSECT identifying your abnormal termination exit routine.
2. Change #GLOBALCSI to the data set name of your global CSI data set.
3. Change #TZONE to the name of your target zone.
4. Add necessary SMP/E PRE information for PTFs you have installed on your system which contain the same part.

CEEWDEXT should run with a condition code of 0.

Identifying the Abnormal Termination Exit (CICS)

Use the CEEWCEXT sample job to specify your own abnormal termination exit in a CICS environment.

Modifying the JCL for CEEWCEXT

1. Replace the comment lines following the ++ SRC statement in CEEWCEXT with your updated CEEEXTAN CSECT identifying your abnormal termination exit routine.

2. Change #GLOBALCSI to the data set name of your global CSI data set.
3. Change #TZONE to the name of your target zone.
4. Add necessary SMP/E PRE information for PTFs you have installed on your system which contain the same part.

CEEWCEXT should run with a condition code of 0.

Creating a Load Notification User Exit

The load notification user exit provides customers who are running applications with LRR active the ability to improve performance by preventing the use count for frequently used modules from dropping below one.

See *z/OS Language Environment Programming Guide* for more information on load notification user exit.

To create a load notification user exit:

1. Create an assembler language routine that conforms to the syntax described in *z/OS Language Environment Programming Guide*.
2. Assemble and link-edit your exit into a library that Language Environment can access at run-time, such as CEE.SCEERUN.
3. Code a CEEBLNUE CSECT that contains a CEEXLRT macro identifying your exit. The macro specifies your routine as a load notification user exit. The CEEBLNUE CSECT can be found in source file CEE.SCEESAMP(CEEBLNUE). See “CEEBLNUE CSECT” for more information.
4. Replace the existing CEEBLNUE CSECT with the updated CEEBLNUE as described in the sections below.

Identifying the Load Notification User Exit

Use the CEEWLNUE sample job to specify your own load notification user exit.

Modifying the JCL for CEEWLNUE

1. Replace the comment lines following the ++ SRC statement in CEEWLNUE with your updated CEEBLNUE CSECT identifying your Load Notification User Exit.
2. Change #GLOBALCSI to the data set name of your global CSI data set.
3. Change #TZONE to the name of your target zone.
4. Add necessary SMP/E PRE information for PTFs you have installed on your system which contain the same part.

CEEWLNUE should run with a condition code of 0.

CEEBLNUE CSECT

CEEBLNUE is a CSECT explicitly linked with Language Environment, and it is the CSECT that you create by coding the CEEXLHD, CEEXLRT, and CEEXLST macros. Specifically, CEEBLNUE is linked with the CEEPLPKA module. CEEBLNUE CSECT is created through the use of the following Language Environment-provided assembler macros:

CEEXLHD

Defines the head of the list. CEEXLHD generates the CSECT statement and any header information required. It has no operands.

CEEXLRT

Identifies the name of the exit to register. Only one name can be provided since only one load notification user exit may be registered. CEEXLRT has only one keyword parameter, LOADXIT=, which is the load name for the load notification user exit. There is a limit of 8 characters for the load name, and no validation of the name is performed by the macro.

CEEXLST

Defines the end of the list. CEEXLST generates the trailer for the CEEBLNUE load notification user exit CSECT. It has no parameters.

Language Environment validates the format of the CEEBLNUE CSECT and issues a load of the name as identified in the table. The LOAD is attempted only during region initialization when Library Routine Retention (LRR) is active. If the LOAD is successful, the exit is called for initialization according to the interface described below. If the LOAD is successful, the exit is registered and called during region initialization, after each successful load, and during region termination. This allows a STEPLIB to either contain or omit the load names.

Only one load notification user exit may be registered.

CEEBLNUE Sample

Figure 6 shows the source for the IBM-supplied CEEBLNUE CSECT. It is provided in the CEE.SCEESAMP data set.

```
*/*****  
*/  
*/*   LICENSED MATERIALS - PROPERTY OF IBM   */  
*/*   5645-001 5688-198                       */  
*/*   (C) Copyright IBM Corp. 1991, 1997     */  
*/*   All Rights Reserved                     */  
*/*   US Government Users Restricted Rights - Use, duplication or */  
*/*   disclosure restricted by GSA ADP Schedule Contract with IBM */  
*/*   Corp.                                   */  
*/*   Status = HMWL810                         */  
*/*****  
CEEXLHD      ,User exit header  
*=====  
*                                           *  
*   To specify a load notification user exit,   *  
*   change the line where CEEXLRT is specified, *  
*   by doing the following:                   *  
*                                           *  
*   1. Change XXXXXXXX to the name of your load notification *  
*       user exit module name. This name must not be longer *  
*       than 8 characters.                       *  
*                                           *  
*   2. Change the asterisk (*) in column 1 to a blank.     *  
*                                           *  
*=====  
*   CEEXLRT LOADXIT=XXXXXXX  
*   CEEXLST      ,Terminate the list
```

Figure 6. Sample of CEEBLNUE Load Notification User Exit CSECT

Creating a Storage Tuning User Exit

The storage tuning user exit provides a programming interface that allows you to collect Language Environment storage tuning information and to set the Language Environment run-time option values for STACK, LIBSTACK, HEAP, ANYHEAP and BELOWHEAP. See *z/OS Language Environment Programming Guide* for more information.

The storage tuning user exit is available on CICS and on non-CICS environments when LRR is used.

To create a storage tuning user exit on CICS:

1. Create an assembler language routine that conforms to the syntax described in *z/OS Language Environment Programming Guide*.
2. Translate your exit with the CICS translator. The SYSEIB translator option must be used.
3. Assemble and link-edit your exit into a library that is in the CICS DFHRPL DD concatenation. The member name of the exit must be CEECSTX.
4. Define program CEECSTX to CICS with LANGUAGE(ASSEMBLER). The definition for the program must be available at CICS start-up.

To create a storage tuning user exit on non-CICS:

1. Create an assembler language routine that conforms to the syntax described in *z/OS Language Environment Programming Guide*.
2. Assemble and link-edit your exit into a library that Language Environment can load at run-time. The member name of the exit must be CEEBSTX.

Chapter 6. Customizing the Cataloged Procedures

You can tailor the cataloged procedures supplied with Language Environment to suit the needs of your site. The procedures are part of the SCEEPROC cataloged procedure library.

You can tailor any of the following:

- If your site uses a prefix other than the IBM-supplied one, you can modify the data set name prefixes by using the LIBPRFX parameter.
- If you place CEE.SCEERUN and CEE.SCEERUN2 in the LNKLSTxx concatenation during installation, remove the STEPLIB DD statements.

Note: Since SCEERUN2 contains module names that do not intersect with any pre-Language Environment run-time library or any existing library, IBM recommends that SCEERUN2 be added to the LNKLST. This will not result in any adverse effects.

- If most of the programs at your site require a larger region for successful execution, change the default region size for the GO steps.
- Change UNIT=SYSDA in CEEWL, CEEWLG, AFHWL, AFHWLG, AFHWN, AFHWRL, and AFHWRLG.
- Tailor your TSO/E LOGON procedure. If you plan to run Language Environment applications under TSO/E, add SCEERUN and SCEERUN2 to the STEPLIB DD of the LOGON procedure, or use the TSO/E command TSOLIB to allocate SCEERUN and SCEERUN2; this is unnecessary if you place SCEERUN and SCEERUN2 into the LNKLST concatenation during installation. For more information, see *TSO/E Command Language Reference*.
- For programs that require the Language Environment Prelinker Utility, see *z/OS Language Environment Programming Guide* and *z/OS C/C++ Programming Guide* for details on changes to link-edit procedures. The requirement to use the Prelinker has been eliminated since the DFSMS/MVS Program Management Binder directly supports input from the Language Environment conforming compilers. By choosing to eliminate usage of the Prelinker, the executable program will be a program object and must reside either in a PDSE or an HFS.

Making the cataloged procedure library available to your jobs

Language Environment is shipped with a procedure library, CEE.SCEEPROC, that contains several procedures that can be used during application development with Language Environment. These procedures are summarized in Table 10:

Table 10. Language Environment Invocation Procedures in CEE.SCEEPROC

Procedure	Purpose
AFHWL	Link-edit a Fortran program.
AFHWLG	Link-edit and run a Fortran program.
AFHWN	Change any external names in conflict between C and Fortran to the Fortran-recognized name.
AFHWRL	Separate the nonshareable and shareable parts of a Fortran object module, and link-edit.
AFHWRLG	Separate the nonshareable and shareable parts of a Fortran object module, link-edit, and execute.

Table 10. Language Environment Invocation Procedures in CEE.SCEEPROC (continued)

Procedure	Purpose
CEEWG	Load and run a non-XPLINK Language Environment-conforming application.
CEEWL	Link-edit a non-XPLINK Language Environment-conforming application.
CEEWLG	Link-edit and run a non-XPLINK Language Environment-conforming application.
CEEXL	Link-edit an XPLINK Language Environment-conforming application.
CEEXLR	Link-edit and run an XPLINK Language Environment-conforming application.
CEEXR	Load and run an XPLINK Language Environment-conforming application.
EDCDLLRN (alias of CRTCP001)	Invoke the DLL RENAME Utility.
EDCGNXL (alias of EDC4P006)	Read a genx1t file and produce the translation table which is stored in the nominated LOADLIB.
EDCICONV (alias of EDC4P007)	Convert the characters from the input file from a coded character set definition to another character set definition and write the characters to the output file.
EDCLIB (alias of CRTCP002)	Maintain a C/C++ object code library.
EDCPL (alias of EDC4P002)	Prelink and link-edit a C/C++ application.

There are three ways to make the procedures available to your jobs. The method you choose depends on the special requirements and policies at your site. Use Table 11 to choose which method to use at your site.

Table 11. Deciding How to Make Cataloged Procedures Available to Your Jobs

If	Then	Result
You plan to use the IBM-supplied defaults and install into the default private procedure library.	Modify the JES2 start procedure.	Makes all procedures in the libraries available to any job in the system.
You are not using all the defaults and you want to choose which of the procedures to make available to general users.	Copy the procedures into a system or private PROCLIB.	Makes the procedures available to your installation jobs.
You are not using all the defaults.	Use the procedures as inline procedures.	Inserts the appropriate procedure into each job.

How to Do It

Modifying the JES2 Start Procedure

You can do EITHER of the following:

- Add a new //PROCnn DD statement for the Language Environment procedure library, CEE.SCEEPROC.
- Concatenate the procedure library to the //PROC00 DD statement.

While testing, you can use the `/*JOBPARM` statement with the `PROCLIB=` parameter to make sure that your jobs use procedures from the correct library. To learn how to do this, see the section on JES2 control statements in *z/OS MVS JCL Reference*.

All procedures in the libraries that are added to the JES2 start procedure are available to any job in the system. The JES2 procedure is usually member JES2 in SYS1.PROCLIB.

Placing Cataloged Procedures in a System or Private PROCLIB

Copy the system procedures from the default libraries into an already-cataloged procedure library. You can use SYS1.PROCLIB as your cataloged procedure library.

The copied procedures are callable by your installation jobs. However, procedures copied into a PROCLIB outside of SMP/E control are more difficult to maintain.

You can use the JCLLIB statement to specify a private PROCLIB. Do this by including the following statement after the JOB card and before the first EXEC statement in the job: `//PROCLIB JCLLIB ORDER=(CEE.SCEEPROC)`

Using Cataloged Procedures as Inline Procedures

Modify the procedure to reflect the high level qualifiers you are using for the installation, and save your changes.

Edit each job before you submit it, and copy the procedure into the job (inline).

Be sure to place `// PEND` at the end of the inline procedure.

Tailoring the Cataloged Procedures and CLISTs to Your Site

Several cataloged procedures and CLISTs are supplied with Language Environment and the Language Environment-conforming compilers. Some of these contain data set names that you may need to customize to your installation.

For information to help you customize the Language Environment cataloged procedures, see the topic discussed in Chapter 6, “Customizing the Cataloged Procedures” on page 39 and the list in Table 10 on page 39.

For a list of names and possible modifications of CLISTs and all other cataloged procedures, see Table 12 on page 42.

Several Fortran and C library routines have identical names. To correctly run existing Fortran applications under Language Environment, it is necessary to resolve all name conflicts. The Language Environment interface validation exit is a routine that automatically resolves conflicting library routine references within Fortran routines.

If the possibility exists of bringing in a Fortran routine when link-editing, activate the binder interface validation exit by modifying each of the cataloged procedures in Table 12 on page 42 that performs a link-edit step to add an LKED parm of `EXITS(INTFVAL(CEEPINTV))`, and provide the following DD statement in the same step:

```
//STEPLIB DD DSN=CEE.SCEELKED,DISP=SHR
```

For further information on resolving conflicting names, see *z/OS Language Environment Programming Guide*.

Table 12. Cataloged Procedures and CLISTs Information

Category	Procedure Names	Possible Modifications
C/C++ cataloged procedures	Procedures found in hlq.SCCNPRC data set.	Modify the procedures to use the release of Language Environment you are using.
COBOL cataloged procedures	IGYWC IGYWCG IGYWCL IGYWCLG IGYWCPG IGYWCP IGYWCPG IGYWPL	Modify the procedures to use the release of Language Environment you are using.
PL/I cataloged procedures	IEL1C IEL1CG IEL1CL IEL1CLG	Modify the procedures to use the release of Language Environment you are using.
Language Environment CLISTs	CMOD CPLINK C370LIB GENXLT ICONV DLLRNAME	<ul style="list-style-type: none"> • If you are not using the IBM-supplied default data set prefix, change the data set prefix symbolic parameter in all CLISTs. • Change parameters in CLISTs to match values at your site. • These procedures can be found in the CEE.SCEECLST data set.

Chapter 7. Placing Language Environment Modules in Link Pack and LIBPACK

Placing routines in the LPA/ELPA reduces the overall system storage requirement by making the routines shareable. Also, initialization/termination (init/term) time is reduced for each application, since load time decreases. For example, if Language Environment modules are not placed in LPA/ELPA, then under z/OS UNIX, every `fork()` call will require approximately 4 MB to be copied into the user address space.

The SCEERUN data set has many modules that are not reentrant, so you cannot place the entire data set in the Link Pack Area (LPALSTxx parmlib). As of OS/390 Release 6, there is a data set called SCEELPA that contains a subset of the SCEERUN modules - those that are reentrant, reside above the line, and are heavily used by z/OS itself. If you put the SCEERUN data set in the linklist (LNKLSTxx), you can place the SCEELPA data set in LPA list (LPALSTxx). Doing this will improve performance. However, if you are using RTLS you cannot place the SCEELPA data set as part of your LPALSTxx.

You can not place the SCEERUN2 data set as part of a LPALSTxx because it is a PDSE. You must use the Dynamic LPA capability to move individual members of SCEERUN2 into the Link Pack Area.

You may also add additional modules to the LPA, using the Modify Link Pack Area (MLPA=) option at IPL. As of OS/390 Release 4, you can also use the Dynamic LPA capability (SET PROG=). Using the Dynamic LPA method avoids the performance degradation that occurs with the use of MLPA.

Choose which routines to put in the LPA/ELPA. See Appendix D, "Modules eligible for the link pack area" on page 199 for a complete list of modules you may place in the LPA/ELPA.

Several members are installed in CEE.SCEESAMP for you to use as examples in creating your IEALPAnn or PROGxx member. Table 13 lists the members and their content.

Table 13. Language Environment Sample IEALPAnn or PROGxx Members in CEE.SCEESAMP

Member Name	Description
CEEWLPA	All Language Environment base modules eligible for the LPA except callable service stubs. Uses Dynamic LPA.
EDCWLPA	All C/C++ component modules eligible for LPA from SCEERUN and SCEERUN2. Uses Dynamic LPA.
IGZWMLP4	All Language Environment COBOL component modules eligible for LPA.
IBMALLP2	All Language Environment PL/I component modules eligible for LPA
IBMPLPA1	MLPA macro for VisualAge PL/I
AFHWMLP2	All Language Environment Fortran modules eligible for LPA

If you want to load modules into the LPA, you do not need to place CEE.SCEERUN or CEE.SCEERUN2 in the LNKLSTnn member. However, if CEE.SCEERUN or CEE.SCEERUN2 is not in the LNKLSTnn member, you need to make modules that

are not included in the link pack areas available to your application programs by copying the modules into a data set that can be either included in the LNKLST nn or used as a STEPLIB.

Using the entire CEE.SCEERUN or CEE.SCEERUN2 data set as a STEPLIB defeats the purpose of placing the modules in the LPA.

Shared Storage Considerations:

- Modules you copy into another (non-LPA) data set are not automatically updated by SMP/E when you apply a service update. You must rerun your copy job after you apply service to Language Environment to make the updated modules available in the LNKLST nn data set or in the STEPLIB.
- Examine the lists carefully to make sure that you are installing the correct module for the national language support you have installed. Comments in CEEWLPA, EDCWLPA, and IBMALLP2 identify the Japanese modules. In IGZWMLP4, remove the module name IGZCMGEN if you do not want U.S. English mixed-case to be in the LPA and add IGZCMGJA if Japanese is installed and you want it to be in the LPA.
- For more information on including modules in the LPA, refer to *z/OS MVS Initialization and Tuning Reference*.

Tailoring the Fortran LIBPACKS

The Fortran component of Language Environment is shipped with individual routines and with groupings of routines called LIBPACKS. A LIBPACK is a load module that contains individual library routines packaged together by the linkage editor into a single load module in order to reduce the time that would otherwise be needed to load the individual routines.

You might want to customize the Fortran LIBPACKS to:

- Shorten the load time for the Fortran LIBPACK by reducing its size
- Minimize the virtual storage required for an application by eliminating seldom-used routines from main storage
- Reduce the number of loads for application programs by adding frequently used routines to Fortran LIBPACKS
- Reduce the size of the contents of shared storage

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

Choices to Make Now

You need to decide whether to modify the Fortran LIBPACKS. If you modify the Fortran LIBPACKS, you make a trade-off between use of storage and faster performance of application programs. See Table 14 below.

Table 14. Making the trade-off: Performance time versus storage use

Type of Fortran LIBPACK	Performance Time	Storage Use
Partially loaded	Slower because more routines are loaded individually	Less virtual and shared storage used
Fully loaded	Faster because no routines loaded individually	More virtual and shared storage used

You can use the information in the following sections and the tables in “Language Environment Fortran component modules” on page 202 to decide which modules to include in your Fortran LIBPACKs.

Language Environment provides four Fortran LIBPACKs, which you can customize either during or following the installation of Language Environment.

AFHPRNAG
AFHPRNBG
AFH5RENA
AFH5RENB

After installation, each LIBPACK contains a default set of routines. You can remove many of the routines if their functions aren’t used frequently at your site, or you can add others that you do use frequently.

Some Examples

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

If you plan to put your Fortran LIBPACK into shared storage and your shared storage space is limited, consider reducing the size of your Fortran LIBPACKs. All modules eligible to be in the Fortran LIBPACKs are reentrant and are therefore eligible to be stored in the shared storage.

Listing the contents of Fortran LIBPACKs

Before tailoring your LIBPACKs, you might want to know their current structure, such as which MODs SMP/E expects to be combined into a particular load module, so that you can decide which ones to add or delete. Use SMP/E sample job AFHWLIST in the SCEESAMP data set to invoke the SMP/E LIST command to list the contents of your LIBPACKs.

Steps for modifying the JCL for AFHWLIST

Perform the following steps to modify the JCL for AFHWLIST.

1. Change #GLOBALCSI to the data set name of your global CSI data set.

2. Change #TZONE to the name of your target zone.

3. Examine the LIBPACK names on the SMP/E LIST statement and remove the comments as appropriate.

When you are done, AFHWLIST should run with a condition code of 0.

Deleting Routines from Fortran LIBPACKs

The sample jobs listed in Table 15 on page 46 each contain SMP/E UCLIN and link-edit JCL that you can modify to delete routines to one of the Fortran LIBPACKs. The sample jobs are in target library CEE.SCEESAMP.

Table 15. SMP/E Sample Jobs for Deleting Routines from Fortran LIBPACKs

For Applications Link-Edited With...	Use Sample Job...	To Delete Routines from LIBPACK...	Which Is Loaded ...
Language Environment	AFHWDERA	AFHPRNAG	above 16 MB
Language Environment	AFHWDERB	AFHPRNBG	below 16 MB
VS FORTRAN	AFHWDVRA	AFH5RENA	above 16 MB
VS FORTRAN	AFHWDVRB	AFH5RENB	below 16 MB

If the IBM-supplied LIBPACKs contain routines that your site does not use often, you can delete them using the SMP/E sample jobs below.

Steps for modifying the JCL to delete routines from a Fortran LIBPACK

Perform the following steps to modify the JCL to delete routines from a Fortran LIBPACK. These steps use the AFHWDERA, AFHWDERB, AFHWDVRA, and AFHWDVRB sample jobs.

1. Change #GLOBALCSI to the data set name of your global CSI data set.

2. Change #TZONE to the name of your target zone.

3. Modify the UCLIN step in the sample job to tell SMP/E to delete routines that you do not want to include in your tailored LIBPACK.
 - Remove the DELETE statement of any routine you want to include in your LIBPACK.
 - Remove the DELETE statement of any routine that is not currently in your LIBPACK.
 - If you run any of the sample jobs shown in Table 15 without modifying them, you receive a minimum LIBPACK without any optional modules.

4. The LINK-EDIT step performs the actual link-edit of the tailored LIBPACK by replacing (deleting) the routines you have specified. The REPLACE statements you keep in the LINK-EDIT step must match the routines you specified in the UCLIN step.

When taking out the REPLACE records, ensure that all alias names (shown with indented REPLACE records) are removed too. For example, if you decide to remove AFHBCMVT, you need to remove AFHBCMVR as well.

5. Check the SYSLMOD DD statement to ensure the data set name is correct.

When you are done, FHWDERA, AFHWDERB, AFHWDVRA, and AFHWDVRB should run with a condition code of 4. Unresolved external references for any optional modules not included in your LIBPACK are expected.

Adding Routines to Fortran LIBPACKs

The sample jobs listed in Table 16 on page 47 each contain SMP/E UCLIN and link-edit JCL that you can modify to add routines to one of the Fortran LIBPACKs. The sample jobs are in target library CEE.SCEESAMP.

Table 16. SMP/E Sample Jobs for Adding Routines to Fortran LIBPACKs

For Applications Link-Edited With...	Use Sample Job...	To Add Routines to LIBPACK...	Which is Loaded...
Language Environment	AFHWAERA	AFHPRNAG	above 16 MB
Language Environment	AFHWAERB	AFHPRNBG	below 16 MB
VS FORTRAN	AFHWAVRA	AFH5RENA	above 16 MB
VS FORTRAN	AFHWAVRB	AFH5RENB	below 16 MB

Note:

The jobs that add routines to the LIBPACKs add the versions of the routines that are in the target libraries.

If the IBM-supplied LIBPACKs exclude routines that your site uses often, you can add them using the SMP/E sample jobs that follow.

Steps for Modifying the JCL for adding routines to a Fortran LIBPACK

Perform the following steps to modify the JCL for adding routines to a Fortran LIBPACK. These steps use the AFHWDERA, AFHWDERB, AFHWDVRA, and AFHWDVVB sample jobs.

1. Change #GLOBALCSI to the data set name of your global CSI data set.

2. Change #TZONE to the name of your target zone.

3. Modify the UCLIN step to tell SMP/E to add the routines you want to include in your tailored LIBPACK.
 - Remove the ADD statement for each routine you are not adding to your tailored LIBPACK.
 - If you run the sample jobs shown in Table 16 without modifying them, you receive a full LIBPACK, including all the required and optional LIBPACK modules.
 - If you attempt to add a routine that is already in the LIBPACK, you receive an SMP/E error message.

4. The LINK-EDIT step performs the actual link-edit of the tailored LIBPACK by including the routines you specify.

The INCLUDE statements you keep in the LINK-EDIT step must match the routines you want to include in your tailored LIBPACK, regardless of whether you add the routine in the UCLIN step above or it is already in the LIBPACK.

5. Check the SYSLMOD DD statement to ensure the data set name is correct.

When you are done, AFHWAERA, AFHWAERB, AFHWAVRA, and AFHWAVRB should run with a condition code of 0 if the LIBPACKs contain all of the optional modules. Otherwise, each of these jobs returns a condition code of 4; unresolved external references for any optional modules not included in the LIBPACKs are expected.

Where to Place the Tailored Fortran LIBPACKS

The sample jobs tailor the LIBPACKS and then use them to replace the LIBPACKS in the Language Environment target library SCEERUN. You could place them in another data set instead, provided that the LOADs issued during run-time can find them. The customized LIBPACKS must be found ahead of (in search-order sequence or in library concatenation), or instead of, those that were installed with the product. If you want to link-edit a LIBPACK into an alternative library, modify and run only the LINK-EDIT step of the sample jobs.

Note: Because SMP/E is only aware of the load modules link-edited into the SCEERUN target library, SMP/E will **not** relink your LIBPACKS automatically when you apply service if you use an alternative library.

Chapter 8. Using Language Environment under CICS

To make sure that CICS can communicate with Language Environment:

- Add the Language Environment required program resource definitions to the CICS System Definition (CSD) file, unless you are using autoinstall for programs; in which case CICS creates the required entries dynamically.
- Ensure the required transient data (TD) queue resource definitions are defined to CICS.

Note: If the resource definitions are already defined in the CSD by the CICS utility, you should ensure they are not removed from the CICS group list used at startup.

- Add the Language Environment Library data sets to the CICS startup job stream.

Add program resource definitions for CICS

Update the CICS system definition (CSD) file using the program definitions in the CEECCSD member in the Language Environment sample (SCEESAMP). This member contains the necessary input to the CSD file utility program to define the Language Environment library routines to the CSD. The CSD group list used during CICS startup must include the CSD group associated with the Language Environment library routines. The group name for Language Environment routines is CEE in the sample CEECCSD.

COBOL Users: The OS/VS COBOL library routines (ILBOs) in Language Environment library SCEERUN are loaded by the operating system and do not require entries in the CSD.

If you are using CICS 4.1 or higher, and the autoinstall option is active, you do not need to specify the Language Environment library routines in the CSD. CICS creates the required entries dynamically. Therefore, you do not need to use this sample job. If you plan to run with program autoinstall and use the Language Environment CLER transaction, you need to define the following using the CEDA transaction:

```
DEFINE PROGRAM(CEL4RT0) GROUP(CEE) LANGUAGE(ASSEMBLER)
EXECKEY(CICS)
DEFINE MAPSET(CELCLEM) GROUP(CEE)
DEFINE MAPSET(CELCLRH) GROUP(CEE)
DEFINE TRANS(CLER) PROG(CEL4RT0) GROUP(CEE)
```

Tip: If you use program autoinstall, Language Environment event handler modules in the range CEEEV001-CEEEV017 that are present in the CEE.SCEERUN might load during CICS/LE initialization. To prevent this from occurring, you should update the CICS CSD with the Language Environment resource definitions in CEECCSD and add code to your autoinstall exit to bypass autoinstall for all CEEEV0* modules. The following autoinstall exit sample demonstrates this:

```
DFHPGADX CSECT
DFHPGADX AMODE 31
DFHPGADX RMODE ANY
DFHREGS ,
*
*       If there is no commarea, return
OC    EIBCALEN,EIBCALEN
BZ    RETURN0
*
*       Address the commarea
```

```

L      R2,DFHEICAP
USING PGAC,R2
*
*      Omit autoinstall for Language Environment modules
CLC   PGAC_PROGRAM(6),=C'CEEEV0'
BE    RETURNDD
*
*      Add user specific code here
*
*      Set the return code to OK
RETURNOK DS   0H
MVI   PGAC_RETURN_CODE,PGAC_RETURN_OK
B     RETURN0
*
*      Branch to this label if you elect not to define
*      the program
RETURNDD DS   0H
MVI   PGAC_RETURN_CODE,
PGAC_RETURN_DONT_DEFINE_PROGRAM
*
RETURN0 DS   0H
EXEC  CICS RETURN,
END   DFHPGADX

```

Table 17. Excluding Programming Language Support under CICS

If you do not run...	Exclude these program definitions from the CEECCSD sample job...
COBOL applications under CICS	CEEEV005, IIGZMSGT, All programs that start with IGZ
C/C++ applications under CICS	CEEEV003, IEDCMSGT, All programs that start with EDC or CEU
PL/I applications under CICS (Also VA PL/I)	CEEEV010, CEEEV011, IIBMSGT, All programs that start with IBM

Guideline: If you use autoinstall and want to exclude one or more languages using this technique, be sure to implement the changes described above in your autoinstall exit to prevent them from being added dynamically.

Note: C was named AD/Cycle C/370 before C++ was added. The sample JCL used the nickname C/370 to refer to either Language Environment-enabled version.

Add destination control table (DCT) entries

The CEECDCT member in the SCEESAMP sample library contains the necessary input to create the transient data queues as extrapartition data queues.

Entries for the transient data queues used by Language Environment are required in the destination control table. Language Environment uses the following transient data queues:

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

- CIGZ: COBOL side file support for CEEDUMPs and Debug Tool. The definition for this queue is required only if you run COBOL programs compiled with the SEPARATE suboption of the TEST compiler option. This is an input-only queue.

In order to use the COBOL side file support on CICS for COBOL programs compiled with the TEST(,SYM,SEPARATE) compiler option, you must define a transient data queue with the name CIGZ. Do not specify a DD for the CIGZ transient data queue in your CICS startup job. The DD will be dynamically allocated and deallocated as needed.

The following is the source that can be used to define CIGZ in the DCT:

```
IGZDBGIN DFHDCT TYPE=SDSCI,          COBOL Side File Support
          DSCNAME=IGZDBGIN,
          TYPEFLE=INPUT
CIGZ     DFHDCT TYPE=EXTRA,          COBOL Side File Support
          DESTID=CIGZ,
          DSCNAME=IGZDBGIN,
          OPEN=DEFERRED
```

Figure 7 illustrates the format for the output transient data queues.

ASA	Terminal ID	Transaction ID	sp	Timestamp YYYYMMDDHHMMSS	sp	Message
1	4	4	1	14	1	132

Figure 7. Format of an Output Transient Data Queue

ASA The American National Standard carriage-control character

Terminal ID

A 4-character terminal identifier

Transaction ID

A 4-character transaction identifier

sp A space

Timestamp

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

Message

The message identifier and message text

These queues can have intrapartition, extrapartition, or indirect destinations. The record length for the transient data queue CESE must be at least 161.

We recommend that you put the required Language Environment entries in the CSD as TDQUEUE resource definitions (introduced in the CICS Transaction Server for z/OS). The Language Environment TD queues are included in the CICS-supplied CSD group called DFHDCTG, which is added to the DFHLIST automatically when initializing or upgrading a CSD. The following are the Language Environment entries created in the DFHDCTG:

```
DEFINE TDQUEUE (CES0)          GROUP(DFHDCTG)
          DESCRIPTION(LE/370 OUTPUT QUEUE)
          TYPE(EXTRA)          TYPEFILE(OUTPUT)
          RECORDSIZE(133)      BLOCKSIZE(137)
          RECORDFORMAT(VARIABLE) BLOCKFORMAT(UNBLOCKED)
```

```

                                DDNAME(CEEOUT)
*
DEFINE TDQUEUE (CESE)          GROUP(DFHDCTG)
                                DESCRIPTION(LE/370 ERROR QUEUE)
                                TYPE(EXTRA)
                                RECORDSIZE(161)
                                RECORDFORMAT(VARIABLE)
                                TYPEFILE(OUTPUT)
                                BLOCKSIZE(165)
                                BLOCKFORMAT(UNBLOCKED)
                                DDNAME(CEEOUT)
*
DEFINE TDQUEUE (CIGZ)          GROUP(DFHDCTG)
                                DESCRIPTION(COBOL SIDE FILE INPUT QUEUE)
                                TYPE(SDCI)
                                TYPEFILE(INPUT)
                                DDNAME(IGZDBGIN)

```

See *CICS System Definition Guide* for information provided by CICS about installing Language Environment support.

If you are using CICS 4.1 or later, use the DFHDCT macro to define the entries for CESE, CESO and CIGZ.

In addition to defining the transient data queues in the DCT, you must make sure that there is a DD statement in the CICS startup job for the transient data queues.

Note: Do not specify a DD for the CIGZ TDQ. It will be dynamically allocated and deallocated as needed.

If you define the CESE and CESO transient data queues as separate extrapartition data queues, the following example shows what you would specify in your CICS startup JCL:

```

//CEEMSG DD DSN=CUSTOMER.CEEMSG,DISP=SHR
//CEEOUT DD DSN=CUSTOMER.CEEOUT,DISP=SHR

```

If you are using CICS 4.1 or later, *CICS Resource Definition (Macro)*, contains further information on the DFHDCT macro and the definitions of the queues and associated buffers.

For other levels of CICS, see *CICS Resource Definition Guide* and *CICS Resource Definition Guide for CICS/ESA 4.1*.

CEEMSG	DFHDCT	TYPE=SDSCI, Language Environment messages, dumps, reports	
		DSCNAME=CEEMSG,	X
		BLKSIZE=165,	X
		RECSIZE=161,	X
		RECFORM=VARUNBA,	X
		TYPEFLE=OUTPUT,	X
		BUFNO=1	
CESE	DFHDCT	TYPE=EXTRA,	X
		DESTID=CESE,	X
		DSCNAME=CEEMSG	
CEEOUT	DFHDCT	TYPE=SDSCI, C/C++ STDOUT stream	X
		DSCNAME=CEEOUT,	X
		BLKSIZE=137,	X
		RECSIZE=133,	X
		RECFORM=VARUNBA,	X
		TYPEFLE=OUTPUT,	X
		BUFNO=1	
CESO	DFHDCT	TYPE=EXTRA,	X
		DESTID=CESO,	X
		DSCNAME=CEEOUT	
IGZDBGIN	DFHDCT	TYPE=SDSCI, COBOL Side File Support	X
		DSCNAME=IGZDBGIN,	X
		TYPEFLE=INPUT	
CIGZ	DFHDCT	TYPE=EXTRA, COBOL Side File Support	X
		DESTID=CIGZ,	X
		DSCNAME=IGZDBGIN	X
		OPEN=DEFERRED	

Note: Xs are in column 72.

Figure 8. Example of DFHDCT Macro

When DFHDCT encounters the entry names CESE, CESO, CIGZ, CEEMSG, and CEEOUT, it might generate messages stating that queue names beginning with the letter 'C' are reserved for CICS. It is normal to receive these messages, and they do not indicate errors.

Add Language Environment-CICS Data Sets to the CICS Startup Job Stream

Before running any CICS transactions under Language Environment, you must add Language Environment to the startup job stream. *CICS System Definition Guide* describes the CICS system startup procedure and provides an example of a CICS startup job stream.

To add the Language Environment-CICS data sets to CICS:

- Update the DFHRPL DD concatenation.

Add the Language Environment run-time library SCEERUN in the DFHRPL DD concatenation of the job that is used to start CICS.

If you are running COBOL programs on CICS, you must also add Language Environment run-time library SCEECICS in the DFHRPL DD concatenation. The SCEECICS library **must** be concatenated before the SCEERUN library.

Any libraries that contain run-time routines from earlier versions of COBOL, PL/I, and C/C++ should be removed from the DFHRPL DD concatenation.

- If SCEERUN is not in LNKST or LPALST, then you will also need to include the proper Language Environment routines into an authorized library that is part of the STEPLIB DD concatenation in the CICS startup job. If SCEERUN is in LNKLST/LPALST, then you do not have to add SCEERUN to the STEPLIB DD concatenation in the CICS startup job. You can either:

1. Authorize the Language Environment run-time library SCEERUN and then include it in the STEPLIB DD concatenation in the CICS startup job. (The SCEERUN2 data set does not need to get added to this concatenation.)
2. Put only those Language Environment routines needed by CICS using the STEPLIB into another library.

If you use the second method, you must make the following Language Environment routines available by using the STEPLIB:

- CEECCICS, CEECTCB
- IGZCWTO: Used for COBOL support.
- IGZCMTUE: Used for COBOL support.
- IGZIDYN: Used for COBOL support.
- ILBO routines: If you are running OS/VS COBOL programs, all of the ILBO routines must be available.

You should remove any libraries that contain run-time routines from earlier versions of COBOL and C/370 from the STEPLIB DD concatenation.

Notes:

1. The previously mentioned library routines required from the STEPLIB might also be available by using the JOBLIB or the LNKLSTnn member.
2. There is no CICS startup option for Language Environment. If CICS locates CEECCICS, it attempts to initialize Language Environment. If the modules have not been installed correctly, Language Environment initialization fails, and CICS generates an error message to that effect.

Language Environment automatic storage tuning for CICS

Language Environment automatic storage tuning for CICS provides automatic storage tuning of Language Environment STACK, LIBSTACK, HEAP, BELOWHEAP and ANYHEAP initial size values. Automatic storage tuning of the Language Environment storage areas can improve the performance of applications running on CICS by reducing the CICS GETMAIN/FREEMAIN activity associated with acquiring Language Environment stack and heap increments. In order to use Language Environment automatic storage tuning for CICS, the CICS system initialization parameter AUTODST must be set to YES. The CICS system initialization parameter AUTODST is available only on:

- CICS Transaction Server Version 1 Release 3 with APARs PQ39052, PQ45031, and PQ55351.
- CICS Transaction Server Version 2.

Note: When Language Environment Automatic Storage Tuning for CICS is used, the capability of the storage tuning user exit is changed. For example, the storage tuning user exit can no longer get storage information. See *z/OS Language Environment Programming Guide* for information about the Language Environment storage tuning user exit.

Enclaves eligible for automatic storage tuning

When running with Language Environment automatic storage tuning for CICS, the actual storage tuning is performed for Language Environment enclaves when one of the following conditions is met:

- The main program is not link-edited with a CEEUOPT.
- The main program is link-edited with a CEEUOPT, and the CEEUOPT does not specify values for any of the following run-time options: STACK, LIBSTACK, HEAP, BELOWHEAP or ANYHEAP.

Notes:

1. A CEEUOPT is present in C/C++ main programs that use #pragma runopts when one of the following compilers were used: z/OS C/C++, OS/390 C/C++, C/C++ Compiler for MVS/ESA, or AD/Cycle C/370.
2. A CEEUOPT is present in PL/I main programs that use PLIXOPT when one of the following compilers are used: VisualAge PL/I for OS/390 or PL/I for MVS & VM.

Automatic storage tuning behavior

Automatic storage tuning values are managed for each load module that is used to start an enclave for Language Environment. For example, transaction ATMW starts program COBOLA (which starts an enclave for Language Environment). COBOLA does a CICS LINK to COBOLB which starts another Language Environment enclave. COBOLB does a dynamic call to COBOLC (when a dynamic call is done, we are still running in the same enclave). In this example, automatic storage tuning will be done for the enclaves started for COBOLA and COBOLB.

When running with Language Environment automatic storage tuning for CICS, Language Environment continuously monitors the amount of Language Environment storage allocated in the enclave for STACK, LIBSTACK, HEAP, ANYHEAP, and BELOWHEAP. When the enclave ends normally, Language Environment will automatically increase the initial size values for STACK, LIBSTACK, HEAP, ANYHEAP, and BELOWHEAP as determined by the amount of storage allocated.

In more detail, Language Environment automatic storage tuning for CICS behaves as follows:

- When a main program starts a Language Environment enclave the first time in a CICS region and the enclave is eligible for automatic storage tuning, Language Environment will use the values for STACK, LIBSTACK, HEAP, ANYHEAP, and BELOWHEAP from the normal search order for run-time options. When a main program starts an eligible enclave a subsequent time, Language Environment will use the initial sizes for STACK, LIBSTACK, HEAP, ANYHEAP, and BELOWHEAP as determined by automatic storage tuning.
- Whenever a Language Environment enclave is initialized and it is eligible for automatic storage tuning, Language Environment will collect the total amount of storage allocated for STACK, LIBSTACK, HEAP, ANYHEAP, and BELOWHEAP while the enclave is active.

Note: When Language Environment automatic storage tuning for CICS is used, Language Environment collects the amount of storage allocated. It does not collect the amount of storage used.

- When the enclave ends with an unhandled condition, Language Environment does not update the automatic storage tuning values. When the enclave ends normally, Language Environment automatic storage tuning will increase the initial size for STACK, LIBSTACK, HEAP, ANYHEAP, and BELOWHEAP to the amount of storage allocated only when the amount of storage allocated is larger than the initial size. The next time the main program initiates a Language Environment enclave, Language Environment automatic storage tuning will use the updated initial size values.
- Language Environment automatic storage tuning never decreases the initial size values.

Altering the automatic storage tuning behavior

To alter the behavior of the Language Environment automatic storage tuning for CICS, the storage tuning user exit can be used. For example, the storage tuning user exit can be used as follows:

- To apply its own logic and determine which programs are eligible for automatic storage tuning.
- To set limits on the initial sizes used by Language Environment automatic storage tuning for CICS.

See *z/OS Language Environment Programming Guide* for information about Language Environment storage tuning user exit.

Chapter 9. Using Language Environment under IMS

If you are running programs that require Language Environment in an IMS/TM dependent region, such as an IMS message processing region, you can improve performance if you use Language Environment library routine retention.

With library routine retention in effect, Language Environment keeps certain resources in memory when an application program ends, making subsequent invocations of programs that use Language Environment much faster because the Language Environment resources left in memory are reused.

Following is a partial list of the resources Language Environment keeps in memory with library routine retention in effect:

- Language Environment run-time load modules
- Language Environment storage associated with the management of the run-time load modules
- Language Environment storage for start-up control blocks

Initializing Library Routine Retention

To use Language Environment library routine retention in an IMS dependent region, you must do the following:

1. In your JCL or procedure used to bring up IMS dependent regions, specify that you want IMS to invoke dependent region preinitialization routines. Do this by specifying a suffix on the PREINIT keyword of the IMS dependent region procedure.
2. In the DFSINTxx member of IMS.PROCLIB (where xx is a suffix specified by the PREINIT keyword), include the name CEELRRIN.

When the module CEELRRIN is invoked by IMS, Language Environment library routine retention is initialized.

Note: The source for module CEELRRIN is available in the SCEESAMP library in member CEELRRIN. If this source does not meet your needs, you can create your own assembler program to initialize Language Environment library routine retention. If you create your own load module to initialize Language Environment library routine retention, you will need to put the name of the module in the DFSINTxx member.

Terminating Library Routine Retention

Language Environment provides a routine called CEELRRTR to terminate library routine retention. However, this routine does not need to be used when running on IMS/TM. If library routine retention is initialized, and the IMS Program Control Task is terminated (for example, due to an ABEND), the operating system will free the Language Environment resources as part of task termination. Then when the IMS Program Control Task is reattached, the preinitialization routines get control before IMS scheduling is attempted.

For more information about specifying IMS dependent region preinitialization routines, see *IMS/ESA Customization Guide: System*. For more information about Language Environment library routine retention, see *z/OS Language Environment Programming Guide*.

Chapter 10. Customizing Language-Specific Features

In addition to tailoring your Fortran LIBPACKs, you may want to customize COBOL, C/C++, Fortran, and PL/I features in order to tune or diagnose the performance of Language Environment for your site.

Choices to Make Now

First, decide which language-specific features you should modify for your site. For more detailed information about the C/C++, Fortran, and PL/I features you can customize, see:

- Appendix C, “Using IBM C/C++ with Language Environment” on page 195
- Appendix B, “Using Fortran with Language Environment” on page 163
- *PL/I for MVS & VM Compiler and Run-Time Migration Guide*

You also need to choose which sample customization jobs you will need to modify and run. Table 18 lists the sample jobs provided on the distribution tape to help you customize COBOL, C/C++, Fortran, and PL/I features. These jobs are part of Language Environment sample library SCEESAMP.

Table 18. Customizing Programming Languages with Sample Customization Jobs

To...	Use This Sample Job
Modify the OS/VS COBOL compatibility library routines	IGZWZAP
Modify the COBOL Reusable Environment	IGZWARRE
Customize the parameter list processing when a COBOL program is invoked with an ATTACH SVC on z/OS	IGZWAPSX
Customize the C/C++ locale time information	EDCLLOCL
Relink OS PL/I Version 2 shared library and OS PL/I Version 1 CICS or tasking shared library	IBMRLSLA
Relink OS PL/I Version 1 non-CICS and non-tasking shared library	IBMRLSLB
Tailor the Language Environment Fortran Unit Attribute Table	AFHWEUAT
Tailor the VS FORTRAN compatibility Unit Attribute Table	AFHWVUAT
Tailor the VS FORTRAN compatibility run-time options defaults	AFHWVPRM
Tailor the VS FORTRAN compatibility Error Option Table	AFHWVOPT

Modifying the OS/VS COBOL compatability library routines

Use the IGZWZAP sample job to modify the OS/VS COBOL compatability library routines. The job lets you apply superzaps to make Language Environment COBOL behave like OS/VS COBOL. See Table 19 for a summary of the modifications you can make with the job. “OS/VS COBOL Considerations” on page 60 explains the superzaps in detail.

Table 19. Using the USERMODs in the IGZWZAP Job to Modify the COBOL Compatibility Library

USERMOD...	Contains superzap(s) to...	For...
IGZWZA1	Continue to force USER ABEND 0100, 0201, 0303, or 0304 and message IFK302I	Certain error situations during VSAM file processing

Table 19. Using the USERMODs in the IGZWZAP Job to Modify the COBOL Compatibility Library (continued)

USERMOD...	Contains superzap(s) to...	For...
IGZWZA2	Force USER ABEND 0295	A serious error detected at run-time
IGZWZA3	Add A, B, and E as valid numeric signs	The IF NUMERIC CLASS TEST

Modifying the JCL for IGZWZAP

1. Change #GLOBALCSI to the data set name of your global CSI data set.
2. Change #TZONE to the name of your target zone.

IGZWZAP should run with a condition code of 0.

OS/VS COBOL Considerations

If the COBOL programmers at your site are familiar with OS/VS COBOL, you may want to modify Language Environment COBOL to make it behave like the OS/VS COBOL run-time. The IGZWZAP member is a sample job provided in CEE.SCEESAMP to apply USERMODs IGZWZA1, IGZWZA2, and IGZWZA3, which are described below. For instructions on modifying the JCL for the IGZWZAP job, see “Modifying the OS/VS COBOL compatability library routines” on page 59.

User modifications for the OS/VS COBOL library also apply for the OS/VS COBOL compatability library routines.

VSAM Considerations

Support for VSAM processing in OS/VS COBOL Release 2 and in the OS/VS COBOL compatability library routines is consistent with the I/O language specified in the COBOL standard, AMERICAN NATIONAL STANDARD (ANS) COBOL, X3.23-1974. However, OS/VS QSAM and VSAM support in OS/VS COBOL Release 1 is not consistent with the standard.

File Status: The FILE STATUS clause is optional. Specifying FILE STATUS for a VSAM file lets you monitor the status of the file's I/O operations by testing the FILE STATUS values. Code the FILE STATUS clause for all appropriate files and test the FILE STATUS (status key) after each input/output statement, including the OPEN statement. FILE STATUS detects error conditions so you can handle them before processing continues.

If you don't specify FILE STATUS and test for the appropriate status key values, you could get undetected errors and erroneous program results.

User Abends: In certain error situations during VSAM file processing, Release 1 of the OS/VS COBOL library modules forced user abends during program execution. OS/VS COBOL Release 2 support eliminated four of these user abends. In place of the abends, a FILE STATUS value is set when an I/O operation fails, and execution continues.

Status key values are set to 90, 93, 95, or 95 rather than the forced USER ABEND 0100, 0201, 0303, or 0304, respectively. The program should test the status key value after each I/O operation to make sure its successful completion. OS/VS COBOL Release 2 support also no longer issues the object-time message 'IKF302I'. In place of this message, the FILE STATUS is set to a value of 30.

Because some users might depend on the previous abends and message, you can apply superzaps as user modifications to continue to force USER ABEND 0100, 0201, 0303, or 0304, and continue to force message IKF302I. The IGZWZA1 USERMOD in the IGZWZAP sample job contains the superzaps to do this.

JOB STEP ERROR COMPLETION CODE (RC12/ABEND U0295)

In OS/VS COBOL, if a COBOL library subroutine detects a serious error at execution time (for example, a SYSOUT DD statement is missing), ILBOSRV1 sets the return code and the JOB STEP COMPLETION/ CONDITION CODE to 12 (CC12) upon terminating the run unit. A return code of 12 is compatible with versions 2 and 3 of ANS COBOL.

If you want to change the default return code, you can overlay the halfword X'000c' at displacement X'0002' into CSECT ILBOSRV with the error completion code of your choice. If the halfword is set to a NEGATIVE value during STOP RUN or GOBACK processing, the program is terminated with the USER ABEND 0295 (ABENDU0295) instead of a return code 12.

Because some users might depend on programs abending in the above conditions, you can apply the superzap as a user modification (IGZWZA2) to force a USER ABEND 0295.

IF NUMERIC CLASS TEST allows only C, D, and F

A, B, and E were valid signs for an IF NUMERIC compare in OS/VS COBOL Release 1, but the current release allows only C, D, and F as valid signs for an IF NUMERIC compare. Because some users might depend on the COBOL NUMERIC CLASS TEST, which includes A, B, and E as valid numeric signs, you can apply a provided superzap (IGZWZA3) as a user modification to add A, B, and E as valid numeric signs.

In any case, incorrect data in a data item used for a numeric class comparison is accepted as valid if its hexadecimal notation contains a valid sign. (For example, EBCDIC 'A', or X'C1', is a valid numeric sign for external decimal; and EBCDIC '%', or X'6C', is a valid numeric sign for internal decimal.)

Modifying the COBOL Parameter List Exit

The COBOL parameter list exit routine IGZEPSX can be modified to alter the parameter list processing when a COBOL main program is invoked by an z/OS ATTACH.

With the IBM supplied default COBOL parameter list exit, if the COBOL main is invoked by using the ATTACH SVC, a halfword-prefixed string is passed to the application after run-time options have been removed. The source of this string is dependent on the environment in which the ATTACH is issued:

- If the ATTACH is issued by z/OS to invoke a batch program, the string is specified using the PARM field of the EXEC statement.
- If the ATTACH is issued by TSO/E to attach a Command Processor (CP), the string is specified as part of the command embedded within the CP parameter of the TSO/E ATTACH CP command.
- If the ATTACH is not issued by z/OS or TSO/E, the string is specified using the PARM field of the ATTACH macro.

If the default behavior does not meet your needs, the COBOL parameter list exit IGZEPSX can be altered to set the parameter list processing so that R1 and the parameter list is passed without change to the main COBOL program.

Use the IGZWAPXS sample job to change the COBOL parameter list exit. You must replace the comment in IGZWAPXS with your source for IGZEPSX. You can copy the source for the IBM-supplied default COBOL parameter list exit from IGZEPSX in SCEESAMP and modify it to suit your needs. Included in IGZEPSX is sample code that can be used to get the same parameter list processing that is done when running COBOL programs with the VS COBOL II run-time library.

Modifying the JCL for IGZWAPXS:

1. Replace the comment lines following the ++ SRC statement in the job with your source program for the COBOL parameter list exit.
2. Change #GLOBALCSI to the data set name of your global CSI data set.
3. Change #TZONE to the name of your target zone.

IGZWAPXS should run with a condition code of 0.

Modifying the COBOL Reusable Environment

The COBOL reusable environment behavior can be modified to control how program checks that occur in the non-Language Environment-conforming driver are handled, as well as to control whether or not COBOL programs can run in a nested enclave in the reusable environment. The COBOL reusable environment is established with the RTEREUS run-time option or a call to either ILBOSTPO or IGZERRE INIT.

With the IBM supplied default setting for COBOL's reusable environment behavior (IGZERREO with REUSENV=COMPAT), when a program check occurs while the reusable environment is dormant (for example, between a GOBACK from a top level COBOL program to the non-Language Environment conforming assembler driver and the next call to a COBOL program), a S0Cx abend will occur. This behavior is compatible with the VS COBOL II and OS/VS COBOL run-times, but it significantly impacts the performance when an Enterprise COBOL for z/OS and OS/390, COBOL for OS/390 & VM, or COBOL for MVS & VM program is invoked repeatedly in a COBOL reusable environment. The performance degradation is caused by Language Environment issuing an ESPIE RESET when the reusable environment becomes dormant and then an ESPIE SET upon reentering the reusable environment.

COBOL's reusable environment behavior can be modified (IGZERREO with REUSENV=OPT) so that all program checks will be intercepted by Language Environment, even those that occur while the reusable environment is dormant. In this case, a program check that occurs while the reusable environment is dormant will result in a 4036 abend from Language Environment. However, since Language Environment does not have to issue the ESPIE RESET and ESPIE SET between invocations of the COBOL program, this can be faster than using REUSENV=COMPAT.

Nested Enclave Behavior

With the IBM-supplied default setting for COBOL's reusable environment behavior (IGZERREO with NESTENC=NO), when a reusable environment is active and a nested enclave is created that contains a COBOL program, COBOL will diagnose this with error message IGZ0168S.

COBOL's reusable environment behavior can be modified (IGZERREO with NESTENC=YES) so that a nested enclave containing a COBOL program will continue to run, even though a reusable environment is still active in the parent enclave.

- When you run a COBOL program in a nested enclave.
- The COBOL program is not part of the reusable environment.
- When the nested enclave ends, all the resources associated with the nested enclave are freed.

If a STOP RUN is done in the nested enclave, it only terminates the nested enclave, and does not terminate the COBOL reusable environment.

Modifying the Behavior of the COBOL Reusable Environment

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

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

To run with optimum performance (for example Language Environment intercepts all program checks that occur when the COBOL reusable environment is dormant and converts them to a 4036 abend, resulting in improved performance), use **IGZRREOP REUSENV=OPT**

To disable nested enclave support in the reusable environment, use **IGZRREOP NESTENC=NO**

To enable nested enclave support in the reusable environment, use **IGZRREOP NESTENC=YES**

Modifying the JCL for IGZWARRE

1. Copy the IGZERREO member from CEE.SCEESAMP into IGZWARRE in place of the comment lines following the ++ SRC statement.
2. Change the REUSENV and NESTENC parameters on the IGZRREOP macro statement to the desired value.
3. Change #GLOBALCSI to the data set name of your global CSI data set.
4. Change #TZONE to the name of your target zone.

IGZWARRE should run with a condition code of 0.

Changing the C/C++ locale time information

Use the EDCLLOCL job to change the C/C++ locale time information for your site. See Appendix C, "Using IBM C/C++ with Language Environment" on page 195 for information on changing the parameters in EDCLLOCL.

Recommendation: You should not install this usermod. The default C/C++ locale (EDC\$\$370) will by default obtain the time zone difference from Greenwich mean time from the system. If your C/C++ application requires a different time zone other

than the one obtained from the system, you can use the `tzset()` and the `TZ` environment variable described in *z/OS C/C++ Run-Time Library Reference*.

Steps for modifying the JCL for EDCLLOCL

Perform the following steps to modify the JCL for EDCLLOCL

1. Change `#GLOBALCSI` to the data set name of your global CSI data set.

2. Change `#TZONE` to the name of your target zone.

When you are done, EDCLLOCL should run with a condition code of 0.

Chapter 11. Customizing for Fortran applications

This section provides information on customizing Language Environment for Fortran applications link-edited with either Language Environment or VS FORTRAN.

Tailoring the Language Environment Fortran Unit Attribute Table

For Fortran applications link-edited with Language Environment, use SMP/E USERMOD AFHWEUAT to change the Unit Attribute Table defaults and DCB information for each I/O unit. Use the information in “Changing the unit attribute table default values” on page 163 to select your default values.

Modifying the JCL for AFHWEUAT

1. Replace the comment lines following the ++ SRC statement with member AFHOUTAG from CEE.SCEESAMP. Make any changes as desired.
2. Change #GLOBALCSI to the data set name of your global CSI data set.
3. Change #TZONE to the name of your target zone.
4. Check the SYSLIB DD statement to ensure the data set names are correct. Also, ensure the SYSLIB concatenation points to non-empty data sets.

AFHWEUAT should run with a condition code of 4. Unresolved external references for module CEESG003 and any optional modules not included in LIBPACK AFHPRNAG are expected.

Tailoring the VS FORTRAN Compatibility Unit Attribute Table

For applications link-edited with VS FORTRAN, use SMP/E USERMOD AFHVVUAT to change the Unit Attribute Table defaults and DCB information for each I/O unit. Use the information in “Changing the Unit Attribute Table Default Values” on page 170 to select your default values.

Modifying the JCL for AFHVVUAT

1. Replace the comment lines following the ++ SRC statement with member AFH5VUAT from CEE.SCEESAMP. Make any changes as desired.
2. Change #GLOBALCSI to the data set name of your global CSI data set.
3. Change #TZONE to the name of your target zone.
4. Check the SYSLIB DD statement to ensure the data set names are correct. Also, ensure the SYSLIB concatenation points to non-empty data sets.

AFHVVUAT should run with a condition code of 0 if LIBPACK AFH5RENA contains all of the optional modules. Otherwise, AFHVVUAT returns a condition code of 4; unresolved external references for any optional modules not included in AFH5RENA are expected.

Tailoring VS FORTRAN Compatibility Run-Time Options

For applications link-edited with VS FORTRAN, use SMP/E USERMOD AFHVVPRM to change the run-time option defaults. Use the information in “Changing VS FORTRAN Run-Time Option Defaults” on page 175 to select your default values.

Modifying the JCL for AFHVVPRM

1. Replace the comment lines following the ++ SRC statement with member AFH5GPRM from CEE.SCEESAMP. Make any changes as desired.
2. Change #GLOBALCSI to the data set name of your global CSI data set.
3. Change #TZONE to the name of your target zone.
4. Check the SYSLIB DD statement to ensure the data set names are correct. Also, ensure the SYSLIB concatenation points to non-empty data sets.

AFHWVPRM should run with a condition code of 0 if LIBPACK AFH5RENA contains all of the optional modules. Otherwise, AFHWVPRM returns a condition code of 4; unresolved external references for any optional modules not included in AFH5RENA are expected.

Tailoring the VS FORTRAN Compatibility Error Option Table

For applications link-edited with VS FORTRAN, use SMP/E USERMOD AFHWVOPT to change error option table defaults such as:

- The number of times the error is allowed to occur before the user program terminates
- The maximum number of times the message can be printed
- Whether or not a traceback map is to be printed
- Whether or not a user-written error exit routine is called

Use the information in “Changing the Error Option Table Defaults” on page 180 to select your default values.

Modifying the JCL for AFHWVOPT

1. Replace the comment lines following the ++ SRC statement with member AFH5UOPT from CEE.SCEESAMP. Make any changes as desired.
2. Change #GLOBALCSI to the data set name of your global CSI data set.
3. Change #TZONE to the name of your target zone.
4. Check the SYSLIB DD statement to ensure the data set names are correct. Also, ensure the SYSLIB concatenation points to non-empty data sets.

AFHWVOPT should run with a condition code of 0 if LIBPACK AFH5RENA contains all of the optional modules. Otherwise, AFHWVOPT returns a condition code of 4; unresolved external references for any optional modules not included in AFH5RENA are expected.

Chapter 12. 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 specific program.

For a table that maps Language Environment run-time options to HLL run-time options to help you plan your customization, see *z/OS Language Environment Run-Time Migration Guide*.

COBOL Compatibility

VS COBOL II supported 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:

```
CBLOPTS=OFF:
```

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

```
CBLOPTS=ON:
```

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

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. The syntax 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 default keywords are indicated for planning information only and appear **above** the main path or options path in the syntax diagrams. In the parameter list, IBM-supplied default choices are underlined.

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 as follows:

- 0** An informational message (or, if the entire token is zero, no information).
- 1** An attention message. Service completed, probably correctly.
- 2** An error message. Correction attempted. Service completed, perhaps incorrectly.
- 3** A severe error message. Service not completed.
- 4** A critical error message. Service not completed and condition signaled. A critical error is a condition that jeopardizes the environment. If a critical error occurs during a Language Environment callable service, it is always signaled to the condition manager instead of being returned synchronously to the caller.

For a complete description of all Language Environment run-time options, see *z/OS Language Environment Programming Reference*.

ABPERC

ABPERC

Derivation

ABnormal PERColation

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 z/OS UNIX, 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.

Non-CICS Default: ABPERC=((NONE),OVR)

Syntax

```
▶▶ ABPERC == ( ( NONE  
               |  
               | abcode ) , ( OVR  
               |  
               | 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. The *abcode* can be specified as:

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

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

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

You can identify only one abend code with this option. However, an abend U0000 is interpreted in the same way as S000.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- ABPERC is ignored under CICS.

Usage Notes

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

For More Information

- For more information about the assembler user exit (CEEBXITA), see *z/OS Language Environment Programming Guide*.

ABTERMENC**Derivation**

ABnormal TERMination of the ENClave

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.

Non-CICS Default: ABTERMENC=((ABEND),OVR)

Syntax

```

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

```

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 z/OS): An abend is issued to terminate the task.

ABTERMENC

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_RETC parameter of the assembler user exit.
- Reason code: Value of the CEEAUE_RSNC 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. The filename is the name defined in the DD card.
- Abend task/step attribute (on z/OS): 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.

RETCODE

Specifies that the enclave terminates with a non-zero return 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_RETC parameter of the assembler user exit. If the assembler user exit does not modify the CEEAUE_RETC 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_RSNC parameter of the assembler user exit.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS considerations

- The default under CICS is ABTERMENC=((ABEND),OVR).

Usage Notes

- z/OS UNIX considerations: In a multithreaded application with ABEND set for ABTERMENC only the main(IPT) thread will be ABENDED and the application terminated, regardless of which thread experienced the unhandled condition. All other threads (the NON-IPT threads) will be terminated normally, including the offending thread, if it is a NON-IPT thread.

For more information

- For information about return code calculation CEEAUE_RETC, CEEAUE_ABND, and assembler user exit CEEBXTA processing, see *z/OS Language Environment Programming Guide*.
- For more information about abend codes and a list of abend code values see *z/OS Language Environment Programming Guide*.

AIXBLD (COBOL Only)

Derivation

Alternate IndeX BuiLD

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.

Non-CICS Default: AIXBLD=((OFF),OVR)

Syntax

```

AIXBLD=( ( ( OFF ) ) , ( OVR ) )
          ( 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.

CICS consideration

- AIXBLD is ignored under CICS.

Usage Notes

- The only valid abbreviations for AIXBLD and NOAIXBLD are AIX and NOAIX, respectively.
- When specifying this option in CEEDOPT, CEEUOPT or CEEROPT, use the syntax AIXBLD(ON) or AIXBLD(OFF). Use AIXBLD and NOAIXBLD only on the command line.
- z/OS 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.

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.

For More Information

- See *Enterprise COBOL for z/OS and OS/390 Programming Guide* or *COBOL for OS/390 & VM Programming Guide* for more details.
- See “MSGFILE” on page 101 for information about the MSGFILE run-time option.

ALL31

Derivation

ALL AMODE 31

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.

It is recommended that ALL31 have the same setting for all enclaves in a process. Language Environment does not support the invocation of a nested enclave requiring ALL31(OFF) from an enclave running with ALL31(ON) in non-CICS environments.

In a multithread environment, Language Environment invokes all start routines, which are specified in a Language Environment 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.

Non-CICS Default: ALL31=((ON),OVR)

Syntax

```

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

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.

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 16 MB line.

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

If you use the setting ALL31(OFF), the STACK and LIBSTACK storage will be allocated from below the 16 MB line, regardless of the value specified on these parameters.

If you use the setting ALL31(OFF), Language Environment preallocates BELOWHEAP instead of ANYHEAP storage.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS considerations

- The default under CICS is ALL31=((ON),OVR).
- In COBOL, EXTERNAL data is allocated in unrestricted storage. Under CICS, Language Environment allocates storage for the common anchor area (CAA) and other control blocks in unrestricted storage.

Usage Notes

- z/OS UNIX consideration — The ALL31 option applies to the enclave.
- COBOL considerations — You must specify ALL31(OFF) if your applications contain one of the following programs:
 - A VS COBOL II NORES program
 - An OS/VS COBOL program (non-CICS program)
 - An AMODE 24 program
- Fortran considerations — Use ALL31(ON) if all of the compile units in the enclave have been compiled with VS FORTRAN Version 1 or Version 2 and there are no requirements for 24-bit addressing mode. Otherwise, use ALL31(OFF).
- When an application is running in an XPLINK environment (that is, either the XPLINK(ON) run-time option was specified, or the initial program contained at least one XPLINK-compiled part), the ALL31 run-time option will be forced to ON. No AMODE 24 routines are allowed in an enclave that uses XPLINK. No message will be issued to indicate this action. In this case, if a Language Environment run-time options report is generated using the RPTOPTS run-time option, the ALL31 option will be reported as "Override" under the LAST WHERE SET column.

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.

For More Information

- See "STACK" on page 130 for information about the STACK run-time option.

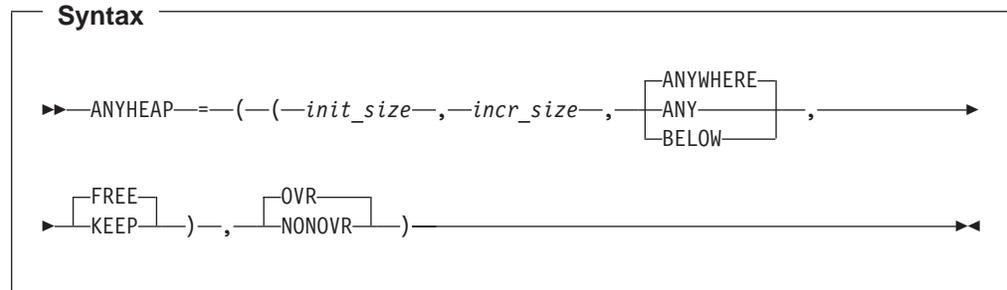
ANYHEAP

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

ANYHEAP

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.

Non-CICS Default: ANYHEAP=((16K,8K,ANYWHERE,FREE),OVR)



init_size

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

incr_size

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

ANYWHERE|ANY

Specifies that heap storage can be allocated anywhere in storage. If there is no storage available above the line, storage is acquired below the 16 MB line.

The only valid abbreviation for ANYWHERE is ANY.

BELOW

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

CICS consideration

- The default under CICS is ANYHEAP=((4K,4080,ANYWHERE,FREE),OVR).
- Both the initial size and the increment size are rounded up to the nearest multiple of 8 bytes. If ANYHEAP(0) is specified, the initial HEAP is obtained on the first use and will be based on the increment size. The maximum initial and increment size for ANYHEAP under CICS is 1 gigabyte (1024 MB).

- The default increment size under CICS is 4080 bytes, rather than 4096 bytes, to accommodate the 16 byte CICS storage check zone. Without this accommodation, an extra page of storage is allocated (only when the storage allocation is below the 16 MB line).

Usage Notes

- z/OS UNIX consideration — The ANYHEAP option applies to the enclave.

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.

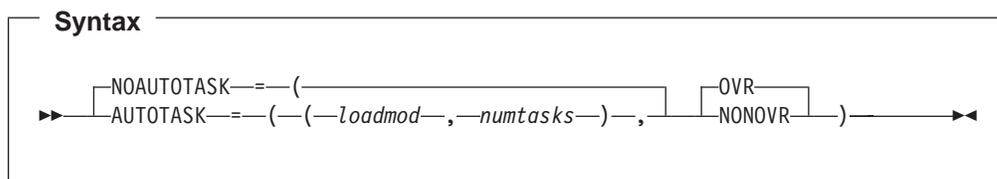
For More Information

- See *z/OS Language Environment Programming Guide* for more information about Language Environment heap storage.
- See “RPTSTG” on page 117 for more information about the RPTSTG run-time option.
- For more information about heap storage tuning with storage report numbers, see *z/OS Language Environment Programming Guide*.

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.

Non-CICS Default: NOAUTOTASK=(OVR)



NOAUTOTASK

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

loadmod

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

numtasks

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

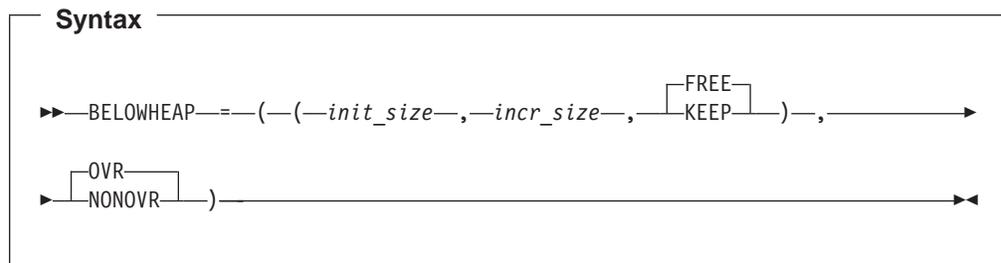
- AUTOTASK is ignored under CICS.

BELOWHEAP

BELOWHEAP

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

Non-CICS Default: BELOWHEAP=((8K,4K,FREE),OVR)



init_size

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

incr_size

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

FREE

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

KEEP

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is BELOWHEAP=((4K,4080,FREE),OVR).
- The default increment size under CICS is 4080 bytes, rather than 4096 bytes, to accommodate the 16 bytes CICS storage check zone. Without this accommodation, an extra page of storage is allocated (only when the storage allocation is below the 16 MB line).

Usage Notes

- Both the initial size and the increment size are rounded to the nearest multiple of 8 bytes. If you specify BELOWHEAP(0), the initial BELOWHEAP is obtained on the first use and will be the increment size.
- z/OS UNIX consideration — The BELOWHEAP option applies to the enclave.

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.

For More Information

- See *z/OS Language Environment Programming Guide* for more information about Language Environment heap storage.
- See “RPTSTG” on page 117 for more information about the RPTSTG run-time option.
- For more information about tuning your application with storage report numbers, see *z/OS Language Environment Programming Guide*.

CBLOPTS (COBOL Only)

Derivation
COBOL OPTionS

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, CEEDOPT or CEEROPT at initialization.

When you specify the ON suboption of CBLOPTS in CEEUOPT, CEEDOPT or CEEROPT, the run-time options and program arguments specified in the JCL or on the command line are honored in the following order, which is the reverse of that usually honored by Language Environment:

program arguments/run-time options

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.

Non-CICS Default: CBLOPTS=((ON),OVR)

Syntax

►► CBLOPTS == ((ON OFF) , (OVR 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.

CBLOPTS

CICS consideration

- CBLOPTS is ignored under CICS.

Usage Note

- If the string contains only run-time options that are invalid, the entire string is interpreted as a program argument. For example, if you pass the string 11/16/1967, 1967 is interpreted as an invalid run-time option. Since there are no other run-time options, the entire string will be interpreted as a program argument.

For More Information

- For more information about CEEUOPT, CEEDOPT or CEEROPT, see Chapter 4, “Customizing Language Environment Run-Time Options” on page 19.

CBLPSHPOP (COBOL Only)

Derivation

COBOL PUSH POP

CBLPSHPOP controls whether CICS PUSH HANDLE and CICS POP HANDLE commands are issued when a COBOL subroutine is called.

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

You can set the CBLPSHPOP run-time option on an enclave basis using CEEUOPT.

CBLPSHPOP is ignored in non-CICS environments.

Non-CICS Default: N/A

Syntax

►► CBLPSHPOP == ((ON / OFF) , (OVR / 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.

CICS consideration

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

For More Information

- For more information about CEEUOPT, see Chapter 4, “Customizing Language Environment Run-Time Options” on page 19.

CBLQDA (COBOL Only)**Derivation**

COBOL QSAM Dynamic Allocation

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 Language Environment formatted dump file (CEEDUMP).

Non-CICS Default: CBLQDA=((OFF),OVR)

Syntax

► CBLQDA = ((OFF) , (OVR))

OFF

Specifies that COBOL QSAM dynamic allocation is not permitted.

ON

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- CBLQDA is ignored under CICS.

Usage Notes

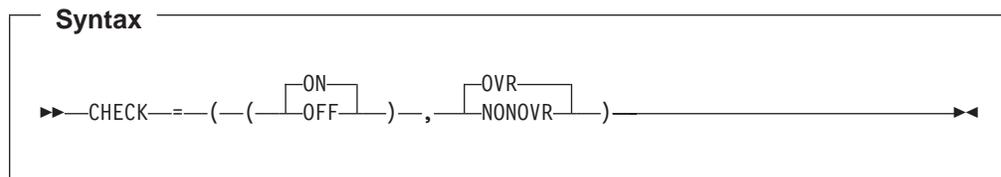
- z/OS consideration — You should use CBLQDA(OFF) under z/OS, because this prevents a temporary data set from being created in case there is a misspelling in your JCL. If you specify CBLQDA(ON) and have a misspelling in your JCL, Language Environment creates a temporary file, writes to it, and then z/OS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

CHECK

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.

Non-CICS Default: CHECK=((ON),OVR)



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.

CICS consideration

- The default under CICS is CHECK=((ON),OVR).

Usage Note

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

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).

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.

Non-CICS Default: COUNTRY=((US),OVR) with US signifying the United States.

Syntax

```

▶▶ COUNTRY=—(—(—country_code—), —OVRNONOVR—)▶▶

```

country_code

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is COUNTRY=((US),OVR) with (US) signifying the United States.

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, CEEDOPT (CEECOPT) and CEEROPT 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 Language Environment 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()`.

- z/OS UNIX consideration — The COUNTRY option sets the initial value for the enclave.

For More Information

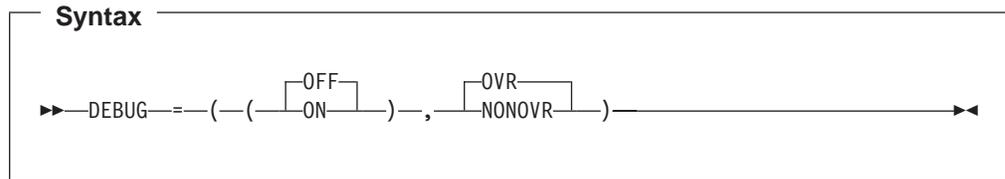
- For more information about the CEE3CTY and the CEESETL callable services, see *z/OS Language Environment Programming Reference*.
- For more information on `setlocale()`, see *z/OS C/C++ Programming Guide*.
- For a list of countries and their codes, see Appendix F, “Language Environment National Language Support Country Codes” on page 223 and *z/OS Language Environment Programming Reference*.

DEBUG (COBOL Only)

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

DEBUG

Non-CICS Default: DEBUG=((OFF),OVR)



OFF

Suppresses the COBOL batch debugging features.

ON

Activates the COBOL batch debugging features.

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is DEBUG=((OFF),OVR).

Usage Note

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

Performance Consideration

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

For More Information

- See *Enterprise COBOL for z/OS and OS/390 Programming Guide* or *COBOL for OS/390 & VM Programming Guide* for more details on the USE FOR DEBUGGING declarative.

DEPTHCONDLMT

Derivation

DEPTH of nested CONDition LiMiT

DEPTHCONDLMT specifies the extent to which conditions can be nested. Figure 9 on page 83 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.

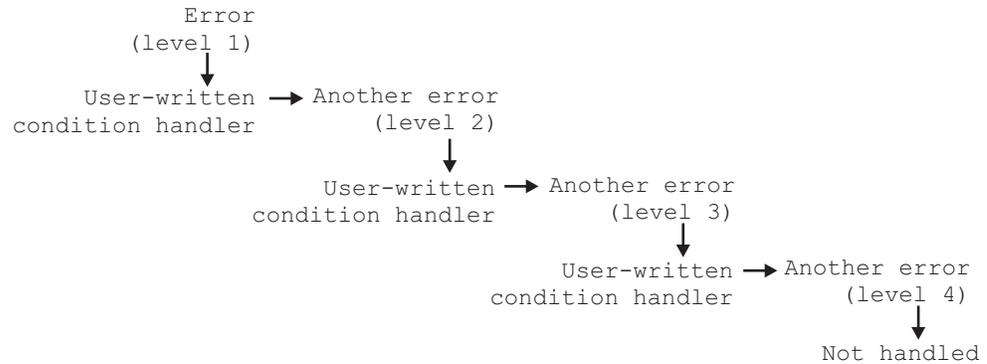
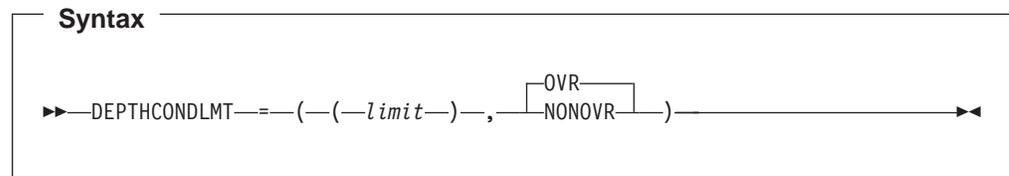


Figure 9. Effect of DEPTHCONDLMT(3) on Condition Handling

Non-CICS Default: DEPTHCONDLMT=((10),OVR)



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 U4087.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is DEPTHCONDLMT=((10),OVR).

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.
- z/OS UNIX consideration — The DEPTHCONDLMT option sets the limit for how many nested synchronous conditions are allowed for a thread. Asynchronous signals do not affect DEPTHCONDLMT.

For More Information

- For more information on nested conditions, see *z/OS Language Environment Programming Guide*.

ENVAR

ENVAR

Derivation

ENvironmental VARiables

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). 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.

Non-CICS Default: ENVAR=(("),OVR)

Syntax

```
ENVAR=( (string) , (OVR|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. The *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.

CICS consideration

- The default under CICS is ENVAR=(('),OVR).

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.

- z/OS UNIX consideration — The environment variables apply to the enclave.

For More Information

- For more information about the RPTOPTS run-time option, see "RPTOPTS" on page 114.

ERRCOUNT

Derivation

ERRor COUNTer

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 ENVCOUNT is reached, no further Language Environment condition management, including CEEHDLR management, is honored.

Non-CICS Default: ENVCOUNT=((0),OVR)

Syntax

```
▶▶—ENVCOUNT—==—(—(—number—)—,—OVRNONOVR—)—▶▶
```

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.

ERRCOUNT

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is `ERRCOUNT=((0),OVR)`.

Usage Notes

- `ERRCOUNT(0)` means that the Language Environment condition handler will not terminate the task regardless of the severity 2, 3, or 4 conditions that are generated. This setting allows previously existing infinite loop or runaway task conditions to persist.
- `ERRCOUNT` only applies when conditions are handled by a user condition handler, signal catcher, PL/I on-units, or a language-specific condition handler. Language Environment does not count severity 0 or 1 messages. However, the COBOL specific run-time library does count its severity 1 (warning) messages. When the limit of 256 IGZnnnnW messages is reached, the COBOL library will issue message IGZ0041W, which indicates that the limit of warning messages has been exceeded. Any further COBOL warning messages are suppressed.
- PL/I consideration — `ERRCOUNT(0)` is recommended for applications containing PL/I. Some conditions, such as `ENDPAGE`, can occur many times in an application. Use `ERRCOUNT(0)` to avoid unnecessary termination of your application.
- 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.
- z/OS UNIX 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`.

For More Information

- For a description of condition severities, see *z/OS Language Environment Programming Guide*.
- For more information about the `CEEHDLR` callable service, or the `CEESGL` callable service, see *z/OS Language Environment Programming Reference*.
- See *z/OS Language Environment Programming Guide* for more information about the facility ID part of messages.

ERRUNIT (Fortran Only)

Derivation <u>ERR</u> or <u>UNIT</u>
--

`ERRUNIT` identifies the unit number to which run-time error information is to be directed. This option is provided for compatibility with the VS Fortran version 2 runtime.

Non-CICS Default: `ERRUNIT=((6),OVR)`

Syntax

```

▶▶ ERRUNIT = ( (number) , ( OVR / 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.

CICS consideration

- ERRUNIT is ignored under CICS.

FILEHIST (Fortran Only)**Derivation**
FILE HISTORY

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.

Non-CICS Default: FILEHIST=((ON),OVR)**Syntax**

```

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

```

ON

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

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

OFF

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

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

- **If you change file definitions during run-time**, the file is treated as if it were being opened for the first time. Before the file definition can be changed, the existing file must be closed.

FILEHIST

- **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.

CICS consideration

- FILEHIST is ignored under CICS.

Usage Notes

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

FILETAG (C/C++ only)

Derivation

FILE TAGging

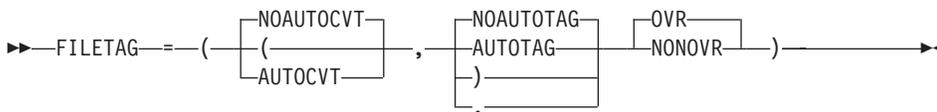
FILETAG run-time option ensures a more granular control of the manner in which untagged HFS files and standard streams opened as terminal files are set up for conversion. It also determines whether certain open functions will tag new or empty HFS files.

Recommendation: You should be familiar with the concept of file tagging, autoconversion, and the CCSID to use the run-time option properly. See *z/OS C/C++ Programming Guide* for more information.

Non-CICS Default: FILETAG=((NOAUTOCVT,NOAUTOTAG),OVR)

CICS Default: FILETAG is ignored under CICS.

Syntax



NOAUTOCVT

Disables behavior indicated below.

AUTOCVT

Enables automatic text conversion for untagged HFS files opened using fopen() or freopen() in text mode. The assumed CCSID for an untagged file will be the EBCDIC CCSID from the CCSID pair specified by the _BPXK_CCIDS environment variable. If the environment variable is not set, a

default CCSID pair is used. See the usage notes for more information. See *z/OS C/C++ Programming Guide* for more information on the `_BPXK_CCSDS` environment variable.

This option also indicates that the standard streams should be enabled for automatic text conversion to the EBCDIC IBM-1047 codepage when they refer to a terminal file (tty). See the usage notes for more information.

This option does not affect untagged HFS files that are automatically tagged by using the AUTOTAG suboption. This is because an HFS file that is tagged is already enabled for automatic text conversion.

Restriction: The automatic text conversion is performed only if one of the following is also true:

- The `_BPXK_AUTOCVT` environment variable value is equal to ON.
- The `_BPXK_AUTOCVT` environment variable is unset and `AUTOCVT(ON)` has been specified in the active `BPXPRMxx` member on your system. See *z/OS C/C++ Programming Guide* for more information on the `_BPXK_AUTOCVT` environment variable.

NOAUTOTAG

Deactivates the automatic tagging of new or empty HFS files.

AUTOTAG

Activates the automatic file tagging, on the first write, of new or empty HFS files open with `fopen()`, `freopen()`, or `popen()`. See the usage notes for more information.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

z/OS UNIX consideration

- FILETAG applies to the enclave. Nested enclaves do not inherit the setting of this run-time option. HFS files that are opened in the nested enclave are not affected.

Usage notes

• **Recommendations:**

- You should avoid the following:
 - Setting this run-time option in the installation-wide defaults (`CEEDOPT`).
 - Setting this run-time option using `_CEE_RUNOPTS` in a default profile for the UNIX shell users.
 - Exporting `_CEE_RUNOPTS` that specifies this run-time option. It can cause unexpected behaviors for the unknowing user or application.
- The application programmer should define this run-time option with the assumption that the application is coded to behave based upon the option's setting.
- The application programmer should specify this run-time option at compile time using `#pragma runopts` or at bind using a `CEEUOPT CSECT` that has been previously created.
- The application user should not override this run-time option because it can change the expected behavior of the application.
- The default CCSID pair is (1047,819), where 1047 indicates the EBCDIC IBM-1047 codepage and 819 indicates the ASCII ISO8859-1 codepage.
- Automatic text conversion is enabled for the standard streams only when the application has been `exec()`ed, for example, when the UNIX shell gives control to

FILETAG

a program entered on the command line, and the standard stream file descriptors are already open, untagged and associated with a tty.

- For the UNIX shell-owned standard streams that are redirected at program execution time, the shell includes added environment variables that control whether the redirected streams are tagged. See *z/OS UNIX System Services Command Reference* for more information.
- Automatic tagging for an HFS file is done at first write by the LFS. The CCSID used for the tag is the program CCSID of the current thread. Both text and binary files are tagged.
- When FILETAG(,AUTOTAG) is in effect, fopen() or freopen() of an HFS file fails if it cannot determine whether the file exists or if it cannot determine the size.

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.

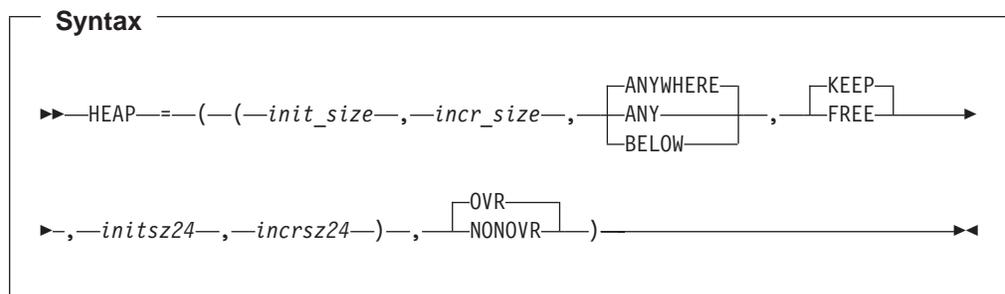
Heaps are storage areas where you allocate memory for user-controlled dynamically allocated variables such as:

- 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.

Non-CICS Default: HEAP=((32K,32K,ANYWHERE,KEEP,8K,4K),OVR)



init_size

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

incr_size

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

ANYWHERE|ANY

Specifies that you can allocate heap storage anywhere in storage. If there is no available storage above the line, storage is acquired below the 16 MB line.

The only valid abbreviation of ANYWHERE is ANY.

BELOW

Specifies that you must allocate heap storage below the 16 MB 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 when an application running AMODE 24 (ALL31(OFF)) requests storage and ANYWHERE has been specified. An AMODE 31 application running with ALL31(OFF) uses the regular heap allocation size. Specify *initsz24* as *n*, *nK*, or *nM* number of bytes. The amount of storage is rounded up to the nearest multiple of 8 bytes.

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

incrsz24

Determines the minimum size of any subsequent increment to the heap area that is obtained when an application running AMODE 24 (ALL31(OFF)) requests storage and ANYWHERE has been specified. An AMODE 31 application running with ALL31(OFF) uses the regular heap allocation size. Specify *incrsz24* as *n*, *nK*, or *nM* number of bytes. The amount of storage is rounded up to the nearest multiple of 8 bytes.

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is HEAP=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR).
- Both the initial HEAP allocation and HEAP increments are rounded to the next higher multiple of 8 bytes (not 4K bytes). If HEAP(0) is specified the initial HEAP is obtained on the first use and will be based on the increment size.
- If HEAP(,ANYWHERE,,) is in effect, the maximum size of a heap segment is 1 gigabyte (1024 MB). These restrictions are subject to change from one release of CICS to another.
- The default increment size under CICS is 4080 bytes, rather than 4096 bytes, to accommodate the 16 bytes CICS storage check zone. Without this accommodation, an extra page of storage is allocated (only when the storage allocation is below the 16 MB line).

HEAP

Usage Notes

- Applications running in AMODE 24 that request heap storage get the storage below the 16 MB 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.
- 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.
- PL/I MTF consideration — In a PL/I MTF application, HEAP specifies the heap storage allocation and management for a PL/I main task.
- z/OS UNIX considerations — The HEAP option applies to the enclave. Under z/OS UNIX, heap storage is managed at the thread level using pthread_key_create, pthread_setspecific, and pthread_getspecific.

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.

For More Information

- See *z/OS Language Environment 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, or the CEEGTST callable service, see *z/OS Language Environment Programming Reference*.
- See “RPTSTG” on page 117 for more information about the RPTSTG run-time option.

HEAPCHK

Derivation

HEAP storage CHecking

HEAPCHK allows the user to run the additional heap check tests.

Non-CICS Default: HEAPCHK=((OFF,1,0,0),OVR)

Syntax

```
HEAPCHK = ( ( OFF | ON ) , frequency , delay , call level ) , ( OVR | NONOVR )
```

OFF

Indicates that heap checking is inactive.

ON

Indicates that heap checking is active.

frequency

The frequency at which heap checks are performed, and is specified in *n*, *nK* or *nM*. A value of one (1) is the default which causes the heap to be checked at each call to a Language Environment heap storage management service. A value of *n* causes the heap to be checked at every *n*th call to a Language Environment heap storage management service. A value of zero causes HEAPCHK to produce only a heap storage diagnostics report. The fourth sub-option (call level) must be set to a value greater than 1.

delay

A value that causes a delay before heap checks are performed, and is specified in *n*, *nK* or *nM*. A value of zero (0) is the default which causes the heap to be checked from the first call to a Language Environment heap storage management service. A value of *n* causes the heap to be checked following the *n*th call to a Language Environment heap storage management service.

call level

This is the number of calls that will be displayed in the traceback for the heap storage diagnostics report and is specified in *n*. A value of zero is the default which turns heap storage diagnostics reporting off. A value of 10 is recommended.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is HEAPCHK=((OFF,1,0,0),OVR).

Usage Notes

- If HEAPCHK(ON) is used with STORAGE(,heap_free_value), the free areas of the heap will also be checked.
- If HEAPCHK(ON) is specified, this will result in a performance degradation due to the additional error checking that is performed.
- A U4042 abend dump will be generated when an error is detected, but no CEEDUMP will be produced.
- To request only a heap storage diagnostics report, specify zero for frequency and a number *n* greater than zero for call level. For example, you could specify HEAPCHK(ON,0,0,10).
- Under normal termination conditions, when the call level is greater than zero and the frequency is set to zero, the heap storage diagnostics report is written to the CEEDUMP report. This is independent of the TERMTHDACT setting.
- If a heap storage diagnostics report is desired while calling CEE3DMP, you must specify the BLOCKS option.

For More Information

- See *z/OS Language Environment Debugging Guide* for more information about creating and using the heap storage diagnostics report.

HEAPPOOLS

HEAPPOOLS (C/C++ only)

Derivation

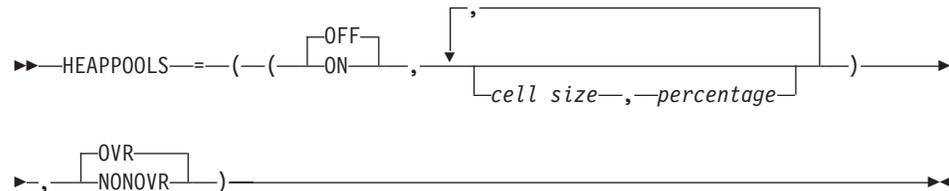
HEAP storage POOLS

The HEAPPOOLS run-time option is used to control an optional heap storage management algorithm known as heap pools. This algorithm is designed to improve performance of multi-threaded C/C++ applications with high usage of `malloc()`, `calloc()`, `realloc()` and `free()`. When active, heap pools virtually eliminates contention for heap storage.

Non-CICS Default:

HEAPPOOLS=((OFF,8,10,32,10,128,10,256,10,1024,10,2048,10),OVR)

Syntax



OFF

Specifies that Language Environment does not use the Heap Pools Manager.

ON

Specifies that Language Environment does use the Heap Pool Manager to manage heap storage requests against the initial heap.

cell size

The size of cells in a heap pool. The cell size must be a multiple of 8 within a range from 8 to 2048. Cell sizes 1K and 2K are also allowed.

percentage

Percentage of the HEAP run-time option init size value to be used as the size for the heap pool and any extents. The percentage must be in a range from 1 to 90.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is **HEAPPOOLS=((OFF,8,10,32,10,128,256,10,1024,10,2048,10),OVR)**.

Usage Notes

- To use less than six heap pools, specify 0 for the cell size after the last heap pool to be used. For example if four heap pools are desired, use 0 for the fifth cell size when setting the HEAPPOOLS run-time option.
- If the percentage of the HEAP run-time option init size values does not allow for at least one cell, then the system will automatically adjust the percentage to enable one cell to be allocated.
- The sum of the percentages may be more or less than 100 percent. This can cause the allocation of a heap pool to require the allocation of a heap increment to satisfy the request.
- Each heap pool is allocated on an as-needed basis. The allocation of a heap pool can result in the allocation of a heap increment to satisfy the request.
- Tuning tips are specified in *z/OS Language Environment Programming Guide*.
- Heap pools and extents are not released back to the heap, and cell sizes are fixed, so care should be taken when specifying the HEAPPOOLS run-time option to avoid wasting storage.
- The HEAPPOOLS run-time option has no effect when the initial heap is allocated below the 16 MB line. This would be the case when BELOW is specified as the location on the HEAP run-time option.
- The FREE sub-option on the HEAP run-time option has no effect on the initial heap or any extents in which a heap pool resides. Each cell in a heap pool can be freed, but the heap pool itself is only released back to the system at enclave termination.
- Mixing of the storage management AWIs (CEEGTST(), CEEFRST() and CEECZST()) and the C/C++ intrinsic functions (malloc(), calloc(), realloc() and free()) is not supported when operating on the same storage address. For example, if you request storage using CEEGTST(), then you may not use free() to release the storage.
- The HEAPPOOLS run-time option applies to the enclave.
- The RPTSTG run-time option will indicate HEAPPOOLS as one of the run-time options which can be adjusted.
- The HEAPCHK run-time option will indicate that individual cells in the cell pools controlled by the HEAPPOOLS run-time option are not validated. It is the heap pool itself which is validated, as it is the actual storage managed by the regular storage manager.

Performance Consideration

To improve the effectiveness of the Heap Pools Manager, use the storage report numbers generated by the RPTSTG run-time option as an aid in determining optimum cell sizes and the initial heap size.

INFOMSGFILTER

Derivation

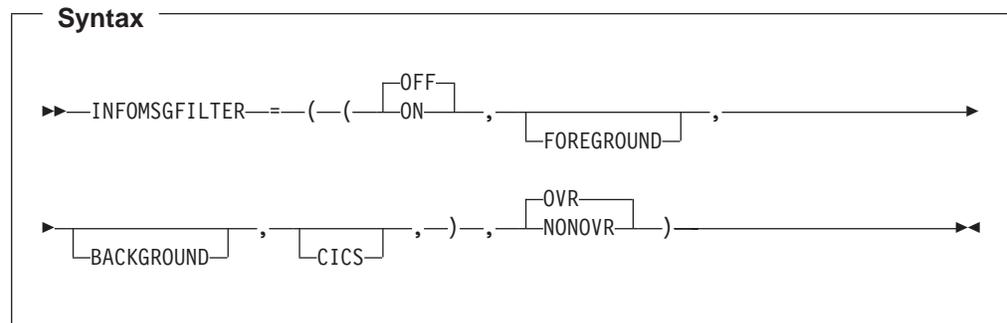
INFOrmational MeSsaGe FILTER

During normal operations, there are times when long lists of informational messages are written to the Language Environment MSGFILE. These messages are not limited to Language Environment (CEE) messages. Informational messages may also be written, using the CEEMSG interface, by other IBM program products or user-written applications. If these messages are routed to the user's terminal,

INFOMSGFILTER

then the user must constantly clear them. If the messages are saved to a data set, they take up disk space and may interfere with a user browsing the output looking for a specific message. INFOMSGFILTER allows the user to activate a filter that eliminates the unwanted and unnecessary informational messages. All informational messages, whether generated by Language Environment or any other source, will be suppressed when the INFOMSGFILTER=(ON) option is in effect.

Non-CICS Default: INFOMSGFILTER=((OFF,,,),OVR)



OFF

Turns off the filtering of messages for all environments.

ON

Turns on the filtering of messages for the specified environments.

FOREGROUND

Selecting this keyword turns message filtering on for the following environments:

- TSO
- CMS
- z/OS UNIX

BACKGROUND

Selecting this keyword turns message filtering on for the following environments:

- MVS Batch
- CMS Batch

CICS

Selecting this keyword turns message filtering on in the CICS environment.

Note: These three keywords are not positional, you can specify them in any order.

The fourth comma is required when coding this option for the installation-wide run-time options CSECT, CEEDOPT (CEEEOPT), or the region-wide run-time options CSECT CEEROPT even though there is no keyword to fill its position. This position is reserved by IBM for future use.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is INFOMSGFILTER=((OFF,,,),OVR).

INQPCOPN (Fortran Only)

Derivation

INQUIRE the Pre-Connected units that are OPeNed

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.

Non-CICS Default: INQPCOPN=((ON),OVR)

Syntax

```

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

```

ON

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

OFF

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- INQPCOPN is ignored under CICS.

Usage Notes

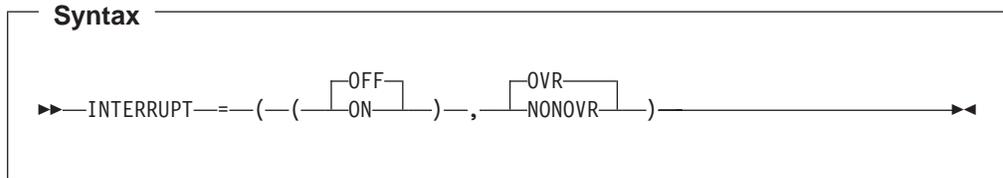
- When specifying this option in CEEDOPT, CEEUOPT or CEEROPT, use the syntax INQPCOPN(ON) or INQPCOPN(OFF).
- Use INQPCOPN and NOINQPCOPN only on the command line.

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.

Non-CICS Default: INTERRUPT=((OFF),OVR)

INTERRUPT



OFF

Specifies that Language Environment does not recognize attention interrupts.

ON

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- INTERRUPT is ignored under CICS.

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 program must be compiled with the INTERRUPT compiler option, and the INTERRUPT run-time option must be in effect.
- 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.
- z/OS UNIX consideration — The INTERRUPT option applies to the enclave. However, only one thread in the enclave is affected for a particular attention interrupt.

For More Information

- See “TEST | NOTEST” on page 142 for more information about the TEST run-time option.
- For more information about the POSIX run-time option, see “POSIX” on page 109.

LIBRARY

Non-CICS Default: LIBRARY=((SYSCEE),OVR)

This option specifies the name of the logical library used for finding RTL S version-controlled modules. The *lib_name* must be defined as a logical library in the CSVRTLxx PARMLIB member.

Syntax

```

▶▶ LIBRARY = ( ( lib_name ) , ( OVR | NONOVR ) ) ▶▶

```

lib_name

The name of the library that matches the name specified in the CSVRTLxx member of PARMLIB. The LIBRARY name must be a string from 1 to 8 characters long. The only valid characters are A-Z, 0-9, #, \$, @, ., -, +, or _ . The code points for the variant characters (\$, @, and #) are assumed to be in code pages 01047 or 00037.

Note: The LIBRARY option is ignored if specified for a nested enclave. The specified name will appear (perhaps erroneously) in the options report for the nested enclave, however. The RTLS library established by the main enclave is used for all nested enclaves.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- LIBRARY is ignored under CICS.

Usage Notes

- This option is ignored when RTLS(OFF) is in effect.
- If CEEGINIT is not available in the LNKLST, LPA, JOBLIB, STEPLIB, or TASKLIB when the Language Environment is invoked, the LIBRARY option has no effect. RTLS will not be used, but the LIBRARY option value will appear in the options report.

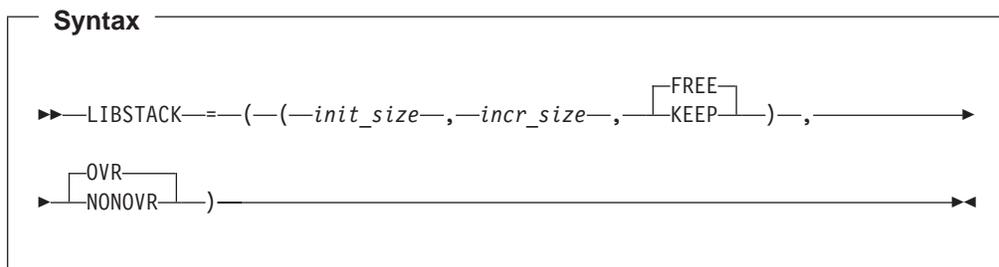
LIBSTACK**Derivation**

LIBrary STACK storage

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 16 MB line.

Non-CICS Default: LIBSTACK=((4K,4K,FREE),OVR)

LIBSTACK



init_size

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

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

In non-CICS environments, an *init_size* of 0 or -0 requests half of the largest block of contiguous storage below the 16 MB line. In addition when STACK(,ANY,,) is in effect, Language Environment does not acquire the initial library stack segment until the first program that requires LIBSTACK runs.

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*, *nK*, or *nM* bytes of storage. The actual amount of allocated storage is the larger of 2 values — *incr_size* or the requested size — rounded up to the nearest multiple of 8 bytes.

If you do not specify *incr_size*, Language Environment uses the Non-CICS 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.

CICS considerations

- The default under CICS is LIBSTACK=((32,4080,FREE),OVR).
- If ALL31(ON) is specified, LIBSTACK will be allocated above the 16 MB line.
- The initial and increment sizes for LIBSTACK are rounded to the next higher multiple of 8 bytes.
- The minimum initial size is 32 bytes; the minimum increment size is 4080.
- The default increment size under CICS is 4080 bytes, rather than 4096 bytes, to accommodate the 16 bytes CICS storage check zone. Without this accommodation, an extra page of storage is allocated (only when the storage allocation is below the 16 MB line).

Usage Notes

- z/OS UNIX consideration — The LIBSTACK option sets the library stack characteristics on each thread.
The recommended setting for LIBSTACK under z/OS UNIX is LIBSTACK=((4K,4K,FREE),OVR).

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.

For More Information

- See “RPTSTG” on page 117 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 *z/OS Language Environment Programming Guide*.

MSGFILE

Derivation

MeSsaGe FILE

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.

Non-CICS Default: MSGFILE=((SYSOUT,FBA,121,0,NOENQ),OVR)

Syntax

```

▶▶ MSGFILE=—(—(—ddname—,—recfm—,—lrecl—,—blksize—,——————▶
▶ [NOENQ] ) , [OVR] ) —————▶
  [ENQ]   [NONOVR]

```

ddname

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

MSGFILE

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.

NOENQ

Serialization around writes to the message file destination specified *ddname* are not performed.

ENQ

Specifies that serialization is performed around writes to the *ddname* specified.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- MSGFILE is ignored under CICS.
- Run-time output under CICS is directed instead to a transient data queue named CESE.

Usage Notes

- The ENQ sub-option should only be used where multiple Language Environment environments are running in the same address space and are sharing the same MSGFILE destination. An example would be a batch job which uses ATTACH to create some number of sub-tasks. Each of these tasks is potentially a distinct Language Environment environment all running with the same default MSGFILE parameters. In this example, each of these environments will share the same MSGFILE destination. To avoid conflicts while writing to the shared MSGFILE destination, it is recommended that the ENQ sub-option be used for each MSGFILE destination that will be shared. Using different *ddname* for each environment would remove the need to use the ENQ sub-option.
- 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, CEEROPT or CEEROPT, then an IBM-supplied default value is used. The default values for *ddname*, *recfm*, *lrecl*, and *blksize* are SYSOUT, FBA, 121, and 0, respectively.
- If there is no block size in the MSGFILE run-time option, in a file definition, or in the label of an existing file, block size is determined as follows:
 - For a *recfm* value that specifies unblocked fixed-length format records (F or FA) or undefined-format records (U or UA), the *blksize* value is the same as the *lrecl* value.
 - For a *recfm* value that specifies unblocked variable-length format records (V or VA), the *blksize* value is the *lrecl* value plus 4.
 - For a DASD device on MVS and a *recfm* value that specifies blocked records (FB, FBA, FBS, FBSA, VB, or VBA), the *blksize* value is left at 0 by Language Environment so that the system can determine the optimum *blksize* value.
 - For a terminal and a *recfm* value that specifies blocked fixed-length format records (FB, FBA, FBS, or FBSA), the *blksize* value is the same as the *lrecl* value.
 - For a terminal and a *recfm* value that specifies blocked variable-length format records (VB or VBA), the *blksize* value is the *lrecl* value plus 4.
 - For all other cases, *blksize* has a value which gives 100 records per block if the *blksize* value wouldn't exceed 32760, otherwise, a value giving the largest number of records per block such that the *blksize* value that doesn't exceed 32760.

Or, to put it another way:

- For a *recfm* value that specifies blocked fixed-length format records (FB, FBA, FBS, or FBSA), the *blksize* value is $lrecl \times bfact$ where *bfact* is the largest integer not exceeding 100 such that the *blksize* value does not exceed 32760.
 - For a *recfm* value that specifies blocked variable-length format records (VB or VBA), the *blksize* value is $(lrecl \times bfact) + 4$ 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 programs 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).
 - Fortran consideration — To get the same message file function as with VS Fortran, specify MSGFILE(FTnnF001,UA,133) where *nn* is the unit number of the error unit. For more information, see the Fortran Run-Time Migration Guide.

MSGFILE

- z/OS UNIX 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 z/OS UNIX 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=*.

z/OS UNIX 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 z/OS UNIX 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 z/OS), 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 `YSOOUT` in the current working directory; or, if there is no current directory, then in the directory `/tmp`.

For More Information

- For more information about the RPTOPTS and RPTSTG run-time options, see “RPTOPTS” on page 114 and “RPTSTG” on page 117.
- For more information about the CEEMSG and CEEMOUT callable services, see *z/OS Language Environment Programming Reference*.
- For details on how HLL compiler options affect messages, see information on HLL I/O statements and message handling in *z/OS Language Environment Programming Guide*.
- For more information about `perror()` and `stderr` see C message output information in *z/OS Language Environment Programming Guide*.
- For more information about the CESE transient data queue, see *z/OS Language Environment Programming Guide*.

MSGQ

Derivation

MeSsaGe Queue

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

Non-CICS Default: MSGQ=((15),OVR)

Syntax

```

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

```

number

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- MSGQ is ignored under CICS.

Usage Note

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

For More Information

- For more information about the CEECM1 callable service, see *z/OS Language Environment Programming Reference*.
- For more information about the ISI, see *z/OS Language Environment Programming Guide*.

NATLANG**Derivation**

NATional LANGuage

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.

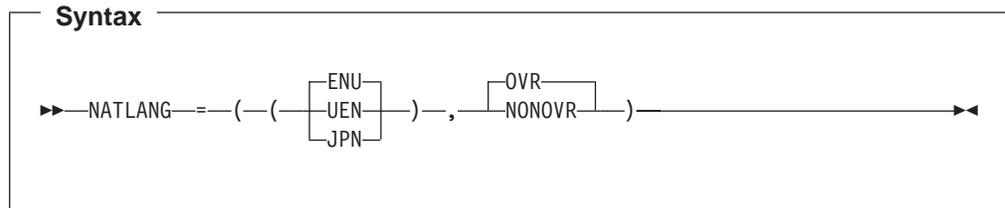
NATLANG affects only the Language Environment NLS and date and time services, not the Language Environment locale callable services.

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.

NATLANG

Non-CICS Default: NATLANG=((ENU),OVR)



ENU

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

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

UEN

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

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

JPN

A 3-character ID specifying Japanese.

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is NATLANG=((ENU),OVR).

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). CEEDOPT (CEECOPT), CEEROPT and CEEUOPT 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.
- z/OS UNIX consideration — The NATLANG option specifies the initial value for the enclave.

For More Information

- For more information about the CEE3LNG callable service, see *z/OS Language Environment Programming Reference*.
- See “MSGQ” on page 104 for more information about the MSGQ run-time option.
- For more information on `setlocale()`, see *z/OS C/C++ Programming Guide*.

OCSTATUS (Fortran Only)

Derivation

Open Close STATUS

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.

Non-CICS Default: OCSTATUS=((ON),OVR)

Syntax

```

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

```

ON

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

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

OFF

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

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

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

In addition, when a preconnected file is disconnected by a CLOSE statement, an OPEN statement is *not* required to reestablish the connection under OCSTATUS(OFF). A sequential READ, WRITE, BACKSPACE, REWIND, or ENDFILE will reconnect the file to a unit. Before the file is reconnected, the INQUIRE statement parameter OPENED will indicate that the unit is

OCSTATUS

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.

CICS consideration

- OCSTATUS is ignored under CICS.

Usage Notes

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

PC (Fortran Only)

Derivation

Private Common blocks

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

Non-CICS Default: PC=((OFF),OVR)

Syntax

► PC = ((OFF) , (OVR))

OFF

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

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

ON

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- PC is ignored under CICS.

Usage Notes

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

PLITASKCOUNT (PL/I Only)

Derivation

PL/I TASK COUNTER

PLITASKCOUNT controls the maximum number of tasks active at one time while you are running PL/I MTF applications. PLITASKCOUNT(20) provides behavior compatible with the PL/I ISASIZE(,20) option.

Non-CICS Default: PLITASKCOUNT=((20),OVR)

Syntax

```

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

```

tasks

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- PLITASKCOUNT is ignored under CICS.

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 z/OS UNIX installation default of the maximum number of threads, Language Environment assumes the z/OS UNIX 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.

POSIX

Derivation

Portable Operating System Interface - X

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

POSIX

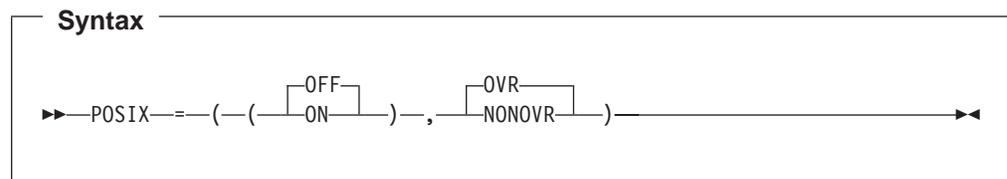
POSIX is an application characteristic that is maintained at the enclave level. After you have established the characteristic during enclave initialization, you cannot change it.

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 programs can access the z/OS UNIX Hierarchical File System (HFS) on MVS 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.

Non-CICS Default: **POSIX=((OFF),OVR)**



OFF

Indicates that the application is not POSIX-enabled.

ON

Indicates that the application is POSIX-enabled.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- POSIX is ignored under CICS.

Usage Notes

- If you set POSIX to ON and you run non-thread-safe languages such as PL/I and C++ in a thread other than the initial thread, the behavior is undefined.
- If you set POSIX to ON when z/OS UNIX is not active, the message file receives a warning, POSIX is set to OFF, but the application continues to run.
- POSIX(ON) does not apply to CICS. If you set POSIX to ON while an application is running under CICS, you receive a warning message, POSIX is set OFF, 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. When nested enclaves are specifying the run-time option POSIX(ON) within one Language Environment process, Language Environment will display a severity 3 error message and let abend 4039 occur with reason code 172.

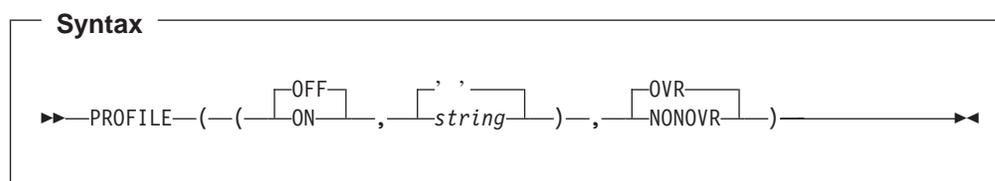
For More Information

- For more information on POSIX functions that have an z/OS UNIX kernel dependency, or a POSIX ON dependency (especially for a failure where the kernel dependency or the POSIX ON setting is not met), see *z/OS C/C++ Run-Time Library Reference*.
- For more information about the INTERRUPT run-time option, see “INTERRUPT” on page 97.

PROFILE

PROFILE controls the use of an optional profiler which collects performance data for the running application.

Non-CICS Default: PROFILE=((OFF,' '),OVR)



OFF

Indicates that the profile facility is inactive.

ON

Indicates that the profile facility is active.

' '

A null string indicates that no options are to be passed to the profiler.

string

Profile options that Language Environment will pass to the profiler installed. You can enclose the string in either single or double quotation marks. The maximum length of the string is 250 bytes when specified on program invocation or via a compiler directive. When establishing installation or programmer defaults using the CEEXOPT macro, the size is limited to 242 bytes.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is PROFILE=((OFF;"),OVR).

Usage Note

- An application cannot run with both the TEST and PROFILE options in effect. If both are specified, an informational message is generated and the Language Environment forces the PROFILE option OFF.

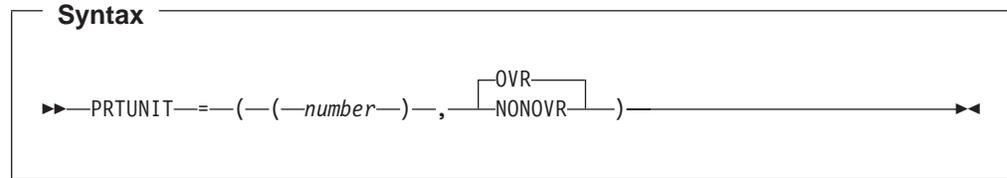
PRTUNIT (Fortran Only)



PRTUNIT

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

Non-CICS Default: PRTUNIT=((6),OVR)



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.

CICS consideration

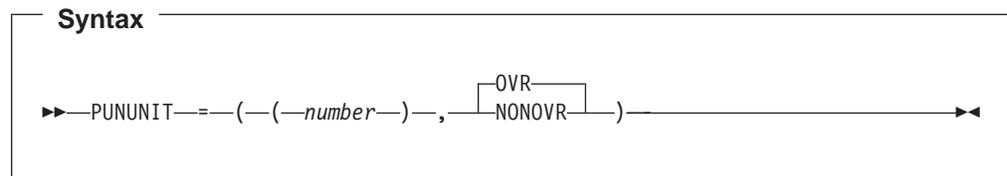
- PRTUNIT is ignored under CICS.

PUNUNIT (Fortran Only)



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

Non-CICS Default: PUNUNIT=((7),OVR)



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.

CICS consideration

- PUNUNIT is ignored under CICS.

RDRUNIT (Fortran Only)

Derivation

ReaDeR UNIT

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

Non-CICS Default: RDRUNIT=((5),OVR)

Syntax

►► RDRUNIT = ((*number*) , (OVR | NONOVR))

number

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- RDRUNIT is ignored under CICS.

RECPAD (Fortran Only)

Derivation

RECord PADding

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

Non-CICS Default: RECPAD=((OFF),OVR)

Syntax

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

OFF|NONE

Specifies that no blank padding be applied when an input list and format

RECPAD

specification requires more data from an input record than the record contains. If more data are 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), fixed blocked (FB), and fixed blocked standard (FBS).
- 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.

CICS consideration

- RECPAD is ignored under CICS.

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.

RPTOPTS

Derivation

RePorT OPTionS

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.

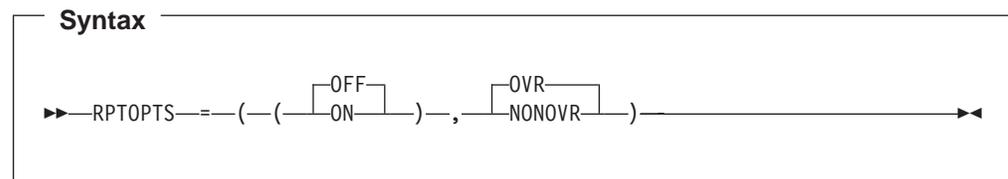
RPTOPTS does not generate the options report if Language Environment abends, but does generate a report in all other cases.

Figure 10 on page 116 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 (CEECOPT), CEEROPT or CEEUOPT. "Programmer default" includes any options specified with C/C++ #pragma runopts, PL/I PLIXOPT, and CEEUOPT. "Override" in the report indicates that Language Environment forced the setting of this specific option, based on either knowledge of the application or a side effect of another run-time option.

Non-CICS Default: RPTOPTS=((OFF),OVR)



OFF

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

ON

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is RPTOPTS=((OFF),OVR).
- With RPTOPTS(ON) under CICS, the options report will be written to the CESE queue when the transaction terminates successfully.

Usage Notes

- z/OS UNIX consideration — The RPTOPTS option reports run-time options for the enclave.
- Version, release and modification level information is included in the Language Environment reports to aid in debugging problems.

Performance Considerations

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

RPTOPTS

Options Report for Enclave main 04/08/02 3:28:53 PM
Language Environment V01 R04.00

LAST WHERE SET	OPTION
Installation default	ABPERC(NONE)
Installation default	ABTERMENC(ABEND)
Installation default	NOAIXBLD
Programmer default	ALL31(ON)
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(OFF)
Installation default	CHECK(ON)
Installation default	COUNTRY(US)
Installation default	NODEBUG
Installation default	DEPTHCONDLMT(10)
Installation default	ENVAR("")
Installation default	ERRCOUNT(0)
Installation default	ERRUNIT(6)
Installation default	FILEHIST
Installation default	FILETAG(NOAUTOCVT,NOAUTOTAG)
Default setting	NOFLOW
Assembler user exit	HEAP(49152,16384,ANYWHERE,KEEP,8192,4096)
Installation default	HEAPCHK(OFF,1,0,0)
Invocation command	HEAPPOLS(ON,8,10,32,10,128,10,256,10,1024,10,2048,10)
Installation default	INFMSGFILTER(OFF,,,))
Installation default	INQPCOPN
Installation default	INTERRUPT(OFF)
Installation default	LIBRARY(SYSCEE)
Programmer default	LIBSTACK(4096,4096,FREE)
Invocation command	MSGFILE(MSGFILE,FBA,121,0,NOENQ)
Installation default	MSGQ(15)
Installation default	NATLANG(ENU)
Mapped	NONIPTSTACK(See THREADSTACK)
Installation default	OCSTATUS
Installation default	NOPC
Installation default	PLITASKCOUNT(20)
Programmer default	POSIX(ON)
Installation default	PROFILE(OFF,"")
Installation default	PRTUNIT(6)
Installation default	PUNUNIT(7)
Installation default	RDRUNIT(5)
Installation default	RECPAD(OFF)
Programmer default	RPTOPTS(ON)
Programmer default	RPTSTG(ON)
Installation default	NORTEREUS
Installation default	RTLS(OFF)
Installation default	NOSIMVRD
Programmer default	STACK(4096,4096,ANYWHERE,FREE,524288,131072)
Programmer default	STORAGE(NONE,NONE,NONE,1024)
Installation default	TERMTHDACT(TRACE,,96)
Installation default	NOTEST(ALL,"*","PROMPT","INSPREF")
Installation default	THREADHEAP(4096,4096,ANYWHERE,KEEP)
Programmer default	THREADSTACK(ON,4096,4096,ANYWHERE,KEEP,131072,131072)
Installation default	TRACE(OFF,4096,DUMP,LE=0)
Installation default	TRAP(ON,SPIE)
Installation default	UPSI(00000000)
Installation default	NOUSRHDLR(,)
Installation default	VCTRSVAVE(OFF)
Installation default	VERSION()
Installation default	XPLINK(OFF)
Installation default	XUFLOW(AUTO)

Figure 10. Options Report Example Produced by Run-Time Option RPTOPTS(ON)

Tip: If automatic storage tuning for CICS changes a storage option setting, the LAST WHERE SET value will be "Automatic Tuning".

For More Information

- See “MSGFILE” on page 101 for more information about the MSGFILE run-time option.
- For more information about the CEE3RPH callable service, or the CEE3CTY callable service, see *z/OS Language Environment Programming Reference*.
- See “COUNTRY” on page 80 for more information about the COUNTRY run-time option.

RPTSTG

Derivation

RePorT SToraGe

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 11 on page 119 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.

Non-CICS Default: RPTSTG=((OFF),OVR)

Syntax

► RPTSTG = ((OFF) , (OVR))

OFF

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

ON

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

RPTSTG

CICS consideration

- The default under CICS is RPTSTG=((OFF),OVR).
- With RPTSTG(ON) under CICS, the storage report will be written to the CESE queue when the transaction terminates successfully.
- The phrases “Number of segments allocated” and “Number of segments freed” represent, on CICS, the number of EXEC CICS GETMAIN and EXEC CICS FREEMAIN requests, respectively.

Usage Notes

- RPTSTG does not generate a storage report if your application terminates abnormally.
- RPTSTG includes PL/I task-level information on stack and heap utilization.
- z/OS UNIX 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.
- Version, release and modification level information is included in the Language Environment reports to aid in debugging problems.

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, THREADSTACK 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 *init_size* and *incr_size* fields, and in the STACK and THREADSTACK *dsinit_size*, *dsincr_size*, *usinit_size* and *usincr_size* fields, for allocating storage.

Storage Report for Enclave main 04/08/02 3:28:53 PM
Language Environment V01 R04.00

```

STACK statistics:
  Initial size:                4096
  Increment size:              4096
  Maximum used by all concurrent threads: 7400
  Largest used by any thread:  7400
  Number of segments allocated: 2
  Number of segments freed:    0
THREADSTACK statistics:
  Initial size:                4096
  Increment size:              4096
  Maximum used by all concurrent threads: 3616
  Largest used by any thread:  3616
  Number of segments allocated: 6
  Number of segments freed:    0
LIBSTACK statistics:
  Initial size:                4096
  Increment size:              4096
  Maximum used by all concurrent threads: 0
  Largest used by any thread:    0
  Number of segments allocated: 0
  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:                49152
  Increment size:              16384
  Total heap storage used (sugg. initial size): 62784
  Successful Get Heap requests:  29
  Successful Free Heap requests: 13
  Number of segments allocated: 2
  Number of segments freed:    0
HEAP24 statistics:
  Initial size:                8192
  Increment size:              4096
  Total heap storage used (sugg. initial size): 0
  Successful Get Heap requests:  0
  Successful Free Heap requests: 0
  Number of segments allocated: 0
  Number of segments freed:    0
ANYHEAP statistics:
  Initial size:                32768
  Increment size:              16384
  Total heap storage used (sugg. initial size): 104072
  Successful Get Heap requests:  30
  Successful Free Heap requests: 15
  Number of segments allocated: 6
  Number of segments freed:    5

```

Figure 11. Storage Report Produced by Run-Time Option RPTSTG(ON) (Part 1 of 4)

```

BELOWHEAP statistics:
  Initial size:                               8192
  Increment size:                             8192
  Total heap storage used (sugg. initial size): 0
  Successful Get Heap requests:                0
  Successful Free Heap requests:               0
  Number of segments allocated:               0
  Number of segments freed:                   0
Additional Heap statistics:
  Successful Create Heap requests:             1
  Successful Discard Heap requests:            1
  Total heap storage used:                     4912
  Successful Get Heap requests:                3
  Successful Free Heap requests:               3
  Number of segments allocated:                2
  Number of segments freed:                   2
HeapPools Statistics:
  Pool 1 size: 8
    Successful Get Heap requests: 1- 8         8
  Pool 2 size: 32
    Successful Get Heap requests: 9- 16        3
    Successful Get Heap requests: 17- 24       5
    Successful Get Heap requests: 25- 32       3
  Pool 3 size: 128
    Successful Get Heap requests: 33- 40       3
    Successful Get Heap requests: 41- 48       3
    Successful Get Heap requests: 49- 56       11
    Successful Get Heap requests: 57- 64       4
    Successful Get Heap requests: 65- 72       3
    Successful Get Heap requests: 73- 80       4
    Successful Get Heap requests: 81- 88       5
    Successful Get Heap requests: 89- 96       4
    Successful Get Heap requests: 97- 104      4
    Successful Get Heap requests: 113- 120     5
    Successful Get Heap requests: 121- 128     4
  Pool 4 size: 256
    Successful Get Heap requests: 129- 136     6
    Successful Get Heap requests: 137- 144     3
    Successful Get Heap requests: 145- 152     4
    Successful Get Heap requests: 153- 160     2
    Successful Get Heap requests: 161- 168     8
    Successful Get Heap requests: 169- 176     5
    Successful Get Heap requests: 177- 184     4
    Successful Get Heap requests: 185- 192     6
    Successful Get Heap requests: 193- 200     3
    Successful Get Heap requests: 201- 208     4
    Successful Get Heap requests: 209- 216     2
    Successful Get Heap requests: 217- 224     3
    Successful Get Heap requests: 225- 232     4
    Successful Get Heap requests: 233- 240     2
    Successful Get Heap requests: 241- 248     2
    Successful Get Heap requests: 249- 256     1

```

Figure 11. Storage Report Produced by Run-Time Option RPTSTG(ON) (Part 2 of 4)

```

Pool 5 size: 1024
Successful Get Heap requests: 257- 264          4
Successful Get Heap requests: 265- 272          1
Successful Get Heap requests: 273- 280          3
Successful Get Heap requests: 281- 288          2
Successful Get Heap requests: 289- 296          2
Successful Get Heap requests: 305- 312          6
Successful Get Heap requests: 313- 320          5
Successful Get Heap requests: 321- 328          4
Successful Get Heap requests: 329- 336          2
Successful Get Heap requests: 337- 344          3
Successful Get Heap requests: 353- 360          2
Successful Get Heap requests: 361- 368          4
Successful Get Heap requests: 369- 376          5
Successful Get Heap requests: 377- 384          2
Successful Get Heap requests: 385- 392          2
Successful Get Heap requests: 393- 400          2
Successful Get Heap requests: 401- 408          5
Successful Get Heap requests: 409- 416          3
Successful Get Heap requests: 417- 424          2
Successful Get Heap requests: 425- 432          1
Successful Get Heap requests: 433- 440          2
Successful Get Heap requests: 441- 448          4
Successful Get Heap requests: 457- 464          1
Successful Get Heap requests: 465- 472          1
Successful Get Heap requests: 473- 480          2
Successful Get Heap requests: 481- 488          1
Successful Get Heap requests: 489- 496          2
Successful Get Heap requests: 497- 504          5
Successful Get Heap requests: 505- 512          2
Successful Get Heap requests: 545- 552          1
Successful Get Heap requests: 577- 584          1
Successful Get Heap requests: 641- 648          2
Successful Get Heap requests: 825- 832          1
Successful Get Heap requests: 913- 920          1
Pool 6 size: 2048
Successful Get Heap requests: 1089-1096         1
Successful Get Heap requests: 1169-1176         1
Successful Get Heap requests: 1185-1192         1
Successful Get Heap requests: 1217-1224         2
Successful Get Heap requests: 1257-1264         1
Successful Get Heap requests: 1377-1384         1
Successful Get Heap requests: 1401-1408         1
Successful Get Heap requests: 1521-1528         1
Successful Get Heap requests: 1537-1544         1
Successful Get Heap requests: 1545-1552         1
Successful Get Heap requests: 1569-1576         1
Successful Get Heap requests: 1665-1672         1
Successful Get Heap requests: 1761-1768         1
Successful Get Heap requests: 1785-1792         1
Successful Get Heap requests: 1929-1936         1
Successful Get Heap requests: 1937-1944         1
Successful Get Heap requests: 1953-1960         1
Requests greater than the largest cell size:    19

```

Figure 11. Storage Report Produced by Run-Time Option RPTSTG(ON) (Part 3 of 4)

RPTSTG

HeapPools Summary:

Cell Size	Extent Percent	Cells Per Extent	Extents Allocated	Maximum Cells Used	Cells In Use
8	10	307	1	2	0
32	10	122	1	3	1
128	10	36	1	10	7
256	10	18	1	11	4
1024	10	4	4	13	12
2048	10	2	2	4	3

Suggested Percentages for current Cell Sizes:
HEAPP(ON,8,1,32,1,128,3,256,6,1024,28,2048,17)

Suggested Cell Sizes:
HEAPP(ON,104,,208,,376,,512,,1264,,1960,)

Largest number of threads concurrently active: 2

End of Storage Report

Figure 11. Storage Report Produced by Run-Time Option RPTSTG(ON) (Part 4 of 4)

Storage Report for Enclave main 04/08/02 3:39:15 PM
Language Environment V01 R04.00

```

STACK statistics:
  Initial size:                4096
  Increment size:              4096
  Maximum used by all concurrent threads: 6864
  Largest used by any thread:  6864
  Number of segments allocated: 2
  Number of segments freed:    0
THREADSTACK statistics:
  Initial size:                4096
  Increment size:              4096
  Maximum used by all concurrent threads: 11232
  Largest used by any thread:  2576
  Number of segments allocated: 6
  Number of segments freed:    0
XPLINK STACK statistics:
  Initial size:                524288
  Increment size:              131072
  Largest used by any thread:  2640
  Number of segments allocated: 1
  Number of segments freed:    0
XPLINK THREADSTACK statistics:
  Initial size:                131072
  Increment size:              131072
  Largest used by any thread:  2752
  Number of segments allocated: 6
  Number of segments freed:    0
LIBSTACK statistics:
  Initial size:                4096
  Increment size:              4096
  Maximum used by all concurrent threads: 0
  Largest used by any thread:  0
  Number of segments allocated: 0
  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:                49152
  Increment size:              16384
  Total heap storage used (sugg. initial size): 64640
  Successful Get Heap requests: 30
  Successful Free Heap requests: 13
  Number of segments allocated: 2
  Number of segments freed:    0

```

Figure 12. Storage Report Produced by RPTSTG(ON) with XPLINK (Part 1 of 4)

```

HEAP24 statistics:
  Initial size:                                8192
  Increment size:                              4096
  Total heap storage used (sugg. initial size): 0
  Successful Get Heap requests:                0
  Successful Free Heap requests:               0
  Number of segments allocated:                0
  Number of segments freed:                   0
ANYHEAP statistics:
  Initial size:                                32768
  Increment size:                              16384
  Total heap storage used (sugg. initial size): 125392
  Successful Get Heap requests:                 31
  Successful Free Heap requests:                15
  Number of segments allocated:                 7
  Number of segments freed:                    6
BELOWHEAP statistics:
  Initial size:                                8192
  Increment size:                              8192
  Total heap storage used (sugg. initial size): 0
  Successful Get Heap requests:                 0
  Successful Free Heap requests:                0
  Number of segments allocated:                 0
  Number of segments freed:                    0
Additional Heap statistics:
  Successful Create Heap requests:              1
  Successful Discard Heap requests:             1
  Total heap storage used:                      4912
  Successful Get Heap requests:                 3
  Successful Free Heap requests:                3
  Number of segments allocated:                 2
  Number of segments freed:                    2
HeapPools Statistics:
  Pool 1 size: 8
    Successful Get Heap requests: 1- 8          8
  Pool 2 size: 32
    Successful Get Heap requests: 9- 16         3
    Successful Get Heap requests: 17- 24        5
    Successful Get Heap requests: 25- 32        3
  Pool 3 size: 128
    Successful Get Heap requests: 33- 40         9
    Successful Get Heap requests: 41- 48         3
    Successful Get Heap requests: 49- 56        11
    Successful Get Heap requests: 57- 64         4
    Successful Get Heap requests: 65- 72         3
    Successful Get Heap requests: 73- 80         4
    Successful Get Heap requests: 81- 88         5
    Successful Get Heap requests: 89- 96         4
    Successful Get Heap requests: 97- 104        4
    Successful Get Heap requests: 113- 120       5
    Successful Get Heap requests: 121- 128       4

```

Figure 12. Storage Report Produced by RPTSTG(ON) with XPLINK (Part 2 of 4)

```

Pool 4 size: 256
  Successful Get Heap requests: 129- 136      6
  Successful Get Heap requests: 137- 144      3
  Successful Get Heap requests: 145- 152      4
  Successful Get Heap requests: 153- 160      2
  Successful Get Heap requests: 161- 168      8
  Successful Get Heap requests: 169- 176      5
  Successful Get Heap requests: 177- 184      4
  Successful Get Heap requests: 185- 192      6
  Successful Get Heap requests: 193- 200      3
  Successful Get Heap requests: 201- 208      4
  Successful Get Heap requests: 209- 216      2
  Successful Get Heap requests: 217- 224      3
  Successful Get Heap requests: 225- 232      4
  Successful Get Heap requests: 233- 240      2
  Successful Get Heap requests: 241- 248      2
  Successful Get Heap requests: 249- 256      1
Pool 5 size: 1024
  Successful Get Heap requests: 257- 264      4
  Successful Get Heap requests: 265- 272      1
  Successful Get Heap requests: 273- 280      3
  Successful Get Heap requests: 281- 288      2
  Successful Get Heap requests: 289- 296      2
  Successful Get Heap requests: 305- 312      6
  Successful Get Heap requests: 313- 320      5
  Successful Get Heap requests: 321- 328      4
  Successful Get Heap requests: 329- 336      2
  Successful Get Heap requests: 337- 344      3
  Successful Get Heap requests: 353- 360      2
  Successful Get Heap requests: 361- 368      4
  Successful Get Heap requests: 369- 376      5
  Successful Get Heap requests: 377- 384      2
  Successful Get Heap requests: 385- 392      2
  Successful Get Heap requests: 393- 400      2
  Successful Get Heap requests: 401- 408      5
  Successful Get Heap requests: 409- 416      3
  Successful Get Heap requests: 417- 424      2
  Successful Get Heap requests: 425- 432      1
  Successful Get Heap requests: 433- 440      2
  Successful Get Heap requests: 441- 448      4
  Successful Get Heap requests: 457- 464      1
  Successful Get Heap requests: 465- 472      1
  Successful Get Heap requests: 473- 480      2
  Successful Get Heap requests: 481- 488      1
  Successful Get Heap requests: 489- 496      2
  Successful Get Heap requests: 497- 504      5
  Successful Get Heap requests: 505- 512      2
  Successful Get Heap requests: 545- 552      1
  Successful Get Heap requests: 577- 584      1
  Successful Get Heap requests: 641- 648      2
  Successful Get Heap requests: 825- 832      1
  Successful Get Heap requests: 913- 920      1

```

Figure 12. Storage Report Produced by RPTSTG(ON) with XPLINK (Part 3 of 4)

```

Pool 6 size: 2048
  Successful Get Heap requests: 1089-1096      1
  Successful Get Heap requests: 1169-1176      1
  Successful Get Heap requests: 1185-1192      1
  Successful Get Heap requests: 1217-1224      2
  Successful Get Heap requests: 1257-1264      1
  Successful Get Heap requests: 1377-1384      1
  Successful Get Heap requests: 1401-1408      1
  Successful Get Heap requests: 1521-1528      1
  Successful Get Heap requests: 1537-1544      1
  Successful Get Heap requests: 1545-1552      1
  Successful Get Heap requests: 1569-1576      1
  Successful Get Heap requests: 1665-1672      1
  Successful Get Heap requests: 1761-1768      1
  Successful Get Heap requests: 1785-1792      1
  Successful Get Heap requests: 1929-1936      1
  Successful Get Heap requests: 1937-1944      1
  Successful Get Heap requests: 1953-1960      1
Requests greater than the largest cell size:    19
HeapPools Summary:
Cell Extent  Cells Per  Extents  Maximum  Cells In
Size Percent Extent   Allocated Cells Used Use
-----
   8    10      307        1         6         0
  32    10     122        1         4         1
 128    10     36         1        26        13
 256    10     18         2        19         4
1024    10     4          4        13        12
2048    10     2          2         4         3
-----
Suggested Percentages for current Cell Sizes:
HEAPP(ON,8,1,32,1,128,8,256,11,1024,28,2048,17)
Suggested Cell Sizes:
HEAPP(ON,96,,208,,376,,512,,1264,,1960,)
Largest number of threads concurrently active:    7
End of Storage Report

```

Figure 12. Storage Report Produced by RPTSTG(ON) with XPLINK (Part 4 of 4)

For More Information

- For more information about the MSGFILE run-time option, see “MSGFILE” on page 101.
- For more information about the CEE3RPH callable service or the CEE3CTY callable service, see *z/OS Language Environment Programming Reference*.
- See “COUNTRY” on page 80 for more information about the COUNTRY run-time option.
- For more information about the ANYHEAP run-time option, see “ANYHEAP” on page 73.
- For more information about the BELOWHEAP run-time option, see “BELOWHEAP” on page 76.
- For more information about the HEAP run-time option, see “HEAP” on page 90.
- For more information about the LIBSTACK run-time option, see “LIBSTACK” on page 99.
- For more information about the STACK run-time option, see “STACK” on page 130.

- For more information about tuning your application with storage numbers, see *z/OS Language Environment Programming Guide*.

RTEREUS (COBOL Only)

Derivation

Run Time Environment REUSE

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, CEEUOPT, CEEROPT or the assembler user exit.

Non-CICS Default: RTEREUS=((OFF),OVR)

Syntax

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

OFF

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

ON

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

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- RTEREUS is ignored under CICS.

Usage notes

- **Recommendation:** Avoid using RTEREUS(ON) as an installation default. If you do use RTEREUS, use it for specific applications only.
- **Restrictions:**
 - RTEREUS(ON) cannot be used with XPLINK(ON).
 - RTEREUS(ON) cannot be used in a z/OS UNIX process.
 - Enterprise COBOL programs compiled with the THREAD compiler option do not run with RTEREUS(ON).
- Under Language Environment, RTEREUS(ON) is only supported in a single enclave environment unless you modify the behavior using the IGZERREO CSECT. With the IBM-supplied default setting for COBOL's reusable environment, applications that attempt to create nested enclaves terminate with error message IGZ0168S.

Nested enclaves can be created by applications that use SVC LINK or CMSCALL to invoke application programs. One example is when an SVC LINK is

RTEREUS

used to invoke an application program under ISPF that is using ISPF services (such as CALL 'ISPLINK' and ISPF SELECT).

- If a Language Environment reusable environment is established (using RTEREUS), attempts to run a C or PL/I main program under Language Environment will fail. For example, when running on ISPF with RTEREUS(ON):
 - The first program invoked by ISPF is a COBOL program. A Language Environment reusable environment is established.
 - At some other point, ISPF invokes a PL/I or C program. The initialization of the PL/I or C program will fail.
- If a large number of COBOL programs are run (using RTEREUS) under the same MVS task, you can encounter out-of-region abends. This is because all storage acquired by Language Environment to run COBOL programs is kept in storage until the MVS task ends or the Language Environment environment is terminated.
- Language Environment storage and run-time options reports are not produced by Language Environment (using RTEREUS) unless a STOP RUN is issued to end the enclave.
- When you specify RTEREUS in CEEDOPT, CEEROPT or CEEUOPT, the only accepted syntax is RTEREUS(ON) or RTEREUS(OFF).
- IMS consideration — RTEREUS is not recommended for use under IMS.
- The IGZERREO CSECT affects the handling of program checks in the non-Language Environment-enabled driver that repeatedly invokes COBOL programs. It also affects the behavior of running COBOL programs in a nested enclave when a reusable environment is active.

Performance considerations

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

The IGZERREO CSECT affects the performance of running with RTEREUS.

Language Environment also offers preinitialization support in addition to RTEREUS.

For more information

- For more information about CEEUOPT, CEEROPT or CEEDOPT, see Chapter 4, “Customizing Language Environment Run-Time Options” on page 19.
- For more information about IGZERREO, see “Modifying the COBOL Reusable Environment” on page 62.
- See *z/OS Language Environment Programming Guide* for more information about preinitialization.

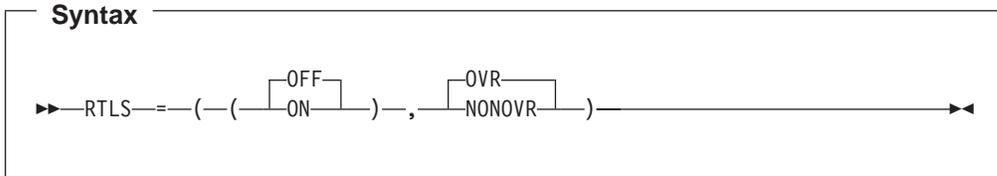
RTLS

Derivation

Run Time Library Services

Non-CICS Default: RTLS=((OFF),OVR)

RTLS allows you to LOAD and DELETE Language Environment modules and any dynamically-loaded user modules using the z/OS module search order or using the active RTLS logical library. See “Setting up run-time library services (RTLS)” on page 8, “LIBRARY” on page 98 and “VERSION” on page 156.



OFF

When RTLS(OFF) is in effect, Language Environment modules and any dynamically-loaded user modules are loaded using the z/OS module search order.

ON

When RTLS(ON) is in effect, most Language Environment modules and dynamically-loaded user modules are first loaded from the active RTLS logical library. If they are not found in the RTLS logical library, they are LOAded using the z/OS module search order.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- RTLS is ignored under CICS.

Usage Notes

- If CEEGINIT is not available in the LNKLST, LPA, JOBLIB, STEPLIB, or TASKLIB when the Language Environment is invoked, the RTLS(ON|OFF) option has no effect. RTLS will not be used, but the RTLS(ON|OFF) option setting will appear in the options report. Load module CEEGINIT can be found in the SCEERTLS data set and must be placed in the z/OS search order in order for this option to take effect. This option is also meaningless under CICS, PIP1, and PIC1. In nested enclaves, this option is inherited from the main enclave. If RTLS(ON) or RTLS(OFF) is specified for the nested enclave, this value is ignored, but will appear in options reports for the nested enclave.
- For RTLS restrictions, see “Restrictions” on page 13.

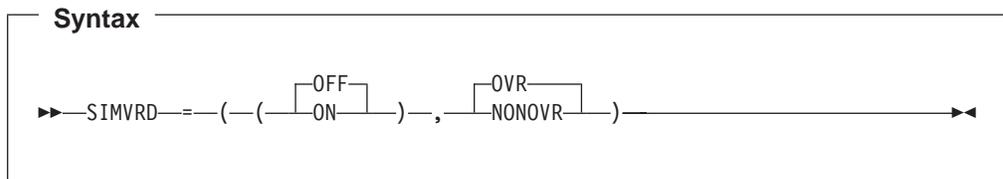
SIMVRD (COBOL Only)



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

Non-CICS Default: SIMVRD=((OFF),OVR)

SIMVRD



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.

CICS consideration

- SIMVRD is ignored under CICS.

Usage Notes

- When you specify SIMVRD in CEEDOPT, CEEROPT or CEEUOPT, the only accepted syntax is SIMVRD(ON) or SIMVRD(OFF).
- Use SIMVRD and NOSIMVRD only on the command line.

For More Information

- See *Enterprise COBOL for z/OS and OS/390 Programming Guide* or *COBOL for OS/390 & VM Programming Guide* for more details.
- For more information about CEEUOPT, CEEROPT or CEEDOPT, see Chapter 4, "Customizing Language Environment Run-Time Options" on page 19.

STACK

STACK controls the allocation of the thread's stack storage for both the upward and downward-growing stacks. Typical items residing in the upward-growing stack are C or PL/I automatic variables, COBOL LOCAL-STORAGE data items, and work areas for COBOL library routines.

The downward growing stack will be allocated only when an application has been built with XPLINK.

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.

Non-CICS Default: STACK=((128K,128K,ANYWHERE,KEEP,512K,128K),OVR)

Syntax

```

▶▶ STACK ::= ( ( ( -usinit_size , -usincr_size , [ ANYWHERE | ANY | BELOW ] ,
▶ [ KEEP | FREE ] , -dsinit_size , -dsincr_size ) , [ OVR | NONOVR ] ) )

```

usinit_size

Determines the size of the initial upward-growing stack segment. The storage is contiguous. You specify the *usinit_size* value as *n*, *nK*, or *nM* bytes of storage. The actual amount of allocated storage is rounded up to the nearest multiple of 8 bytes.

usinit_size can be preceded by a minus sign. In environments 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 16 MB line.

usincr_size

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

If you specify *usincr_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, *usincr_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.

ANYWHERE | ANY | BELOW

Specifies the storage location. For downward growing stack, this option is ignored and the storage is always placed above 16 MB.

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. If there is no storage available above the line, Language Environment acquires storage below the 16 MB line.

KEEP | FREE

Determines the disposition of the storage increments when the last stack frame in the increment segment is freed.

STACK

KEEP

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

FREE

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

dsinit_size

Determines the size of the initial downward growing stack segment. The storage is contiguous. You specify the *dsinit_size* value as n, nK, or nM bytes of storage. The actual amount of allocated storage is rounded up to the nearest multiple of 16 bytes.

dsincr_size

Determines the minimum size of any subsequent increment to the downward growing stack area. You can specify this value as n, nK, or nM bytes of storage. The actual amount of allocated storage is the larger of two values-- *dsincr_size* or the requested size--rounded up to the nearest multiple of 16 bytes.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is `STACK=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR)`.
- *dsinit_size* and *dsincr_size* sub-options are ignored under CICS.
- The maximum initial and increment size for CICS above 16 MB is 1 gigabyte (1024 MB). This restriction is subject to change from one release of CICS to another.
- Both the initial size and the increment size are rounded up to the nearest multiple of 8 bytes. The initial size minimum is 4 KB.
- If you do not specify `STACK`, Language Environment assumes the default value of 4 KB. Under CICS, `STACK(0)`, `STACK(-0)`, and `STACK(-n)` are all interpreted as `STACK(4K)`.
- The default increment size under CICS is 4080 bytes, rather than 4096 bytes, to accommodate the 16 bytes CICS storage check zone. Without this accommodation, an extra page of storage is allocated (only when the storage allocation is below the 16 MB line).

Usage Notes

- When an application is running in an XPLINK environment (that is, either the `XPLINK(ON)` run-time option was specified, or the initial program contained at least one XPLINK-compiled part), the `STACK` run-time option will be forced to `STACK(,ANY,,)`. Only the third suboption of the `STACK` run-time option is changed by this action, to indicate that stack storage can be allocated anywhere in storage. No message will be issued to indicate this action. In this case, if a Language Environment run-time options report is generated using the `RPTOPTS` run-time option, the `STACK` option will be reported as "Override" under the `LAST WHERE SET` column.
- The *dsinit_size* and *dsincr_size* values are the amounts of storage that can be used for downward growing stack frames (plus the stack header, approximately 20 bytes). The actual size of the storage getmained will be 4K (8K if a 4K page alignment cannot be guaranteed) larger to accommodate the guard page.

- The downward growing stack is only initialized in an XPLINK supported environment, and only when an XPLINK application is active in the enclave. Otherwise the suboptions for the downward growing stack are ignored.
- Applications running with ALL31(OFF) must specify STACK(,BELOW,,) to ensure that stack storage is addressable by the application.
- PL/I consideration — PL/I automatic storage above the 16 MB 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 16 MB. Stack frame extensions are also constrained to 16 MB. Therefore, the size of an automatic aggregate, temporary variable, or dummy argument cannot exceed 16 MB. 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 “THREADSTACK” on page 146.
- z/OS UNIX 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 thread’s 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 z/OS UNIX is STACK=((12K,12K,ANYWHERE,KEEP,512K,128K),OVR).

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 sizes for STACK.

For More Information

- See “ALL31” on page 72, for more information about the ALL31 run-time option.
- See “RPTSTG” on page 117, for more information about the RPTSTG run-time option.
- See “THREADSTACK” on page 146, for more information about the THREADSTACK run-time option.
- For more information about using the storage reports generated by the RPTSTG run-time option to tune the stacks, see *z/OS Language Environment Programming Guide*.

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.

STORAGE

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.

Non-CICS Default: STORAGE=((NONE,NONE,NONE,0K),OVR)

Syntax

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

heap_alloc_value

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

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

heap_free_value

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

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

dsa_alloc_value

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

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

You can specify *dsa_alloc_value* as:

- A single character enclosed in quotes. If you specify a single character, any dynamically acquired stack storage allocated by the storage manager is initialized to that character's EBCDIC equivalent. For example, if you specify 'A' as the *dsa_alloc_value*, stack storage is initialized to X'C1'. A *dsa_alloc_value* of 'F' initializes stack storage to X'C6', 'd' to X'84'.

- Two hex digits without quotes. If you specify two hex digits, any dynamically-acquired stack storage is initialized to that value. For example, if you specify FE as the *dsa_alloc_value*, stack storage is initialized to X'FE'. A *dsa_alloc_value* of 00 initializes stack storage to X'00', FF to X'FF'.
- **NONE**. If you specify NONE, the stack storage is not initialized.

reserve_size

The amount of storage for the Language Environment storage manager to reserve in the event of an out-of-storage condition. You can specify the *reserve_size* value as *n*, *nK*, or *nM* bytes of storage. The amount of storage is rounded to the nearest multiple of 8 bytes.

The default *reserve_size* is 0, so no reserve segment is allocated. If you do not specify a reserve segment and your application exhausts storage, the application terminates with abend 4088 and a reason code of 1024.

If you specify *reserve_size* as 0, no reserve segment is allocated. If you do not specify a reserve segment and your application exhausts storage, the application terminates with abend 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 uses the reserve stack as the new segment and signals a CEEOPD out of storage condition. This allows a user-written condition handler to gain control for this signal and release storage. If the reserve stack segment overflows while this is happening, Language Environment terminates with abend 4088 and reason code of 1004. The reserve stack segment is not freed until thread termination. It is acquired from 31-bit storage if the STACK(,ANY,,) run-time option is set or 24-bit storage when STACK(,BELOW,,) is requested. If a determination is made to activate the reserve stack, the reserve size should be set to a minimum of 32 KB to support Language Environment condition handling and messaging internal routines as well as the user condition handler. When using the reserve stack in a multi-threaded environment, it is recommended that the ALL31(ON) and STACK(,ANY,,) options also be in effect.

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 terminates with abend 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.

CICS consideration

- The default under CICS is STORAGE=((NONE,NONE,NONE,0K),OVR).
- The out-of-storage condition is not raised under CICS.

Usage Notes

- The behavior of the *dsa_alloc_value* sub-option of the STORAGE runtime option will be different for an XPLINK stack. The DSA will only be initialized for routines

STORAGE

that perform an explicit check for stack overflow. (For C/C++, the compiler option XPLINK(NOGUARD) can be used to force the compiler to generate prologs with explicit checks for stack overflow.)

- *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('""')
```

- z/OS UNIX consideration — A reserve stack of the size specified by the *reserve_size* suboption of STORAGE is allocated for each thread.
- COBOL consideration — If using WSCLEAR in VS COBOL II, STORAGE(00,NONE,NONE,0K) is recommended.

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.

TERMTHDACT

Derivation

TERMinating THread ACTions

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 TRACE, UATRACE, DUMP and UADUMP suboptions of TERMTHDACT.

The following CEE3DMP options are passed for TRACE and UATRACE:

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

The following options are passed for DUMP and UADUMP:

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

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.

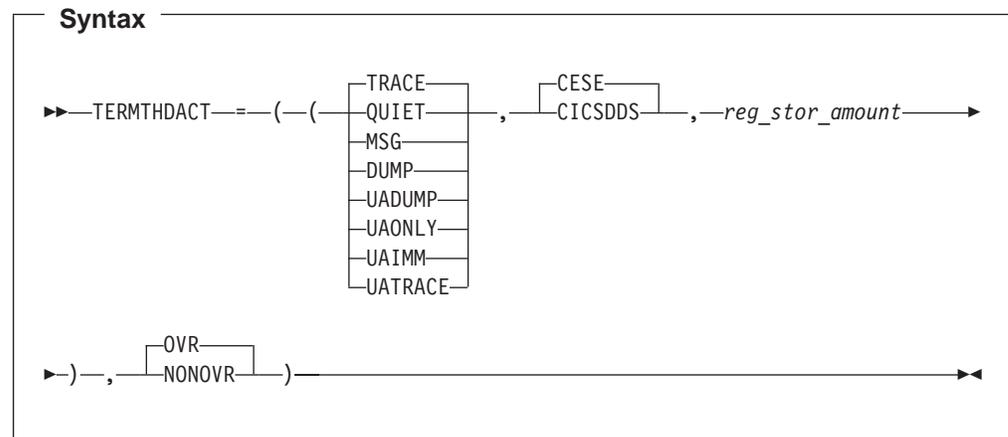
TERMTHDACT

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 normal termination, the following dump contents are generated:

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

Non-CICS Default: TERMTHDACT=((TRACE,CESE,96),OVR)



TRACE

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

QUIET

Specifies that Language Environment does not generate a message when a thread terminates due to an unhandled condition of severity 2 or greater.

MSG

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

DUMP

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

UADUMP

Specifies that when a thread terminates due to an unhandled condition of severity 2 or greater, Language Environment generates a message indicating the cause of the termination, a Language Environment dump, and generates a U4039 abend which allows a system dump of the user address space to be generated. Under non-CICS, if the appropriate DD statement is used, you will get a system dump of your user address space. Under CICS, you will get a CICS transaction dump.

UATRACE

Specifies that when a thread terminates due to an unhandled condition of severity 2 or greater, Language Environment generates a message indicating

TERMTHDACT

the cause of the termination, a trace of the active routines on the activation stack, and generates a U4039 abend which allows a system dump of the user address space to be generated. Under non-CICS, if the appropriate DD statement is used, you will get a system dump of your user address space. Under CICS, you will get a CICS transaction dump.

UAONLY

Specifies that when a thread terminates due to an unhandled condition of severity 2 or greater, Language Environment generates a U4039 abend which allows a system dump of the user address space to be generated. Under non-CICS, if the appropriate DD statement is used, you will get a system dump of your user address space. Under CICS, you will get a CICS transaction dump.

UAIMM

Specifies to Language Environment that prior to condition management processing, for abends and program interrupts that are conditions of Severity 2 or higher, Language Environment will immediately request the operating system to generate a system dump of the original abend/program interrupt of the user address space. Due to an unhandled condition of severity 2 or greater, Language Environment generates a U4039 abend which allows a system dump of the user address space to be generated. Under non-CICS, if the appropriate DD statement is used, you will get a system dump of your user address space. After the dump is taken by the operating system, Language Environment condition manager can continue processing. If the thread terminates due to an unhandled condition of Severity 2 or higher, then Language Environment will terminate as if TERMTHDACT(QUIET) was specified.

Note: For software-raised conditions or signals, UAIMM behaves the same as UAONLY. When TRAP(ON,SPIE) is in effect, UAIMM will yield UAONLY behavior.

CESE

Specifies that Language Environment dump output will be written to the CESE queue.

CICSDDS

Specifies that Language Environment dump output will be written to the CICS transaction dump data set that contains both CICS and CEEDUMP data. For program checks or ABENDs, the CICSDDS option directs Language Environment to place the message output in the CICS dump dataset created for the failure. For software-raised errors, like subscript range exceeded, the CESE queue remains the destination for the output (since there may be no transaction dump for these). CICSDDS can be specified with any of the first TERMTHDACT settings except DUMP and UADUMP. Attempts to request this combination will result in an error in building the options module.

reg_stor_amount

Controls the amount of storage to be dumped around registers. This amount can be in the range from 0 to 256 bytes. The amount specified will be rounded up to the nearest multiple of 32. The default amount is 96 bytes.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is TERMTHDACT=((TRACE,CESE,96),OVR).

TERMTHDACT

- All TERMTHDACT output is written to the data queue based on the setting of CESE or CICSDDS.

See the following tables for help in understanding the results of the different options that are available.

TERMTHDACT

For program checks or ABENDs in a CICS environment:

Table 20. Condition Handling of OCx ABENDS

Options	TERMTHDACT(option,CESE,)	TERMTHDACT(option,CICSDDS,)
QUIET	<ul style="list-style-type: none"> No output. ASRA or user ABEND issued. 	<ul style="list-style-type: none"> No output. ASRA or user ABEND issued.
MSG	<ul style="list-style-type: none"> Message written to CESE queue or MSGFILE. ASRA or user ABEND issued. 	<ul style="list-style-type: none"> Message written to CESE queue or MSGFILE. ASRA or user ABEND issued.
TRACE	<ul style="list-style-type: none"> Message written to CESE queue. Traceback written to CESE queue. ASRA or user ABEND issued. 	<ul style="list-style-type: none"> Message written to CESE or MSGFILE. Traceback included in CICS transaction dump for this ABEND. ASRA or user ABEND issued.
DUMP	<ul style="list-style-type: none"> Message written to CESE queue. Traceback written to CESE queue. CEEDUMP to CESE queue. ASRA or user ABEND issued. 	<ul style="list-style-type: none"> Invalid sub-option combination. Not supported.
UATRACE	<ul style="list-style-type: none"> Message written to CESE queue. Traceback included in CICS transaction dump for this ABEND. U4039 transaction dump in CICS dump data set. ASRA or user ABEND issued. 	<ul style="list-style-type: none"> Message written to CESE queue. Traceback written to CESE queue. U4039 transaction dump in CICS dump data set. ASRA or user ABEND issued.
UADUMP	<ul style="list-style-type: none"> Message written to CESE queue. Traceback written to CESE queue. CEEDUMP written to CESE queue. U4039 transaction dump in CICS dump data set. ASRA or user ABEND issued. 	<ul style="list-style-type: none"> Invalid sub-option combination. Not supported.
UAONLY	<ul style="list-style-type: none"> U4039 transaction dump in CICS dump data set. 	<ul style="list-style-type: none"> No changes in behavior for CICSDDS.
UAIMM	<ul style="list-style-type: none"> U4039 transaction dump in CICS dump data set. 	<ul style="list-style-type: none"> No changes in behavior for CICSDDS.

Note: Program checks end in ASRx (most commonly ASRA) CICS abend with a CICS dump in the dump data set. Abends end with the abend code provided on the EXEC CICS ABEND command with a CICS dump in the dump data set if the NODUMP option was NOT specified.

For software raised errors of severity 2 or higher in a CICS environment:

Table 21. Handling of software raised conditions

Options	TERMTHDACT(option,CESE,)	TERMTHDACT(option,CICSDDS,)
QUIET	<ul style="list-style-type: none"> No output. U4038 abend issued with CANCEL and NODUMP options. 	<ul style="list-style-type: none"> No output. U4038 abend issued with CANCEL and NODUMP options.

Table 21. Handling of software raised conditions (continued)

Options	TERMTHDACT(option,CESE,)	TERMTHDACT(option,CICSDDS,)
MSG	<ul style="list-style-type: none"> • Message written to CESE queue or MSGFILE. • U4038 abend issued. 	<ul style="list-style-type: none"> • Message written to CESE queue or MSGFILE. • U4038 abend issued.
TRACE	<ul style="list-style-type: none"> • Message written to CESE queue or MSGFILE. • Traceback written to CESE queue. • U4038 abend issued. 	<ul style="list-style-type: none"> • Message written to CESE queue or MSGFILE. • Traceback written to CESE queue. • U4038 abend issued.
DUMP	<ul style="list-style-type: none"> • Message written to CESE queue or MSGFILE. • Traceback written to CESE queue. • CEEDUMP written to CESE queue. • U4038 abend issued. 	<ul style="list-style-type: none"> • Invalid sub-option combination. Not supported.
UATRACE	<ul style="list-style-type: none"> • Message written to CESE queue or MSGFILE. • Traceback written to CESE queue. • U4039 transaction dump in CICS dump data set. • U4038 abend issued. 	<ul style="list-style-type: none"> • Message written to CESE queue or MSGFILE. • Traceback written to CESE queue. • U4039 transaction dump in CICS dump data set. • U4038 abend issued.
UADUMP	<ul style="list-style-type: none"> • Message written to CESE queue or MSGFILE. • Traceback written to CESE queue. • CEEDUMP written to CESE queue. • U4039 transaction dump in CICS dump data set. • U4038 abend issued. 	<ul style="list-style-type: none"> • Invalid sub-option combination. Not supported.
UAONLY	<ul style="list-style-type: none"> • U4039 transaction dump in CICS dump data set. • U4038 abend issued. 	<ul style="list-style-type: none"> • No changes in behavior for CICSDDS.
UAIMM	<ul style="list-style-type: none"> • U4039 transaction dump in CICS dump data set. • U4038 abend issued. 	<ul style="list-style-type: none"> • No changes in behavior for CICSDDS.

Notes:

1. See *z/OS Language Environment Run-Time Messages* for more complete details regarding the U4039 abend.
2. When assembling a CEECOPT, CEEROPT, or CEEUOPT, the CICSDDS option cannot be issued with DUMP, or UADUMP. Doing this results in a RC=8 and the following message from CEEXOPT is issued and the setting is forced to TRACE:

8,The TERMTHDACT level setting of DUMP
8,conflicts with the CICSDDS suboption.
8,A level of TRACE or less must be used with CICSDDS.
8,The TERMTHDACT level suboption
8,was set to TRACE.
3. Running with something like TERMTHDACT(TRACE,CICSDDS) in the CEECOPT or CEEROPT and then creating a CEEUOPT without specifying the

TERMTHDACT

second operand (for example, TERMTHDACT(DUMP)) results in the CICS dump data set as the output destination and the following message occurs in the CESE queue:

```
CEE3627I The following messages pertain to the programmer default
run-time options.
CEE3775W A conflict was detected between the TERMTHDACT suboptions
CICSDDS and DUMP.
The TERMTHDACT level setting has been set to TRACE.
```

and the traceback is written to the CICS transaction dump data set.

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.
- COBOL consideration — TERMTHDACT(UADUMP) produces debugging information that is similar to the information produced by previous levels of COBOL.
- z/OS UNIX 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.
- A run-time options report will be generated and placed at the end of the enclave information whenever the TRACE, UATRACE, DUMP and UADUMP options are invoked.

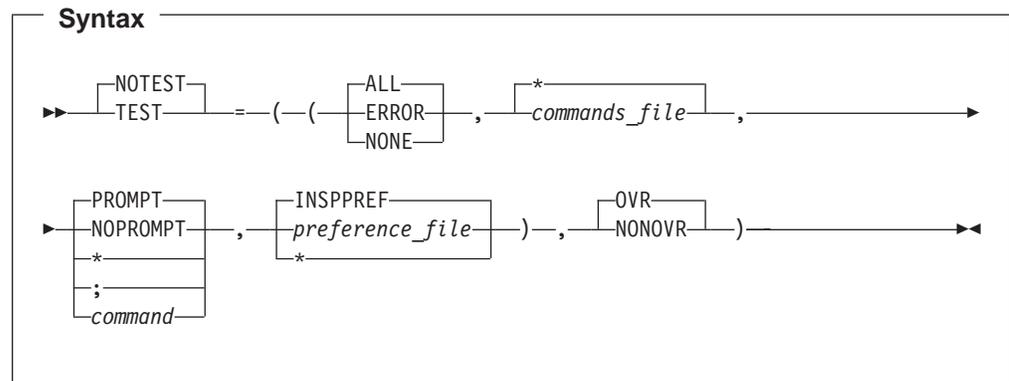
For More Information

- See "TRACE" on page 149, for more information about the TRACE run-time option.
- For more information about the CEE3DMP service and its parameters, see *z/OS Language Environment Programming Reference*.
- See *z/OS Language Environment Programming Guide* for more information about the TERMTHDACT run-time option and condition message.
- For more information about CESE, see *z/OS Language Environment Programming Guide*.

TEST | NOTEST

TEST specifies the conditions under which a debug tool (such as the Debug Tool supplied with z/OS) 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.

Non-CICS Default: NOTEST=((ALL,*,PROMPT,INSPREF),OVR)

**ALL**

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

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

ERROR

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

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

NONE

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

commands_file

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

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

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

* (asterisk — in place of *commands_file*)

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

PROMPT

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

NOPROMPT

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

TEST | NOTEST

* (asterisk — in place of PROMPT/NOPROMPT)

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

; (semicolon — in place of PROMPT/NOPROMPT)

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

command

A character string that specifies a valid debug tool command. The command list can be enclosed in single or double quotes to distinguish it from the rest of the TEST parameter list; it cannot contain DBCS characters. Quotes are needed whenever the command list contains embedded blanks, commas, semicolons, or parentheses. The list can have a maximum of 250 characters.

preference_file

A valid *ddname*, data set name (MVS), or file name (CMS), specifying the preference file to be used. A preference file is a type of commands file that you can use to specify settings for your debugging environment. It is analogous to creating a profile for a text editor, or initializing an S/370 terminal session.

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

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

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

* (asterisk — in place of *preference_file*)

Specifies that no *preference_file* is supplied.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is NOTEST=((ALL,*,PROMPT,INSPREF),OVR).

Usage Notes

- You can specify parameters on the NOTEST option. If NOTEST is in effect when the application gains control, it is interpreted as TEST(NONE,,*,). If Debug Tool is initialized using a CALL CEETEST or equivalent, the initial test level, the initial *commands_file*, and the initial *preference_file* are taken from the NOTEST run-time option setting.
- z/OS UNIX 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.

Performance Consideration

To improve performance, use this option only while debugging.

For More Information

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

THREADHEAP

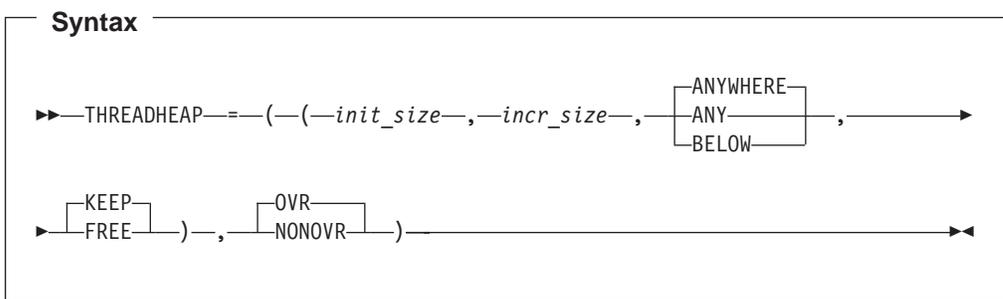
Derivation
THREAD level HEAP storage

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.

Non-CICS Default: THREADHEAP=((4K,4K,ANYWHERE,KEEP),OVR)



init_size

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

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

incr_size

The minimum size of any subsequent increment to the noninitial heap storage is specified in n, nK, or nM. The actual amount of allocated storage is the larger of two values, *incr_size* or the requested size, rounded up to the nearest multiple of 8 bytes.

If you specify *incr_size* as 0, only the amount of the storage needed at the time of the request (rounded up to the nearest 8 bytes) is obtained.

ANYWHERE|ANY

Specifies that the heap storage can be allocated anywhere in storage. If there is no available storage above the line, storage is acquired below the 16 MB line.

The only valid abbreviation of ANYWHERE is ANY.

BELOW

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

THREADHEAP

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.

CICS consideration

- THREADHEAP is ignored under CICS.
- Even though this option is ignored under CICS, the default increment size under CICS has changed from 4 KB (4096 bytes) to 4080 bytes, to accommodate the 16 byte CICS storage check zone.

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 16 MB line based upon the HEAP(,,,initsz24,incrsz24) settings.
- PL/I MTF considerations — The thread-level heap is allocated only in applications that use the PL/I MTF. For PL/I MTF applications, controlled and based variables specified in subtasks are located in the thread-level heap.
If the main program is AMODE 24 and THREADHEAP(,ANY,) is in effect, heap storage is allocated below the 16 MB 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.
- THREADHEAP(4K,4K,ANYWHERE,KEEP) provides behavior compatible with the PL/I TASKHEAP option.
- The initial thread heap segment is never released until the thread terminates.
- THREADHEAP has no effect on C/C++ or VS Fortran MTF applications.

THREADSTACK

Derivation

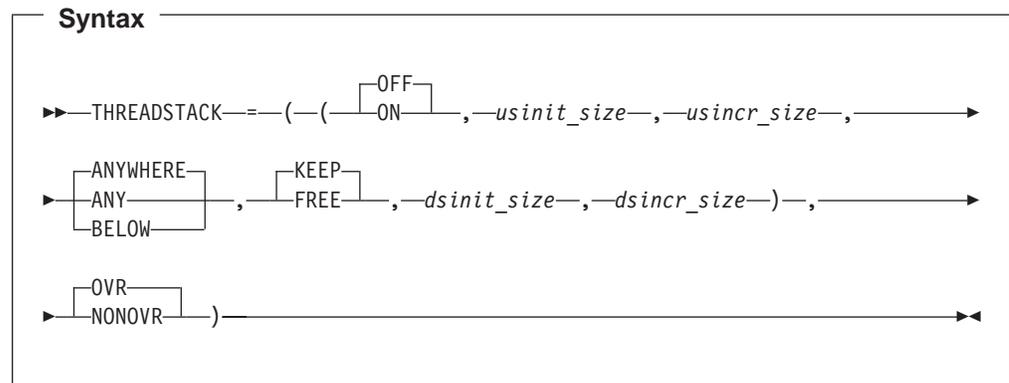
THREAD level STACK storage

THREADSTACK controls the allocation of the thread's stack storage for both the upward and downward-growing stacks, except the initial thread in a multi-threaded application.

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 THREADSTACK option.

Non-CICS Default:

THREADSTACK=((OFF,4K,4K,ANYWHERE,KEEP,128K,128K),OVR)



OFF

Indicates that the allocation suboptions of the STACK run-time option are used for thread stack allocation. Any other suboption specified with THREADSTACK is ignored.

ON

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

usinit_size

Determines the size of the initial upward-growing stack segment. The storage is contiguous. You specify the *usinit_size* value as *n*, *nK*, or *nM* bytes of storage. The actual amount of allocated storage is rounded up to the nearest multiple of 8 bytes.

usinit_size can be preceded by a minus sign. In environments other than, 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 16 MB line.

usincr_size

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

If you specify *usincr_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, *usincr_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.

ANYWHERE | ANY | BELOW

Specifies the storage location. For downward growing stack, this option is ignored and the storage is always placed above 16M.

BELOW

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

THREADSTACK

ANYWHERE|ANY

Specifies that stack storage can be allocated anywhere in storage. If there is no storage available above the line, Language Environment acquires storage below the 16 MB line.

KEEP | FREE

Determines the disposition of the storage increments when the last stack frame in the increment segment is freed.

KEEP

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

FREE

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

dsinit_size

Determines the size of the initial downward growing stack segment. The storage is contiguous. You specify the *init_size* value as n, nK, or nM bytes of storage. The actual amount of allocated storage is rounded up to the nearest multiple of 16 bytes.

dsincr_size

Determines the minimum size of any subsequent increment to the downward growing stack area. You can specify this value as n, nK, or nM bytes of storage. The actual amount of allocated storage is the larger of two values-- *incr_size* or the requested size--rounded up to the nearest multiple of 16 bytes.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- THREADSTACK is ignored under CICS.

Usage Notes

- The *dsinit_size* and *dsincr_size* values are the amounts of storage that can be used for downward growing stack frames (plus the stack header, approximately 20 bytes). The actual size of the storage getmained will be 4K (8K if a 4K page alignment cannot be guaranteed) larger to accommodate the guard page.
- The downward growing stack is only initialized in an XPLINK supported environment, and only when an XPLINK application is active in the enclave. Otherwise the suboptions for the downward growing stack are ignored.
- The THREADSTACK option replaces the **NONIPTSTACK** and **NONONIPTSTACK** options.
- All storage allocated to THREADSTACK segments are 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 THREADSTACK(OFF,...), but they are ignored. If you override the THREADSTACK(OFF,...) suboption with THREADSTACK(ON) and you omit suboptions, then the suboptions you specified with THREADSTACK(OFF,...) remain in effect. If you respecify THREADSTACK(OFF,...) with different suboptions, they override the defaults.

- PL/I MTF consideration — THREADSTACK(ON,4K,4K,BELOW,KEEP,,) provides PL/I compatibility for stack storage allocation and management for each subtask in the application.
- PL/I considerations — For multitasking or multithreaded environments, the stack size for a subtask or non-Initial Process Thread (non-IPT) is taken from the THREADSTACK option unless THREADSTACK(OFF) is specified. THREADSTACK(OFF) specifies that the values in the STACK option be used.
- In the multithreaded environment, you can explicitly specify the stack size in the thread attribute object; it will be used instead of the value specified with THREADSTACK or STACK.

For More Information

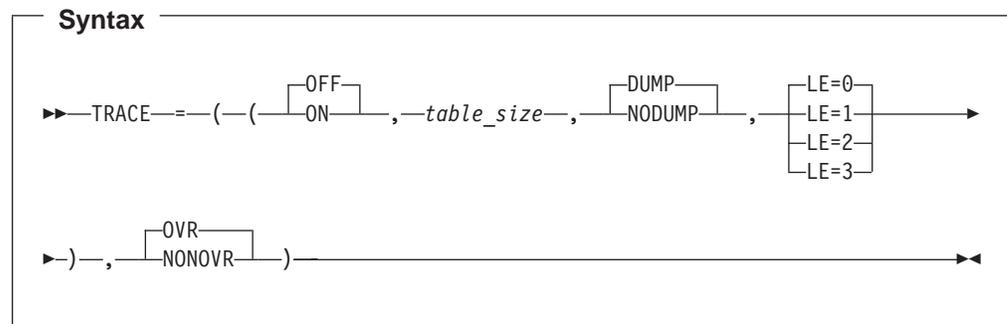
- For more information about the STACK run-time option, see “STACK” on page 130.
- For more information about the ALL31 run-time option, see “ALL31” on page 72.

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.

Non-CICS Default: TRACE=((OFF,4K,DUMP,LE=0),OVR)



OFF

Indicates that the tracing facility is inactive.

ON

Indicates that the tracing facility is active.

table_size

Determines the size of the tracing table as specified in bytes (*nK* or *nM*). The upper limit is 16 MB.

TRACE

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.

CICS consideration

- The default under CICS is TRACE=((OFF,4K,DUMP,LE=0),OVR).

Usage Notes

- 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.
- Under abnormal termination, the following dump contents are generated:
 - TERMTHDACT(TRACE) — generates a dump containing the trace table and the traceback and options report.
 - 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)and an options report.
 - TERMTHDACT(UADUMP) — generates a system dump of the user address space and an options report.

For More Information

- For more information about the dump contents, see “TERMTHDACT” on page 136.

- For more information about using the tracing facility, see *z/OS Language Environment Debugging Guide*.

TRAP

TRAP specifies how Language Environment programs handle abends and program interrupts.

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:

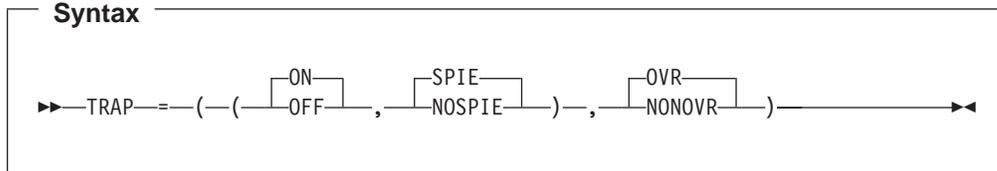
Table 22. TRAP Run-Time Option Settings

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. TRAP is set OFF if both options are negative.
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.

Non-CICS Default: TRAP=((ON,SPIE),OVR)

Syntax



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.

SPIE

SPIE specifies that Language Environment issue an ESPIE macro to handle program interrupts. The SPIE sub-option is ignore when specified with the OFF sub-option.

TRAP

NOSPIE

NOSPIE specifies that Language Environment will NOT issue the ESPIE macro. When you specify the ON sub-option, Language Environment will handle program interrupts and abends via an ESTAE. The NOSPIE sub-option is ignored when specified with the OFF sub-option.

Due to the restrictions and side-effects when running TRAP(OFF) stated in the usage notes below, IBM highly recommends running TRAP(ON,SPIE) in all environments.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is TRAP=((ON,SPIE),OVR)
- Since Language Environment never sets a SPIE or STAE, the SPIE|NOSPIE sub-option is ignored on CICS.

Usage Notes

- The SPIE | NOSPIE run-time option offered by C and PL/I does not affect the TRAP sub-options SPIE and NOSPIE.
- 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, an ESPIE is not issued, but an 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 z/OS UNIX, 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 z/OS C++ try/throw/catch condition handling mechanisms to work.
- When TRAP(ON) is in effect, and the abend code is in the CEEAUE_A_AB_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(ON,NOSPIE) is specified, Language Environment will handle program interrupts and abends via an ESTAE. This feature is useful when you do not want Language Environment to issue an ESPIE macro. If you do not want Language Environment to issue an ESPIE, you must specify TRAP(OFF).
When TRAP(OFF), (TRAP(OFF,SPIE) or TRAP(OFF,NOSPIE) is specified and there is a program interrupt, the user exit for termination is not driven.
- z/OS UNIX consideration — The TRAP option applies to the entire enclave and all threads within.

For More Information

- See “ABTERMENC” on page 69 for more information about the ABTERMENC run-time option.
- See *z/OS Language Environment Programming Reference* for more information about the CEESGL callable service, or the CEEHDLR callable service.
- See *z/OS Language Environment Programming Guide* for more information about the CEEBXITA assembler user exit.

UPSI (COBOL Only)

Derivation
User Programmable Status Indicator

UPSI sets the eight UPSI switches on or off for applications that use COBOL programs.

Non-CICS Default: UPSI=((00000000),OVR)

Syntax

▶▶ UPSI= (((nnnnnnnn)) , (OVR / 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.

UPSI

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is UPSI=((00000000),OVR).

Usage Note

- When you specify this option in CEEDOPT(CEECOPT), CEEROPT 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.

For More Information

- For more information on how COBOL programs access the UPSI switches, see *Enterprise COBOL for z/OS and OS/390 Programming Guide* or *COBOL for OS/390 & VM Programming Guide*.

USRHDLR

Derivation

USeR condition HanDLeR

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.

Non-CICS Default: NOUSRHDLR=((),OVR)

Syntax

►► NOUSRHDLR USRHDLR = ((lmname , lmname2) , 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.

lmname2

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 to get control after the enablement phase and before any other user condition handler.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

CICS consideration

- The default under CICS is NOUSRHDLR=((),OVR).
- When specifying USRHDLR under CICS, *Imname* and *Imname2* must be defined in the CICS PPT.

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 *Imname* is invoked for conditions that are still unhandled after being presented to condition handlers for the main program.
- The user condition handler *Imname2* is invoked for each condition after the condition completes the enablement phase but before any other registered user condition handlers is given control.
- Restriction—If user condition handler *Imname* is in effect, it is unsupported to resume execution in the program in which the condition occurs. This includes calls in the condition handler to CEEMRCR and CEEMRCE. This restriction does not apply to user condition handler *Imname2*.

Imname2 is the second parameter added to the USRHDLR run-time option. It specifies a user written handler that is commonly referred to as the Super Condition Handler. It is invoked like handlers registered by CEEHDLR, so you can resume execution in the program in which the condition occurs. *Imname*, the first parameter of the USRHDLR run-time option still has the restriction that it cannot resume execution in the program in which the condition occurs. Thus the restriction for not being able to use CEEMRCE and CEEMRCR is for *Imname* only.

- 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 to handle conditions at stack frame 0 using the USRHDLR run time option is not presented to any other user condition handler.
- The loading of the user condition handlers *Imname* and *Imname2* occurs only when that user condition handler needs to be invoked the first time.
- If the load of either *Imname* or *Imname2* fails, an error message is issued.

For More Information

- For information on registering a user condition handler and its interfaces, see the CEEHDLR callable service in *z/OS Language Environment Programming Reference*.

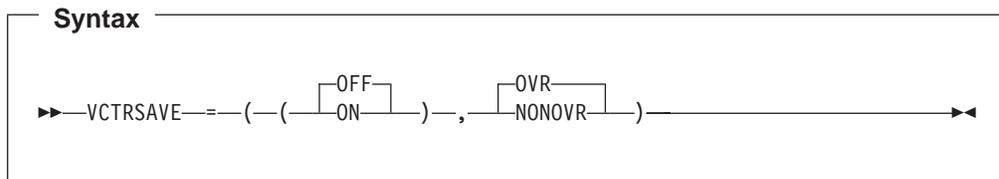
VCTRSAVE**Derivation**

VeCToR environment to be SAVEd

VCTRSAVE specifies whether any language in the application uses the vector facility when user-written condition handlers are called.

Non-CICS Default: VCTRSAVE=((OFF),OVR)

VCTRSAVE



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.

CICS consideration

- VCTRSAVE is ignored under CICS.

Usage Note

- z/OS UNIX consideration — The VCTRSAVE option applies to the entire enclave and all threads within.

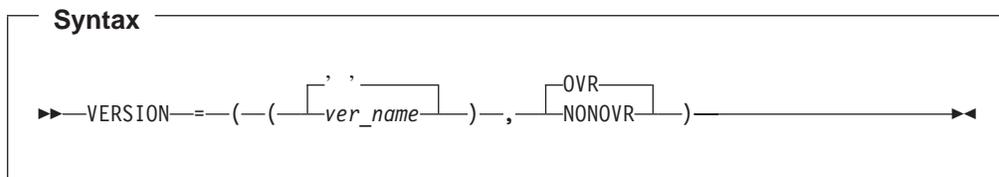
Performance Considerations

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.

VERSION

The VERSION option specifies the version of the RTLS logical library used for finding RTLS version-controlled modules. If not defaulted to an empty string (''), the *ver_name* must be defined in the CSVRTLxx PARMLIB member as a version of the logical library specified by the LIBRARY option.

Non-CICS Default: VERSION=((' '),OVR)



' ' Use the default version of the logical library. The CSVRTLxx PARMLIB member defines a default version for each logical library.

ver_name

The VERSION name must be a string from 1 to 8 characters long. The only

VERSION

valid characters are A-Z, 0-9, #, \$, @, ., -, +, or _ . The code points for the variant characters (A-Z, 0-9, #, \$, @, ., -, +, or _) are assumed to be in code pages 01047 or 00037.

OVR

Specifies that the option can be overridden.

NONOVR

Specifies that the option cannot be overridden.

If this option is specified as ' ', or is defaulted to ' ', the default version of the logical library is used. This default version is defined in the CSVRTLxx PARMLIB member.

CICS consideration

- VERSION is ignored under CICS.

Usage Notes

- This option is ignored when RTLS(OFF) is in effect.
- If CEEGINIT is not available in the LNKLST, LPA, JOBLIB, STEPLIB, or TASKLIB when the Language Environment is invoked, the VERSION option has no effect. RTLS will not be used, but the VERSION option value will appear in the options report.

XUFLOW

Derivation

eXponent Under FLOW

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.

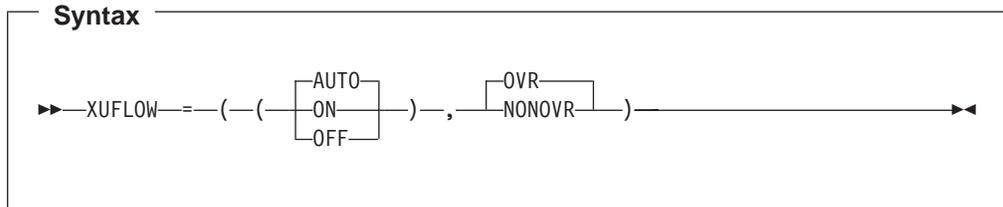
The underflow setting is determined at enclave initialization and is updated when new languages are introduced into the application (via fetch or dynamic call, for example). Otherwise, it does not vary while the application is running.

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 program does not cause a condition to be raised.

Non-CICS Default: XUFLOW=((AUTO),OVR)

XUFLOW



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.

CICS consideration

- The default under CICS is XUFLOW=((AUTO),OVR).

Usage Notes

- PL/I consideration — When setting XUFLOW to OFF, be aware that the semantics of PL/I require the underflow to be signaled.
- z/OS UNIX consideration — The XUFLOW option applies to the entire enclave and all threads within.

Appendix A. Customizing Language Environment run-time options using z/OS msys for Setup

In z/OS Release 1.3, z/OS Managed System Infrastructure for Setup (msys for Setup) offers support for customizing Language Environment run-time options. It can significantly reduce the complexity of customization by:

- Guiding you in defining your Language Environment needs and generating an appropriate set of run-time options based on IBM-recommended settings.
- Allowing you to further customize advanced options, if needed.
- Allowing you to customize region specific run-time options for CICS and IMS regions which will produce CEECOPT, CEEDOPT and CEEROPT members.

Who should use msys for Setup?

Generally, msys for Setup is intended for users customizing Language Environment for the first time or migrating to a new release. However, msys for Setup can also be used if your system changes or if you want to customize additional regions. Once msys for Setup has been used once, you should continue to use msys for Setup for additional changes. When you migrate to a new release, msys for Setup will make the appropriate changes for any new IBM-recommended defaults.

What is the Language Environment customization task?

The Language Environment customization task is divided into multiple steps: CUSTOMIZE, UPDATE, and COMMIT. During the CUSTOMIZE step, the user is asked to enter configuration data for CICS, non-CICS and region-specific run-time options through a series of self-explanatory msys panels. The first time a user tries to CUSTOMIZE, he or she will be directed to the Language Environment msys wizard, which will ask the minimum set of questions needed to set up a basic Language Environment configuration. From there, the user is able to further customize more advanced options by entering information into the property sheets. After the wizard is successfully completed, the user will always be directed to the main property sheet panel, but will have the ability to re-run the wizard, and reset msys values to the default.

Upon completion of the CUSTOMIZE step, this data will be stored in LDAP. When the UPDATE command is issued, msys constructs the information from LDAP and stores the results as a CEECOPT/CEEDOPT member in a user-specified data set. If the user has specified any CEEROPT members, the UPDATE task will construct a file for each set of options in the /tmp directory.

During the COMMIT step, the files will be submitted from /tmp and the resulting load modules will be placed in the data set specified by the user in the msys panels. Upon successful completion of the COMMIT step, the user will still need to take action to activate these changes. The user can click "Browse User Actions" to find out what further steps need to be taken.

Recommendations when using msys for Setup for Language Environment customization

msys for Setup provides many Language Environment settings based on best practices. While you may change these default settings, it is recommended that you use the values provided. When adjusting settings, the msys for Setup help will provide you with background on each run-time option as well as suggestions for optimal settings.

Restrictions when using msys for Setup for Language Environment customization

The following run-time options are set to IBM-recommended defaults and cannot be further modified with msys for Setup:

- ABPERC
- AIXBLD
- ARGPARSE
- CBLQDA
- DEBUG
- ENV
- ERRUNIT
- FILEHIST
- FILETAG
- FLOW
- LIBRARY
- MSGFILE
- MSGQ
- PC
- PLIST
- POSIX
- PROFILE
- PRTUNIT
- PUNUNIT
- RDRUNIT
- REDIR
- RPTOPTS
- RTEREUS
- RTLS
- STORAGE
- TEST
- TRACE
- TRAP
- VCTRSAVE
- VERSION
- XUFLOW

Additionally, msys for Setup imposes practical limits to certain settings that are different from Language Environment limits. Consult the msys for Setup help panels for a full description of limits on individual settings.

Note: If you change any run-time options without using msys for Setup, the changes will not be recognized by msys for Setup.

Where to find information about msys for Setup

For more information about msys for Setup, see *z/OS Managed System Infrastructure for Setup User's Guide*.

For more Information about Language Environment customization, see *z/OS Language Environment Customization*.

Appendix B. Using Fortran with Language Environment

This appendix provides information for tuning and customizing your Language Environment Fortran run-time routines within Language Environment. The customization information in this appendix is intended to help you enhance system performance and provide certain I/O characteristics.

This appendix contains the following sections:

- Customizing for Fortran applications link-edited with Language Environment
- “Customizing for Fortran Applications Link-Edited with VS FORTRAN” on page 169
- “Customizing Fortran LIBPACKs” on page 184

Customizing for Fortran applications link-edited with Language Environment

This section provides information on how to customize Language Environment for Fortran applications that are link-edited with Language Environment. You can customize, or not customize:

- Unit Attribute Table default values (See Changing the unit attribute table default values.)
- Language Environment run-time options (See Chapter 4, “Customizing Language Environment Run-Time Options” on page 19.)

For information on customizing Language Environment if you have Fortran applications that were link-edited with VS FORTRAN Version 1 or 2 for running in load mode, see “Customizing for Fortran Applications Link-Edited with VS FORTRAN” on page 169.

Changing the unit attribute table default values

Module AFHOUTAG contains the Unit Attribute Table defaults and DCB information for each I/O unit. You can accept the IBM-supplied defaults, shown in Figure 13 on page 168, or you can supply your own defaults. To customize AFHOUTAG for your site, use the IBM-supplied job AFHWEUAT, and modify the AFHOUTCM, AFHOUNTM, and AFHODCBM macro instructions in an SMP/E USERMOD. The following sections describe the syntax and operands of the macro instructions.

Starting the unit attribute table definition using the AFHOUTCM macro

Use the AFHOUTCM macro to start and to end the Unit Attribute Table definition. In addition, you can specify default values for information required by the run-time input/output routines of the Fortran component of Language Environment. This section shows the syntax of the operands used for starting the Unit Attribute Table definition.

Syntax of AFHOUTCM Macro Instruction

```
AFHOUTAG  AFHOUTCM  [UNTABLE={ highunit 99 } ]  
[,DEVICE={ device-name | SYSDA } ]
```

UNTABLE=*highunit*

Specifies the largest unit number that can be used in any Fortran program in I/O statements other than the CLOSE and INQUIRE statements. *highunit* must be an integer between 8 and 2000, inclusive. If the UNTABLE parameter is omitted, the default value of *highunit* is 99.

DEVICE=*device-name*

Specifies where dynamically allocated data sets are placed if there is no overriding value given through an invocation of the FILEINF callable service. *device-name* can be a unit address, a group name, or a device type for a DASD device. A unit address is 3 or 4 hexadecimal digits consisting of the channel, control unit, and device number. A group name is any name that is defined during MVS system generation for a DASD device such as SYSDA or DISK. The device type is the IBM-supplied name such as 3380 or 3390.

If the DEVICE parameter is omitted, the default value is SYSDA.

Associating Units with DCB Characteristics Using the AFHOUNTM Macro

Use the AFHOUNTM macro to specify a single unit, or group of units, that is to be associated with a set of DCB default values. Use the AFHOUNTM macro in conjunction with the AFHODCBM macro.

Syntax of AFHOUNTM Macro Instruction

```
AFHOUNTM { unitno | (unitno, qty) | RDRUNIT | PRTUNIT | PUNUNIT }
,DCBSET=label
```

unitno

The unit number, or the first in a series of consecutive unit numbers, for which the set of default DCB characteristics referenced by the DCBSET parameter is to be applied. If *unitno* is the number of the error message unit (or if the error message unit is included in the range covered by *qty*, following), the specification is ignored for the error message unit.

qty

The number of consecutive unit numbers, beginning with *unitno*, for which the set of default DCB characteristics referenced by the DCBSET parameter is to be applied.

RDRUNIT

Indicates that the set of default DCB characteristics referenced by the DCBSET parameter is to be applied to the standard input unit. The *standard input unit* is the unit to which a READ statement applies when the unit identifier is given as *. The number of the standard input unit is the value given by the RDRUNIT run-time option or its default.

Even though there may also be a AFHOUNTM macro instruction that refers to the standard input unit by its unit number (that is, with the *unitno* form of specification), the AFHOUNTM with the RDRUNIT parameter takes precedence and applies to the standard input unit.

If there is no AFHOUNTM macro instruction with a RDRUNIT parameter, then the default DCB characteristics for the standard input unit are those referenced by a AFHOUNTM macro instruction that refers to this unit with the *unitno* form of specification.

PRTUNIT

Indicates that the set of default DCB characteristics referenced by the DCBSET parameter is to be applied to the print unit.

The *print unit* is one of the standard output units and is the unit to which either a WRITE statement with a unit identifier of * or a PRINT statement applies. The number of the print unit is the value given by the PRTUNIT run-time option or its default if the number of the print unit is different than the number of the error message unit.

The *error message unit* the unit to which output such as error messages and dumps from services such as CDUMP and SDUMP is directed. The number of the error message unit is the value given by the ERRUNIT run-time option or its default.

The *punch unit* is one of the standard output units and is the unit to which a PUNCH statement applies. The number of the punch unit is the value given by the PUNUNIT run-time option or its default.

Even though there may also be a AFHOUNTM macro instruction that refers to the print unit by its unit number (that is, with the *unitno* form of specification), the AFHOUNTM with the PRTUNIT parameter takes precedence and applies to the print unit.

If there is no AFHOUNTM macro instruction with a PRTUNIT parameter and if the print unit and the error message units are different units, then the default DCB characteristics for the print unit are those referenced by a AFHOUNTM macro instruction that refers to this unit with the *unitno* form of specification.

PUNUNIT

Indicates that the set of default DCB characteristics referenced by the DCBSET parameter is to be applied to the punch unit. The *punch unit* is one of the standard output units and is the unit to which a PUNCH statement applies. The number of the punch unit is the value given by the PUNUNIT run-time option or its default.

Even though there may also be a AFHOUNTM macro instruction that refers to the punch unit by its unit number (that is, with the *unitno* form of specification), the AFHOUNTM with the PUNUNIT parameter takes precedence and applies to the punch unit.

If there is no AFHOUNTM macro instruction with a PUNUNIT parameter, then the default DCB characteristics for the punch unit are those referenced by a AFHOUNTM macro instruction that refers to this unit with the *unitno* form of specification.

DCBSET=*label*

The identifier of the DCB attributes to associate with this unit, set of units, or standard I/O unit. This is the name given in the associated AFHODCBM macro instruction.

Specifying the DCB Characteristics Using the AFHODCBM Macro

Use the AFHODCBM macro to specify default DCB information for the I/O units that have a DCBSET=*label* parameter on the AFHOUNTM macro.

Syntax of AFHODCBM Macro Instruction

```
[label] AFHODCBM [,SFBUFNO=number | 2]  
[,SUBUFNO=number | 2]  
[,SFBLKSI=number | 800]  
[,SUBLKSI=number | 800]  
[,SFLRECL=number | 800]  
[,SULRECL=number | -1]  
[,SFRECFM=char | U]  
[,SURECFM=char | VS]  
[,SFMAXRE=number | 100]  
[,SUMAXRE=number | 100]  
[,DMAXRE=number | 100]
```

label

The name specified in the DCBSET parameter of one or more AFHOUNTM macro instructions to relate the I/O units to this set of DCB default values.

If *label* is omitted, the DCB data is assigned to all units defined in the Unit Attribute Table by the AFHOUTCM macro instruction that do not have a AFHOUNTM macro instruction. If any of the units in the Unit Attribute Table do not have their own AFHOUNTM macro instruction, then you must provide a AFHODCBM macro instruction **without a label** to apply defaults to these units.

SFBUFNO=*number* | 2

Specifies the default value for the number of buffers for sequential formatted files on DASD or tape. *number* must be a value greater than or equal to 1 and less than or equal to 255.

SUBUFNO=*number* | 2

Specifies the default value for the number of buffers for sequential unformatted files on DASD or tape. *number* must be a value greater than or equal to 1 and less than or equal to 255.

SFBLKSI = *number* | 800

Specifies the block size for sequential formatted files. *number* is an integer expression of length 4 bytes; valid range of the blocksize is from 1 to 32760.

SUBLKSI = *number* | 800

Specifies the block size for sequential unformatted files. *number* is an integer expression of length 4 bytes; valid range of the blocksize is from 1 to 32760.

SFLRECL = *number* | 800

Specifies the logical record length for sequential formatted files. *number* is an integer expression of length 4 bytes; valid range is from 1 to 32756 for variable record formats (SURECFM= V, VA, VB, or VBA), or 1 to 32760 for all other record formats.

SULRECL = *number* | -1

Specifies the logical record length for sequential unformatted files. *number* is an integer expression of length 4 bytes; valid range is from 1 to 32756 for variable record formats (SURECFM= V, VA, VB, VBA, VS, or VBS), or 1 to 32760 for all other record formats or -1, which specifies an unlimited record length. -1 is valid for SURECFM=VS or VBS formats.

SFRECFM = *char* | U

Specifies the record format for sequential formatted files. The value of *char*

must be F, FA, FB, FBA, V, VA, VB, VBA, U, or UA. For more information on I/O, see *VS FORTRAN Version 2 Programming Guide for CMS and MVS*

SURECFM = char | VS

Specifies the record format for sequential unformatted files. The value of *char* must be F, FA, FB, FBA, V, VA, VB, VBA, VS, VBS, U, or UA. For more information on I/O, see *VS FORTRAN Version 2 Programming Guide for CMS and MVS*

SFMAXRE = number | 100

Specifies the amount of space to be converted into blocks in a sequential formatted file. It is only valid for new DASD files; if specified for an existing file, it will be ignored. *number* is an integer expression of length 4. See MAXREC in *VS FORTRAN Version 2 Programming Guide for CMS and MVS* for information on how space is converted to blocks.

SUMAXRE = number | 100

Specifies the amount of space to be converted into blocks in a sequential unformatted file. It is only valid for new DASD files; if specified for an existing file, it will be ignored. *number* is an integer expression of length 4. See MAXREC in *VS FORTRAN Version 2 Programming Guide for CMS and MVS* for information on how space is converted to blocks.

DMAXRE = number | 100

Specifies the amount of space to be converted into blocks in a direct file. It is only valid for new DASD files; if specified for an existing file, it will be ignored. *number* is an integer expression of length 4. See *VS FORTRAN Version 2 Programming Guide for CMS and MVS* for information on how space is converted to blocks.

CAUTION: If you change the IBM-supplied default DCB values, any existing Fortran programs that depend on the original defaults might not work.

Ending the Unit Attribute Table Definition Using the AFHOUTCM Macro

The AFHOUTCM macro is used to start and to end the Unit Attribute Table definition. This section shows the syntax of the operands used for ending the Unit Attribute Table definition.

Syntax of AFHOUTCM Macro Instruction: Final Statement

```
AFHOUTCM TYPE=FINAL
```

IBM-Supplied Unit Attribute Table Default Values

The macro instructions shown in Figure 13 are provided in the module AFHOUTAG. This module is used to set up the IBM-supplied default values for the standard I/O units, and file characteristics such as the DCB information.

```

AFHOUTAG AFHOUTCM UNTABLE=99,
          DEVICE=SYSDA

AFHOUNTM RDRUNIT,DCBSET=DCBRDR
AFHOUNTM PRTUNIT,DCBSET=DCBPRT
AFHOUNTM PUNUNIT,DCBSET=DCBPUN

DCBRDR AFHODCBM SFRECFM=F,SFLRECL=80,SFBLKSI=80,
          SURECFM=F,SULRECL=80,SUBLKSI=80

DCBPRT AFHODCBM SFRECFM=UA,SFLRECL=133,SFBLKSI=133

DCBPUN AFHODCBM SFRECFM=F,SFLRECL=80,SFBLKSI=80,
          SURECFM=F,SULRECL=80,SUBLKSI=80

AFHODCBM SFRECFM=U,SFLRECL=800,SFBLKSI=800,SFMAXRE=100,
          SURECFM=VS,SULRECL=-1,SUBLKSI=800,SUMAXRE=100,
          DMAXRE=100

AFHOUTCM TYPE=FINAL

```

Figure 13. IBM-Supplied Macro Instructions

Note: The above format is given for readability purposes. Remember to add the necessary continuation flags in column 72, and to begin continued lines in column 16.

The three AFHOUNTM macro instructions indicate that the standard input unit, the print unit, and the punch unit have the default DCB information provided on the first three AFHODCBM macro instructions. Note that the last AFHODCBM macro does not have a label; its set of defaults apply to all units except the standard I/O units. Refer to *z/OS Language Environment Programming Reference* for more information on the RDRUNIT, ERRUNIT, PRTUNIT, and PUNUNIT run-time options, which are used to specify the unit numbers of these standard I/O units.

Examples of Changing Unit Attribute Table Default Values

The following example shows how you can modify the IBM-supplied defaults for your own environment. You can alter instructions by typing over existing data, or you can remove or add AFHOUNTM and AFHODCBM macro instructions.

Example: In this example, we have specified device name SYSSQ and assigned a unique set of DCB attributes to units 1 through 4 for dynamically allocated data sets.

```

AFHOUTAG AFHOUTCM UNTABLE=99,
          DEVICE=SYSSQ

          AFHOUNTM RDRUNIT,DCBSET=DCBRDR
          AFHOUNTM PRTUNIT,DCBSET=DCBPRT
          AFHOUNTM PUNUNIT,DCBSET=DCBPUN
          AFHOUNTM (1,4),DCBSET=USERDCB

DCBRDR AFHODCBM SFRECFM=F,SFLRECL=80,SFBLKSI=80,
          SURECFM=F,SULRECL=80,SUBLKSI=80

DCBPRT AFHODCBM SFRECFM=UA,SFLRECL=133,SFBLKSI=133

USERDCB AFHODCBM SFRECFM=FB,SFLRECL=50,SFBLKSI=250,
          SFMAXRE=200,SURECFM=FB,SULRECL=50,
          SUBLKSI=250,SUMAXRE=200,DMAXRE=200

DCBPUN AFHODCBM SFRECFM=F,SFLRECL=80,SFBLKSI=80,
          SURECFM=F,SULRECL=80,SUBLKSI=80

          AFHODCBM SFRECFM=U,SFLRECL=800,SFBLKSI=800,SFMAXRE=100,
          SURECFM=VS,SULRECL=-1,SUBLKSI=800,SUMAXRE=100,
          DMAXRE=100

          AFHOUTCM TYPE=FINAL

```

Figure 14. Modified IBM-Supplied Macro Instructions

Note: The above format is given for readability purposes. Remember to add the necessary continuation flags in column 72, and to begin continued lines in column 16.

AFHOUTCM, AFHOUNTM, and AFHODCBM must all be coded, in that order, followed by the AFHOUTCM TYPE=FINAL statement.

Customizing for Fortran Applications Link-Edited with VS FORTRAN

This section contains information on how to customize Language Environment if you have Fortran applications that were link-edited with VS FORTRAN Version 1 or 2 for running in load mode.

If you have such applications, you can customize, or not customize:

- VS FORTRAN Unit Attribute Table defaults (See “Changing the Unit Attribute Table Default Values” on page 170 following.)
- VS FORTRAN run-time option defaults (See “Changing VS FORTRAN Run-Time Option Defaults” on page 175.)
- VS FORTRAN Error Option Table defaults (See “Changing the Error Option Table Defaults” on page 180.)

Note: Language Environment provides a VS FORTRAN compability library for running Fortran applications that are not link-edited with Language Environment.

You can customize Language Environment to provide certain run-time characteristics for Fortran applications that were link-edited with VS FORTRAN for running in load mode. You use macros with the same names as you used in VS FORTRAN Version 2 Release 6. These macros are VSF2UAT, VSF2UNIT, VSF2DCB, VSF2PARAM, and VSF2UOPT. Each of these macros is available in Language Environment with these macro names as aliases for members with

names beginning with AFH5. The use of these macros is identical to that in VS FORTRAN Version 2 Release 6; therefore, if you have assembler language source files that you used in the past, you can use these same source files to customize Language Environment.

Changing the Unit Attribute Table Default Values

Module AFH5VUAT contains the Unit Attribute Table defaults and DCB information for each I/O unit of the VS FORTRAN compatibility library. You can accept the IBM-supplied defaults, shown in Figure 15 on page 174, or you can supply your own defaults. To customize AFH5VUAT for your site, use the IBM-supplied job AFH5VUAT, and modify the VSF2UAT, VSF2UNIT, and VSF2DCB macro instructions in an SMP/E USERMOD. The following sections describe the syntax and operands of the macro instructions.

Starting the Unit Attribute Table Definition Using the VSF2UAT Macro

The VSF2UAT macro is used to start and to end the Unit Attribute Table definition. Fortran component of Language Environment. In addition, you can specify default values for information required by the run-time input/output routines of the VS FORTRAN compatibility library. This section shows the syntax of the operands used for starting the Unit Attribute Table definition.

Syntax of VSF2UAT Macro: Statement Form

```
[name] VSF2UAT [DECIMAL=PERIOD | COMMA]  
[,PUNCH=number | 7 ]  
[,ERRMSG=number | 6 ]  
[,PRINTER=number | 6 ]  
[,READER=number | 5 ]  
[,UNTABLE=number | 99]  
[,DEVICE=device-name | SYSDA]
```

See “Ending the Unit Attribute Table Definition Using the VSF2UAT Macro” on page 173 for the form of VSF2UAT as the final macro instruction.

The IBM-supplied default values are underlined in the following option list. If an option is not specified, its default value will be used.

name

Specifies a name, such as AFBVUAT or AFH5UAT. *name* is ignored, and the CSECT name becomes AFH5VUAT automatically.

DECIMAL = PERIOD | COMMA

Specifies the character to be used as the decimal indicator in printed output.

PUNCH = *number* | 7

Specifies, for LANGLVL(66) only, the standard I/O unit number for the PUNCH statement to send data to the card punch. The number specified must be between 0 and 99 or the value specified for the UNTABLE parameter, for UNTABLE values less than or equal to 99. It must not be the same as the number specified for ERRMSG, PRINTER, or READER.

ERRMSG = *number* | 6

Specifies the standard I/O unit number for the error messages generated by VS FORTRAN Version 2 Library. The number specified must be between 0 and 99 or the value specified for the UNTABLE parameter, for UNTABLE values less

than or equal to 99. It must not be the same as the number specified for PUNCH or READER; it can be the same number specified for PRINTER.

PRINTER = *number* | 6

Specifies the standard I/O unit number for the print statement, and with any WRITE statement specifying an installation-dependent form of the unit. The number specified must be between 0 and 99 or the value specified for the UNTABLE parameter, for UNTABLE values less than or equal to 99. It must not be the same as that specified for PUNCH and READER. It can be the same number specified for ERRMSG.

READER = *number* | 5

Specifies the standard I/O unit number for any READ statement specifying an installation-dependent form of the unit. The number specified must be between 0 and 99 or the value specified for the UNTABLE parameter, for UNTABLE values less than or equal to 99. It must not be the same as the number specified for either PUNCH, ERRMSG, or PRINTER.

UNTABLE = *number* | 99

Specifies the largest unit number you can include in a VS FORTRAN program. It can be specified as any integer between 8 and 2000.

DEVICE = *device-name* | SYSDA

Specifies where dynamically allocated data sets are placed if there is no overriding value given through an invocation of the FILEINF callable service. *device-name* can be a unit address, a group name, or a device type for a DASD device. A unit address is 3 or 4 hexadecimal digits consisting of the channel, control unit, and device number. A group name is any name that is defined during MVS system generation for a DASD device such as SYSDA or DISK. The device type is the IBM-supplied name such as 3380 or 3390.

If the DEVICE parameter is omitted, the default value is SYSDA.

Note: In Fortran, the units described by the PUNCH, ERRMSG, PRINTER and READER parameters are called standard I/O units.

Associating Units with DCB Characteristics Using the VSF2UNIT Macro

Use the VSF2UNIT macro to specify a single unit, or group of units, that is to be associated with a set of DCB default values. The VSF2UNIT macro is used in conjunction with the VSF2DCB macro.

Syntax of VSF2UNIT Macro

```
VSF2UNIT { unitno |  
( unitno [,qty] ) } ,DCBSET = label
```

unitno

Specifies the unit number, or the first in a series of consecutive unit numbers, that are to have DCB default values assigned.

qty

Specifies, if there is more than one, the number of consecutive unit numbers, beginning with *unitno*, that are to have DCB default values assigned.

DCBSET=*label*

Specifies the identifier of the DCB attributes to associate with this unit or set of units. This is the name given in the associated VSF2DCB macro instruction.

Specifying the DCB Characteristics Using the VSF2DCB Macro

Use the VSF2DCB macro to specify DCB default information for the I/O units that have DCBSET=*label* parameter of the VSF2UNIT macro.

Syntax of VSF2DCB Macro

```
[label] VSF2DCB [,SFBUFNO=number | 2]  
[,SUBUFNO=number | 2]  
[,SFBLKSI=number | 800]  
[,SUBLKSI=number | 800]  
[,SFLRECL=number | 800]  
[,SULRECL=number. | -1]  
[,SFRECFM=char | U]  
[,SURECFM=char | VS]  
[,SFMAXRE=number | 100]  
[,SUMAXRE=number | 100]  
[,DMAXRE=number | 100]
```

label

Specified in the VSF2UNIT macro to identify the I/O units that are to be assigned DCB default values.

If *label* is omitted, the DCB data is assigned to all units defined in the default table by the VSF2UAT macro, but which have not been defined by the VSF2UNIT macro. If any of the units defined in the attribute table do not have their own associated DCBSET coded, you must provide a VSF2DCB macro **without a label** to apply defaults to these units.

SFBUFNO=*number* | 2

Specifies the default value for the number of buffers for sequential formatted files on DASD or tape. *number* must be a value greater than or equal to 1 and less than or equal to 255.

SUBUFNO=*number* | 2

Specifies the default value for the number of buffers for sequential unformatted files on DASD or tape. *number* must be a value greater than or equal to 1 and less than or equal to 255.

SFBLKSI = *number* | 800

Specifies the block size for sequential formatted files. *number* is an integer expression of length 4 bytes; valid range of the blocksize is from 1 to 32760.

SUBLKSI = *number* | 800

Specifies the block size for sequential unformatted files. *number* is an integer expression of length 4 bytes; valid range of the blocksize is from 1 to 32760.

SFLRECL = *number* | 800

Specifies the logical record length for sequential formatted files. *number* is an integer expression of length 4 bytes; valid range is from 1 to 32756 for variable record formats (SURECFM= V, VA, VB, or VBA), or 1 to 32760 for all other record formats.

SULRECL = *number* | -1

Specifies the logical record length for sequential unformatted files. *number* is an integer expression of length 4 bytes; valid range is from 1 to 32756 for variable record formats (SURECFM= V, VA, VB, VBA, VS, or VBS), or 1 to 32760 for all other record formats or -1, which specifies an unlimited record length. -1 is valid for SURECFM=VS or VBS formats.

SFRECFM = *char* | U

Specifies the record format for sequential formatted files. The value of *char* must be F, FA, FB, FBA, V, VA, VB, VBA, U, or UA. For more information on I/O, see *VS FORTRAN Version 2 Programming Guide for CMS and MVS*

SURECFM = *char* | VS

Specifies the record format for sequential unformatted files. The value of *char* must be F, FA, FB, FBA, V, VA, VB, VBA, VS, VBS, U, or UA. For more information on I/O, see *VS FORTRAN Version 2 Programming Guide for CMS and MVS*

SFMAXRE = *number* | 100

Specifies the amount of space to be converted into blocks in a sequential formatted file. It is only valid for new DASD files; if specified for an existing file, it will be ignored. *number* is an integer expression of length 4. See MAXREC in *VS FORTRAN Version 2 Programming Guide for CMS and MVS* for information on how space is converted to blocks.

SUMAXRE = *number* | 100

Specifies the amount of space to be converted into blocks in a sequential unformatted file. It is only valid for new DASD files; if specified for an existing file, it will be ignored. *number* is an integer expression of length 4. See MAXREC in *VS FORTRAN Version 2 Programming Guide for CMS and MVS* for information on how space is converted to blocks.

DMAXRE = *number* | 100

Specifies the amount of space to be converted into blocks in a direct file. It is only valid for new DASD files; if specified for an existing file, it will be ignored. *number* is an integer expression of length 4. See *VS FORTRAN Version 2 Programming Guide for CMS and MVS* for information on how space is converted to blocks.

CAUTION: If you change the IBM-supplied default DCB values, the existing Fortran programs that depend on the original defaults may not work. For more information on DCB values, refer to *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

Ending the Unit Attribute Table Definition Using the VSF2UAT Macro

Use the VSF2UAT macro to start and to end the Unit Attribute Table definition. Use the following form of VSF2UAT as the final macro instruction in the Unit Attribute Table definition.

Syntax of VSF2UAT Macro: Final Statement

```
VSF2UAT    TYPE=FINAL
```

TYPE = FINAL

Is the required last statement of the VSF2UAT macro.

IBM-Supplied Unit Attribute Table Default Values

The macro instructions shown in Figure 15 on page 174 are provided in the module AFH5VUAT. This module is used to set up the IBM-supplied default values for the standard I/O units, and file characteristics such as the DCB information.

```

AFH5VUAT VSF2UAT UNTABLE=99,
                DECIMAL=PERIOD,
                READER=5,
                ERRMSG=6,
                PRINTER=6,
                PUNCH=7,
                DEVICE=SYSDA

                VSF2UNIT 5,DCBSET=DCBRDR
                VSF2UNIT 6,DCBSET=DCBPRT
                VSF2UNIT 7,DCBSET=DCBPUN

DCBRDR VSF2DCB  SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                SURECFM=F,SULRECL=80,SUBLKSI=80

DCBPRT VSF2DCB  SFRECFM=UA,SFLRECL=133,SFBLKSI=133

DCBPUN VSF2DCB  SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                SURECFM=F,SULRECL=80,SUBLKSI=80

                VSF2DCB  SFRECFM=U,SFLRECL=800,SFBLKSI=800,SFMAXRE=100,
                SURECFM=VS,SULRECL=-1,SUBLKSI=800,SUMAXRE=100,
                DMAXRE=100

                VSF2UAT  TYPE=FINAL

```

Figure 15. IBM-Supplied Macro Instructions

Note: The above format is given for readability purposes. Remember to add the necessary continuation flags in column 72, and to begin continued lines in column 16.

The three VSF2UNIT macro instructions indicate that units 5, 6, and 7 have the default DCB information provided on the first three VSF2DCB macro instructions. Note that the last VSF2DCB macro does not have a label; its set of defaults apply to all units except 5, 6, and 7. Refer to *VS FORTRAN Version 2 Programming Guide for CMS and MVS* for more information on the RDRUNIT, ERRUNIT, PRTUNIT, and PUNUNIT run-time options, which are used to specify the unit numbers of these standard I/O units.

Examples of Changing Unit Attribute Table Default Values

The following examples show how you could modify the IBM-supplied defaults for your own environment. You can alter instructions by typing over existing data, or you can add more VSF2UNIT and VSF2DCB macro instructions.

Example 1: In this example, we have specified device name SYSSQ for dynamically allocated data sets and assigned a unique set of DCB attributes to units 1 through 4. The DCB Information for both sequential formatted and unformatted files written on these units is indicated in the first VSF2DCB macro instruction (“USERDCB”) shown in Figure 16 on page 175.

```

AFH5VUAT  VSF2UAT  DEVICE=SYSSQ
          VSF2UNIT (1,4),DCBSET=USERDCB
          VSF2UNIT 5,DCBSET=DCBRDR
          VSF2UNIT 6,DCBSET=DCBPRT
          VSF2UNIT 7,DCBSET=DCBPUN

USERDCB   VSF2DCB   SFRECFM=FB,SFLRECL=50,SFBLKSI=250,SFMAXRE=200,
          SURECFM=FB,SULRECL=50,SUBLKSI=250,SUMAXRE=200,
          DMAXRE=200

DCBRDR    VSF2DCB   SFRECFM=F,SFLRECL=80,SFBLKSI=80,
          SURECFM=F,SULRECL=80,SUBLKSI=80

DCBPRT    VSF2DCB   SFRECFM=UA,SFLRECL=133,SFBLKSI=133

DCBPUN    VSF2DCB   SFRECFM=F,SFLRECL=80,SFBLKSI=80,
          SURECFM=F,SULRECL=80,SUBLKSI=80

          VSF2DCB   SFRECFM=U,SFLRECL=800,SFBLKSI=800,SFMAXRE=100,
          SURECFM=VS,SULRECL=-1,SUBLKSI=800,SUMAXRE=100,
          DMAXRE=100

          VSF2UAT   TYPE=FINAL

```

Figure 16. Modified IBM-Supplied Macro Instructions

Note: The above format is given for readability purposes. Remember to add the necessary continuation flags in column 72, and to begin continued lines in column 16.

VSF2UAT, VSF2UNIT, and VSF2DCB must all be coded, in that order, followed by the VSF2UAT TYPE=FINAL statement.

Example 2: If you want to change the unit numbers of the standard input unit, the error message unit, the print unit, and the punch unit, to 1, 2, 3, 4, respectively, modify the IBM-supplied macros as shown in Figure 17.

```

AFH5VUAT  VSF2UAT  DECIMAL=PERIOD,
                  READER=1,
                  ERRMSG=2,
                  PRINTER=3,
                  PUNCH=4,
                  DEVICE=SYSDA

          VSF2UNIT 1,DCBSET=DCBRDR
          VSF2UNIT 2,DCBSET=DCBTERM
          VSF2UNIT 3,DCBSET=DCBPRT
          VSF2UNIT 4,DCB=DCBPUN

```

Figure 17. Modified IBM-Supplied Macro Instructions

Changing VS FORTRAN Run-Time Option Defaults

Module AFH5GPRM contains the set of run-time option defaults for running with the VS FORTRAN compatibility library. You can accept the IBM-supplied defaults, shown in this section, or you can supply your own defaults. To customize AFBVGPRM for your site, use the IBM-supplied job AFHWVPRM, and modify the VSF2PARAM macro instruction in an SMP/E USERMOD. The syntax and operands of the VSF2PARAM macro instruction are described in this section.

Use the AFH5PARM macro to change the IBM-supplied default values for VS FORTRAN run-time options. The default values you assign are assumed if you do not override them.

There are no operands to set the default values for the run-time options AUTOTASK, PARALLEL, and PARTRACE; therefore these options cannot be changed during installation. They can, however, be changed at run time.

There are no operands in the VSF2PARM macro to set the default values for the run-time options ERRUNIT, RDRUNIT, PRTUNIT, and PUNUNIT. The default I/O unit values for these units can be changed during installation through the Unit Attribute Table.

Syntax of VSF2PARM Macro Instruction

```
VSF2PARM SCOPE = GLOBAL  
[,ABSDUMP | NOABSDUMP]  
[,CNVIOERR | NOCNVIOERR]  
[,DEBUG | NODEBUG]  
[,DEBUNIT(s1[,s2,...]) | NODEBUNIT]  
[,ECPACK | NOECPACK]  
[,FAIL(ABEND | RC | ABENDRC)]  
[,FILEHIST | NOFILEHIST]  
[,INQPCOPN | NOINQPCOPN]  
[,IOINIT | NOIOINIT]  
[,OCSTATUS | NOOCSTATUS]  
[,RECPAD[(ALL)] | NORECPAD]  
[,SPIE | NOSPIE]  
[,STAE | NOSTAE]  
[,XUFLOW | NOXUFLOW]
```

The IBM-supplied default values are underlined in the following option list. If an option is not specified, its default will be used, with the exception of the SCOPE option, which must always be specified.

SCOPE = GLOBAL

Required to replace the global run-time options table AFBVGPRM, which supplies default values for all users of the VS FORTRAN compatibility library.

There is no default value for this option. Thus SCOPE=GLOBAL must always be specified.

ABSDUMP | NOABSDUMP

Specifies whether or not the post-abend symbolic dump information is printed.

ABSDUMP

Causes the post-abend symbolic dump information to be printed in the event of an abnormal termination.

NOABSDUMP

Suppresses the printing of the post-abend symbolic dump information.

CNVIOERR | NOCNVIOERR

Specifies whether input conversion errors will be treated as I/O errors.

CNVIOERR

Causes ERR and IOSTAT to recognize conversion errors as I/O errors.

NOCNVIOERR

Causes conversion errors not to be treated as I/O errors. ERR and IOSTAT have no effect for these errors.

DEBUG | NODEBUG

Specifies whether or not interactive debug will be invoked.

Note: This option does not apply to the Language Environment VS FORTRAN compatibility library. If you want to use the VS FORTRAN Interactive Debugger, then run your program with the VS FORTRAN Version 2 library rather than with Language Environment.

DEBUNIT | NODEBUNIT

Specifies whether or not Fortran unit numbers will be treated as if connected to a terminal device.

Note: This option does not apply to the Language Environment VS FORTRAN compatibility library. If you want to use the VS FORTRAN Interactive Debugger, then run your program with the VS FORTRAN Version 2 library rather than with Language Environment.

ECPACK | NOECPACK

Specifies whether a data space should be filled with as many extended common blocks as possible before a new data space is allocated.

ECPACK

Specifies extended common blocks be placed into the fewest possible number of data spaces. This option reduces some of the overhead associated with referencing data spaces.

NOECPACK

Specifies that each extended common block be placed into a separate data space. As a result, reference errors made beyond the bounds of an extended common block might be more easily detected.

FAIL (ABEND | RC | ABENDRC)

Indicates how applications that fail are to be terminated: either by a nonzero return or by an abnormal termination (ABEND). The suboption of the FAIL option may have the following meanings.

ABEND

Causes the program to end by an abnormal termination (ABEND) with a user completion code of 240.

RC

Causes the program to end normally but with a nonzero return code (16).

ABENDRC

Causes the program to end by abnormal termination (ABEND) when failure is because of a condition for which the operating system would usually cause an ABEND; and to end with a nonzero return code when failure is by some condition detected by VS FORTRAN.

FILEHIST | NOFILEHIST

Specifies whether to allow the file definition of a file referred to by a ddname to be changed at run time.

FILEHIST

Causes the history of a file to be used in determining its existence. In particular 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).

When FILEHIST is specified, you cannot change the file definition of a file at run time and have the same results produced as previous VS FORTRAN releases.

NOFILEHIST

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

If you specify NOFILEHIST you should consider:

- **If you change file definitions at run time:** the file is treated as if it was being opened for the first time. Note that before the file definition can be changed, the existing file must be closed.
- **If you do not change file definitions at 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.

INQPCOPN | NOINQPCOPN

Specifies whether or not a unit is connected to a file when executing an INQUIRE by unit.

INQPCOPN

Specifies that, if a unit is connected to a file, even if it was preconnected and no I/O statement has been executed, a value of true is returned in the variable or an array element given in the OPENED specifier from an INQUIRE by unit statement.

NOINQPCOPN

Indicates that, if and only if an unit is internally open, a value of true is returned in the variable or an array element given in the OPENED specifier for an INQUIRE by unit statement.

"Internally open" means that the unit is connected to a file by an OPEN statement, or if the unit has been preconnected, that a READ, WRITE, PRINT, REWIND, or ENDFILE statement has been successfully executed.

IOINIT | NOIOINIT

Specifies whether or not the normal initialization for I/O processing will occur during initialization of the run-time environment.

IOINIT

Causes the normal initialization for I/O processing to occur during initialization of the run-time environment.

NOIOINIT

Suppresses initialization for I/O processing. This means that the error message unit will not be opened during initialization of the run-time environment. However, this does not prevent I/O from occurring on this or on any other unit. (Such I/O may fail if proper DD statements are not given.)

OCSTATUS | NOOCSTATUS

Specifies whether file existence will be checked during the running of OPEN statements, whether files are deleted from their storage media, and whether files that have been closed can be reconnected without an OPEN statement.

OCSTATUS

Specifies:

1. File existence will be checked for consistency with the OPEN statement specifiers STATUS='OLD' and STATUS='NEW'.

2. File deletion will occur when the CLOSE statement specifier STATUS='DELETE' is given (on devices which allow deletion).
3. A preconnected file will be disconnected when a CLOSE statement is given or when another file is opened on the same unit. It can be reconnected only by an OPEN statement when there is no other file currently connected to that unit.

NOOCSTATUS

Specifies:

1. File existence will not be checked for consistency with the OPEN statement specifiers STATUS='OLD' and STATUS='NEW'.
2. File deletion will not occur when the CLOSE statement specifier STATUS='DELETE' is given.
3. A preconnected file will be disconnected when a CLOSE statement is given or when another file is opened on the same unit. It can be reconnected by a sequential READ or WRITE, BACKSPACE, OPEN, REWIND, or ENDFILE statement when there is no other file currently connected to that unit.

RECPAD[(ALL)] | NORECPAD

Specifies whether a formatted input record is padded with blanks.

RECPAD

Causes a formatted input record within an internal file or a varying/undefined length record (RECFM=U or V) external file to 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. If ALL is specified, a formatted input record is padded regardless of the record format of the file.

NORECPAD

Specifies that an input list and format specification must not require more data from an input record than the record contains. If more data is required, condition FOR1002E is raised.

SPIE | NOSPIE

Specifies whether or not the run-time environment will take control when a program interrupt occurs.

SPIE

Specifies that the run-time environment takes control when a program interrupt occurs.

NOSPIE

Specifies that the run-time environment does not take control when a program interrupt occurs. If you specify NOSPIE, various run-time functions that depend on a return of control after a program interrupt are not available. These include the following:

- The messages and corrective action for a floating-point overflow
- The messages and corrective action for a floating-point underflow interrupt (unless the underflow is to be handled by the hardware based upon the XUFLOW option)
- The messages and corrective action for a floating-point or fixed-point divide exception
- The simulation of extended precision floating-point operations on processors that do not have these instructions

- The realignment of vector operands that are not on the required storage boundaries and the re-running of the failed instruction.

Instead of the corrective action, abnormal termination results. In this case, the STAE or NOSTAE option that is in effect governs whether or not the VS FORTRAN run-time environment gains control at the time of the abend.

STAE | NOSTAE

Specifies whether or not the run-time environment will take control if an abnormal termination occurs.

STAE

Specifies that the run-time environment will take control when an abnormal termination occurs.

NOSTAE

Specifies that the run-time environment does not take control when an abnormal termination occurs. If NOSTAE is specified, abnormal termination is handled by the operating system rather than by the VS FORTRAN run-time environment. In this case the following occurs:

- Message AFB240I, which shows the PSW and register contents at the time of the abend, is not printed. However, this information will be provided by the operating system.
- The indication of which Fortran statement caused the failure will not be printed.
- The traceback of the routines will not be printed.
- The post-abend symbolic dump will not be printed even with the option ABSDUMP in effect.
- Certain exceptional conditions handled by the run-time environment or by the debugging device cause system abends rather than VS FORTRAN messages. For example, some errors that occur during running of an OPEN statement result in a system abend rather than the printing of message AFB219I, which allows the program to possibly continue running.
- An MTF subtask that terminates unexpectedly causes a user ABEND 922 in the main task rather than message AFB922I.

XUFLOW | NOXUFLOW

Specifies whether or not an exponent underflow will cause a program interrupt.

XUFLOW

Allows an exponent underflow to cause a program interrupt, followed by a message from the VS FORTRAN Version 2 Library, followed by a standard fixup.

NOXUFLOW

Suppresses the program interrupt caused by an exponent underflow. The hardware sets the result to zero.

Changing the Error Option Table Defaults

Module AFH5UOPT contains the Error Option Table defaults. You can accept the IBM-supplied defaults, or you can supply your own defaults. To customize AFH5UOPT for your site, use the IBM-supplied job AFHWVOPT, and modify the VSF2UOPT macro instructions in an SMP/E USERMOD. The syntax and operands of the VSF2UOPT macro instructions are described in this section.

If you have Fortran applications that are link-edited with Language Environment, then there is no error option table to customize.

Use the VSF2UOPT macro to customize the Error Option Table as follows:

- Adding new error messages to the table, without changing existing ones, by coding the **VSF2UOPT Required Macro Instruction**, followed by an END statement.
- Changing existing error messages in the table, with or without adding new ones, by coding the **VSF2UOPT Required Macro Instruction**, followed by the necessary number of optional macro instructions, followed by an END statement.

For information on IBM-supplied error messages, refer to “Extended Error-Handling Subroutines and Error Option Table” in *VS FORTRAN Version 2 Language and Library Reference*.

Syntax of VSF2UOPT Required Macro Instruction

```
VSF2UOPT [ADDNTRY = n]
```

ADDNTRY=*n*

Is a positive integer specifying the number of new error message numbers to be added to the error option table. Additional error message numbers will begin at 500 and continue sequentially, up to a maximum of 899. If you want to change existing messages but do not want to add new ones, omit ADDNTRY=*n*.

n is a positive integer between 1 and 598.

Syntax of VSF2UOPT Optional Macro Instruction

```
VSF2UOPT MSGNO = (ermsno [, qty])  
[, ALLOW = errs]  
[, INFOMSG = YES | NO]  
[, IOERR = YES | NO]  
[, MODENT = YES | NO]  
[, PRINT = prmsg]  
[, PRTBUF = YES | NO]  
[, TRACBAK = YES | NO]  
[, USREXIT = exitname]
```

The MSGNO option must always be specified. The default values of the five options INFOMSG, IOERR, MODENT, PRTBUF, and TRACBAK vary according to the following conditions:

- If the value of MSGNO specifies an IBM-supplied message number, and *none* of the five options is changed, then the default values are found in “Extended Error-Handling Subroutines and Error Option Table” of *VS FORTRAN Version 2 Language and Library Reference*.
- If either
 - the value of MSGNO specifies an IBM-supplied message number, and *one or more* of the five options is changed, or
 - the value of MSGNO specifies a new message number,

then the default values for the *unspecified* options are the following:

INFOMSG

NO

IOERR

NO

MODENT

YES

PRTBUF

NO

TRACBAK

YES

MSGNO = (*ermsno* [, *qty*])

Specifies which error messages are affected by the default changes.

ermsno

Specifies either one message number, or the first error message number in a series of consecutive numbers.

qty

Specifies, if there is more than one, the number of consecutive error message numbers, beginning with *ermsno*.

For example, if the option is coded MSGNO=(153), then the default values for message 153 will be changed. If the option is coded MSGNO=(153,4), then the default values for messages 153 through 156 will be changed.

ALLOW = *errs*

Specifies the number of times the error may occur before the program is terminated.

errs

Specifies the number of errors allowed. To specify an exact number of errors allowed, *errs* must be a positive integer with a maximum of 255. A zero, or any number greater than 255, means the error can occur an unlimited number of times.

Note: Be aware that altering an error option table entry to allow “unlimited” error occurrence may cause a program to loop indefinitely.

If the value of MSGNO specifies an IBM-supplied message number, the default value for this option is listed in “Extended Error-Handling Subroutines and Error Option Table” of *VS FORTRAN Version 2 Language and Library Reference*. If the value of MSGNO specifies a new message number, the default value is 10.

INFOMSG = YES | NO

Specifies whether the message is an informational or an error message.

YES

Specifies that the message is informational only. In this case the following occurs:

- No user error exit is taken.
- The value of ALLOW is ignored. Running will not terminate, even if it reaches the designated number of errors allowed.
- The error summary printed after termination of your program does not include a count of the number of times the condition occurred.

NO

Specifies that the message is an error message.

IOERR = YES | NO

Specifies whether or not this error message represents an I/O error for which error counting is to be suppressed when an ERR or IOSTAT option is given on the I/O statement.

YES

Specifies that if an ERR or IOSTAT option is given, the occurrence of the error is not to be counted toward the maximum number specified by the ALLOW option above. This should be specified only for those errors listed in *VS FORTRAN Version 2 Language and Library Reference* for which the ERR and IOSTAT options are honored.

NO

Specifies that the error occurrence is to be counted toward the maximum number of errors allowed.

MODENT = YES | NO

Specifies whether or not the ERRSET subroutine may be used to modify the error option table entry for this message.

YES

Specifies that the entry may be modified.

NO

Specifies that the entry may not be modified.

If you code a YES value for an IBM-supplied error message whose default is NO, and you subsequently modify this entry using the ERRSET subroutine, you may receive undesirable results. Check the chapter “Extended Error-Handling Subroutines and Error Option Table” of *VS FORTRAN Version 2 Language and Library Reference* to find out which message numbers have a “Modifiable Entry” value of NO.

PRINT = *prmsg*

Specifies the number of times the error message is to be printed. Subsequent occurrences of the error do not cause the message to be printed again.

prmsg

Specifies the number of times the message is to be printed. To specify an exact number of times printed, *prmsg* must be a positive integer, with a maximum of 254. A “0” means the message will not be printed. Specifying 255 means the message can be printed an unlimited number of times.

If the value of MSGNO specifies an IBM-supplied message number, the default value for this option is listed in the chapter “Extended Error-Handling Subroutines and Error Option Table” in *VS FORTRAN Version 2 Language and Library Reference*. If the value of MSGNO specifies a new message number, the default value is 5.

PRTBUF = YES | NO

Specifies whether or not the I/O buffer is to be printed following certain I/O errors.

YES

Specifies that the contents of the buffer are to be printed.

NO

Specifies that the contents of the buffer are not to be printed.

This option applies only to IBM-supplied error messages. Do not code YES unless the IBM-supplied default for this error message number already allows

the buffer to be printed. Check the chapter “Extended Error-Handling Subroutines and Error Option Table” in *VS FORTRAN Version 2 Language and Library Reference* to find out which message numbers have a “Print Buffer” value of YES.

TRACBAK = YES | NO

Specifies whether or not a module traceback listing is to be printed following the error message.

YES

Specifies that the traceback listing is to be printed.

NO

Specifies that the traceback listing is not to be printed.

USREXIT = *exitname*

Specifies the user error exit routine that is invoked following the printing of the error message.

exitname

Specifies the entry point name of the user error exit routine. The routine should not be written in VS FORTRAN and should be reentrant.

If the routine is specified here, instead of being specified as an option passed to the ERRSET subroutine, the routine is invoked when the error occurs for any user. In this case, the routine will be invoked, regardless of whether the ERRSET routine was used or not. (However, unless a MODENT value of NO is in effect, programs can still call ERRSET dynamically to specify their own exit routine instead of the one specified by USREXIT.)

For programs operating in link mode, the user error exit routine must be link-edited with all users’ programs.

To make the user error exit routine available to users who operate in load mode, the routine must be included in the composite module AFH5RENA. Then, if the user error exit routine must communicate with the program in which the error was detected, it must do so using a dynamic common area, not a static one.

Customizing Fortran LIBPACKs

The Fortran LIBPACKs are collections of individual modules that are packaged into a single load module in order to reduce the time that would otherwise be needed to load the individual modules.

Language Environment provides four Fortran LIBPACKs, which you can customize either during or following the installation of Language Environment.

Table 23. Fortran LIBPACKs

For Applications Link-Edited With...	Customize LIBPACK...	Which Is Loaded...
Language Environment	AFHPRNAG	above 16 MB
Language Environment	AFHPRNBG	below 16 MB
VS FORTRAN	AFH5RENA	above 16 MB
VS FORTRAN	AFH5RENB	below 16 MB

The following tables give the names of the individual modules that can be included with or excluded from the LIBPACKs. In the tables, the terms **required** and **optional** are defined as follows:

required

Means that this module must be a part of the LIBPACK. It is not possible to exclude it.

optional

Means that this module may be either included or excluded from the LIBPACK. If the function indicated for the module is frequently used at your installation, the module should generally be included in order to avoid having to load it individually for each enclave.

For LIBPACKs loaded above the 16 MB line, the optional modules are included in the IBM-supplied default LIBPACK. For LIBPACKs loaded below the 16 MB line, only the required modules are included in the IBM-supplied default LIBPACK. Each optional LIBPACK module is also present individually. It will be loaded if that module is not included in the LIBPACK.

Refer to “Tailoring the Fortran LIBPACKs” on page 44 for information on how to tailor these LIBPACKs.

Contents of the Fortran LIBPACK AFHPRNAG

Table 24 lists routines you can include in the Fortran LIBPACK AFHPRNAG and briefly describes each to help you determine which to include in your tailored LIBPACK.

Note: For all entries in Table 24, the link-edited AMODE is 31 and the link-edited RMODE is ANY.

Table 24. Routines Eligible for Inclusion in the Fortran LIBPACK AFHPRNAG

Name	Description	Required or Optional
AFHALBCG	Library common work area	Required
AFHBCITT	Character intrinsic functions	Optional
AFHBCMPT	Complex/character compare routine	Optional
AFHBCMVT	Character move routine	Optional
AFHBCNCT	Character concatenation routine	Optional
AFHBCSTT	IBCLR/IBSET/BTEST functions	Optional
AFHBDPRT	Double/Extended precision product	Optional
AFHBFIFT	Real to integer intrinsic function	Optional
AFHBIBTT	IBITS using INTEGER*1 or INTEGER*2 argument	Optional
AFHBIDXT	Character index function	Optional
AFHBLOGT	Bit intrinsic functions, INTEGER*4 arguments	Optional
AFHBLXCT	Lexical comparison routines	Optional
AFHBMVBT	MVBITS (move bits) subroutine	Optional
AFHBMV8T	MVBITS (move bits) routine, INTEGER*8 arguments	Optional
AFHBMXDT	Maximum/minimum function, REAL*8 arguments	Optional
AFHBMXIT	Maximum/minimum function, INTEGER*4 arguments	Optional
AFHBMXRT	Maximum/minimum function, REAL*4 arguments	Optional

Table 24. Routines Eligible for Inclusion in the Fortran LIBPACK AFHPRNAG (continued)

Name	Description	Required or Optional
AFHBSHCT	ISHFTC function, all integer argument types	Optional
AFHBSHFT	ISHFT bit shift function, INTEGER*1 or INTEGER*2 arguments	Optional
AFHBMST	Exponent underflow control function	Optional
AFHCBFBE	Condition token ownership	Optional
AFHCENAE	Fortran condition enablement	Required
AFHCGETT	Qualifying data retrieval function	Optional
AFHCLC1E	Locator text construction	Optional
AFHCLC2E	Message text construction	Optional
AFHCLOCT	Qualifying data address	Optional
AFHCLSHE	Language specific condition handler for math routines	Required
AFHCPUTT	Qualifying data update	Optional
AFHCQFBE	Feedback code query function	Optional
AFHCSERT	Compiler detected error processing at run time	Optional
AFHCSGLE	Condition signaling processor	Required
AFHCTMHE	MTF termination condition handler	Optional
AFHCTOHE	I/O termination condition handler	Optional
AFHCTRAT	ERRTRA processing	Optional
AFHCXITE	Exit DSA activation	Optional
AFHDASGT	ASSIGNM (DCBS character) processor	Required
AFHDBGVE	DCBS given byte	Required
AFHDBMOE	DCBS assignment (move)	Required
AFHDBMVE	DCBS move string	Required
AFHDBPAE	DCBS pad string	Required
AFHDBTRE	DCBS truncate string	Required
AFHDBTTE	DCBS translate and test	Required
AFHFGSTL	Math glue code generator	Optional
AFHGDIRE	Direct symbol table lookup	Optional
AFHGFORT	TEST option debug interface	Optional
AFHGISDE	Init symbol dictionary default	Optional
AFHGSQLE	Sequential lookup service	Optional
AFHIABDT	SYSABD processing	Optional
AFHIABNT	SYSABN processing	Optional
AFHIEINE	Enclave initialization	Required
AFHIETRE	Enclave termination	Required
AFHIEXTT	CALL EXIT processing	Optional
AFHIMTRT	Main program termination	Required
AFHIPAUT	PAUSE processing	Optional
AFHIPINE	Process initialization	Required

Table 24. Routines Eligible for Inclusion in the Fortran LIBPACK AFHPRNAG (continued)

Name	Description	Required or Optional
AFHIRCST	SYSRCS processing	Optional
AFHIRCTT	SYSRCT processing	Optional
AFHIRCXT	SYSRCX processing	Optional
AFHISTPT	STOP processing	Required
AFHITINE	Thread initialization	Required
AFHITTRE	Thread termination	Required
AFHLNABE	Find NAB and build dummy DSA	Required
AFHMOCBE	MTF run-time options for subtask	Required
AFHOASTE	Asynchronous I/O file close at termination routine	Optional
AFHOASYT	Asynchronous I/O request processing routine	Optional
AFHOBDSE	Build descriptor from parse tree	Optional
AFHOBNTE	Build nest table, implied DO in iolist item	Optional
AFHOBTRE	Build parse tree	Optional
AFHOCLOT	CLOSE processing routine	Optional
AFHOCMFE	I/O to terminal or to other device processing routine	Optional
AFHOCNTT	Control statement processing routine	Optional
AFHOCVIE	Copy parse tree or descriptor	Optional
AFHODCBE	DCB attributes resolution routine	Required
AFHODICT	DEFINE FILE processing routine	Optional
AFHODYNG	Dynamic file allocation	Optional
AFHOFINT	FILEINF processing routine	Optional
AFHOFMPE	Formatted I/O record processing routine	Optional
AFHOFMTT	Formatted I/O service request routing routine	Optional
AFHOFSCG	File name scan	Optional
AFHOIBCT	Pre-VS FORTRAN I/O services routing routine	Optional
AFHOINIE	I/O support initialization	Required
AFHOINQT	INQUIRE statement processing routine	Optional
AFHOINTE	Internal file I/O service processing routine	Optional
AFHOLDFT	Pre-VSF 1.4.0 list-directed I/O parameter list processor	Optional
AFHOLDRT	List-directed I/O processing routine	Optional
AFHOLDTE	Pre-VSF 1.4.0 list-directed I/O processing routine	Optional
AFHONAMT	Pre-VSF 1.4.0 NAMELIST I/O parameter processor routine	Optional
AFHONLLE	Namelist I/O for static debug	Optional
AFHONLTE	Pre-VSF 1.4.0 NAMELIST I/O processing routine	Optional
AFHONMLT	Namelist I/O processing routine	Optional
AFHOOPNT	OPEN statement processing routine	Optional
AFHOSCOT	Pre-VSF 1.4.0 I/O services routing routine	Optional
AFHOSIIE	Get scalar intrinsic items	Optional
AFHOSTAG	Default I/O units allocation	Required

Table 24. Routines Eligible for Inclusion in the Fortran LIBPACK AFHPRNAG (continued)

Name	Description	Required or Optional
AFHOSYSE	STOP/PAUSE message display routine	Required
AFHOTRFE	Close all files at termination routine	Required
AFHOUFMT	Unformatted I/O processing routine	Optional
AFHOUFOE	Pre-VSF 1.4.0 unformatted I/O processing routine	Optional
AFHOUNIT	UNTANY/UNTNOFD processing	Optional
AFHOUTAG	Unit attribute table	Required
AFHPINIE	Program management initialization	Required
AFHPLVDE	LIBVEC descriptor	Required
AFHPRNAG	AFHPRNAG LIBPACK CSECT	Required
AFHRABTT	ABORT processing routine	Optional
AFHSDYAT	Obtain storage for ALLOCATE statement routine	Optional
AFHSDYDT	Free storage for DEALLOCATE statement routine	Optional
AFHSFREE	Storage free	Optional
AFHSGETE	Storage get	Optional
AFHSMIRE	Storage management initialization	Required
AFHSSG1T	Signal condition FOR0311S	Optional
AFHSSG2T	Signal condition FOR0312S	Optional
AFHSSG3T	Signal condition FOR0313S	Optional
AFHSVFAT	VSF version ALLOCATE/DEALLOCATE statements routine	Optional
AFHTCNIE	External input to internal format conversion routine	Optional
AFHTCNOE	Internal format to external output conversion routine	Optional
AFHTCVSE	I/O data conversion routing routine	Optional
AFHTCVTE	I/O data conversion routing routine adcon form	Optional
AFHTTENE	Powers of ten constants tables	Optional
AFHUDMAE	Dump file attributes event handler	Optional
AFHUDM2E	Dump variable event handler	Optional
AFHUDUMT	Dump processing	Optional
AFHUSDMT	SDUMP processing	Optional
AFHVSPIT	Obtain compile-time required vector temporaries routine	Optional
AFHXARGT	Get argument string	Optional
AFHXBSDE	New direct symbol table lookup routine	Optional
AFHXCDME	Common block directory maintenance routine	Optional
AFHXCMNT	Obtain dynamic common blocks storage routine	Optional
AFHXCPTV	CPU time processing routine	Optional
AFHXCUIE	Compiled unit identification routine	Optional
AFHXCVDI	Convert and dump program symbols routine	Optional
AFHXDCLE	Save area classification routine	Optional
AFHXDEST	Signal extended common request routine	Optional
AFHXDIVT	DIV requests processing routine	Optional

Table 24. Routines Eligible for Inclusion in the Fortran LIBPACK AFHPRNAG (continued)

Name	Description	Required or Optional
AFHXDOCT	Divide check/overflow test routine	Optional
AFHXDPET	Signal parallel execution request routine	Optional
AFHXDSPT	Old form calculate array span/dimension factor routine	Optional
AFHXDTME	Termination exit to close DIV objects	Optional
AFHXDYLT	Dynamic loading processing routine	Optional
AFHXEINE	LCWA init for environment and run-time options	Required
AFHXEV7E	Fortran event handler routine	Required
AFHXFAIT	LCP initialize associated variable pointer routine	Optional
AFHXFAUT	LCP update associated variable routine	Optional
AFHXFFEE	Identify entry point type routine	Optional
AFHXFMTT	LCP define file processing routine	Optional
AFHXIGNT	IGNORE FILE HISTORY processing routine	Optional
AFHXLNKT	Nonshareable to shareable CSECT linkage routine	Optional
AFHXOWNE	Save area ownership routine	Optional
AFHXPMLT	Subprogram parameter list checker routine	Optional
AFHXSIDE	Obtain ISN or sequence number id routine	Optional
AFHXSISE	Convert item to vib_desc_fmt	Optional
AFHXSPNT	Calculate array span/dimension factor routine	Optional
AFHXSQLE	New sequential symbol table retrieval	Optional
AFHXSTIE	Obtain symbol table information routine	Optional
AFHXTIMT	Date/time information routine	Optional
AFHXUSDE	Update symbol table retrieval	Optional
AFHX8SMT	New compiler i*8 simulator routine	Optional

Contents of the Fortran LIBPACK AFHPRNGB

Table 25 on page 189 lists routines you can include in the Fortran LIBPACK AFHPRNGB and briefly describes each to help you determine which to include in your tailored LIBPACK.

Note: For all entries in Table 25, the link-edited AMODE is ANY and the link-edited RMODE is 24.

Table 25. Routines Eligible for Inclusion in the Fortran LIBPACK AFHPRNGB

Name	Description	Required or Optional
AFHLCLNE	Clear Fortran dummy DSA	Required
AFHOASUG	Asynchronous I/O subtask routine	Optional
AFHOBDRE	Direct I/O processing routine	Optional
AFHOBSQE	Sequential I/O processing routine	Required
AFHOFSTG	File status	Required
AFHOSTRE	Striped I/O processing routine	Optional

Table 25. Routines Eligible for Inclusion in the Fortran LIBPACK AFHPRNBG (continued)

Name	Description	Required or Optional
AFHOVKYE	VSAM KSDS (keyed I/O) services routine	Optional
AFHOVSMG	VSAM (RRDS, ESDS) I/O services routine	Optional
AFHPRNBG	AFHPRNBG LIBPACK CSECT	Required

Contents of the Fortran LIBPACK AFH5RENA

Table 26 lists routines you can include in the Fortran LIBPACK AFH5RENA and briefly describes each to help you determine which to include in your tailored LIBPACK.

Note: For all entries in Table 26, the link-edited AMODE is 31 and the link-edited RMODE is ANY.

Table 26. Routines Eligible for Inclusion in the Fortran LIBPACK AFH5RENA

Name	Description	Required or Optional
AFH5ABEX	VSF ABEND handler (ESTAE)	Required
AFH5ALOP	VAL function routine	Optional
AFH5AMEP	VSF NAMELIST I/O parmlist decoder	Optional
AFH5AREN	VSF VRENA vector table	Required
AFH5ARGP	VSF 2.6 ARG obtain argument string routine	Optional
AFH5ASYP	Asynchronous I/O services driver routine	Optional
AFH5BALG	Vector boundary alignment routine	Optional
AFH5BCOP	Old FORTRAN library services interface routine	Optional
AFH5BLN\$	VSF build nest table stub	Required
AFH5BLNT	Build nest table I/O service routine	Optional
AFH5CDM\$	VSF dynamic COMMON routine special stub	Required
AFH5CDMA	VSF COMMON block directory maintenance	Optional
AFH5CLOP	VSF CLOSE services routine	Optional
AFH5CNI\$	VSF conversion routine special stub	Required
AFH5CNO\$	VSF conversion routine special stub	Required
AFH5COM\$	VSF COMH special stub	Required
AFH5COMH	VSF formatted I/O processor	Optional
AFH5CONI	VSF convert external to internal format	Optional
AFH5CONO	VSF convert internal to external format	Optional
AFH5CPTP	VSF CPUTIME routine	Optional
AFH5CVT\$	VSF CVTH special stub	Required
AFH5CVTH	VSF conversion routine	Optional
AFH5DEB\$	VSF DEBU special stub	Required
AFH5DFCP	VSF DEFINEFILE processing routine	Optional
AFH5DFIP	VSF pre-1.4.0 list-directed I/O decoder	Optional
AFH5DIO\$	VSF DIOS special stub	Required

Table 26. Routines Eligible for Inclusion in the Fortran LIBPACK AFH5RENA (continued)

Name	Description	Required or Optional
AFH5DIVP	VSF Data-In-Virtual services processor	Optional
AFH5DOCP	VSF divide check/overflow test routine	Optional
AFH5DYLP	VSF dynamic binder routine	Optional
AFH5DYN\$	VSF dynamic allocation special stub	Required
AFH5DYNA	VSF dynamic file allocation routine	Optional
AFH5EMG\$	VSF error message special stub	Required
AFH5EMGN	VSF message build routine	Optional
AFH5ERE\$	VSF EEH special stub	Required
AFH5ERRE	VSF object time error summary	Required
AFH5ERS\$	VSF exit/return code special stub	Required
AFH5EXIP	VSF return code and exiting routine	Optional
AFH5FINP	VSF file information routine	Optional
AFH5FISC	VSF file name scan routine	Optional
AFH5FNTH	VSF program interrupt handler	Required
AFH5GMFM	VSF getmain/freemain routine	Required
AFH5GPRM	VSF global parmlist	Required
AFH5IAD\$	VSF IAD interface special stub	Required
AFH5IIO\$	VSF internal I/O special stub	Required
AFH5IIOS	VSF internal I/O routine	Optional
AFH5INI\$	VSF Vector common init special stub	Required
AFH5INQP	VSF INQUIRE processing routine	Optional
AFH5INTH	VSF vector program interrupt handler	Optional
AFH5INTP	VSF init/term routine	Required
AFH5IOCP	VSF I/O control processing	Optional
AFH5IOFP	VSF formatted I/O router routine	Optional
AFH5IOLP	VSF list-directed processor	Optional
AFH5IONP	VSF NAMELIST processor	Optional
AFH5IOUP	VSF unformatted I/O processor	Optional
AFH5KIO\$	VSF keyed I/O special stub	Required
AFH5LBC0	VSF library common work area	Required
AFH5LINP	VSF shareable code load routine	Optional
AFH5LOAD	VSF load/delete service routine	Required
AFH5LOC\$	VSF offset locate special stub	Required
AFH5LOCA	VSF offset locator routine	Optional
AFH5MIN\$	VSF MTF init special stub	Required
AFH5MMA\$	VSF MTF map and attach special stub	Required
AFH5MOPP	VSF extended error handling routine	Optional
AFH5MPR\$	MTF subparameter parser special stub	Required
AFH5MSKL	VSF message skeletons	Optional

Table 26. Routines Eligible for Inclusion in the Fortran LIBPACK AFH5RENA (continued)

Name	Description	Required or Optional
AFH5OCMP	VSF dynamic COMMON processor routine	Optional
AFH5OPEP	VSF OPEN processor routine	Optional
AFH5PARM	VSF run-time parameter list scan routine	Required
AFH5PIO\$	VSF striped I/O special stub	Required
AFH5POS\$	VSF post-ABEND processor special stub	Required
AFH5RDCB	VSF DCB resolution routine	Required
AFH5SCOP	VSF pre-1.4 I/O interface	Optional
AFHFSPAP	VSF array span calculator	Optional
AFH5SPBP	VSF 1.4 array span calculator	Optional
AFH5SPIE	VSF SPIE set routine	Required
AFH5STAE	STAE set routine	Required
AFH5STIO	VSF standard I/O setup routine	Required
AFH5TIMP	VSF obtain date and time routine	Optional
AFH5TRC\$	VSF traceback special stub	Required
AFH5TRCH	VSF traceback routine	Optional
AFH5TRMF	VSF termination file close routine	Required
AFH5UNIN	VSF vector unnorm argument exception handler	Optional
AFH5UOPT	VSF error message options table	Required
AFH5VDMQ	VSF PDUMP/CPDUMP service routine	Optional
AFH5VINI	VSF vector common area initializer	Optional
AFH5VIO\$	VSF non-keyed VSAM special stub	Required
AFH5VTEN	VSF floating point conversion constants	Optional
AFH5VUAT	VSF UNIT Attribute Table	Required

Contents of the Fortran LIBPACK AFH5RENB

Table 27 lists routines you can include in the Fortran LIBPACK AFH5RENB and briefly describes each to help you determine which to include in your tailored LIBPACK.

Note: For all entries in Table 27, the link-edited AMODE is ANY and the link-edited RMODE is 24.

Table 27. Routines Eligible for Inclusion in the Fortran LIBPACK AFH5RENB

Name	Description	Required or Optional
AFH5ASUB	Asynchronous I/O services subtask routine	Optional
AFH5BREN	VSF VRENB locator table	Required
AFH5DIOS	VSF direct access I/O routine	Optional
AFH5FIST	VSF file info status routine	Required
AFH5KIOS	VSF keyed I/O processor	Optional
AFH5SIOS	VSF sequential I/O routine	Required

Table 27. Routines Eligible for Inclusion in the Fortran LIBPACK AFH5RENB (continued)

Name	Description	Required or Optional
AFH5VIOS	VSF non-keyed VSAM routine	Optional
IBMPEV11 CEEEV011	VisualAge PL/I library	ANY

Appendix C. Using IBM C/C++ with Language Environment

This appendix describes the C/C++ locale and its properties you can customize at installation time. It also describes the C system programming facilities that are packaged and shipped with Language Environment.

Planning to Customize Locale Time Information

Recommendation: You should not install this usermod. The default C/C++ locale (EDC\$S370) obtains the time zone difference from Greenwich mean time from the system. If your C/C++ application requires a different time zone other than the one obtained from the system, you can use the `tzset()` function and the `TZ` environment variable described in *z/OS C/C++ Run-Time Library Reference*.

This section describes the time information options that you can change at installation time for the C/C++ locale. When C/C++ initializes its environment, it uses the C/C++ locale as its default locale. The only category of locale that you can change at installation is the `LC_TOD` field in the EDC\$S370 locale, or the IBM-supplied default locale. See *z/OS C/C++ Programming Guide* for information on how to change categories other than `LC_TOD` and how to create new locales.

The `LC_TOD` category defines variables that describe zone difference, time zone name, and Daylight Savings Time (DST) start and end. The `LC_TOD` variables are used by the `mktime` and `localtime` functions for determining local time. The time functions use the time zone difference from the system as the default.

Note: For information on using `TZ` in Non-POSIX C/C++ applications see *z/OS C/C++ Programming Guide*.

Customizing the Locale Time Information

You can set up your default run-time environment and customize time information for your installation's default C/C++ locale. The `EDCLLOCL JCL` defines a `USERMOD` that allows for changes to the time zone and Daylight Savings Time parameters listed below. To set these time parameters, update the `EDCLLOCL JCL` with the appropriate time information for your installation and apply the resulting `USERMOD`.

Figure 18 on page 196 is a hypothetical example. The time zone name in the example is `EST`. `TZDIFF` is 300 (minutes), which means it is 5 hours greater than (West of) Greenwich mean time. If `TZDIFF` is greater than 1440, the time zone difference from Greenwich mean time is obtained from the system. The Daylight Savings Time (DST) information in the example is:

- DST starts in April, in the second week on Sunday.
- DST begins at 2 am.
- DST shifts 1 hour.
- DST ends in October, in the second week on Sunday.
- DST ends at 2 am.
- DST time zone name is `EDT`.

```
TZDIFF=300,TNAME='EST',
DSTSTM=4,DSTSTW=2,DSTSTD=0,STARTTM=7200,SHIFT=3600,
DSTENM=10,DSTENW=2,DSTEND=0,ENDTM=7200,DSTNAME='EDT',
```

Figure 18. Example of Time Zone and Daylight Savings Time Information in Module EDCLOCTZ

Time Information Options Reference

TZDIFF Time zone difference expressed in minutes. If the local time zone is west of the Greenwich Meridian, this value must be positive. If the local time zone is east of the Greenwich Meridian, this value must be negative. An absolute value that is greater than 1440 (the number of minutes in a day) for this field tells the C/C++ Library to get the time zone difference from the system.

TNAME Time zone name such as PST (Pacific Standard Time) specified within quotation marks. The default for this field is a NULL string.

DSTNAME

A Daylight Savings Time zone name such as PDT (Pacific Daylight Time) specified within quotation marks, if available. If DST information is not available, this is set to NULL, which is also the IBM-supplied default. This field must have a value if Daylight Savings Time information (as provided by the other fields) is to be taken into account by the `mktime` and `localtime` functions. These functions ignore DST if this field is set to NULL.

DSTSTM Month of the year when DST (Daylight Savings Time) comes into effect. This value ranges from 1 through 12 inclusive, with 1 corresponding to January and 12 corresponding to December. If DST is not applicable to a locale, this is set to 0, which is also the IBM-supplied default.

DSTENM Month of the year when Daylight Savings Time ceases to be in effect. Semantics similar to DSTSTM.

DSTSTW Week of the month when DST comes into effect. Acceptable values range from -4 to +4. A value of 4 means the fourth week of the month, while a value of -4 means fourth week of the month, counting back from the end of the month. Sunday is considered the start of the week. If DST is not applicable to a locale, DSTSTW is set to 0, which is also the IBM-supplied default.

DSTENW Week of the month when DST ceases to be in effect. Semantics similar to DSTSTW.

Note: DSTSTW and DSTENW need not be used. The DSTSTD and DSTEND fields can specify either day of week or day of month. If day of month is specified, DSTSTW and DSTENW become redundant.

DSTSTD Dependent on the value of DSTSTW. If DSTSTW is not equal to 0, this is the day of the week when DST comes into effect. It ranges from 0 through 6 inclusive, with 0 corresponding to Sunday and 6 corresponding to Saturday. If DSTSTW equals 0, DSTSTD is the day of the month (for the current year) when DST comes into effect. It ranges from 1 to the last day of the month inclusive. The last day of the month is 31 for January, March, May, July, August, October, and December. It is 30 for April, June, September, and November. For February, it is 28 on non-leap years and 29 on leap years. If DST is not applicable to a locale, DSTSTD is set to 0, which is also the IBM-supplied default.

DSTEND The day of the week or the day of the month when DST ceases to be in effect. Semantics similar to DSTSTD.

STARTTM

Seconds after 12 midnight, local standard time, when DST comes into effect. For example, if DST is to start at 2:00 AM, STARTTM is assigned the value 7200; for 12:00 AM (midnight), STARTTM is 0; for 1:00 AM, it is 3600.

ENDTM Seconds after 12 midnight, local standard time, when DST ceases to be in effect. Semantics similar to STARTTM.

SHIFT DST time shift, expressed in seconds. Default is 3600, for 1 hour.

System Programming Facilities

C system programming facilities are installed along with Language Environment. The modules for C system programming facilities use aliases that are the same as some of the C component module names. Therefore, the C system programming facilities are installed into their own target and distribution libraries:

- SCEESPC
- AEDCMOD2

For more information on the system programming facilities of C, see *z/OS C/C++ Programming Guide*. For information on how to build these free-standing applications after compiling, see *z/OS Language Environment Programming Guide*.

Appendix D. Modules eligible for the link pack area

The modules listed in the following table can be put in the LPA or the ELPA, depending on their RMODE:

- If the RMODE is ANY, the module can reside in the link pack area or in the extended link pack area (above or below the 16 MB address line).
- If the RMODE is 24, the module can reside only in the link pack area (below the 16 MB line).

If you are considering placing the modules listed in this appendix in the LPA or the ELPA, then IBM highly recommends that you place the SCEELPA data set in the LPA list (LPALSTxx). This data set contains modules that are reentrant, reside above the line and are heavily used by z/OS itself.

The specific HLL sections contains tables of modules eligible for the LPA or the ELPA above and beyond what is found in the SCEELPA data set. You will need to use the Dynamic LPA or MLPA approach to move these modules into the LPA/ELPA. You do not need to include recommended modules if they contain functions your installation does not use. Language Environment modules not listed in these tables can be moved into LPA/ELPA at your discretion.

Language Environment base modules

Modules and *aliases* listed in Table 28 can be moved into LPA/ELPA using the sample job CEEWLPA found in the SCEESAMP data set.

Table 28. Language Environment Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area

Language Environment Module Name	Description	RMODE
CEEBINIT CEEBLIBM	Initialization/termination for batch	24
CEEBLIIA IBMBLIIA IBMBPIIA	OS PL/I and C load module compatibility	24
CEEBLRR	Library Retention Routine	ANY
CEEBPICI	Initialization/termination routines for preinitialization compatibility	24
CEEGINIT ¹ CEEBINIT	Initialization / termination when using RTLS.	24
CEELRRIN	LRR initialization	ANY
CEELRRTR	LRR termination	ANY
CEEMENU0	Message file with mixed-case English; messages 000-999	ANY
CEEMENU2	Message file with mixed-case English; messages 2000-2999	ANY
CEEMENU3	Message file with mixed-case English; messages 3000-3999	ANY
CEEMENU4	Message file with mixed-case English; messages 4000-4999	ANY
CEEMENU5	Message file with mixed-case English; messages 5000-5999	ANY
CEEMJPN0	Message file with Japanese; messages 000-999	ANY
CEEMJPN2	Message file with Japanese; messages 2000-2999	ANY

Table 28. Language Environment Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Language Environment Module Name	Description	RMODE
CEEMJPN3	Message file with Japanese; messages 3000-3999	ANY
CEEMJPN4	Message file with Japanese; messages 4000-4999	ANY
CEEMJPN5	Message file with Japanese; messages 5000-5999	ANY
CEEMUEN0	Message file with uppercase English; messages 000-999	ANY
CEEMUEN2	Message file with uppercase English; messages 2000-2999	ANY
CEEMUEN3	Message file with uppercase English; messages 3000-3999	ANY
CEEMUEN4	Message file with uppercase English; messages 4000-4999	ANY
CEEMUEN5	Message file with uppercase English; messages 5000-5999	ANY
CEEPIPI	Initialization/termination routines for the Language Environment pre-initialization facility	24

Note: If CEEGINIT is to be placed in LPA, then its alias CEEBINIT, found in the SCEERTLS data set should also be placed in LPA, and not the CEEBINIT module found in SCEERUN. In addition, IBM recommends CEEBINSS should never be placed in LPA.

Language Environment C/C++ component modules

The C/C++ component modules and *aliases* listed in Table 29 can be moved into LPA/ELPA using the sample job EDCWLPA found in the SCEESAMP data set. The Language Environment base modules listed in Table 28 on page 199 should also be moved into LPA/ELPA.

Table 29. C/C++ Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area

C/C++ Module Name	Description	RMODE
EDCHDMNP DEMANGLE	Demangler	ANY
EDCNINSP	Debug tool interface	ANY
EDCPRLK	Prelink utility	ANY
EDCRNLIB EDCRNLST	Rename list	ANY
EDCZEMSG	Mixed-case U.S. English messages	ANY
EDCZJMSG	Japanese messages	ANY
EDCZUMSG	Uppercase English messages	ANY
IEDCMSGT	C/370 message table	ANY
CELHDCPP SCEECPP	DLL for XPLINK C++ applications	ANY
CELHV003	DLL for XPLINK C applications	ANY

Notes:

1. EDCNINSP is highly recommended for inclusion in the LPA or ELPA if the Debug tool is heavily used.
2. EDCPRLK is highly recommended for inclusion in the LPA or ELPA if the prelink utility is heavily used.

- The default code page converters or locale modules, or customized code page converters or locale modules (the ones applicable for the user's country), should be included in the LPA or ELPA.

Language Environment COBOL component modules

The COBOL component modules and *aliases* listed in Table 30 can be moved into LPA/ELPA using the sample job IGZWMLP4 found in the SCEESAMP data set. The Language Environment base modules listed in Table 28 on page 199 should also be moved into LPA/ELPA.

Additional modules that exist for OS/VS COBOL compatibility (ILBO) are not described here. Refer to the OS/VS COBOL documentation for information about these modules.

Table 30. COBOL Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area

COBOL Module Name	Description	RMODE
IGZCEV5 CEEEV005	COBOL event handler	ANY
IGZCA2D	DBCS data manipulation	ANY
IGZCD2A	DBCS data manipulation	ANY
IGZCMTUE	COBOL WTO error messages	ANY
IGZCPAC	COBPACK	ANY
IGZPCO	COBPACK	ANY
IGZINSH	Formatted dump and Debug Tool support	ANY
IGZEPL	COBOL termination (VS COBOL II and OS/VS COBOL only)	24
IGZERRE	COBOL reusable environment	ANY
IGZEWTO	COBOL: write error message to operator's console	ANY
IGZCWTO	COBOL write error message	ANY
IGZCD24	COBOL dynamic call to AMODE(24) programs	24
IGZCMGUE	COBOL (IGZ) messages in uppercase English	ANY
IGZCMGEN	COBOL (IGZ) messages in English	ANY
IGZCMGJA	COBOL (IGZ) messages in Japanese	ANY
IGZCMLT IGZMSGT	COBOL message tables	ANY
IGZEPLF	COBOL environment initialization (VS COBOL II and OS/VS COBOL only)	24
IGZCBUG	Used for debugging	24
IGZCLNC	Linkage manager for OS/VS COBOL and IGZBRDGE (dynamic call and cancel)	24
IGZCLNK	Linkage manager for VS COBOL II and COBOL/370 (dynamic call and cancel)	24
IGZCULE	User I/O logic error handler	24
IGZCXFR	I/O declarative transfer	24
IGZEDMR	Reusable environment deactivation	24
IGZEINI	Environment initialization	24

Table 30. COBOL Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

COBOL Module Name	Description	RMODE
IGZEINP	Accept input reader	24
IGZEOPN	OPENS SYSIN and SYSPUNCH in the Initial Program Thread (IPT)	24
IGZEOU	Display output writer	24
IGZEQBL	QSAM initialization transmission verbs, error exits	24
IGZEQOC	QSAM OPEN/CLOSE	24
IGZERCO	OS/V S COBOL TERMINATION	24
IGZESMG	Sort/Merge interface	24
IGZEVAM	VSAM-to-IDCAMS interface	24
IGZEVEX	VSAM exit module for SYNAD and LERAD	24
IGZESCD	SORT-CONTROL I/O handling routine	24
IGZETRM	Environment termination	24

Language Environment Fortran component modules

The Fortran component modules and *aliases* listed in Table 31 can be moved into LPA/ELPA using the sample job AFHWMLP2 found in the SCEESAMP data set. The Language Environment base modules listed in Table 28 on page 199 should also be moved into LPA/ELPA.

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area

Fortran Module Name	Description	RMODE
AFHBCITT AFHBACHK AFHBACIK AFHBACJK AFHBACKK AFHBCHAR AFHBCH2R AFHBCH8R AFHBIACK AFHBICHR AFHBJACK AFHBJCHR AFHBLNR AFHBLN8R	Character intrinsic functions	ANY
AFHBCMPT AFHBCMPR AFHBXMPR	Complex/character compare routine	ANY
AFHBCMVT AFHBCMVR	Character move routine	ANY
AFHBCNCT AFH-BCNCK AFHBCNCR	Character concatenation routine	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFHBCSTT AFHBHCLK AFHBHCLR AFHBHSTK AFHBHSTR AFHBHTSK AFHBHTSR AFHBKCLK AFHBKCLR AFHBKSTK AFHBKSTR AFHBKTSK AFHBKTSR	IBCLR/IBSET/BTEST functions	ANY
AFHBDPRT AFHBDPRR AFHBQPRR	Double/Extended precision product	ANY
AFHBFIFT AFHBIDTR AFHBIFIR AFHBINTR	Real to integer intrinsic function	ANY
AFHBIBTT AFHBHBTK AFHBHBTR AFHBKBTK AFHBKBTR	IBITS using INTEGER*1 or INTEGER*2 argument	ANY
AFHBIDXT AFHBIDXK AFHBIDXR AFHBJDXK AFHBJDXR	Character index function	ANY
AFHBLOGT AFHBHEOR AFHBHNDR AFHBHNOR AFHBHORR AFHBIEOR AFHBINDR AFHBINOR AFHBIORR AFHBJEOR AFHBJNDR AFHBJNOR AFHBJORR	Bit intrinsic functions, INTEGER*4 arguments	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFHBLXCT AFHBLGEEK AFHBLGER AFHBLGTK AFHBLGTR AFHBLLEK AFHBLLER AFHBLGTK AFHBLLTR AFHB8GEEK AFHB8GER AFHB8GTK AFHB8GTR AFHB8LEK AFHB8LER AFHB8LTK AFHB8LTR	Lexical comparison routines	ANY
AFHBMVBT AFHBHMBK AFHBIMBK AFHBIMBR AFHBKMBK	MVBITS (move bits) subroutine	ANY
AFHBMV8T AFHBJMBK	MVBITS (move bits) routine, INTEGER*8 arguments	ANY
AFHBMXDT AFHBDMNR AFHBDMXR	Maximum/minimum function, REAL*8 arguments	ANY
AFHBMXIT AFHBIANR AFHBIAXR AFHBIMNR AFHBIMXR	Maximum/minimum function, INTEGER*4 arguments	ANY
AFHBMXRT AFHBRANR AFHBRAXR AFHBRMNR AFHBRMXR	Maximum/minimum, REAL*4 arguments	ANY
AFHBSHCT AFHBISCK AFHBISCR AFHBJSCK AFHBJSCR AFHBKSCK AFHBKSCR AFHBHSCK AFHBHSCR	ISHFTC function, all integer argument types	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFHBSHFT AFHBHLSK AFHBHLSR AFHBHRSK AFHBHRSR AFHBHSHK AFHBHSHR AFHBKLSK AFHBKLSR AFHBKRSK AFHBKRSR AFHBKSHK AFHBKSHR	ISHFT bit shift function, INTEGER*1 or INTEGER*2 arguments	ANY
AFHBXMST AFHBXMSR	Exponent underflow control function	ANY
AFHCBFBE AFHCBFBR	Condition token ownership	ANY
AFHCGETT AFHCGETR	Qualifying data retrieval function	ANY
AFHCLC1E AFHCLC1R	Locator text construction	ANY
AFHCLC2E AFHCLC2R	Message text construction	ANY
AFHCMSGE AFHCMSGR IFORMSGT	Fortran message table header	ANY
AFHCMS1E AFHCMS1R	Mixed-case English message file 1	ANY
AFHCMS1J	Japanese message file 1	ANY
AFHCMS1U	Uppercase English message file 1	ANY
AFHCMS2E AFHCMS2R	Mixed-case English message file 2	ANY
AFHCMS2J	Japanese message file 2	ANY
AFHCMS2U	Uppercase English message file 2	ANY
AFHCMS3E AFHCMS3R	Mixed-case English message file 3	ANY
AFHCMS3J	Japanese message file 3	ANY
AFHCMS3U	Uppercase English message file 3	ANY
AFHCMS4E AFHCMS4R	Mixed-case English message file 4	ANY
AFHCMS4J	Japanese message file 4	ANY
AFHCMS4U	Uppercase English message file 4	ANY
AFHCPUTT AFHCPUTR	Qualifying data update	ANY
AFHCQFBE AFHCQFBR	Feedback code query function	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFHCSERT AFHCSERR	Compiler detected error processing at run time	ANY
AFHCTMHE AFHCTMHR	MTF termination condition handler	ANY
AFHCTOHE AFHCTOHR	I/O termination condition handler	ANY
AFHCTRAT AFHCTRAR	ERRTRA processing	ANY
AFHCXITE AFHCXITR	Exit DSA activation	ANY
AFHFGSTL AFHFGSTR	Math glue code generator	ANY
AFHGDIRE AFHGDIRR	Direct symbol table lookup	ANY
AFHGFORT AFHGSTNR AFHGSTXR AFHGTRCR	TEST option debug interface	ANY
AFHGISDE AFHGISDR	Init symbol dictionary default	ANY
AFHGSQLE AFHGSQLR	Sequential lookup service	ANY
AFHIABDT AFHIABDR	SYSABD processing	ANY
AFHIABNT AFHIABNR	SYSABN processing	ANY
AFHIEXTT AFHIEXTR	CALL EXIT processing	ANY
AFHIPAUT AFHIPAUK AHHIPAUR	PAUSE processing	ANY
AFHIRCST AFHIRCSR	SYSRCS processing	ANY
AFHIRCTT AFHIRCTR	SYSRCT processing	ANY
AFHIRCXT AFHIRCXR	SYSRCX processing	ANY
AFHMAAG AFHMAAR	MTF map and ATTACH routine	24
AFHMSTCT AFHMSTCR	MTF subtask control	24
AFHMTFAG ³	MTF LIBPACK	ANY
AFHOASTE AFHOASTR	Asynchronous I/O file close at termination routine	ANY
AFHOASUG AFHOASUR	Asynchronous I/O subtask routine	24

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFHOASYT AFHOAINR AFHOAOUR AFHOAWTR	Asynchronous I/O request processing routine	ANY
AFHOBDRE AFHOBDRR	Direct I/O processing routine	24
AFHOBDSE AFHOBDSE	Build descriptor from parse tree	ANY
AFHOBNTE AFHOBNTR	Build nest table, implied DO in iolist item	ANY
AFHOBTRE AFHOBTTR	Build parse tree	ANY
AFHOCLOT AFHOCLOR	CLOSE processing routine	ANY
AFHOCMFE AFHOCMFR	I/O to terminal or to other device processing routine	ANY
AFHOCNTT AFHOCBSR AFHOCDLR AFHOCEFR AFHOCRWR	Control statement processing routine	ANY
AFHOCVIE AFHOCVIR	Copy parse tree or descriptor	ANY
AFHODICT AFHODICR	DEFINE FILE processing routine	ANY
AFHODYNG AFHODYNR	Dynamic file allocation	ANY
AFHOFINT AFHOFINR	FILEINF processing	ANY
AFHOFMPE AFHOFMPR	Formatted I/O record processing routine	ANY
AFHOFMTT AFHOCSTR AFHODSFR AFHOESFR AFHOFXFR AFHOIXFR AFHOQKFR AFHORDFR AFHORIFR AFHORKFR AFHORSFR AFHOSXFR AFHOUVFR AFHOWDFR AFHOWIFR AFHOWKFR AFHOWSFR	Formatted I/O service request routing routine	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFHOFSCG AFHOFSCR	File name scan	ANY
AFHOIBCT AFHOIAFR AFHOIANR AFHOIBSR AFHOIEFR AFHOIENR AFHOILFR AFHOILNR AFHOINFR AFHOIPAR AFHOIRFR AFHOIRNR AFHOIRWR AFHOIWFR AFHOIWRN	Pre-VS FORTRAN I/O services routing routine	ANY
AFHOINQT AFHOINQR	INQUIRE statement processing routine	ANY
AFHOINTE AFHOINTR	Internal file I/O service processing routine	ANY
AFHOLDFT AFHOLFAR AFHOLFERR AFHOLFRL AFHOLFRR AFHOLFWR AFHOLVAR AFHOLVER AFHOLVLR AFHOLVRR AFHOLVWR	Pre-VSF 1.4.0 list-directed I/O parameter list processor	ANY
AFHOLDRT AFHOCSLR AFHODSLR AFHOESLR AFHOFXLR AFHOIXLR AFHORILR AFHORSLR AFHOWILR AFHOWSLR	List-directed I/O processing routine	ANY
AFHOLDTE AFHOAXLR AFHOLXLR AFHOMXLR AFHOTXLR	Pre-VSF 1.4.0 list-directed I/O processing routine	ANY
AFHONAMT AFHONFRR AFHONFWR AFHONVRR AFHONVWR	Pre-VSF 1.4.0 NAMELIST I/O parameter processing routine	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFHONLLE AFHONLWR	Namelist I/O for static debug	ANY
AFHONLTE AFHOSSNR AFHOXSNR	Pre-VSF 1.4.0 NAMELIST I/O processing routine	ANY
AFHONMLT AFHOCSNR AFHOESNR AFHORINR AFHORSNR AFHOWINR AFHOWSNR	Namelist I/O processing routine	ANY
AFHOSCOT AFHOVAFR AFHOVANR AFHOVBKR AFHOVEFR AFHOVENR AFHOVLFR AFHOVLNR AFHOVNFR AFHOVRFR AFHOVRNR AFHOVRWR AFHOVWFR AFHOVWNR	Pre-VSF 1.4.0 I/O services routing routine	ANY
AFHOSIIE AFHOSIIR	Get scalar intrinsic items	ANY
AFHOSTRE AFHOSTRR	Striped I/O processing routine	24
AFHOUFMT AFHOFDUR AFHOFXUR AFHOIXUR AFHOQKUR AFHORDUR AFHORKUR AFHORSUR AFHOSXUR AFHOUVUR AFHOWDUR AFHOWKUR AFHOWSUR	Unformatted I/O processing routine	ANY
AFHOUFOE AFHOEXUR AFHOLXUR AFHOMXUR AFHOPXUR	Pre-VSF 1.4.0 unformatted I/O processing routine	ANY
AFHOUNIT AFHOUNFR AFHOUNTR	UNTANY/UNTNOFD processing	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFHOVKYE AFHOVKYR	VSAM KSDS (keyed I/O) services routine	24
AFHOVSMG AFHOVDIR AFHOVSQR	VSAM (RRDS, ESDS) I/O services routine	24
AFHPRNAG ¹ CEEEV007	AFHPRNAG LIBPACK CSECT	ANY
AFHPRNBG ¹	AFHPRNBG LIBPACK CSECT	24
AFHRABTT AFHRABTK	ABORT processing routine	ANY
AFHSDYAT AFHSDYAR	Obtain storage for ALLOCATE statement routine	ANY
AFHSDYDT AFHSDYDR AFHSDYFK	Free storage for DEALLOCATE statement routine	ANY
AFHSFREE AFHSFRER	Storage free	ANY
AFHSGETE AFHSGETR	Storage get	ANY
AFHSSG1T AFHSSG1R	Signal condition FOR0311S	ANY
AFHSSG2T AFHSSG2R	Signal condition FOR0312S	ANY
AFHSSG3T AFHSSG3R	Signal condition FOR0313S	ANY
AFHSVFAT AFHSVALK AFHSVALR AFHSVA4K AFHSVA4R AFHSVA8K AFHSVA8R AFHSVDEK AFHSVDER	VSF version ALLOCATE/DEALLOCATE statements routine	ANY
AFHTCVSE AFHTFAOR AFHTFCOR AFHTFDOR AFHTFEOR AFHTFGOR AFHTFIOR AFHTFLOR AFHTFQOR AFHTFZOR	I/O data conversion routing routine	ANY
AFHTCVTE AFHTCVTR	I/O data conversion routing routine adcon form	ANY
AFHUDMAE AFHUDMAR	Dump file attributes event handler	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFHUDM2E AFHUDM2R	Dump variable event handler	ANY
AFHUDUMT AFHUCDMR AFHUCPDR AFHUDUMR	Dump processing	ANY
AFHUSDMT AFHUSDMR	SDUMP processing	ANY
AFHVSPIT AFHVSPIR	Obtain compile-time required vector temporaries routine	ANY
AFHXARGT AFHXARGR	Get argument string	ANY
AFHXBSDE AFHXBSDR	New direct symbol table lookup routine	ANY
AFHXCDME AFHXCDMR	Common block directory maintenance routine	ANY
AFHXCMNT AFHXCMNR AFHXCMSR AFHXDCDR AFHXDCFR AFHXDCGR AFHXDCIR AFHXSDCR	Obtain dynamic common blocks storage routine	ANY
AFHXCPTV AFHXCPtr	CPU time processing routine	ANY
AFHXCUIE AFHXCUIR	Compiled unit identification routine	ANY
AFHXCVEDE AFHXCVDRE	Convert and dump program symbols routine	ANY
AFHXDCLE AFHXDCLE	Save area classification routine	ANY
AFHXDEST AFHXDESR	Signal extended common request routine	ANY
AFHXDIVT AFHXDCMR AFHXDNFR AFHXDNVR AFHXDRSR AFHXDSVR AFHXDTFR AFHXDTVR AFHXDWFR AFHXDWVR	DIV requests processing routine	ANY
AFHXDOCT AFHXDVKR AFHXOVER	Divide check/overflow test routine	ANY
AFHXDPET AFHXDPER	Signal parallel execution request routine	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFHXDSPT AFHXDSNR AFHXDS2R	Old form calculate array span&slash dimension factor routine	ANY
AFHXDTME AFHXDTMR	Termination exit to close DIV objects	ANY
AFHXDYLT AFHXDYLK AFHXDYLR	Dynamic loading processing routine	ANY
AFHXFAIT AFHXFAIR	LCP initialize associated variable pointer routine	ANY
AFHXFAUT AFHXFAUR	LCP update associated variable routine	ANY
AFHXFFEE AFHXFFER	Identify entry point type routine	ANY
AFHXFMTT AFHXFMTR	LCP define file processing routine	ANY
AFHXIGNT AFHXIGDR AFHXIGUR	IGNORE FILE HISTORY processing routine	ANY
AFHXLNKT AFHXLIMK AFHXLIMR AFHXLISK AFHXLISR	Nonshareable to shareable CSECT linkage routine	ANY
AFHXOWNE AFHXOWNR	Save area ownership routine	ANY
AFHXPMLT AFHXPMLK AFHXPMLR AFHXPMMK	Subprogram parameter list checker routine	ANY
AFHXSIDE AFHXSIDR	Obtain ISN or sequence number id routine	ANY
AFHXSISE AFHXSISR	Convert item to vib_desc_fmt	ANY
AFHXSPNT AFHXSP4R AFHXSP5R	Calculate array span/dimension factor routine	ANY
AFHXSQLE AFHXSQLR	New sequential symbol table retrieval	ANY
AFHXSTIE AFHXSTIR	Obtain symbol table information routine	ANY
AFHXTIMT AFHXCLKR AFHXCLXR AFHXDMTR AFHXDTXR	Date/time information routine	ANY
AFHXUSDE AFHXUSDR	Update symbol table retrieval	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFH5ALOP AFBVALOP	VAL function routine	ANY
AFH5AMEP AFBNAMEP IFYNAMEP	VSF NAMELIST I/O parmlist decoder	ANY
AFH5ARGP AFBVARGP	VSF 2.6 ARG obtain argument string routine	ANY
AFH5ASUB AFBVASUB	Asynchronous I/O services subtask routine	24
AFH5ASYP AFBVASYP IFYVASYP	Asynchronous I/O services driver routine	ANY
AFH5BALG AFBVBALG	Vector boundary alignment routine	ANY
AFH5BCOP AFBIBCOP IFYIBCOP	Old FORTRAN library services interface routine	ANY
AFH5BLNT AFBVBLNT IFYVBLNT	Build nest table I/O service routine	ANY
AFH5CDMA AFBVCDMA	VSF COMMON block directory maintenance	ANY
AFH5CLOP AFBVCLOP IFYVCLOP	VSF CLOSE services routine	ANY
AFH5COMH AFBVCOMH	VSF formatted I/O processor	ANY
AFH5CONI AFBVCONI	VSF convert external to internal format	ANY
AFH5CONO AFBVCONO	VSF convert internal to external format	ANY
AFH5CPTP AFBCCPTP AFBVCPTP	VSF CPUTIME routine	ANY
AFH5CVTH AFBVCVTH	VSF conversion routine	ANY
AFH5DFCP AFBDIOCP IFYDIOCP	VSF DEFINEFILE processing routine	ANY
AFH5DFIP AFBLDFIP IFYLDFIP	VSF pre-1.4.0 list-directed I/O decoder	ANY
AFH5DIOS AFBVDIOS	VSF direct access I/O routine	24
AFH5DIVP AFBVDIVP	VSF Data-In-Virtual services processor	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFH5DOCP AFBVDOCP	VSF divide check/overflow test routine	ANY
AFH5DYLP AFBVDYLP	VSF dynamic binder routine	ANY
AFH5DYNA AFBCDYNA AFBVDYNA	VSF dynamic file allocation routine	ANY
AFH5EMGN AFBVEMGN	VSF message build routine	ANY
AFH5EXIP AFBVEXIP	VSF return code and exiting routine	ANY
AFH5FINP AFBVFIMP	VSF file information routine	ANY
AFH5FISC AFBCFISC AFBVFISC	VSF file name scan routine	ANY
AFH5IIOS AFBVIOS	VSF internal I/O routine	ANY
AFH5INQP AFBVINQP IFYVINQP	VSF INQUIRE processing routine	ANY
AFH5INTH AFBVINTH	VSF vector program interrupt handler	ANY
AFH5IOCP AFBVIOCP IFYVIOCP	VSF I/O control processing	ANY
AFH5IOFP AFBVIOFP IFYVIOFP	VSF formatted I/O router routine	ANY
AFH5IOLP AFBVIOLP IFYVIOLP	VSF list-directed processor	ANY
AFH5IONP AFBVIONP IFYVIONP	VSF NAMELIST processor	ANY
AFH5IOUP AFBVIOUP IFYVIOUP	VSF unformatted I/O processor	ANY
AFH5KIOS AFBVKIOS	VSF keyed I/O processor	24
AFH5LINP AFBVLINP IFYVLINP	VSF shareable code load routine	ANY
AFH5LOCA AFBVLOCA	VSF offset locator routine	ANY

Table 31. Fortran Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

Fortran Module Name	Description	RMODE
AFH5MOPP AFBVMOPP IFYVMOPP	VSF extended error handling routine	ANY
AFH5MSKL AFBVMSKL	VSF message skeletons	ANY
AFH5OCMP AFBDDCMP AFBVOCMP IFYDDCMP	VSF dynamic COMMON processor routine	ANY
AFH5OPEP AFBVOPEP IFYVOPEP	VSF OPEN processor routine	ANY
AFH5RENA ¹ AFBVRENA	AFH5RENA LIBPACK CSECT	ANY
AFH5RENB ¹ AFBVRENB	AFH5RENB LIBPACK CSECT	24
AFH5RENP ¹ AFBVRENP	AFH5RENP LIBPACK CSECT	ANY
AFH5SCOP AFBVSCOP IFYVSCOP	VSF pre-1.4 I/O interface	ANY
AFH5SPAP AFBVSPAP IFYVSPAP	VSF array span calculator	ANY
AFH5SPBP AFBDSPAP IFYDSPAP	VSF 1.4 array span calculator	ANY
AFH5TIMP AFBVTIMP	VSF obtain date and time routine	ANY
AFH5TRCH AFBVTRCH	VSF traceback routine	ANY
AFH5UNIN AFBVUNIN	VSF vector unnorm argument exception handler	ANY
AFH5VDMQ AFBVDUMQ IFYVDUMQ	VSF PDUMP/CPDUMP service routine	ANY
AFH5VINI AFBVVINI	VSF vector common area initializer	ANY
AFH5VIOS AFBVVIOS	VSF non-keyed VSAM routine	24
AFH5VTEN AFBVVTEN	VSF floating point conversion constants	ANY
Note: AFH5RENA, AFH5RENB, and AFH5RENP are used only for applications that were link-edited with VS FORTRAN Version 1 or 2 for execution in load mode.		

Language Environment PL/I Component Modules

The PL/I component modules and *aliases* listed in Table 32 can be moved into LPA/ELPA using the sample job IBMALLP2 or IBMPLPA1 found in the SCEESAMP data set. The Language Environment base modules listed in Table 28 on page 199 should also be moved into LPA/ELPA.

Table 32. PL/I Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area

PL/I Module Name	Description	RMODE
IBMREV10 CEEEV010	PL/I event handler	ANY
CEEEV011	VisualAge PL/I event handler	ANY
IBMRCCLA IBMCCLA	Conversion director (complex strings)	24
IBMRCCRA IBMCCRA	Conversion director (non-complex strings)	24
IBMRCOMP	Conversion routines vector	24
IBMRDMPJ	Dump formatter for Japanese	ANY
IBMRDMPM	Dump formatter for mixed-case U.S. English	ANY
IBMRDMPU	Dump formatter for uppercase English	ANY
IBMREDOA IMBEDOA	Open diagnostic file module	24
IBMREDTA IMBEDTA	Diagnostic file transmitter	24
IBMREDWA IMBEDWA	Console transmitter	24
IBMREMT IIBMMSGT	Message table	ANY
IBMREOCA IMBEOCA	ON-code module / ON-code calculator	ANY
IBMRKDBA IMBKDBA	Dump file transmitter	24
IBMRKDOA IMBKDOA	Open dump file	24
IBMRKDTA IMBKDTA	Dump file transmitter	24
CEEKMRA IMBKMRA IBMRKMRA	Link to main dump control module	24
IBMRKPTA IMBKPTA	Dump parameter translate module	24
IBMLANA IMBLANA	Language table (mixed-case U.S. English)	24
IBMLANN IMBLANN	Language table (Japanese)	24
IBMLANU IMBLANU	Language table (uppercase English)	24
IBMLIB1	Lib pack (below the line)	24

Table 32. PL/I Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

PL/I Module Name	Description	RMODE
IBMLNNTA IBMLNNTA	Language table (mixed-case U.S. English)	24
IBMLNNTN IBMLNNTN	Language table (Japanese)	24
IBMLNNTU IBMLNNTU	Language table (uppercase English)	24
IBMRMCTA IBMRMCTA	ERF/ERFC (extended float)	24
IBMROCAA IBMROCAA	Close module	24
IBMROPEA IBMROPEA	Open routine (VSAM)	24
IBMROPZA IBMROPZA	Direct output file formatter	24
IBMRPDBA IBMRPDBA	Debugger interface module	24
IBMRPESA IBMRPESA	ABEND analyzer	24
IBMRPEVA IBMRPEVA	ABEND diagnostic message module	24
IBMRPTLA IBMRPTLA	Transient library level data	24
IBMRRAAA IBMRRAAA	IBMRRAI: regional sequential output	24
IBMRRABA IBMRRABA	REG(1) sequential unbuffered transmitter	24
IBMRRACA IBMRRACA	BSAM LOAD REG(2) buffered F-format transmitter	24
IBMRRADA IBMRRADA	REG(2) SEQ. unbuffered transmitter	24
IBMRRAEA IBMRRAEA	REG(3) buffered F-format transmitter	24
IBMRRAFA IBMRRAFA	REG(3) sequential unbuffered F-format transmitter	24
IBMRRAGA IBMRRAGA	REG(3) buffered U+V-format transmitter	24
IBMRRAHA IBMRRAHA	REG(3) sequential unbuffered U+V-format transmitter	24
IBMRRAIA IBMRRAIA	REG(3) buffered VS-format transmitter	24
IBMRRBAA IBMRRBAA	BSAM REG(1) buffered F-format transmitter	24
IBMRRBBA IBMRRBBA	BSAM REG(1) unbuffered F-format transmitter	24

Table 32. PL/I Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

PL/I Module Name	Description	RMODE
IBMRRBCA IBMBRBCA	REG(2)+(3) buffered F-format transmitter	24
IBMRRBDA IBMBRBDA	REG(2)+(3) unbuffered F-format transmitter	24
IBMRRBEA IBMBRBEA	BSAM REG(3) buffered U+V-format transmitter	24
IBMRRBFA IBMBRBFA	BSAM REG(3) update U+V-format transmitter	24
IBMRRBGA IBMBRBGA	BSAM REG(3) input/update VS-format transmitter	24
IBMRRCAA IBMBRCAA	BSAM (consecutive) F-format transmitter	24
IBMRCBBA IBMRRCBBA	BSAM (consecutive) U-format transmitter	24
IBMRRCCA IBMBRCCA	BSAM (consecutive) V-format transmitter	24
IBMRRCDA IBMBRCDA	Consecutive unbuffered OMR transmitter	24
IBMRRCEA IBMBRCEA	Consecutive unbuffered device associated F-format transmitter	24
IBMRRDAA IBMBRDAA	REG(1) direct F-format transmitter	24
IBMRRDBA IBMBRDBA	REG(2)+(3) direct F-format transmitter	24
IBMRRDCA IBMBRDCA	REG(3) direct U-format transmitter	24
IBMRRDDA IBMBRDDA	REG(3) direct V+VS-format transmitter	24
IBMRRDAA IBMBREAA	Consecutive buffered record I/O error modules	24
IBMRRDAA IBMBREBA	QISAM+BISAM record I/O error modules	24
IBMRRDAA IBMBRECA	REG+SEQ+T.P. files record I/O error modules	24
IBMRRDAA IBMBREEA	VSAM record I/O error modules	24
IBMRRDAA IBMBREFA	Record I/O endfile module	24
IBMRRJAA IBMBRJAA	QISAM (SCAN) F-format transmitter	24
IBMRRJBA IBMBRJBA	QISAM (SCAN) V-format transmitter	24
IBMRRKAA IBMBRKAA	IBMRRKC: indexed direct non-exclusive	24

Table 32. PL/I Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

PL/I Module Name	Description	RMODE
IBMRRKBA IBMBRKBA	BISAM F-format transmitter	24
IBMRRKCA IBMBRKCA	BISAM V-format transmitter	24
IBMRLAA IBMBRLAA	QISAM (LOAD) F-format transmitter	24
IBMRLBA IBMBRLBA	QISAM (LOAD) V-format transmitter	24
IBMRRQAA IBMBRQAA	QSAM F-format transmitter	24
IBMRRQBA IBMBRQBA	QSAM V-format transmitter	24
IBMRRQCA IBMBRQCA	QSAM U-format transmitter	24
IBMRRQDA IBMBRQDA	QSAM paper tape transmitter	24
IBMRRQEA IBMBRQEA	Buffered consecutive spanned record format input	24
IBMRRQFA IBMBRQFA	Buffered consecutive spanned record format output	24
IBMRRQGA IBMBRQGA	Buffered consecutive record format update	24
IBMRRQHA IBMBRQHA	Consecutive buffered OMR	24
IBMRRQIA IBMBRQIA	Consecutive buffered device associated	24
IBMRRTPA IBMBRTPA	Teleprocessing buffered input/output files	24
IBMRRVAA IBMBRVAA	ESDS transmitter	24
IBMRRVGA IBMBRVGA	KSDS sequential output	24
IBMRRVHA IBMBRVHA	KSDS or PATH input/update/direct	24
IBMRRVIA IBMBRVIA	VSAM RRDS	24
IBMRRXAA IBMBRXAA	REG(1) direct F-format exclusive transmitter	24
IBMRRXBA IBMBRXBA	REG(2)+(3) direct F-format exclusive transmitter	24
IBMRRXCA IBMBRXCA	REG(3) direct U-format exclusive transmitter	24
IBMRRXDA IBMBRXDA	REG(3) direct V+VS-format exclusive transmitter	24

Table 32. PL/I Modules Eligible for Inclusion in the Link Pack Area and the Extended Link Pack Area (continued)

PL/I Module Name	Description	RMODE
IBMRRYAA IBMBRYAA	BISAM F-format transmitter	24
IBMRRYBA IBMBRYBA	BISAM FB-format transmitter	24
IBMRRYCA IBMBRYCA	BISAM V-format transmitter	24
IBMRRYDA IBMBRYDA	BISAM VB-format transmitter	24
IBMRSAP IBMESAP	CICS bootstrap	24
IBMRSICA IBMBLICA	Conversational input transmitter	24
IBMRSOCA IBMBSOCA	Conversational output transmitter	24
IBMRSOFA IBMBSOFA	Output file transmitter (F-format)	24
IBMRSOUA IBMBSOUA	Output file transmitter (U-format)	24
IBMRSOVA IBMBSOVA	Output file transmitter (V-format)	24
IBMRSPCA IBMBSPCA	Conversational file formatting	24
IBMRSFTA IBMBSTFA	Print file transmitter (F-record)	24
IBMRSFTA IBMBSTFA	Input file transmitter	24
IBMRSFTA IBMBSTFA	Print file transmitter (U-record)	24
IBMRSFTA IBMBSTFA	Print file transmitter (V-record)	24
IBMSOPAA IBMBOPAA	Open	24
IBMUPJRO IBMTPJRA	OS PL/I multitasking load module compatibility	24
IBM9LMSA	NLS mixed-case message source	ANY
IBM9LMSN	NLS Japanese message source	ANY
IBM9LMSU	NLS uppercase message source	ANY
IBM9LM2A	NLS mixed-case message	ANY
IBM9LM2N	NLS Japanese message	ANY
IBM9LM2U	NLS uppercase English message	ANY

Appendix E. Modifying the JCL for Japanese National Language Support

The table below specifies additional changes you will need to make in the sample customization jobs if you want to install Language Environment Japanese NLS on the MVS platform.

Table 33. Japanese National Language Support (NLS) JCL Modifications

For this MVS Job...	Modify the JCL like this...
CEEWDXIT CEEWCXIT CEEWUXIT	Change the NATLANG run-time option default in the CEEXOPT macro to NATLANG=(JPN).
IGZWMLP4	To store the Japanese module in the link-pack area, remove the IGZCMGEN module name and add the IGZCMGJA module name.

Appendix F. Language Environment National Language Support Country Codes

The following table contains valid country identifiers along with their respective countries:

Table 34. Country Codes

Code	Country/Region	Code	Country/Region
AD	Andorra	AE	United Arab Emirates
AF	Afghanistan	AG	Antigua and Barbuda
AL	Albania	AN	Netherlands Antilles
AO	Angola	AR	Argentina
AT	Austria	AU	Australia
BA	Bosnia/ Herzegovina	BB	Barbados
BD	Bangladesh	BE	Belgium
BF	Burkina Faso (Upper Volta)	BG	Bulgaria
BH	Bahrain	BI	Burundi
BJ	Benin	BM	Bermuda
BN	Brunei Darussalam	BO	Bolivia
BR	Brazil	BS	Bahamas
BU	Burma	BW	Botswana
CA	Canada	CF	Central African Republic
CG	Congo	CH	Switzerland
CI	Ivory Coast	CL	Chile
CM	Cameroon	CN	People's Republic of China
CO	Colombia	CR	Costa Rica
CS	Czechoslovakia	CU	Cuba
CY	Cyprus	CZ	Czech Republic
DE	Germany	DK	Denmark
DO	Dominican Republic	DZ	Algeria
EC	Ecuador	EE	Estonia
EG	Egypt	ES	Spain
ET	Ethiopia	FI	Finland
FR	France	GA	Gabon
GB	United Kingdom	GH	Ghana
GM	Gambia	GN	Guinea
GR	Greece	GT	Guatemala
GW	Guinea-Bissau	GY	Guyana
HK	China (Hong Kong S.A.R.)	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	Lichtenstein	LK	Sri Lanka
LR	Liberia	LS	Lesotho
LT	Lithuania	LU	Luxembourg
LV	Latvia	LY	Libya
MA	Morocco	MC	Monaco

Table 34. Country Codes (continued)

Code	Country/Region	Code	Country/Region
MG	Madagascar	MK	Macedonia
ML	Mali	MO	China (Macau S.A.R.)
MR	Mauritania	MT	Malta
MU	Mauritius	MW	Malawi
MX	Mexico	MY	Malaysia
MZ	Mozambique	NA	Namibia
NC	New Caledonia	NG	Nigeria
NE	Niger	NI	Nicaragua
NL	Netherlands	NO	Norway
NZ	New Zealand	OM	Oman
PA	Panama	PE	Peru
PG	Papua New Guinea	PH	Philippines
PK	Pakistan	PL	Poland
PR	Puerto Rico	PT	Portugal
PY	Paraguay	QA	Qatar
RO	Romania	RU	Russian Federation
SA	Saudi Arabia	SC	Seychelles
SD	Sudan	SE	Sweden
SG	Singapore	SI	Slovenia
SK	Slovakia	SL	Sierra Leone
SN	Senegal	SO	Somalia
SR	Surinam	SU	Union of Soviet Socialist Republics
SV	El Salvador	SY	Syria
SZ	Swaziland	TD	Chad
TG	Togo	TH	Thailand
TN	Tunisia	TR	Turkey
TT	Trinidad and Tobago	TW	Taiwan
TZ	Tanzania	UG	Uganda
US	United States	UY	Uruguay
VE	Venezuela	VU	Vanuatu
WS	Western Samoa	YE	Yemen
YU	Yugoslavia	ZA	South Africa
ZM	Zambia	ZR	Zaire
ZW	Zimbabwe		

Appendix G. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This book documents intended Programming Interfaces that allow the customer to write programs to obtain the services of Language Environment in OS/390. This publication also documents information that is NOT intended to be used as a Programming Interface of Language Environment.

Trademarks

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

AD/Cycle
AS/400
C++ / MVS
C/370
C/400
CICS/ESA
COBOL/370
DB2
DFSMS/MVS
DFSORT
IBM
IBMLink
IMS/ESA
Language Environment
MVS/ESA
Resource Link
VisualAge
z/OS
z/OS.e
z/Series

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and other countries.

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

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

Bibliography

This section lists the books in the Language Environment library and other publications that may be helpful when using Language Environment.

Language Products Publications

z/OS Language Environment

- *z/OS Language Environment Concepts Guide*, SA22-7567
- *z/OS Language Environment Programming Guide*, SA22-7561
- *z/OS Language Environment Programming Reference*, SA22-7562
- *z/OS Language Environment Customization*, SA22-7564
- *z/OS Language Environment Debugging Guide*, GA22-7560
- *z/OS Language Environment Run-Time Migration Guide*, GA22-7565
- *z/OS Language Environment Writing Interlanguage Communication Applications*, SA22-7563
- *z/OS Language Environment Run-Time Messages*, SA22-7566
- *z/OS Language Environment Vendor Interfaces*, SA22-7568

z/OS C/C++

- *C/C++ Language Reference*, SC09-4815
- *z/OS C/C++ Compiler and Run-Time Migration Guide*, GC09-4913
- *z/OS C/C++ Programming Guide*, SC09-4765
- *z/OS C/C++ User's Guide*, SC09-4767
- *z/OS C/C++ Run-Time Library Reference*, SA22-7821
- *z/OS C/C++ Messages*, GC09-4819
- *IBM Open Class Library User's Guide*, SC09-4811
- *IBM Open Class Library Reference, Vol. 1*, SC09-4812
- *IBM Open Class Library Reference, Vol. 2*, SC09-4813
- *IBM Open Class Library Reference, Vol. 3*, SC09-4814

Enterprise COBOL for z/OS and OS/390

- *Enterprise COBOL for z/OS and OS/390 Licensed Program Specifications*, GC27-1410
- *Enterprise COBOL for z/OS and OS/390 Customization*, GC27-1410
- *Enterprise COBOL for z/OS and OS/390 Language Reference*, SC27-1408
- *Enterprise COBOL for z/OS and OS/390 Programming Guide*, SC27-1412
- *Enterprise COBOL for z/OS and OS/390 Migration Guide*, GC27-1409

COBOL for OS/390 & VM

- *COBOL for OS/390 & VM Licensed Program Specifications*, GC26-9044
- *COBOL for OS/390 & VM Customization under OS/390*, GC26-9045
- *COBOL for OS/390 & VM Language Reference*, SC26-9046
- *COBOL for OS/390 & VM Programming Guide*, SC26-9049
- *COBOL for OS/390 & VM Compiler and Run-Time Migration Guide*, GC26-4764

COBOL for MVS & VM (Release 2)

- *Licensed Program Specifications*, GC26-4761
- *Programming Guide*, SC26-4767
- *Language Reference*, SC26-4769
- *Compiler and Run-Time Migration Guide*, GC26-4764
- *Installation and Customization under MVS*, SC26-4766
- *Reference Summary*, SX26-3788
- *Diagnosis Guide*, SC26-3138

VS COBOL II

VS COBOL II Application Programming Guide for MVS and CMS, SC26-4045

Debug Tool

- *Debug Tool User's Guide and Reference, SC09-2137*

VS FORTRAN Version 2

- *Language Environment Fortran Run-Time Migration Guide, SC26-8499*
- *Language and Library Reference, SC26-4221*
- *Programming Guide for CMS and MVS, SC26-4222*

Enterprise PL/I for z/OS and OS/390, V3R1

- *Enterprise PL/I for z/OS and OS/390 Licensed Program Specifications, GC27-1456*
- *Enterprise PL/I for z/OS and OS/390 Programming Guide, SC27-1457*
- *Enterprise PL/I for z/OS and OS/390 Language Reference, SC27-1460*
- *Enterprise PL/I for z/OS and OS/390 Migration Guide, GC27-1458*
- *Enterprise PL/I for z/OS and OS/390 Messages and Codes, SC27-1461*
- *Enterprise PL/I for z/OS and OS/390 Diagnosis Guide, SC27-1459*

VisualAge PL/I

- *VisualAge PL/I for OS/390 Licensed Program Specifications, GC26-9471*
- *VisualAge PL/I for OS/390 Programming Guide, SC26-9473*
- *VisualAge PL/I Language Reference, SC26-9476*
- *VisualAge PL/I for OS/390 Compiler and Run-Time Migration Guide, SC26-9474*
- *VisualAge PL/I Messages and Codes, SC26-9478*
- *VisualAge PL/I for OS/390 Diagnosis Guide, SC26-9475*

PL/I for MVS & VM

- *PL/I for MVS & VM Licensed Program Specifications, GC26-3116*
- *PL/I for MVS & VM Programming Guide, SC26-3113*
- *PL/I for MVS & VM Language Reference, SC26-3114*
- *PL/I for MVS & VM Reference Summary, SX26-3821*
- *PL/I for MVS & VM Compiler and Run-Time Migration Guide, SC26-3118*
- *PL/I for MVS & VM Installation and Customization under MVS, SC26-3119*
- *PL/I for MVS & VM Compile-Time Messages and Codes, SC26-3229*
- *PL/I for MVS & VM Diagnosis Guide, SC26-3149*

High Level Assembler for MVS & VM

- *Programmer's Guide, MVS & VM Edition, SC26-4941*

Related Publications

CICS

- *CICS Transaction Server for OS/390 Installation Guide, GC34-5985*
- *CICS Operations and Utilities Guide, SC34-5991*
- *CICS Problem Determination Guide, GC34-6002*
- *CICS Resource Definition Guide, SC34-5990*
- *CICS Data Areas, LY33-6100*
- *CICS Application Programming Guide, SC34-5993*
- *CICS Application Programming Reference, SC34-5994*
- *CICS System Definition Guide, SC34-5988*

DB2

Database 2 Application Programming and SQL Guide, SC26-4377

DFSMS/MVS

z/OS DFSMS Program Management, SC27-1130
z/OS DFSMS DFM Guide and Reference, SC26-7395

IPCS

- *z/OS MVS IPCS User's Guide*, SA22-7596
- *z/OS MVS IPCS Commands*, SA22-7594
- *z/OS MVS IPCS Customization*, SA22-7595

DFSORT

DFSORT Application Programming Guide R14, SC33-4035

IMS/ESA

IMS/ESA Application Programming: Design Guide, SC26-8728
IMS/ESA Application Programming: Database Manager, SC26-8727
IMS/ESA Application Programming: Transaction Manager, SC26-8729
IMS/ESA Application Programming: EXEC DLI Commands for CICS and IMS, SC26-8726

msys for Setup

- *z/OS Managed System Infrastructure for Setup User's Guide*, SC33-7985

z/OS

- *z/OS Introduction and Release Guide*, GA22-7502
- *z/OS Program Directory*, GI10-0670
- *z/OS and z/OS.e Planning for Installation*, GA22-7504
- *z/OS Information Roadmap*, SA22-7500
- *z/OS Hot Topics Newsletter*, GA22-7501
- *z/OS Licensed Program Specifications*, GA22-7503
-
- *z/OS ISPF Dialog Tag Language Guide and Reference*, SC34-4824
- *z/OS ISPF Planning and Customizing*, GC34-4814
- *z/OS ISPF Dialog Developer's Guide and Reference*, SC34-4821
-
- *z/OS UNIX System Services User's Guide*, SA22-7801
- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803
- *z/OS UNIX System Services Planning*, GA22-7800
-
- *z/OS TSO/E Customization*, SA22-7783
- *z/OS TSO/E Programming Services*, SA22-7789
- *z/OS TSO/E System Programming Command Reference*, SA22-7793

z/OS.e

- *z/OS.e Overview*, GA22-7869
- *z/OS.e Licensed Program Specifications*, GA22-7503

Softcopy Publications

z/OS Collection, SK3T-4269

Index

A

abend
 CEEEXTAN CSECT 33
 CICS 35
 customization job 34
 identifying 33
 non-CICS (MVS batch) 35
abnormal termination exit
 CEEEXTAN CSECT 33
 CICS 35
 customization job 34
 identifying 33
 non-CICS (MVS batch) 35
ABPERC run-time option 68
ABTERMENC run-time option 69
accessibility 225
AIXBLD run-time option 71
ALL31 run-time option 72
ANYHEAP run-time option 73
AUTOTASK run-time option 75

B

BELOWHEAP run-time option 76

C

C
 customizing locale time 195
 options 196, 197
 system programming facilities 197
cataloged procedure
 list of 39
 making available to your jobs 40
CBLOPTS run-time option 77
CBLPSHPOP run-time option 78
CBLQDA run-time option 79
CEEBDATX abnormal termination exit 33
CEECOPT 21
CEEDOPT 21
CEEROPT 26
CEEUOPT 21
CEEWCOPT 21
CEEWCXIT 27, 28
CEEWDEXT 27, 28, 33
CEEWDOPT 21
CEEWDXIT 27, 28
CEEWHLLX 27, 28, 32
CEEWROPT 26
CEEWUOPT 21
CEEWUXIT 27, 28
CEEXOPT macro 21
CHECK run-time option 80
CICS
 customizing
 abnormal termination exit 35
 run-time options 26

CICS (*continued*)
 installing Language Environment support for 49
COBOL
 compatibility with Language Environment
 options 67
 run-time options specific to 67
COBPACK xviii
command
 syntax diagrams xiii
condition nesting 82
COUNTRY run-time option 80, 223
CSECT
 CEECOPT, CICS macro
 modifying 26
 sample of 23
 CEEDOPT, non-CICS macro
 modifying 25
 sample of 22
customizing
 Fortran 65
 high-level language user exit 32
 job samples 21
 procedure 1, 64
Customizing locale time 195

D

Daylight Saving Time (DST) C option 196
DCT (destination control table). 50
DEBUG run-time option 81
DEPTHCONDLMT run-time option 82
Description of Language Environment Target
 Libraries 3
destination control table (DCT) 50
disability 225
documents, licensed xv
DSTEND (C option) 197
DSTENM (C option) 196
DSTENW (C option) 196
DSTNAME (C option) 196
DSTSTD (C option) 196
DSTSTM (C option) 196
DSTSTW (C option) 196

E

ELPA (extended link pack area) 199
ENDTM (C option) 197
ENVAR run-time option 84
ERRCOUNT run-time option 85
ERRUNIT run-time option 86
exit
 abnormal termination 33
 high-level language user 32

F

- FILEHIST run-time option 87
- FILETAG run-time option 88
- Fortran
 - customizing for Fortran applications 65, 66, 163, 184
 - LIBPACKs
 - tailoring Fortran LIBPACKs 44, 48, 184, 195

H

- HEAP run-time option 90
- high-level language user exit 32

I

- IMS
 - performance considerations 57
- INQPCOPN run-time option 97
- installation
 - support for CICS 49
- INTERRUPT run-time option 97

J

- JCL (Job Control Language)
 - common modifications
 - for customization 1
- Job Control Language (JCL)
 - common modifications
 - for customization 1

K

- keyboard 225

L

- Language Environment
 - customizing 1
- LC_TOD, C locale time information 195
- library 39
- library routine retention 57
- LIBSTACK run-time option 99
- licensed documents xv
- link pack area (LPA).
 - installing Language Environment into 199
- locale time information, C 195
- LookAt message retrieval tool xvi
- LPA (link pack area)
 - installing Language Environment into 199

M

- message retrieval tool, LookAt xvi
- MSGFILE run-time option 101
- MSGQ run-time option 104

Multiple Virtual System (MVS)

- customizing
 - C locale time 195
- installing
 - in a link pack area 199

N

- National Language Support (NLS) 221
- NATLANG run-time option 105
- nested conditions
 - limiting 82
- Notices 227

O

- OCSTATUS run-time option 107
- OS/VS COBOL 60

P

- passing
 - parameters at invocation 67, 77
 - run-time options at invocation 67, 77
- PC run-time option 108
- performance
 - considerations for IMS 57
- procedure, cataloged
 - list of 39
 - making available to your jobs 40
- PRTUNIT run-time option 111
- PUNUNIT run-time option 112

R

- RDRUNIT run-time option 113
- RECPAD run-time option 113
- RPTOPTS run-time option 114
- RPTSTG run-time option 117
- RTEREUS run-time option 127
- run-time options 67
 - ABPERC—percolate an abend 68
 - ABTERMENC—determine how an enclave terminates 69
 - AIXBLD—invoke AMS for COBOL 71
 - ALL31—indicate whether application runs in AMODE(31) 72
 - ANYHEAP—control unrestricted library heap storage 73
 - AUTOTASK—specify whether Fortran MTF is to be used 75
 - BELOWHEAP—control library heap storage below 16 MB 76
 - CBLOPTS—specify format of COBOL argument 77
 - CBLPSHPOP—control CICS commands 78
 - CBLQDA—control COBOL QSAM 79
 - CHECK—detect checking errors 80
 - COBOL-specific 67
 - COUNTRY—specify default date/time formats 80
 - DEBUG—activate COBOL batch debugging 81

run-time options (*continued*)

DEPTHCONDLMT—limit extent of nested conditions 82
ENVAR—set initial values for environment variables 84
ERRCOUNT—specify number of errors allowed 85
ERRUNIT—specify unit number to which error information is directed 86
FILEHIST—specify whether to allow a file definition to be changed at run time 87
FILETAG—specify whether to allow AUTOTAG / AUTOCVT. 88
HEAP—control allocation of heaps 90
HEAPCK—runs additional heap check tests 92
HEAPOOLS—control allocation of optional heap pools storage 94
INFOMSGFILTER—eliminates unwanted informational messages 95
INQPCOPN—control value in OPENED specifier of INQUIRE by unit statement 97
INTERRUPT—cause attentions to be recognized by Language Environment 97
LIBRARY—logical library name for finding RTLS modules 98
LIBSTACK—control library stack storage 99
MSGFILE—specify ddname of diagnostic file 101
MSGQ—specify number of ISI blocks allocated 104
NATLANG—specify national language 105
OCSTATUS—control checking of file existence and whether file deletion occurs 107
PC—control whether Fortran status common blocks are shared among load modules 108
PLITASKCOUNT—control the maximum number of active tasks 109
POSIX—specify whether enclave runs with POSIX semantics 109
PROFILE—controls optional PROFILE use 111
PRTUNIT—specifies unit number used for PRINT and WRITE statements 111
PUNUNIT—specifies unit number used for PUNCH statements 112
RDRUNIT—specifies unit number used for READ statements 113
RECPAD—specifies whether a formatted input record is padded with blanks 113
RPTOPTS—generate a report of run-time options used 114
RPTSTG—generate a report of storage used 117
RTEREUS—initialize a reusable COBOL environment 127
RTLS—specifies version-controlled libraries 128
SIMVRD—specify VSAM KSDS for COBOL 129
STACK—allocate stack storage 130
STORAGE—control storage 133
TERMTHDACT—specify type of information generated with unhandled error 136
TEST—indicate debug tool to gain control 142
THREADHEAP—control the allocation of thread-level heap storage 145
THREADSTACK—control the allocation of stack storage 146

run-time options (*continued*)

TRACE—activate Language Environment run-time library tracing 149
TRAP—handle abends and program interrupts 151
UPSI—set UPSI switches. 153
USRHDLR—register a user condition handler at stack frame 0 154
VCTRSAVE—use vector facility 155
VERSION—specifies version for finding RTLS modules 156
XUFLOW—specify program interrupt due to exponent underflow 157

S

sample job
 Fortran 65
 high-level language user exit 32
 job samples 21
 procedure 1, 64
SHIFT C option 197
shortcut keys 225
SIMVRD run-time option 129
SPC (system programmer C) 197
STACK run-time option 130
STARTTM C option 197
storage
 required for MVS
 link pack area for MVS 199
STORAGE run-time option 133
syntax diagrams
 how to read xiii
system programming facilities, C 197

T

TERMTHDACT run-time option 136
TEST run-time option 142
THREADSTACK run-time option 146
TNAME C option 196
TRACE run-time option 149
TRAP run-time option 151
TSO/E LOGON procedure 39
TZDIFF C option 196

U

UPSI run-time option 67
 ABPERC—percolate an abend 68
 ABTERMENC—determine how an enclave terminates 69
 AIXBLD—invoke AMS for COBOL 71
 ALL31—indicate whether application runs in AMODE(31) 72
 ANYHEAP—control unrestricted library heap storage 73
 AUTOTASK—specify whether Fortran MTF is to be used 75
 BELOWHEAP—control library heap storage below 16 MB 76
 CBLOPTS—specify format of COBOL argument 77

UPSI run-time option (*continued*)

- CBLPSHPOP—control CICS commands 78
- CBLQDA—control COBOL QSAM 79
- CHECK—detect checking errors 80
- COBOL-specific 67
- COUNTRY—specify default date/time formats 80
- DEBUG—activate COBOL batch debugging 81
- DEPTHCONDLMT—limit extent of nested conditions 82
- ENVAR—set initial values for environment variables 84
- ERRCOUNT—specify number of errors allowed 85
- ERRUNIT—specify unit number to which error information is directed 86
- FILEHIST—specify whether to allow a file definition to be changed at run time 87
- FILETAG—specify whether to allow AUTOTAG / AUTOCVT. 88
- HEAP—control allocation of heaps 90
- HEAPCK—runs additional heap check tests 92
- HEAPOOLS—control allocation of optional heap pools storage 94
- INFOMSGFILTER—eliminates unwanted informational messages 95
- INQPCOPN—control value in OPENED specifier of INQUIRE by unit statement 97
- INTERRUPT—cause attentions to be recognized by Language Environment 97
- LIBRARY—logical library name for finding RTLS modules 98
- LIBSTACK—control library stack storage 99
- MSGFILE—specify ddname of diagnostic file 101
- MSGQ—specify number of ISI blocks allocated 104
- NATLANG—specify national language 105
- OCSTATUS—control checking of file existence and whether file deletion occurs 107
- PC—control whether Fortran status common blocks are shared among load modules 108
- PLITASKCOUNT—control the maximum number of active tasks 109
- POSIX—specify whether enclave runs with POSIX semantics 109
- PROFILE—controls optional PROFILE use 111
- PRTUNIT—specifies unit number used for PRINT and WRITE statements 111
- PUNUNIT—specifies unit number used for PUNCH statements 112
- RDRUNIT—specifies unit number used for READ statements 113
- RECPAD—specifies whether a formatted input record is padded with blanks 113
- RPTOPTS—generate a report of run-time options used 114
- RPTSTG—generate a report of storage used 117
- RTEREUS—initialize a reusable COBOL environment 127
- RTLS—specifies version-controlled libraries 128
- SIMVRD—specify VSAM KSDS for COBOL 129
- STACK—allocate stack storage 130
- STORAGE—control storage 133

UPSI run-time option (*continued*)

- TERMTHDACT—specify type of information generated with unhandled error 136
- TEST—indicate debug tool to gain control 142
- THREADHEAP—control the allocation of thread-level heap storage 145
- THREADSTACK—control the allocation of stack storage 146
- TRACE—activate Language Environment run-time library tracing 149
- TRAP—handle abends and program interrupts 151
- UPSI—set UPSI switches. 153
- USRHDLR—register a user condition handler at stack frame 0 154
- VCTRSAVE—use vector facility 155
- VERSION—specifies version for finding RTLS modules 156
- XUFLOW—specify program interrupt due to exponent underflow 157

user

- exit
 - high-level language 32

USRHDLR run-time option 154

V

VCTRSAVE run-time option 155

W

worksheet

- changing run-time option defaults on MVS 19

X

XUFLOW run-time option 157

Readers' Comments — We'd Like to Hear from You

**z/OS
Language Environment
Customization**

Publication No. SA22-7564-03

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>				

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<input type="checkbox"/>				

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? Yes No

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

Name

Address

Company or Organization

Phone No.



Fold and Tape

Please do not staple

Fold and Tape



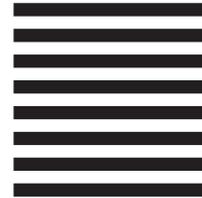
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400



Fold and Tape

Please do not staple

Fold and Tape



Program Number: 5694-A01, 5655-G52

Printed in U.S.A.

SA22-7564-03

