

Enterprise COBOL for z/OS



# Customization Guide

*Version 5.2*



Enterprise COBOL for z/OS



# Customization Guide

*Version 5.2*

**Note**

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 83.

**Fourth edition (March 2019)**

- | This edition applies to Version 5 Release 2 of IBM Enterprise COBOL for z/OS (program number 5655-W32) and to
- | all subsequent releases and modifications until otherwise indicated in new editions. Make sure that you are using
- | the correct edition for the level of the product.

You can view or download softcopy publications free of charge at [www.ibm.com/shop/publications/order/](http://www.ibm.com/shop/publications/order/).

© Copyright IBM Corporation 1996, 2018.

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

---

# Contents

<b>Figures</b> . . . . .	<b>v</b>
--------------------------	----------

<b>Tables</b> . . . . .	<b>vii</b>
-------------------------	------------

<b>Preface</b> . . . . .	<b>ix</b>
--------------------------	-----------

About this information . . . . .	ix
How to read the syntax diagrams . . . . .	ix
Using the macro planning worksheets . . . . .	x
Summary of changes . . . . .	xi
How to send your comments . . . . .	xii
Accessibility . . . . .	xii
Interface information . . . . .	xiii
Keyboard navigation . . . . .	xiii
Accessibility of this information . . . . .	xiii
IBM and accessibility . . . . .	xiii

## Chapter 1. Planning to customize

<b>Enterprise COBOL</b> . . . . .	<b>1</b>
-----------------------------------	----------

Making changes after installation: why customize? . . . . .	1
Planning to modify compiler option default values . . . . .	1
Making compiler options fixed . . . . .	2
Modifying compiler options and phases . . . . .	4
Planning to place compiler phases in shared storage . . . . .	6
Why place the compiler phases in shared storage? . . . . .	7
Compiler phases and their defaults . . . . .	8
Planning to create an additional reserved word table . . . . .	12
Why create additional reserved word tables? . . . . .	12
Controlling use of nested programs . . . . .	13
Reserved word tables supplied with Enterprise COBOL . . . . .	13
Using product registration to enable or disable Enterprise COBOL . . . . .	14

<b>Chapter 2. Enterprise COBOL compiler options</b> . . . . .	<b>15</b>
---	-----------

Specifying COBOL compiler options . . . . .	15
Conflicting compiler options . . . . .	15
Compiler options for standards conformance . . . . .	16
Compiler options syntax and descriptions . . . . .	16
ADATA . . . . .	17
ADEXIT . . . . .	17
ADV . . . . .	18
AFP . . . . .	18
ALOWCBL . . . . .	19
ARCH . . . . .	19
ARITH . . . . .	21
AWO . . . . .	21
BLOCK0 . . . . .	22
BUF . . . . .	22
CICS . . . . .	23
CODEPAGE . . . . .	23
COMPILE . . . . .	24
COPYRIGHT . . . . .	24
CURRENCY . . . . .	25

DATA . . . . .	26
DBCS . . . . .	27
DBCSXREF . . . . .	27
DECK . . . . .	28
DIAGTRUNC . . . . .	29
DISPSIGN . . . . .	29
DLL . . . . .	30
DYNAM . . . . .	31
EXPORTALL . . . . .	31
FASTSRT . . . . .	32
FLAG . . . . .	32
FLAGSTD . . . . .	33
HGPR . . . . .	35
INEXIT . . . . .	36
INITCHECK . . . . .	36
INTDATE . . . . .	37
LANGUAGE . . . . .	38
LIBEXIT . . . . .	38
LINECNT . . . . .	39
LIST . . . . .	39
LITCHAR . . . . .	40
LVLINFO . . . . .	40
MAP . . . . .	41
MAXPCF . . . . .	41
MDECK . . . . .	42
MSGEXIT . . . . .	43
NAME . . . . .	43
NSYMBOL . . . . .	44
NUM . . . . .	44
NUMCHECK . . . . .	45
NUMCLS . . . . .	47
NUMPROC . . . . .	47
OBJECT . . . . .	48
OFFSET . . . . .	49
OPTIMIZE . . . . .	49
OUTDD . . . . .	50
PGMNAME . . . . .	50
PRTEXIT . . . . .	51
QUALIFY . . . . .	51
RENT . . . . .	52
RMODE . . . . .	53
RULES . . . . .	54
SERVICE . . . . .	55
SEQ . . . . .	55
SOURCE . . . . .	56
SPACE . . . . .	56
SQL . . . . .	56
SQLCCSID . . . . .	57
SQLIMS . . . . .	58
SSRANGE . . . . .	58
STGOPT . . . . .	60
TERM . . . . .	60
TEST . . . . .	61
THREAD . . . . .	62
TRUNC . . . . .	63
VBREF . . . . .	64

VLR . . . . .	65
VSAMOPENFS . . . . .	66
WORD . . . . .	66
XMLPARSE . . . . .	67
XREFOPT . . . . .	68
ZONECHECK . . . . .	69
ZONEDATA . . . . .	69
ZWB . . . . .	70

## Chapter 3. Customizing Enterprise

### COBOL . . . . . 73

Summary of user modifications. . . . .	73
Changing the defaults for compiler options. . . . .	74
Changing the compiler options default module . . . . .	75
Overriding options specified as fixed . . . . .	75
Changing reserved words . . . . .	76
Creating or modifying a reserved word table . . . . .	77
Coding control statements . . . . .	77
Rules for coding control statements . . . . .	78
Coding operands in control statements . . . . .	78

Rules for coding control statement operands . . . . .	79
ABBR statement . . . . .	79
INFO statement . . . . .	79
RSTR statement . . . . .	80
Modifying and running non-SMP/E JCL . . . . .	80
Running JCL that creates a reserved word table . . . . .	81
Placing Enterprise COBOL modules in shared storage . . . . .	81
Tailoring the cataloged procedures to your site . . . . .	82

### Notices . . . . . 83

Programming interface information . . . . .	85
Trademarks . . . . .	85

### List of resources . . . . . 87

Enterprise COBOL for z/OS. . . . .	87
Related publications . . . . .	87

### Index . . . . . 89

---

## Figures

- |    |  |    |
|----|--|----|
| 1. | Syntax format for IGYCOPT compiler options<br>and phases macro . . . . . | 4  |
| 2. | Syntax format for reserved word control<br>statements . . . . .          | 78 |





---

## Tables

1.	IGYCDOPT worksheet for options . . . . .	5	6.	Effect of RENT and RMODE on residency mode . . . . .	53
2.	IGYCDOPT program worksheet for compiler phases . . . . .	11	7.	Effect of RMODE and RENT   NORENT on residency mode . . . . .	54
3.	Conflicting compiler options . . . . .	15	8.	Summary of user modification jobs for Enterprise COBOL . . . . .	73
4.	DISPLAY output with the DISPSIGN=COMPAT option or the DISPSIGN=SEP option specified: .	30			
5.	Entries for the LANGUAGE compiler option	38			



---

## Preface

---

### About this information

This information is intended for systems programmers who are responsible for customizing IBM® Enterprise COBOL for z/OS® for their location. It provides information needed to plan for and customize Enterprise COBOL under z/OS. This information can also help you assess the value of Enterprise COBOL to your organization.

In this information, the generic term "operating system" is used to refer to z/OS.

To use this information, and ensure successful customization, you should have a knowledge of Enterprise COBOL and of your system's operating environment.

### How to read the syntax diagrams

This section describes how to read the syntax diagrams in this information.

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following table shows the meaning of symbols at the beginning and end of syntax diagram lines.

Symbol	Indicates
>>—	The syntax diagram starts here
—>	The syntax diagram is continued on the next line
>—	The syntax diagram is continued from the previous line
—><	The syntax diagram ends here

Diagrams of syntactical units other than complete statements start with the >— symbol and end with the —> symbol.

- Required items appear on the horizontal line (the main path).

▶▶—STATEMENT—required item————▶▶

- Optional items appear below the main path.

▶▶—STATEMENT—  
                  └ optional item ┘————▶▶

- When you can choose from two or more items, they appear vertically in a stack.

If you *must* choose one of the items, one item of the stack appears on the main path. The default, if any, appears above the main path and is chosen by the IGYCOPT macro if you do not specify another choice. In some cases, the default is affected by the system in which the program is being run.

▶▶—STATEMENT—  
                  ┌ default-item ┐  
                  └ required choice 1 ┘  
                  └ required choice 2 ┘————▶▶

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



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



A repeat arrow above a stack indicates that you can make more than one choice from the stacked items, or repeat a single choice.

- Keywords appear in uppercase letters (for example, PRINT). They must be spelled exactly as shown. Variables appear in an italic font (for example, *item*). They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or such symbols are shown, they must be entered as part of the syntax.
- Use at least one blank or comma to separate parameters.

For a description of the meaning of the asterisk (\*) in syntax diagrams, and for further information, see “Compiler options syntax and descriptions” on page 16.

## Using the macro planning worksheets

The planning worksheets in this information (“IGYCDOPT worksheet for compiler options” on page 4 and “IGYCDOPT worksheet for compiler phases” on page 11) will help you prepare to customize Enterprise COBOL. By completing the worksheets, you will be able to easily identify those values that you want to change from the IBM-supplied defaults. You might then want to use the worksheets as a source from which to customize the IBM-supplied default values.

The headings in each worksheet differ somewhat from each other. See the following list of definitions for an explanation of the column headings in the worksheet for compiler options.

### Compiler option

The options contained within a specific installation macro. This column represents the options exactly as they are in the macro.

### Enter \* for fixed

The options that cannot be overridden by an application programmer. Enter an asterisk (\*) only for those options that you want to be fixed.

### Enter selection

The value associated with each option. In the space provided, enter the value that you want to assign to each option. To assist you in selecting the appropriate value, see the reference in the **Syntax description** column.

### IBM-supplied default

The value that is supplied for the specified installation macro if the option is not altered. If the IBM-supplied default is the value that you want, you do not need to modify that option within that specific macro.

### Syntax description

The topic that contains the syntax diagram and more specific information about the given option.

After you have completed the worksheets, identify those options that are different from the IBM-supplied defaults. These are the items that you must code in the installation macros. The worksheet entries are positioned such that the order of the entries is consistent with the actual coding semantics.

## Summary of changes

This section lists the key changes that have been made to this document for Enterprise COBOL Version 5 Release 2 and Version 5 Release 2 with PTFs installed. The latest technical changes are marked within >| and |< in the HTML version, or marked by vertical bars (|) in the left margin in the PDF version.

### Version 5 Release 2 with PTFs installed

- The following compiler options are new:
  - PI40822: ZONECHECK (“ZONECHECK” on page 69)
  - PI69197: INITCHECK (“INITCHECK” on page 36)
  - PI81006: NUMCHECK (“NUMCHECK” on page 45)
  - PI85868: VSAMOPENFS (“VSAMOPENFS” on page 66)
- The following compiler options are modified:
  - PI40853: ZONEDATA: New suboption of NOPFD is added to the ZONEDATA compiler option. ZONEDATA=NOPFD lets the compiler generate code that performs comparisons of zoned decimal data in the same manner as COBOL V4 does when using NUMPROC=NOPFD|PFD with COBOL V4 (“ZONEDATA” on page 69).
  - PI53044: SSRANGE: New suboptions ZLEN and NOZLEN are added to the SSRANGE compiler option to control how the compiler checks reference modification lengths (“SSRANGE” on page 58).
  - PI86343: SSRANGE: New suboptions MSG and ABD are added to the SSRANGE compiler option to control how the compiler checks reference modification lengths. (“SSRANGE” on page 58)
  - PI90458: ZONEDATA: The ZONEDATA option is updated to affect the behaviour of MOVE statements, comparisons, and computations for USAGE DISPLAY or PACKED-DECIMAL data items that could contain invalid digits, an invalid sign code, or invalid zone bits (“ZONEDATA” on page 69).
  - PI97835: NUMCHECK(PAC): For packed decimal (COMP-3) data items that have an even number of digits, the unused bits are checked for zeros. (“NUMCHECK” on page 45)
  - PH01241: NUMCHECK(ZON): New suboptions ALPHNUM | NOALPHNUM are added to the NUMCHECK(ZON) option to control whether the compiler will generate code for an implicit numeric class test for zoned decimal data items that are being compared with an alphanumeric data item, alphanumeric literal or alphanumeric figurative constant. (“NUMCHECK” on page 45)
- The following compiler option is removed:
  - PI81006: ZONECHECK is deprecated and can no longer be specified in IGYCDOPT. NUMCHECK=(ZON) gives the same results as ZONECHECK used to. (“ZONECHECK” on page 69)

### Version 5 Release 2

- Several changes are made to compiler options:
  - The following compiler options are new:
    - COPYRIGHT (“COPYRIGHT” on page 24)
    - QUALIFY=COMPAT | EXTEND (“QUALIFY” on page 51)

- RULES ("RULES" on page 54)
- SERVICE ("SERVICE" on page 55)
- SQLIMS ("SQLIMS" on page 58)
- VLR=COMPAT | STANDARD ("VLR" on page 65)
- XMLPARSE=XMLSS | COMPAT ("XMLPARSE" on page 67)
- ZONEDATA=PFD | MIG ("ZONEDATA" on page 69)
- The following compiler options are modified:
  - ARCH: ARCH(6) is no longer accepted. A new higher level of ARCH(11) is accepted, and ARCH(7) is the default ("ARCH" on page 19).
  - MAP: New suboptions HEX and DEC are added to the MAP compiler option to control whether hexadecimal or decimal offsets are shown for MAP output in the compiler listing ("MAP" on page 41).
- The following compiler option is removed:
  - SIZE
- XML COMPAT support is restored. You can specify the XMLPARSE(XMLSS|COMPAT) compiler option to choose between parsing with the compatibility-mode COBOL XML parser from the COBOL library, or with the z/OS XML System Services parser. It can ease your migration to the Enterprise COBOL V5 compiler.

---

## How to send your comments

Your feedback is important in helping us to provide accurate, high-quality information. If you have comments about this information or any other Enterprise COBOL documentation, contact us in one of these ways:

- Use the Online Readers' Comments Form at [www.ibm.com/software/awdtools/rcf/](http://www.ibm.com/software/awdtools/rcf/).
- Send your comments to the following address: [compinfo@cn.ibm.com](mailto:compinfo@cn.ibm.com).

Be sure to include the name of the document, the publication number, the version of Enterprise COBOL, and, if applicable, the specific location (for example, the page number or section heading) of the text that you are commenting on.

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

---

## Accessibility

Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use software products successfully. The accessibility features in z/OS provide accessibility for Enterprise COBOL.

The major accessibility features in z/OS are:

- Interfaces that are commonly used by screen readers and screen-magnifier software
- Keyboard-only navigation
- Ability to customize display attributes such as color, contrast, and font size

## Interface information

Assistive technology products work with the user interfaces that are found in z/OS. For specific guidance information, see the documentation for the assistive technology product that you use to access z/OS interfaces.

## Keyboard navigation

Users can access z/OS user interfaces by using TSO/E or ISPF. For information about accessing TSO/E or ISPF interfaces, see the following publications:

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Volume I*

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.

## Accessibility of this information

The English-language XHTML format of this information that will be provided in the IBM Knowledge Center at [www.ibm.com/support/knowledgecenter/en/SS6SG3\\_5.2.0/welcome.html](http://www.ibm.com/support/knowledgecenter/en/SS6SG3_5.2.0/welcome.html) is accessible to visually impaired individuals who use a screen reader.

To enable your screen reader to accurately read syntax diagrams, source code examples, and text that contains the period or comma PICTURE symbols, you must set the screen reader to speak all punctuation.

## IBM and accessibility

See the IBM Human Ability and Accessibility Center at [www.ibm.com/able](http://www.ibm.com/able) for more information about the commitment that IBM has to accessibility.





---

## Chapter 1. Planning to customize Enterprise COBOL

When you plan the customization of Enterprise COBOL, you need to consider whether to modify compiler-option default values, whether to place compiler phases in shared storage, and whether to create an additional reserved-word table.

The following information helps you plan your customization:

- “Making changes after installation: why customize?”
- “Planning to modify compiler option default values”
- “Planning to place compiler phases in shared storage” on page 6
- “Planning to create an additional reserved word table” on page 12
- “Using product registration to enable or disable Enterprise COBOL” on page 14

If you're installing IBM Debug Tool, you can decide whether to place its modules in shared storage, and whether to set up your CICS® environment to work with Debug Tool.

For the actual customization procedures, see Chapter 3, “Customizing Enterprise COBOL,” on page 73.

This information also contains worksheets to help you plan modifications to the IBM-supplied default values within macros. For an explanation about the planning sheets, see “Using the macro planning worksheets” on page x.

**Important:** Confer with the application programmers at your site while you plan the customization of Enterprise COBOL. Doing so will ensure that the modifications you make serve their needs and support the applications being developed.

---

### Making changes after installation: why customize?

When you install Enterprise COBOL, you receive IBM-supplied defaults for compiler options and phases, and for the reserved word table. You might want to customize Enterprise COBOL to better suit the needs of application programmers at your site.

After you install Enterprise COBOL, you can:

- Modify the default values of compiler options: see “Planning to modify compiler option default values.”
- Make compiler options fixed: see “Making compiler options fixed” on page 2.
- Modify the residency values of the compiler phases: see “Planning to place compiler phases in shared storage” on page 6.
- Create additional reserved word tables: see “Planning to create an additional reserved word table” on page 12.

---

### Planning to modify compiler option default values

Compiler option defaults and residency for compiler phases are set in the IGYCDOPT program.

IGYCDOPT is link-edited with AMODE 31 and RMODE ANY during installation.

For compiler option defaults and residency for compiler phases set in the IGYCDOPT program, see Table 1 on page 5 and Table 2 on page 11, respectively.

When you assemble COBOL customization parts, such as IGYCDOPT, you need access to a system MACLIB. Typically, the MACLIB is found in SYS1.MACLIB. You also need access to the COBOL MACLIB IGY.V6R2M0.SIGYMAC.

The IGYCDOPT program has two purposes: it lets you select and fix the defaults for compiler options, and specify which compiler phases are in shared storage. You can accept the IBM-supplied compiler option values when you install Enterprise COBOL, or you can modify them to better suit the needs of programmers at your location. You can also choose whether your application programmers will have the ability to override these options.

**Note:** The high-level qualifier IGY.V6R2M0 might have been changed when Enterprise COBOL was installed.

If you identify compiler phases that reside in shared system storage, the compiler can use the storage in the region for work areas. For a more detailed description of why you might want to modify the phase defaults, see “Why place the compiler phases in shared storage?” on page 7.

## Making compiler options fixed

These sections explain why you might want to make a compiler option fixed, and explain how you can fix a compiler option and bypass a fixed option.

Enterprise COBOL can help you to set up your site's unique programming standards. For example, many sites might want RENT as the preferred compiler option setting, and so want to enforce its use.

With Enterprise COBOL, you use the IGYCDOPT program to specify that an option is fixed and cannot be changed or overridden at compile time. Then, at compile time, an attempt to override a fixed option results in a diagnostic message with a nonzero compiler return code.

When certain options are fixed for consistent usage, there might be special conditions that require the ability to bypass a fixed option. This change can be made by assembling a temporary copy of the IGYCDOPT program with different parameters. At compile time, programmer can use a JOBLIB or STEPLIB that contains the required IGYCDOPT module to bypass the fixed option.

For example, if you select the OPT=1 option to be fixed (indicating that you always want the COBOL compiler to generate optimized object code), and then need to exempt an application from this requirement, you must reassemble the IGYCDOPT program after you remove the asterisk parameter from the option. You then place the resulting IGYCDOPT module in a temporary library to be accessed as a JOBLIB or STEPLIB at compile time.

### Sample installation jobs

Enterprise COBOL provides two sample installation jobs that you can modify and then use to change the defaults for compiler options. One sample job, IGYWDOPT, provides an example of how to change the IBM-supplied defaults for compilers. The other sample job, IGYWUOPT, provides an example of how to override compiler options that have been fixed. These jobs are located in the COBOL sample data set IGY.V6R2M0.SIGYSAMP.

## IGYWDOPT

Use this sample installation job to change the IBM-supplied defaults using SMP/E.

If IGYWDOPT is being run for the first time, complete the following steps:

1. Change the job card to meet your system requirements.
2. Change these items:
  - #globalcsi (Make it the CSI name of the installation site)
  - #tzone (Make it the TARGET ZONE name of the installation site)
3. Copy member SIGYSAMP(IGYCDOPT) into SIGYSAMP(IGYWDOPT) in place of the comment lines following the ++ SRC statement in step DOPT.
4. Modify the IGYCDOPT text that was just copied in so that it contains the list of compiler options that need to be overridden. For example:

```
COPY IGYCOPT
      IGYCDOPT CSECT
      IGYCDOPT AMODE ANY
      IGYCDOPT RMODE ANY
              IGYCOPT ARCH=10,
                      OPTIMIZE=*2,
                      ZONECHECK=MSG,
                      ZONEDATA=NOPFD
      END    IGYCDOPT
```

Use the continuation "X" in column 72 as needed.

5. Run IGYWDOPT to receive and apply the usermod to create a customized version of MOD(IGYCDOPT).
6. **Important:** Save a copy of the modified SIGYSAMP(IGYWDOPT) for future reference.

**CAUTION:** Do not ACCEPT the usermod! Accepting the usermod makes it impossible to RESTORE it later in SMP/E when needed.

If MOD(IGYCDOPT) is being changed by an IBM PTF and requires a rerun of the SIGYSAMP(IGYWDOPT) job, complete the following steps:

1. RESTORE the usermod created by the IGYWDOPT job. This is done via the SMP/E command RESTORE SELECT (IGYWDOP). Doing this will restore MOD(IGYCDOPT) back to the previous IBM PTF level, which will then allow the new IBM PTF to apply properly.
2. Apply the IBM PTF.
3. Change the rework date by changing the REWORK parameter on the ++ USERMOD statement to the date the changes are being made.
4. Add the proper "PRE( )" statement after the "FMID( )" statement. This is typically the PTF number that was just applied. See the technote to determine what PRE statement to add if an error occurs.
5. Using the SIGYSAMP(IGYWDOPT) backup member as a reference and referring to any options that may have been added or modified in SIGYMAC(IGYCOPT), update IGYWDOPT so that it contains the list of compiler options that need to be overridden.
6. Rerun IGYWDOPT to receive and apply the usermod to recreate a customized version of MOD(IGYCDOPT).
7. **Important:** Save off a copy of the modified SIGYSAMP(IGYWDOPT) for future reference.

IGYWDOPT should run with a condition code of 0.  
 Check the IGYNNNN informational messages in the ASSEMBLER  
 SYSPRINT data set to verify the options that will be in effect when the  
 new IGYCDOPT module is used.

### IGYWUOPT

Use this sample installation job to create a module outside of SMP/E in  
 which you can specify different defaults if it becomes necessary to override  
 compiler options that have been fixed with the IGYCDOPT program.

These jobs are located in the COBOL sample data set IGY.V6R2M0.SIGYSAMP.

## Modifying compiler options and phases

If you plan to modify the values for compiler options and compiler phases, use the  
 IGYCOPT syntax format.

The IBM-supplied default values are shown both on the planning worksheets and  
 immediately following each syntax diagram. The syntax diagrams also show the  
 default as explained in "How to read the syntax diagrams" on page ix.

Compiler options and phases, and their defaults, are described in the following  
 information. Review these options and phases and their default values to  
 determine the values that are most suitable for your applications.

### IGYCOPT format

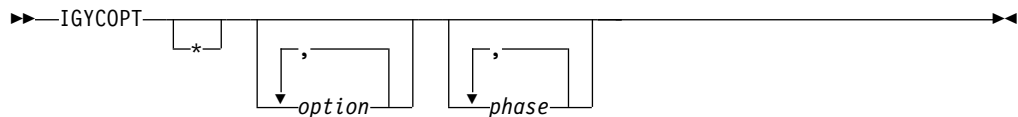


Figure 1. Syntax format for IGYCOPT compiler options and phases macro

### IGYCDOPT worksheet for compiler options

The IGYCDOPT worksheet will help you plan and code the compiler options  
 portion of the IGYCDOPT program.

To complete the worksheet, fill in the "Enter \* for fixed" and the "Enter selection"  
 columns.

The IGYCDOPT worksheet also includes a section for compiler phases. That  
 section of the worksheet can be found in "IGYCDOPT worksheet for compiler  
 phases" on page 11, after the description of compiler phases.

#### Note:

- Coding the asterisk [ \* ] when you modify a compiler option default value indicates that the option is to be fixed and cannot be overridden by an application programmer.
- The ALLOWCBL, DBCSXREF, LVLINFO, and NUMCLS options cannot be overridden at compile time. Therefore, the "Enter \* for fixed" worksheet entries for these options are blank.
- The IBM-supplied default value for ADEXIT, INEXIT, LVLINFO, LIBEXIT, MSGEXIT, and PRTEXTIT is null. Therefore, the "IBM-supplied default" entries for these options are blank.

- The DUMP compiler option cannot be set through the IGYCDOPT program. Unless changed at compile time, DUMP is always set to NODUMP.
- The OPTFILE compiler option cannot be set through the IGYCDOPT program.

Table 1. IGYCDOPT worksheet for options

Compiler option	Enter * for fixed	Enter selection	IBM-supplied default	Syntax description
ADATA=	_____	_____	<u>NO</u>	"ADATA" on page 17
ADEXIT=	_____	_____		"ADEXIT" on page 17
ADV=	_____	_____	<u>YES</u>	"ADV" on page 18
AFP=	_____	_____	<u>VOLATILE</u>	"AFP" on page 18
ALOWCBL=	_____	_____	<u>YES</u>	"ALOWCBL" on page 19
ARCH=	_____	_____	<u>7</u>	"ARCH" on page 19
ARITH=	_____	_____	<u>COMPAT</u>	"ARITH" on page 21
AWO=	_____	_____	<u>NO</u>	"AWO" on page 21
BLOCK0=	_____	_____	<u>NO</u>	"BLOCK0" on page 22
BUF=	_____	_____	<u>4K</u>	"BUF" on page 22
CICS=	_____	_____	<u>NO</u>	"CICS" on page 23
CODEPAGE=	_____	_____	<u>1140</u>	"CODEPAGE" on page 23
COMPILE=	_____	_____	<u>NOC(S)</u>	"COMPILE" on page 24
COPYRIGHT=	_____	_____	<u>NO</u>	"COPYRIGHT" on page 24
CURRENCY=	_____	_____	<u>NO</u>	"CURRENCY" on page 25
DATA=	_____	_____	<u>31</u>	"DATA" on page 26
DBCS=	_____	_____	<u>Yes</u>	"DBCS" on page 27
DBCSXREF	_____	_____	<u>NO</u>	"DBCSXREF" on page 27
DECK=	_____	_____	<u>NO</u>	"DECK" on page 28
DIAGTRUNC=	_____	_____	<u>NO</u>	"DIAGTRUNC" on page 29
DISPSIGN=	_____	_____	<u>COMPAT</u>	"DISPSIGN" on page 29
DLL=	_____	_____	<u>NO</u>	"DLL" on page 30
DYNAM=	_____	_____	<u>NO</u>	"DYNAM" on page 31
EXPORTALL=	_____	_____	<u>NO</u>	"EXPORTALL" on page 31
FASTSRT=	_____	_____	<u>NO</u>	"FASTSRT" on page 32
FLAG=	_____	_____	<u>(I,I)</u>	"FLAG" on page 32
FLAGSTD=	_____	_____	<u>NO</u>	"FLAGSTD" on page 33
HGPR=	_____	_____	<u>PRESERVE</u>	"HGPR" on page 35
INEXIT=	_____	_____		"INEXIT" on page 36
INITCHECK=	_____	_____	<u>NO</u>	"INITCHECK" on page 36
INTDATE=	_____	_____	<u>ANSI</u>	"INTDATE" on page 37
LANGUAGE=	_____	_____	<u>EN</u>	"LANGUAGE" on page 38
LIBEXIT=	_____	_____		"LIBEXIT" on page 38
LINECNT=	_____	_____	<u>60</u>	"LINECNT" on page 39
LIST=	_____	_____	<u>NO</u>	"LIST" on page 39
LITCHAR=	_____	_____	<u>QUOTE</u>	"LITCHAR" on page 40
LVLINFO=	_____	_____		"LVLINFO" on page 40
MAP=	_____	_____	<u>NO</u>	"MAP" on page 41
MAXPCF=	_____	_____	<u>60000</u>	"MAXPCF" on page 41
MDECK=	_____	_____	<u>NO</u>	"MDECK" on page 42
MSGEXIT=	_____	_____		"MSGEXIT" on page 43
NAME=	_____	_____	<u>NO</u>	"NAME" on page 43
NSYMBOL=	_____	_____	<u>NATIONAL</u>	"NSYMBOL" on page 44
NUM=	_____	_____	<u>NO</u>	"NUM" on page 44
NUMCHECK=	_____	_____	<u>(NO)</u>	"NUMCHECK" on page 45
NUMCLS=	_____	_____	<u>PRIM</u>	"NUMCLS" on page 47
NUMPROC=	_____	_____	<u>NOFPD</u>	"NUMPROC" on page 47

Table 1. IGYCDOPT worksheet for options (continued)

Compiler option	Enter * for fixed	Enter selection	IBM-supplied default	Syntax description
OBJECT=	_____	_____	<u>YES</u>	"OBJECT" on page 48
OFFSET=	_____	_____	<u>NO</u>	"OFFSET" on page 49
OPTIMIZE=	_____	_____	<u>0</u>	"OPTIMIZE" on page 49
OUTDD=	_____	_____	<u>SYSOUT</u>	"OUTDD" on page 50
PGMNAME=	_____	_____	<u>COMPAT</u>	"PGMNAME" on page 50
PRTEXT=	_____	_____	_____	"PRTEXT" on page 51
I QUALIFY=	_____	_____	<u>COMPAT</u>	"QUALIFY" on page 51
RENT=	_____	_____	<u>YES</u>	"RENT" on page 52
RMODE=	_____	_____	<u>AUTO</u>	"RMODE" on page 53
I RULES=	_____	_____	<u>NO</u>	"RULES" on page 54
SEQ=	_____	_____	<u>YES</u>	"SEQ" on page 55
I SERVICE=	_____	_____	<u>NO</u>	"SERVICE" on page 55
SOURCE=	_____	_____	<u>YES</u>	"SOURCE" on page 56
SPACE=	_____	_____	<u>1</u>	"SPACE" on page 56
SQL=	_____	_____	<u>NO</u>	"SQL" on page 56
SQLCCSID=	_____	_____	<u>YES</u>	"SQLCCSID" on page 57
SQLIMS=	_____	_____	<u>NO</u>	"SQLIMS" on page 58
SSRANGE=	_____	_____	<u>NO</u>	"SSRANGE" on page 58
STGOPT=	_____	_____	<u>NO</u>	"STGOPT" on page 60
TERM=	_____	_____	<u>NO</u>	"TERM" on page 60
TEST=	_____	_____	<u>(NO, NODWARF)</u>	"TEST" on page 61
THREAD=	_____	_____	<u>NO</u>	"THREAD" on page 62
TRUNC=	_____	_____	<u>STD</u>	"TRUNC" on page 63
VBREF=	_____	_____	<u>NO</u>	"VBREF" on page 64
I VLR=	_____	_____	<u>COMPAT</u>	"VLR" on page 65
I VSAMOPENFS=	_____	_____	<u>COMPAT</u>	"VSAMOPENFS" on page 66
WORD=	_____	_____	<u>NO</u>	"WORD" on page 66
I XMLPARSE=	_____	_____	<u>XMLSS</u>	"XMLPARSE" on page 67
XREFOPT=	_____	_____	<u>FULL</u>	"XREFOPT" on page 68
I ZONECHECK=	_____	_____	<u>NO</u>	"ZONECHECK" on page 69
I ZONEDATA	_____	_____	<u>PFD</u>	"ZONEDATA" on page 69
ZWB=	_____	_____	<u>YES</u>	"ZWB" on page 70

## Planning to place compiler phases in shared storage

The following sections define shared storage, describe the various compiler phases, and explain why you might want to place the compiler phases in shared storage.

You might want to make some program objects resident in a link-pack area in order to minimize the search for them when an Enterprise COBOL program is run or when the objects will be shared. You might also want to make some or all of the compiler phases resident.

The term *shared storage* is used generally to describe the link-pack area (LPA), the extended link-pack area (ELPA), or modified link-pack area (MLPA). Except where specifically otherwise stated, all three terms are implied when the term *link-pack area* is used in this information.

## Why place the compiler phases in shared storage?

This section defines shared storage, and explains which compiler phases can be placed in shared storage and why you might want to place them there.

Shared storage is an area of storage that is the same for each virtual address space. Information stored there does not have to be loaded in the user region because it is the same space for all users. By sharing the information, more space is made available for the compiler work area.

All compiler modules, except the run dump modules (IGYCRDPR and IGYCRDSC) and the reserved word utility (IGY8RWTU), are eligible for placement in the shared storage of z/OS machines. All compiler phases except IGYCRCTL and IGYCSIMD have RMODE ANY and AMODE 31. Since IGYCRCTL and IGYCSIMD have RMODE 24, they can be placed in the LPA or MLPA, but not in the ELPA.

**Note:** The Enterprise COBOL, COBOL for MVS™ & VM, COBOL for OS/390® & VM, and VS COBOL II compilers have modules with the same names; thus, only one set of phases can be placed in the LPA for any given initialization of the operating system. Exception is that Enterprise COBOL V5 modules cannot be placed in shared storage by using LPALST. Because the compiler modules are in a PDSE data set, the module names cannot be in the LPALST concatenation during IPL. You can dynamically add them to LPA at the end of IPL.

The IGYCDOPT program indicates where each compiler phase (except for phases IGYCBE, IGYECWI, IGYMSGK and IGYMSGGE) is loaded, either inside (IN) or outside (OUT) the user region. By placing compiler phases in the MLPA, the compiler has more storage available for the user's program.

If you indicate that a phase will not reside in the user region, you must ensure that you actually place the phase in shared storage. This information is used by the compiler to determine how much storage to leave for the system to load compiler phases in the user region.

For a description of how to place a phase in shared storage, see the *Initialization and Tuning* references listed in "Related publications" on page 87.

It is recommended that the following phases be placed in a shared storage area:

### **IGYCRCTL**

Because it is resident in the user region throughout compilation.

### **IGYCSIMD**

Because it is resident in the user region throughout compilation.

### **IGYCBE**

Because it is the largest compiler phase.

### **IGYECWI**

Because it is the second largest compiler phase.

### **IGYCPGEN**

Because it is the third largest compiler phase.

### **IGYCSCAN**

Because it is the fourth largest compiler phase.

You can select any or all compiler phases to be placed in shared storage based on frequency of concurrent use and phase size. If your facility seldom uses the

compiler, there might be no advantage to installing any phases in shared storage. However, if there are frequent compilations and sufficient MLPA storage is available, making the entire compiler resident might be advantageous. If sufficient shared storage is not available, priority must be given to IGYCRCTL and IGYCSIMD, the two phases that are always resident in the user region during compilation; and also to IGYCBE, the largest compiler phase.

Another advantage of placing compiler phases in shared storage is that, at compile time, the initialization logic allocates in the user region a storage block of sufficient size to contain the largest phase not resident in shared storage. Minimizing the space allocation for any given user region size means more space for the compilation process (which allows larger programs to be compiled within a given user region) and possibly a more efficient compilation. The IGYCBE and IGYECWI compiler phases are approximately 10 MB larger than the next largest compiler phase. Shared storage can make a significant difference if you are compiling using the minimum region size.

## Compiler phases and their defaults

This section explains how to indicate where a compiler phase is to be loaded, describes the phases, and includes a worksheet to help you plan changes to the phases.

To indicate where each compiler phase is loaded in relation to the user region, specify either IN or OUT.

For more information about why you might or might not want to change these defaults, see “Why place the compiler phases in shared storage?” on page 7.

**IN** Indicates that the compiler phase is loaded into the user region from a library available at compile time.

Even though IN is specified for a compiler phase, the phase still can be placed into the shared system area. However, the compiler control phase ensures that the main storage area reserved for compiler phases is large enough to contain the largest phase for which IN is specified. This option will cause some storage to be unused.

**OUT** Indicates that the compiler phase is not loaded into the user region from the library, and therefore must reside in a shared system area, such as the MLPA.

### IGYCDGEN

The data generation phase. It generates the appropriate WCode instructions to describe data items in the program.

#### Syntax

►► DGEN= 

IN
OUT

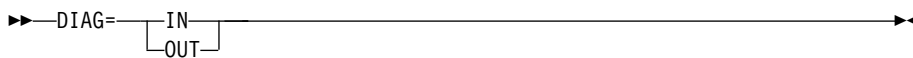
 ◀◀

### IGYCDIAG

The diagnostic phase. It processes E-form text and generates compiler diagnostics for source program errors. It includes IGYCDIAG plus the following message modules: IGYCxx\$D, IGYCxx\$1, IGYCxx\$2, IGYCxx\$3, IGYCxx\$4, IGYCxx\$5, and IGYCxx\$8, where xx is EN, UE, or JA.



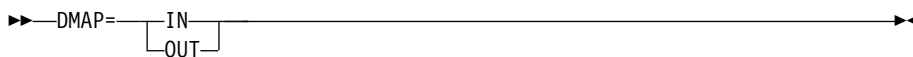
## Syntax



## IGYCDMAP

The DMAP phase. It prepares text for output requested by the MAP option.

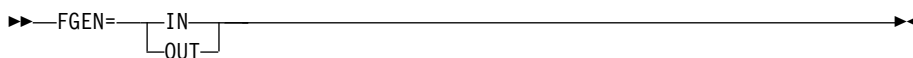
## Syntax



## IGYCFGEN

The file generation phase. It generates the control blocks for the FDs and SDs defined in the program.

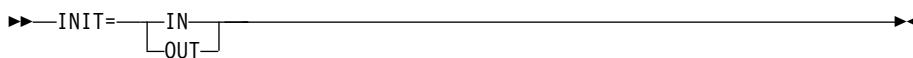
## Syntax



## IGYCINIT

The initialization phase. It does housekeeping to prepare for running of the processing phases.

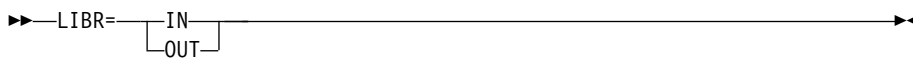
## Syntax



## IGYCLIBR

The COPY phase. It processes library source text and does a syntax check of the COPY, BASIS, and REPLACE statements.

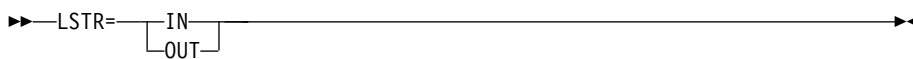
## Syntax



## IGYCLSTR

The source listing phase. It prints the source listing with embedded cross-reference and diagnostic information.

## Syntax



## IGYCMMSGT

Represents the header text table and diagnostic message level tables. It includes the following modules: IGYCxx\$R, IGYCLVL0, IGYCLVL1, IGYCLVL2, IGYCLVL3, and IGYCLVL8, where xx is EN, UE, or JA.

### Syntax

►►MSGT=—IN—  
          └─OUT─┘

### IGYCOSCN

The option scanning phase. It determines the default options, processes the EXEC PARM options, and processes the PROCESS (CBL) statements.

### Syntax

►►OSCN=—IN—  
          └─OUT─┘

### IGYCPGEN

The procedure generation phase. It supplies code for all procedure source verbs.

### Syntax

►►PGEN=—IN—  
          └─OUT─┘

### IGYCRCTL

The resident control phase. It establishes the size of compiler common and working storage, and performs initialization of program common storage.

### Syntax

►►RCTL=—IN—  
          └─OUT─┘

### IGYCRWT

The normal reserved word table.

### Syntax

►►RWT=—IN—  
          └─OUT─┘

### IGYCSCAN

The scanning phase. It does syntax and semantic analysis of the source program and translates the source to intermediate text.

### Syntax

►►SCAN=—IN—  
          └─OUT─┘

### IGYCSIMD

The system interface phase for the Enterprise COBOL compiler. This phase is called by all other compiler phases to perform system-dependent functions.

### Syntax

►►SIMD= 

IN
OUT

►►

### IGYCXREF

The XREF phase. It sorts user-names and procedure-names in EBCDIC collating sequence.

### Syntax

►►XREF= 

IN
OUT

►►

## IGYCDOPT worksheet for compiler phases

The IGYCDOPT worksheet helps you plan and code the phases portion of the IGYCDOPT program.

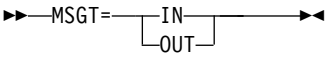
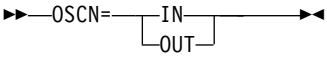
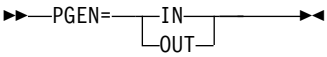
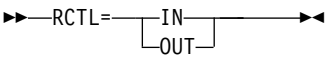
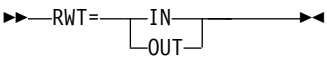
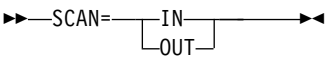
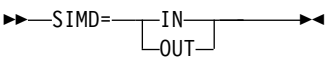
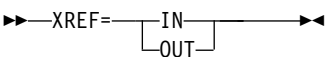
In the following worksheet, circle the value that you plan to assign to each phase. For more information on the values that can be assigned to each phase, see “Compiler phases and their defaults” on page 8.

**Note:** All phase defaults are initially set to IN.

Table 2. IGYCDOPT program worksheet for compiler phases

Phase	Circle selection	Syntax description		
DGEN=	<u>IN</u> / OUT	►►—DGEN=— <table><tr><td>IN</td></tr><tr><td>OUT</td></tr></table> —►►	IN	OUT
IN				
OUT				
DIAG=	<u>IN</u> / OUT	►►—DIAG=— <table><tr><td>IN</td></tr><tr><td>OUT</td></tr></table> —►►	IN	OUT
IN				
OUT				
DMAP=	<u>IN</u> / OUT	►►—DMAP=— <table><tr><td>IN</td></tr><tr><td>OUT</td></tr></table> —►►	IN	OUT
IN				
OUT				
FGEN=	<u>IN</u> / OUT	►►—FGEN=— <table><tr><td>IN</td></tr><tr><td>OUT</td></tr></table> —►►	IN	OUT
IN				
OUT				
INIT=	<u>IN</u> / OUT	►►—INIT=— <table><tr><td>IN</td></tr><tr><td>OUT</td></tr></table> —►►	IN	OUT
IN				
OUT				
LIBR=	<u>IN</u> / OUT	►►—LIBR=— <table><tr><td>IN</td></tr><tr><td>OUT</td></tr></table> —►►	IN	OUT
IN				
OUT				
LSTR=	<u>IN</u> / OUT	►►—LSTR=— <table><tr><td>IN</td></tr><tr><td>OUT</td></tr></table> —►►	IN	OUT
IN				
OUT				

Table 2. IGYCDOPT program worksheet for compiler phases (continued)

Phase	Circle selection	Syntax description
MSGT=	<u>IN</u> / OUT	
OSCN=	<u>IN</u> / OUT	
PGEN=	<u>IN</u> / OUT	
RCTL=	<u>IN</u> / OUT	
RWT=	<u>IN</u> / OUT	
SCAN=	<u>IN</u> / OUT	
SIMD=	<u>IN</u> / OUT	
XREF=	<u>IN</u> / OUT	

## Planning to create an additional reserved word table

The following sections describe why you might want to create additional reserved word tables, explain how you can restrict the use of nested programs by modifying a reserved word table, and list the reserved word tables supplied with Enterprise COBOL.

You can create additional reserved word tables after installation. During compilation, the value of the WORD compiler option determines which reserved word table is used.

## Why create additional reserved word tables?

This section describes the benefits of creating additional reserved word tables.

You can create additional reserved word tables to:

- Translate the reserved words into another language, such as French or German.
- Prevent application programmers from using a particular Enterprise COBOL instruction, such as GO TO.

- Control the usage of nested programs.
- Flag words that are not supported under CICS, such as READ and WRITE.

## Controlling use of nested programs

To restrict the use of nested programs without restricting any other COBOL language features, modify the reserved word table.

Do this by using the INFO and RSTR control statements. For instructions on how to make these modifications, see “Creating or modifying a reserved word table” on page 77.

## Reserved word tables supplied with Enterprise COBOL

Enterprise COBOL provides reserved word tables on the installation medium.

The reserved word tables are:

- Default reserved word table
- CICS reserved word table

### Default reserved word table (IGYCRWT)

About the default reserved word table provided for your entire facility, see the *Enterprise COBOL Language Reference*.

### CICS reserved word table (IGYCCICS)

Enterprise COBOL provides an alternate reserved word table for CICS application programs so that COBOL words that are not supported under CICS are flagged by the compiler.

The CICS reserved word table is the same as the default reserved word table except that the following COBOL words are marked as restricted (RSTR):

- CLOSE
- DELETE
- FD
- FILE
- FILE-CONTROL
- INPUT-OUTPUT
- I-O-CONTROL
- MERGE
- OPEN
- READ
- RERUN
- REWRITE
- SD
- SORT
- START
- WRITE

**SORT users:** Enterprise COBOL supports an interface for the SORT statement under CICS. If you intend to use the SORT statement under CICS, you must modify the CICS reserved word table before using it. The words that are underlined above must be removed from the list of words marked as restricted, because they are required for the SORT function.

### Using the table:

To use the CICS reserved word table, you must specify the WORD(CICS) compiler option.

To have the CICS reserved word table used as the default, you must set the default value of the WORD compiler option to WORD=CICS.

### Location of the table:

The data used to create the CICS reserved word table is in member IGY8CICS in IGY.V6R2M0.SIGYSAMP.

**Note:** The high-level qualifier IGY.V6R2M0 might have been changed when Enterprise COBOL was installed.

---

## Using product registration to enable or disable Enterprise COBOL

The default behavior for Enterprise COBOL V5 is to run on every z/OS system, but you can use the IFAPRDxx member of SYSx.PARMLIB to disable COBOL V5 from running on selected z/OS systems.

If you want to disable COBOL V5, add the following code to the active IFAPRDxx member:

```
PRODUCT OWNER('IBM CORP')
        NAME('ENTERPRISE COBOL')
        ID(5655-W32)
        VERSION(05) RELEASE(*) MOD(*)
        FEATURENAME(*)
        STATE(DISABLED)
```

In this case, the compiler stops with RC=16 and a write-to-operator message.

If you want to explicitly enable COBOL V5, add the following code to the active IFAPRDxx member:

```
PRODUCT OWNER('IBM CORP')
        NAME('ENTERPRISE COBOL')
        ID(5655-W32)
        VERSION(05) RELEASE(*) MOD(*)
        FEATURENAME(*)
        STATE(ENABLED)
```

However, because COBOL V5 is enabled by default, you don't have to explicitly enable COBOL V5 by adding the previous statements to the active IFAPRDxx member.

---

## Chapter 2. Enterprise COBOL compiler options

This information describes the compiler options whose default values can be changed.

The notes that accompany some of the descriptions provide additional information about these options, such as how they interact with other options during compilation.

This information might help you to make decisions about which default values are appropriate for your installation.

For more information about the compiler options, see *Compiler options* in the *Enterprise COBOL Programming Guide*.

### Important:

Confer with the application programmers at your site while you plan the customization of Enterprise COBOL. Doing so will ensure that the modifications you make serve their needs and support the applications that are being developed.

---

## Specifying COBOL compiler options

When you specify compiler options in the IGYCOPT macro, both the option name and its value must be specified in uppercase.

If you don't specify the option name in uppercase, both the option name and its value are ignored and the default value is used instead. No error message is issued. If only the option value is not in uppercase, an error message will be issued indicating that an invalid option value has been specified.

---

## Conflicting compiler options

If you specify certain compiler option values, a conflict with other compiler options might result. This topic describes possible conflicts between compiler options.

Table 3. Conflicting compiler options

Compiler option	Conflicts with:
CICS=YES	RENT=NO DYNAM=YES
DBCS=NO	NSYMBOL=NATIONAL
DBCSXREF=(other than NO)	XREFOPT=NO
DLL=NO	EXPORTALL=YES
DLL=YES	DYNAM=YES RENT=NO

Table 3. Conflicting compiler options (continued)

Compiler option	Conflicts with:
DYNAM=YES	CICS=YES DLL=YES EXPORTALL=YES
EXPORTALL=YES	DLL=NO DYNAM=YES RENT=NO
FLAGSTD=(other than NO)	WORD=xxx
LIST=YES	OFFSET=YES
NSYMBOL=NATIONAL	DBCS=NO
OBJECT=NO	TEST=(other than NO)
OFFSET=YES	LIST=YES
OPTIMIZE=0	INITCHECK=YES
RENT=NO	CICS=YES DLL=YES EXPORTALL=YES THREAD=YES
THREAD=YES	RENT=NO
WORD=xxx	FLAGSTD=(other than NO)
XREFOPT=NO	DBCSXREF=(other than NO)

## Compiler options for standards conformance

Several compiler options are required to conform with the 85 COBOL Standard.

For details, see *Option settings for 85 COBOL Standard conformance* in the *Enterprise COBOL Programming Guide*.

## Compiler options syntax and descriptions

The syntax diagrams in the following topics describe each modifiable compiler option. The text after each diagram describes the effect of selecting a specific parameter.

### Note:

- The DUMP option is not in this list. Unless you change DUMP at compile time, it is always set to NODUMP. This option is not for general use; it is used only at the request of an IBM representative.



- The OPTFILE option is not in this list. It can be specified only as a compiler invocation PARM option, or in a PROCESS or CBL statement in the COBOL source program.
- Coding the asterisk (\*) when you modify a compiler option default value indicates that the option is to be fixed and cannot be overridden by an application programmer.

---

## ADATA

ADATA affects whether an Associated Data file is produced during compilation.

### Syntax



### Default

ADATA=NO

### YES

Produces the Associated Data file with the appropriate records.

**NO** Does not produce the Associated Data file.

### Note:

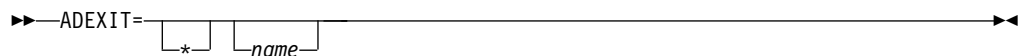
- The ADATA option can be specified only at invocation through the option list, on the PARM field of JCL, as a command option, or as an installation default.
- Selection of the Japanese language option might result in DBCS characters written records in the Associated Data file.
- Specification of NOCOMPILE(W|E|S) might stop compilation prematurely, resulting in a loss of specific Associated Data records.
- If you use the INEXIT option, the compilation source module is not identified in the SYSADATA (Associated Data file) information.

---

## ADEXIT

ADEXIT designates a module to be called for each record that is written to the SYSADATA file.

### Syntax



### Default

No exit is specified. Equivalent to specifying the NOADEXIT suboption of the EXIT compiler option. If ADEXIT=\* is coded without the *name* parameter, NOADEXIT cannot be overridden.

### *name*

Identifies a module to be used with the EXIT compiler option. If the suboption for this user exit is specified, the compiler loads the named module and calls it for each record that is written to the SYSADATA file.

For more information about the EXIT compiler option, see *EXIT compiler option* in the *Enterprise COBOL Programming Guide*.

---

## ADV

ADV affects WRITE ... ADVANCING statements, determining whether one byte is added to the record length for the printer control character.

### Syntax



### Default

ADV=YES

### YES

Adds one byte to the record length for the printer control character. This option might be useful to programmers who use WRITE ... ADVANCING in their source files. The first character of the record does *not* have to be explicitly reserved by the programmer.

**NO** Does not adjust the record length for WRITE ... ADVANCING. The compiler uses the first character of the specified record area to place the printer control character. The application programmer must ensure that the record description allows for this additional byte.

### Note:

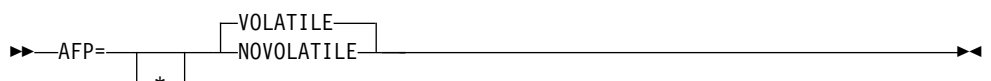
- With ADV=YES, the record length on the physical device is one byte larger than the record description length in the source program.
- If the record length for the output file is not defined in the source code, COBOL ensures that the DCB parameters are appropriately set.
- If ADV=YES is specified, and the record length for the output file has been defined in the source code, the programmer *must* specify the record description length as one byte larger than the source program record description. The programmer *must* also specify the block size in correct multiples of the larger record size.
- If the LINAGE clause is specified in a file description (FD), the compiler treats that file as if ADV=YES has been specified.

---

## AFP

The AFP option controls the compiler usage of the Additional Floating Point (AFP) registers that are provided by z/Architecture<sup>®</sup> processors.

### Syntax



Default is: AFP=VOLATILE

The Enterprise COBOL compiler generates code that uses the full complement of 16 floating point registers (FPR) provided by a z/Architecture processor. These FPRs are as follows:

- Original FPRs, which are numbered 0, 2, 4, and 6
- AFP registers, which are numbered 1, 3, 5, 7, and 8-15

**Note:** If your code runs on a version of CICS Transaction Server that is earlier than V4.1, you must specify AFP=VOLATILE.

#### **AFP=VOLATILE**

If you specify AFP=VOLATILE, the AFP registers 8-15 are considered volatile, which means that they might be changed by a called subprogram. Therefore, the COBOL compiler generates extra code to protect the values in these registers.

#### **AFP=NOVOLATILE**

If you specify AFP=NOVOLATILE, the AFP registers 8-15 are considered nonvolatile, which means that they are known to be unchanged or preserved by every called subprogram. Therefore, the compiler can generate more efficient code sequences for programs with floating point operations. It is the normal z/OS architecture convention.

---

## **ALLOWCBL**

ALLOWCBL affects whether PROCESS (or CBL) statements can be used in COBOL programs.

#### **Syntax**

➡➡ ALLOWCBL= 

YES
NO

 ➡➡

#### **Default**

ALLOWCBL=YES

#### **YES**

Allows the use of the PROCESS (or CBL) statements in COBOL programs.

**NO** Diagnoses the use of PROCESS (or CBL) statements in a program as an error.

#### **Note:**

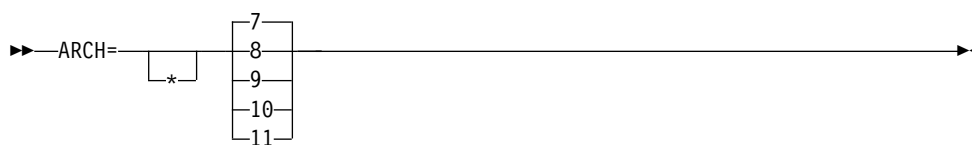
- ALLOWCBL cannot be overridden at compile time because it cannot be included in the PROCESS (or CBL) statement.
- The PROCESS (or CBL) statement specifies compiler-option parameters within source programs. If your installation requirements do not allow compiler options to be specified in a source program, specify ALLOWCBL=NO.

---

## **ARCH**

The ARCH option specifies the machine architecture for which the executable program instructions are to be generated.

## Syntax



If you specify a higher ARCH level, the compiler generates code that uses newer and faster instructions. Your application might abend if it runs on a processor with an architecture level lower than what you specify with the ARCH option. Use the ARCH level that matches the lowest machine architecture where your application runs.

Current supported architecture levels and groups of models are as follows:

### Default

ARCH=7

- 7 Produces code that uses instructions available on the 2096-xxx (IBM System z9<sup>®</sup> BC) and 2094-xxx (IBM System z9 EC) models in z/Architecture mode.

Specifically, these ARCH(7) machines and their follow-ons add instructions supported by the following facilities:

- Extended-immediate facility
- Decimal floating point facility. These instructions might be generated if decimal data is used in numeric operations.

- 8 Produces code that uses instructions available on the 2097-xxx (IBM System z10<sup>®</sup> EC) models in z/Architecture mode.

Specifically, these ARCH(8) machines and their follow-ons add instructions supported by the general instruction extensions facility.

- 9 Produces code that uses instructions available on 2817-xxx (IBM zEnterprise<sup>®</sup> 196) and 2818-xxx (IBM zEnterprise 114) models in z/Architecture mode.

Specifically, these ARCH=9 machines and their follow-ons add instructions supported by the following facilities:

- High-word facility
- Interlocked access facility
- Load/store-on-condition facility
- Distinct-operands facility
- Population-count facility

- 10 Produces code that uses instructions available on the 2827-xxxx (IBM zEnterprise EC12) models in z/Architecture mode.

Specifically, these ARCH(10) machines and their follow-ons add instructions supported by the following facilities:

- Execution-hint facility
- Load-and-trap facility
- Miscellaneous-instructions-extension facility
- Transactional-execution facility

- Enhanced decimal floating point facility that enables more efficient conversions between zoned decimal data items and decimal floating point data items. Instead of converting zoned decimal data items to packed decimal data items to perform arithmetic, the compiler converts zoned decimal data items directly to decimal floating point data items, and then back again to zoned decimal data items after the computations are complete.

**11** Produces code that uses instructions available on the 2964-xxx (IBM z13<sup>®</sup>) models in z/Architecture mode.

Specifically, these ARCH(11) machines and their follow-ons add instructions with support of the following facilities:

- Enhanced decimal floating point facility that enables more efficient conversions between packed-decimal data items and decimal floating point intermediate result data items
- Exploitation of the new vector extension facility (SIMD) instructions for some INSPECT REPLACING and INSPECT TALLYING statements

**Note:** A higher ARCH level includes the facilities of the lower ARCH level. For example, ARCH=11 includes all the facilities of the lower ARCH levels.

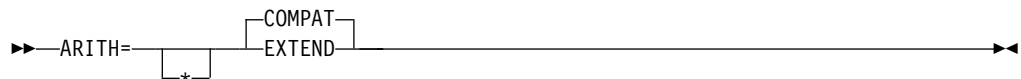
For more information about these facilities, see *z/Architecture Principles of Operation*.

---

## ARITH

ARITH affects the maximum number of digits that can be coded for integers, and the number of digits used in fixed-point intermediate results.

### Syntax



### Default

ARITH=COMPAT

### COMPAT

Specifies 18 digits as the maximum precision for decimal data.

### EXTEND

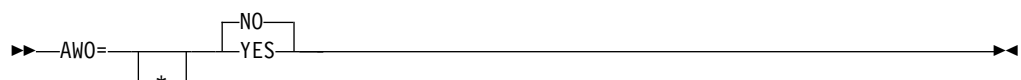
Specifies 31 digits as the maximum precision for decimal data.

---

## AWO

AWO affects whether the APPLY-WRITE-ONLY clause is activated for physical-sequential files that have variable blocked format.

### Syntax



### Default

AWO=NO

## YES

Activates the APPLY-WRITE-ONLY clause for any file within the program that is physical sequential with variable block format regardless of whether or not the APPLY-WRITE-ONLY clause is specified in the program.

**Performance consideration:** Using AWO=YES generally results in fewer calls to Data Management Services for runtime files when handling input and output.

**NO** Does not activate the APPLY-WRITE-ONLY clause for any file within the program that is physical sequential with variable block format unless the APPLY-WRITE-ONLY clause is specified in the program.

---

## BLOCK0

BLOCK0 affects whether the default blocking specification for QSAM files is changed from unblocked to blocked.

### Syntax



### Default

BLOCK0=NO

## YES

Changes the default blocking specification for QSAM files that specify neither BLOCK CONTAINS nor RECORDING MODE U in the file description entry. BLOCK0=YES activates the BLOCK CONTAINS 0 clause for such files, causing them to have a system-determined block size at run time.

**Performance consideration:** Using BLOCK0=YES could result in enhanced processing speed and minimized storage requirements for QSAM output files. But see the recommendation below.

**NO** Does not activate the BLOCK CONTAINS 0 clause by default for any file.

**Recommendation:** Adding a BLOCK CONTAINS 0 clause to file descriptions in existing programs could result in a change of behavior in those programs, including some undesirable effects for files opened as INPUT. For this reason, it is recommended that BLOCK0=YES not be set as an installation default.

For further details, see *BLOCK0* in the *Enterprise COBOL Programming Guide*.

---

## BUF

BUF specifies the amount of dynamic storage to be used during compilation.

### Syntax



### Default

BUF=4K

### ***integer***

Specifies the amount of dynamic storage, in bytes, to be allocated to each compiler work file buffer. The minimum value is 256 bytes.

**Performance consideration:** Using a large buffer size usually improves the performance of the compiler.

### ***integerK***

Specifies the amount of dynamic storage to be allocated to buffers in increments of 1K (1024) bytes.

### **Note:**

- BUF cannot exceed the track capacity for the device used, nor can it exceed the maximum allowed by data management services.

---

## **CICS**

CICS affects whether a COBOL source program that contains CICS statements is to be processed by the integrated CICS translator.

### **Syntax**



### **Default**

CICS=NO

### **YES**

If a COBOL source program contains CICS statements and has not been preprocessed by the CICS translator, the YES option must be specified.

**NO** When the NO option is specified, any CICS statements that are found in the source program are diagnosed and discarded.

### **Note:**

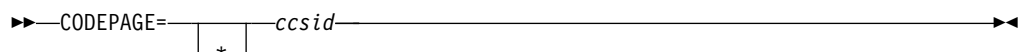
- The CICS compiler option can contain CICS suboptions. The CICS suboptions delimiters can be quotation marks or apostrophes. CICS suboptions cannot be specified as a COBOL installation default.
- You can specify the CICS compiler option in any of the compiler option sources: installation defaults, compiler invocation, or PROCESS or CBL statements.

---

## **CODEPAGE**

CODEPAGE affects the coded character set identifier (CCSID) for an EBCDIC code page for processing compile-time and runtime COBOL operations that are sensitive to character encoding.

### **Syntax**



### **Default**

CODEPAGE=1140

**ccsid**

Specifies a valid coded character set identifier (CCSID) integer that identifies an EBCDIC code page.

The default CCSID 1140 is the equivalent of CCSID 37 (EBCDIC Latin-1, USA), but additionally includes the euro symbol.

**Recommendation:** To avoid unnecessary conversions and associated performance overhead on systems that use both COBOL and DB2®, use the same CODEPAGE compiler option setting as in your DB2 subsystem parameters and application programming defaults (specify in DSNHDECP).

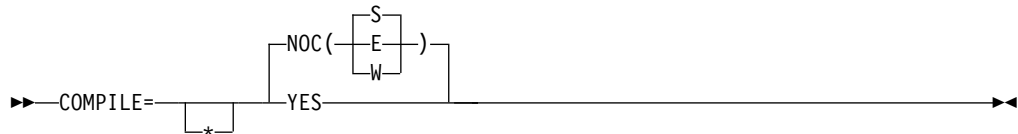
**Note:** If you specify the TEST option, you must set the CODEPAGE option to the CCSID that is used for the COBOL source program. In particular, programs that use Japanese characters in DBCS literals or DBCS user-defined words must be compiled with the CODEPAGE option set to a Japanese codepage CCSID.

For further details, see *CODEPAGE* in the *Enterprise COBOL Programming Guide*.

---

## COMPILE

COMPILE determines whether compilation continues if diagnostic messages of a specified severity occur.

**Syntax****Default**

COMPILE=NOC(S)

**NOC**

Indicates that you want only a syntax check.

**NOC(W)****NOC(E)****NOC(S)**

Specifies an error message level: W is warning; E is error; S is severe. When an error of the level specified or of a more severe level occurs, compilation stops, and only syntax checking is done for the balance of the compilation.

**YES**

Indicates that you want full compilation, including diagnostics and object code.

Specifying NOCOMPILE might affect the Associated Data file by stopping compilation prematurely, resulting in loss of specific messages.

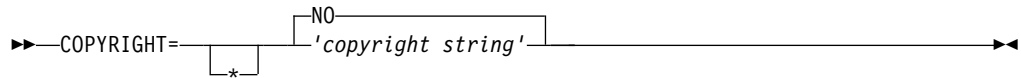
---

## COPYRIGHT

Use COPYRIGHT to place a string in the object module if the object module is generated. If the object is linked into a program object, the string is loaded into memory with this program object.



## Syntax



## Default

COPYRIGHT=NO

The *copyright string* is limited to 64 characters in length.

---

# CURRENCY

CURRENCY determines whether an alternate currency symbol will be used; the default is dollar sign (\$).

## Syntax



## Default

CURRENCY=NO

## literal

Represents the default currency symbol that you want to use in your program.

The literal must be a nonnumeric literal representing a 1-byte EBCDIC character that must not be any of the following items:

- Digits zero (0) through nine (9)
- Uppercase alphabetic characters: A B C D P R S V X Z
- Lowercase alphabetic characters a through z
- The space
- Special characters: \* + - / , . ; ( ) = "
- Uppercase alphabetic character G, if the COBOL program defines a DBCS item with the PICTURE symbol G. The PICTURE clause will not be valid for that DBCS item because the symbol G is considered to be a currency symbol in the PICTURE clause.
- Uppercase alphabetic character N, if the COBOL program defines a DBCS item with the PICTURE symbol N. The PICTURE clause will not be valid for that DBCS item because the symbol N is considered to be a currency symbol in the PICTURE clause.
- Uppercase alphabetic character E, if the COBOL program defines an external floating-point item. The PICTURE clause will not be valid for the external floating-point item because the symbol E is considered to be a currency symbol in the PICTURE clause.

The literal (including hex literal) syntax rules are as follows:

- The literal delimiters can be either quotation marks or apostrophes regardless of whether the APOST or QUOTE option is in effect.
- When an apostrophe (') is to be the currency sign, the embedded apostrophe must be doubled, that is, two apostrophes must be coded to represent one apostrophe within the literal. For example:

'''' or ''''

- The format for a hex literal specification is as follows:

X'H1H2' or X"H1H2"

where H1H2 is a valid hexadecimal value representing a 1-byte EBCDIC character conforming to the rules for the currency sign literal as described above. Alphabetic characters in the hex literal must be in uppercase.

**Note:** Hex values of X'20' or X'21' are not allowed.

- NO** Indicates that no alternate default currency sign is provided through the CURRENCY option, and the dollar sign will be used as the default currency sign for the program if the CURRENCY option is not specified at compile time.

The value NO provides the same results for the source program as omitting the CURRENCY SIGN clause in the COBOL source program.

**Note:**

- You can use the CURRENCY option as an alternative to the CURRENCY SIGN clause (which is specified in the COBOL source program) for selecting the currency symbol that you use in the PICTURE clause of your COBOL program.
- When both the CURRENCY option and the CURRENCY SIGN clause are used in a program, the symbol that is specified in the CURRENCY SIGN clause is the currency symbol in a PICTURE clause when that symbol is used, even if the CURRENCY option is fixed (\*).

---

## DATA

DATA affects whether storage for dynamic data areas and other dynamic runtime storage is obtained from above or below the 16 MB line.

### Syntax



### Default

DATA=31

- 24** Causes allocation of user data areas in virtual addresses below 16 MB in storage acquired by a GETMAIN with the LOC=BELOW option.

Specify DATA=24 for programs compiled with the RENT option that are passing data parameters to programs in 24-bit mode. This includes the following cases:

- A COBOL program is passing items in its WORKING-STORAGE to an AMODE 24 program.
- A COBOL program is passing, by reference, data items received from its caller to an AMODE 24 program. DATA=24 is required even when the data received is below the 16 MB line.

Otherwise, the data might not be addressable by the called program.

DATA does not affect the location of LOCAL-STORAGE data; the STACK runtime option controls that location instead, along with the AMODE of the program.

- 31 Causes allocation of user data areas, such as WORKING-STORAGE and FD record areas, from unrestricted storage or in space acquired by a GETMAIN with the LOC=ANY option. Specifying this option can result in storage being acquired in virtual addresses either above or below the 16 MB line. The operating system generally satisfies the request with space in virtual addresses above the 16 MB line, if it is available.

**Note:**

- If a program is compiled with the RENT option, the DATA option controls how space for WORKING-STORAGE and parameter lists is acquired.
- The DATA option has no effect on programs compiled with the NORENT option.

---

## DBCS

DBCS affects whether the compiler recognizes X'0E' and X'0F' in a nonnumeric literal and treats them as shift-out and shift-in control characters for delimiting DBCS data.

**Syntax**



**Default**

DBCS=YES

**YES**

Recognizes X'0E' and X'0F' in a nonnumeric literal and treats them as shift-out and shift-in control characters for delimiting DBCS data.

**NO** Does not recognize X'0E' and X'0F' as shift-out and shift-in control characters in a nonnumeric literal.

**Note:**

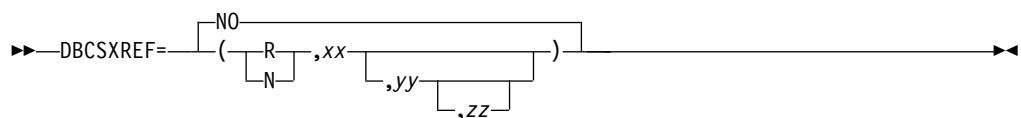
- The presence of DBCS data inside the nonnumeric literal might cause the compiler to disallow certain uses of that literal. For example, DBCS characters are not allowed as program names or DDNAMES.
- DBCS=NO conflicts with NSYMBOL(NATIONAL).

---

## DBCSXREF

DBCSXREF indicates that an ordering program is to be used for cross-referencing of DBCS names.

**Syntax**



**Default**

DBCSXREF=NO

- R** Specifies that the DBCS Ordering Support Program (DBCSOS) is loaded into the user region.
- N** Specifies that the DBCS Ordering Support Program (DBCSOS) is loaded into a shared system area such as the MLPA.
- xx** Names a program object of the relevant ordering program to produce DBCS cross-references. It must be eight characters in length.
- yy** Names an ordering type. It must be two characters in length. The default ordering type defined by the specified ordering program occurs if this parameter is omitted.
- zz** Names the encode table that the specified ordering type uses. It must be eight characters in length. The default encode table that is associated with the particular ordering type occurs if this parameter is omitted.
- NO** Specifies that no ordering program is used for cross-reference of DBCS names. If the XREF phase is specified, a cross-reference listing of DBCS names is provided based on their physical order in the program.

**Note:**

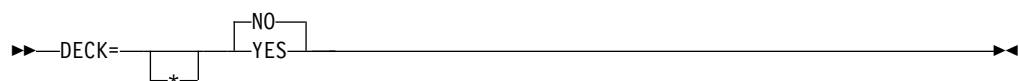
- The DBCS Ordering Support Program (DBCSOS) must be installed to specify anything other than DBCSXREF=NO.
- If R is specified, ensure that the user region is large enough to accommodate both the compiler and the ordering program.
- Specifying both XREFOPT=NO and DBCSXREF with an ordering program results in a nonzero return code while attempting to assemble the customization macro.
- The assembly process terminates when validation diagnoses:
  - A parameter length that is not valid
  - Characters other than 'R' and 'N'
  - Missing parameters after a comma
  - Missing *yy* when *zz* is specified

---

## DECK

DECK determines whether 80-column object-code records are produced in a file defined by the SYSPUNCH DD statement.

### Syntax



### Default

DECK=NO

### YES

Places the generated object code in a file defined by SYSPUNCH.

**NO** Sends no object code to SYSPUNCH.

---

## DIAGTRUNC

DIAGTRUNC affects whether the compiler issues a severity-4 (warning) diagnostic message for MOVE statements with numeric receivers when the receiving data item has fewer integer positions than the sending data item or literal.

### Syntax



### Default

DIAGTRUNC=NO

### YES

Causes the compiler to issue a severity-4 (warning) diagnostic message for MOVE statements with numeric receivers when the receiving data item has fewer integer positions than the sending data item or literal.

**NO** Does not cause the compiler to produce a severity-4 message.

### Note:

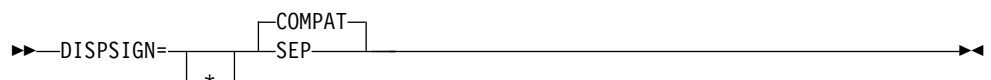
- The diagnostic message is also issued for moves to numeric receivers from alphanumeric data names or literal senders, except when the sending field is reference modified.
- There is no diagnostic message for COMP-5 receivers, nor for binary receivers when you specify the TRUNC(BIN) option.

---

## DISPSIGN

The DISPSIGN option controls output formatting for DISPLAY of signed numeric items.

### Syntax



Default is: DISPSIGN=COMPAT

### DISPSIGN=COMPAT

If you specify DISPSIGN=COMPAT, formatting for displayed values of signed numeric items is compatible with prior versions of Enterprise COBOL. Overpunch signs are generated in some cases.

### DISPSIGN=SEP

If you specify DISPSIGN=SEP, the displayed values for signed binary, signed packed-decimal, or overpunch signed zoned-decimal items are always formatted with a leading separate sign.

The following example shows the DISPLAY output with the DISPSIGN=COMPAT option or the DISPSIGN=SEP option specified:

*Table 4. DISPLAY output with the DISPSIGN=COMPAT option or the DISPSIGN=SEP option specified:*

Data items	DISPLAY output with the DISPSIGN=COMPAT option specified	DISPLAY output with the DISPSIGN=SEP option specified
Unsigned binary	111	111
Positive binary	111	+111
Negative binary	11J	-111
Unsigned packed-decimal	222	222
Positive packed-decimal	222	+222
Negative packed-decimal	22K	-222
Zoned-decimal unsigned	333	333
Zoned-decimal trailing positive	33C	+333
Zoned-decimal trailing negative	33L	-333
Zoned-decimal leading positive	C33	+333
Zoned-decimal leading negative	L33	-333

## DLL

DLL affects whether an object module generated by the compiler is enabled for dynamic link library (DLL) support.

### Syntax



### Default

DLL=NO

### YES

Generates an object module that is enabled for dynamic link library (DLL) support. DLL enablement is required if the program is part of a DLL, references DLLs, or contains object-oriented COBOL syntax ( for example, INVOKE statements, or class definitions).

Specification of the DLL option requires that the NODYNAM option and RENT options are also used.

**NO** Generates an object module that is not enabled for DLL usage.

### RELATED REFERENCES

CALLINTERFACE (*Enterprise COBOL Language Reference*)

---

## DYNAM

DYNAM affects whether the compiler dynamically loads subprograms that are invoked through the *CALL literal* statement.

### Syntax



### Default

DYNAM=NO

### YES

Dynamically loads subprograms that are invoked through the *CALL literal* statement.

**Performance consideration:** Using DYNAM=YES eases subprogram maintenance because the application is not relink-edited if the subprogram is changed. However, individual applications with *CALL literal* statements can experience some performance degradation due to a longer path length.

**NO** Includes, in the calling program, the text files of subprograms called with a *CALL literal* statement into a single program object.

### Note:

- The DYNAM option has no effect on the *CALL identifier* statement at compile time. The *CALL identifier* statement always compiles to a dynamic call.
- Do not specify DYNAM=YES for applications running under CICS.

For further details, see *DYNAM* in the *Enterprise COBOL Programming Guide*.

### RELATED REFERENCES

CALLINTERFACE (*Enterprise COBOL Language Reference*)

---

## EXPORTALL

EXPORTALL affects whether the compiler automatically exports certain symbols when the object deck is link-edited to form a DLL.

### Syntax



### Default

EXPORTALL=NO

### YES

Automatically exports the program-name and alternate entry-point names when the object deck is link-edited to form a DLL.

Specification of EXPORTALL requires that the DLL, RENT, and NODYNAM options are also used.

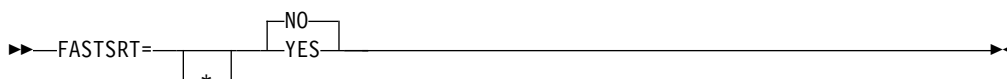
**NO** Does not export any symbols.

---

## FASTSRT

FASTSRT determines whether DFSORT or comparable product performs input and output for sort and merge, or whether they are performed by Enterprise COBOL. It applies only to sorting files by using the format 1 SORT statement.

### Syntax



### Default

FASTSRT=NO

### YES

Specifies that the IBM DFSORT licensed program or comparable product performs input and output when you use either the USING or GIVING option.

**Performance consideration:** Using FASTSRT=YES eliminates the overhead, in terms of CPU time usage, of returning to Enterprise COBOL after each record is processed. However, there are restrictions that you must follow if you choose to use this option. (For a detailed description of the restrictions, see *Improving sort performance with FASTSRT* in the *Enterprise COBOL Programming Guide*.)

**NO** Specifies that Enterprise COBOL does the input and output for the sort and merge.

### Note:

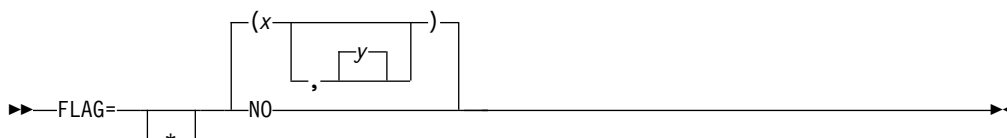
- If FASTSRT is in effect at compile time, the compiler verifies that the FASTSRT interface can be used for all restrictions except these two:
  - A device other than a direct-access device must be used for sort work files.
  - The DCB parameter of the DD statement for the input file or output file must match the file description (FD) of the file.
- If FASTSRT cannot be used, the compiler generates a diagnostic message and prevents the sort program from performing I/O when using either the USING or GIVING options. Therefore, it might be to your advantage to specify YES as the default.

---

## FLAG

FLAG affects whether the compiler produces diagnostic messages at or above a specified severity level.

### Syntax



### Default

FLAG=(I,I)



**Note:** The second severity level used in this syntax must be equal to or higher than the first.

**x** I|W|E|S|U

Specifies that errors at or above the severity level specified are flagged and written at the end of the source listing.

ID	Type	Return code
I	Information	0
W	Warning	4
E	Error	8
S	Severe error	12
U	Unrecoverable error	16

**y** I|W|E|S|U

The optional second severity level specifies the level of syntax messages embedded in the source listing in addition to being at the end of the listing.

**NO** Indicates that no error messages are flagged.

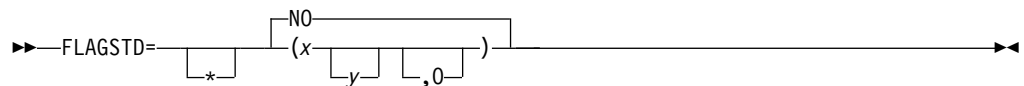
**Note:**

- If the messages are to be embedded, SOURCE must be specified at compile time. Embedded messages enhance productivity because they are placed after the referenced source statement.
- Specification of FLAG(W|E|S) might result in the loss of entire classes of messages from the Events records in the Associated Data (SYSADATA) file. For further details, see *FLAG* in the *Enterprise COBOL Programming Guide*.

## FLAGSTD

FLAGSTD affects the subset of the 85 COBOL Standard language elements that are regarded as conforming, and affects the flagging of informational messages about the language elements used.

**Syntax**



**Default**

FLAGSTD=NO

**x** Can be M, I, or H to specify flagging for a FIPS COBOL subset or standard:

**M** = ANS minimum subset of Standard COBOL

**I** = ANS intermediate subset, composed of those additional intermediate subset language elements that are not part of the ANS minimum subset

**H** = ANS high subset, composed of those additional high subset language elements that are not part of the ANS intermediate subset

**y** Can be any one or two combinations of D, N, or S to further define the level of flagging produced:

**D** Specifies ANS debug module level 1

- N Specifies ANS segmentation module level 1
- S Specifies ANS segmentation module level 2 (where S is a superset of N)
- 0 Specifies that obsolete elements occurring in any of the sets above are flagged
- NO Specifies that no FIPS flagging is to be done

**Note:**

- The following elements are flagged as nonconforming nonstandard IBM extensions to the 85 COBOL Standard:
  - Language syntax used by the COBOL automatic date-processing facilities
  - Language syntax for object orientation and improved interoperability with C/C++
  - Use of the PGMNAME=LONGMIXED compiler option
- When FIPS flagging is specified, informational messages in the source program listing identify:
  - Whether the language element is obsolete, nonconforming standard, or nonconforming nonstandard (language elements that are both obsolete and nonconforming are flagged as obsolete only)
  - The clause, statement, or header containing the nonconforming or obsolete syntax
  - The source program line and an indication of the starting column within that line
  - The level or optional module to which the language element belongs
- FIPS flagging is suppressed when any error diagnosed as level E or higher occurs.
- Interaction of FLAGSTD and other compiler options:
  - If the following compiler options are explicitly or implicitly specified in a program, FLAGSTD=(other than NO) causes a compiler FIPS message to be issued :
    - ADV=NO
    - BLOCK0=YES
    - CICS=YES
    - DLL=YES
    - DYNAM=NO
    - EXPORTALL=YES
    - FASTSRT=YES
    - LITCHAR=APOST
    - NAME=NO
    - NUMPROC=PFD
    - PGMNAME=LONGMIXED
    - QUALIFY=EXTEND
    - THREAD=YES
    - TRUNC=OPT or BIN
    - VLR=COMPAT
    - WORD=(other than NO or RWT)
    - ZONEDATA=MIG
    - ZWB=NO

- Specifying the following options together with FLAGSTD=(other than NO), while attempting to assemble the customization macro, results in a nonzero return code:
  - ADV=NO
  - DBCS=YES
  - DYNAM=NO
  - LITCHAR=APOST
  - NUM=YES
  - NUMPROC=PFD
  - QUALIFY=EXTEND
  - SEQ=YES
  - TRUNC=OPT or BIN
  - VLR=COMPAT
  - WORD=(other than NO or RWT)
  - ZONEDATA=MIG
  - ZWB=NO
- FLAGSTD might produce events records in the Associated Data file for FIPS standard conformation messages. Error messages are not guaranteed to be sequential with respect to source record numbers.

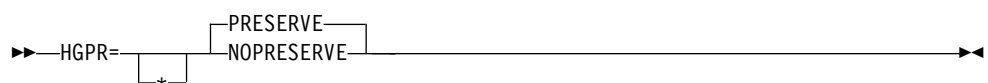
FLAGSTD messages can be converted into diagnostic messages, or suppressed. For details, see "MSGEXIT" on page 43.

---

## HGPR

The HGPR option controls the compiler usage of the 64-bit registers provided by z/Architecture processors.

### Syntax



Default is: HGPR=PRESERVE

The Enterprise COBOL compiler uses the 64-bit width of the z/Architecture General Purpose Registers (GPRs). HGPR stands for "High-halves of 64-bit GPRs", which means the use of native 64-bit instructions.

### HGPR=PRESERVE

If you specify HGPR=PRESERVE, the compiler preserves the high halves of the 64-bit GPRs that a program is using, by saving them in the prolog for the function and restoring them in the epilog. The PRESERVE suboption is necessary only if the caller of the program is not Enterprise COBOL, Enterprise PL/I, or z/OS XL C/C++ compiler-generated code.

## HGPR=NOPRESERVE

If you specify HGPR=NOPRESERVE, the compiler omits preserving the high-halves of the 64-bit GPRs that a program is using, which improves performance.

---

## INEXIT

INEXIT designates a module to be called to obtain source statements instead of reading the SYSIN data set.

### Syntax

►► INEXIT= \* name ►►

### Default

No exit is specified. Equivalent to specifying the NOINEXIT suboption of the EXIT compiler option. If INEXIT=\* is coded without the *name* parameter, NOINEXIT cannot be overridden.

### *name*

Identifies a module to be used with the EXIT compiler option. If the suboption for this user exit is specified, the compiler loads the named module and calls it to obtain source statements instead of reading the SYSIN data set. If the option is supplied, the SYSIN data set is not opened.

For more information about the EXIT compiler option, see *EXIT compiler option* in the *Enterprise COBOL Programming Guide*.

---

## INITCHECK

Use the INITCHECK option to have the compiler check for uninitialized data items and issue warning messages when they are used without being initialized.

### Syntax

►► INITCHECK= \* NO YES ►►

### Default

INITCHECK=NO

**NO** The compiler will not issue any warning messages for uninitialized data items.

### **YES**

The compiler will check for uninitialized data items and issue a warning message when a data item is used without being initialized. However, if a data item is possibly initialized when it is used in a statement, no warning message will be issued.

### Restrictions:

- The INITCHECK option analyzes data items in the WORKING-STORAGE SECTION and LOCAL-STORAGE SECTION only. In particular, it does not analyze data items in the LINKAGE SECTION or FILE SECTION.
- The INITCHECK option does not track external or global data items.

- The INITCHECK option does not track individual elements in tables independently. Instead, if one element of a table is initialized, all corresponding elements of the table are considered to be initialized. This applies to both fixed-length and variable-length tables.
- The INITCHECK analysis does not track the initialization of items if it happens through a pointer. For example, if a pointer to an uninitialized data item is created by using ADDRESS-OF, and that data item is initialized through that pointer, the INITCHECK analysis might also issue a warning message.
- For uninitialized data items being passed BY REFERENCE, no warning messages will be issued. However, the INITCHECK analysis will warn about uninitialized data items being passed BY CONTENT and BY VALUE.
- The INITCHECK and OPTIMIZE=0 compiler options are mutually exclusive. Setting OPTIMIZE=0 and INITCHECK=YES results in a nonzero return code and an error message during assembly of the customization macro.

**Note:**

- All of the INITCHECK analyses occur at compile time only.
- The INITCHECK option has no effect on the behavior or performance of the program after it has been compiled.
- Use of the INITCHECK option might increase compile time and memory consumption.

## INTDATE

INTDATE affects the starting date that is used for date intrinsic functions.

### Syntax



### Default

INTDATE=ANSI

### ANSI

Uses the ANSI COBOL Standard starting date for integer date format dates used with date intrinsic functions. Day 1 = Jan 1, 1601.

With INTDATE(ANSI), the date intrinsic functions return the same results as in COBOL/370 Release 1.

### LILIAN

Uses the Language Environment® Lilian starting date for integer date format dates used with date intrinsic functions. Day 1 = Oct 15, 1582.

With INTDATE(LILIAN), the date intrinsic functions return results compatible with the Language Environment date callable services. These results are different from those in COBOL/370 Release 1.

### Note:

- When INTDATE(LILIAN) is in effect, CEECBLDY is not usable because you have no way to turn an ANSI integer into a meaningful date using either intrinsic functions or callable services. If you code a CALL literal statement with CEECBLDY as the target of the call with INTDATE(LILIAN) in effect, the compiler diagnoses this and converts the call target to CEEDAYS.

- If you set your installation option to INTDATE(LILIAN), you should recompile all of your COBOL/370 Release 1 programs that use intrinsic functions to ensure that all of your code uses the lilian integer date standard. This method is the safest, because you can store integer dates, pass them between programs, and even pass them from PL/I to COBOL to C programs and have no problems.

## LANGUAGE

LANGUAGE affects the language used for compiler output messages.

### Syntax

►► LANGUAGE= \* XX ◀◀

### Default

LANGUAGE=EN

**XX** Specifies the language for compiler output messages. Entries for this parameter might be selected from the following list.

*Table 5. Entries for the LANGUAGE compiler option*

Entry	Language
EN or ENGLISH	Mixed case U.S. English
JA, JP, or JAPANESE	Japanese
UE or UENGLISH	Uppercase U.S. English

### Note:

- The LANGUAGE option name must consist of at least the first two identifying characters. Other characters after the first two can be used; however, only the first two are used to determine the language name.
- This compiler option does not affect the language in which runtime messages are displayed. For more information about runtime options and messages, see the *z/OS Language Environment Programming Guide*.
- Some printers use only uppercase and might not accept output in mixed case (LANGUAGE=ENGLISH).
- To specify the Japanese language option, the Japanese National Language Feature must be installed.
- To specify the English language option (mixed-case English), the U.S. English Language Feature must be installed.
- If your installation provides a language other than those listed above, and you select it as your installation's default, you must specify at least the first two characters of the language name. These two characters must be alphanumeric.
- The selection of Japanese together with specification of the ADATA option might result in DBCS characters being written to error identification records in the Associated Data file.

## LIBEXIT

LIBEXIT designates a module to be called to obtain COPY statements instead of reading the SYSLIB or library-name data set.

## Syntax

►► LIBEXIT= ☐\* ☐name ◀◀

### Default

No exit is specified. Equivalent to specifying the NOLIBEXIT suboption of the EXIT compiler option. If LIBEXIT=\* is coded without the *name* parameter, NOLIBEXIT cannot be overridden.

### *name*

Identifies a module to be used with the EXIT compiler option. If the suboption for this user exit is specified, the compiler loads the named module and calls it to obtain COPY statements instead of reading the SYSLIB or library-name data set. If the option is supplied, the SYSLIB and library-name data sets are not opened.

For more information about the EXIT compiler option, see *EXIT compiler option* in the *Enterprise COBOL Programming Guide*.

---

## LINECNT

LINECNT affects the number of lines to be printed on each page of the compiler source listing.

## Syntax

►► LINECNT= ☐\* ☐60 ☐integer ◀◀

### Default

LINECNT=60

### *integer*

Specifies the number of lines to be printed on each page of the compiler source code listing. Three of the lines are used to generate headings. For example, if you specify LINECNT=60, 57 lines of source code are printed on each page of the output listing, and 3 lines are used for headings.

The LINECNT installation option is equivalent to the LINECOUNT compiler option.

---

## LIST

LIST affects whether an assembler-language expansion is produced in source listings.

## Syntax

►► LIST= ☐\* ☐NO ☐YES ◀◀

### Default

LIST=NO

**YES**

Produces a listing that includes:

- The assembler-language expansion of source code
- Information about working storage
- Global tables
- Literal pools

**NO** Suppresses this listing.

The LIST and OFFSET compiler options are mutually exclusive. Setting OFFSET=YES and LIST=YES results in a nonzero return code and an error message during assembly of the customization macro.

---

## LITCHAR

LITCHAR affects whether the QUOTE figurative constant represents quotation marks or apostrophes.

**Syntax**

```

  >>—LITCHAR=—*QUOTEAPOST—————>>

```

**Default**

LITCHAR=QUOTE

**APOST**

Use APOST if you want the figurative constant [ALL] QUOTE or [ALL] QUOTES to represent one or more apostrophe (') characters.

**QUOTE**

Use QUOTE if you want the figurative constant [ALL] QUOTE or [ALL] QUOTES to represent one or more quotation mark (") characters. QUOTE conforms to the 85 COBOL Standard.

**Note:**

- Either quotation marks or apostrophes can be used as literal delimiters, regardless of whether the APOST or QUOTE option is in effect.
- The delimiter character used as the opening delimiter for a literal must be used as the closing delimiter for that literal.

---

## LVLINFO

LVLINFO affects whether one to eight characters of compiler-level information are inserted into the listing header and object program.

**Syntax**

```

  >>—LVLINFO=—*XXXXXXXX—————>>

```

**Default**

No characters are specified.

**XXXXXXXX**

Identifies the one to eight alphanumeric characters that are inserted into the



listing header following the release number, and also the last eight bytes of the Compiler Options and Program Information Section of the object program. This option might be used to identify “compiler level” information within the listing header and object program.

## MAP

The MAP option affects whether map information about the DATA DIVISION items and all implicitly declared items is shown in the listing. The option also controls whether hexadecimal or decimal offsets are shown for MAP output in the listing.

### Syntax



### Default

MAP=NO

### HEX or DEC

Maps items that are declared in the DATA DIVISION. Map output includes:

- DATA DIVISION map
- Nested program structure map, and program attributes
- Size of the program's WORKING-STORAGE and LOCAL-STORAGE and its location in the object code if the program is compiled with the NORENT option

If you specify MAP=HEX, data item offsets within groups will be in hexadecimal notation.

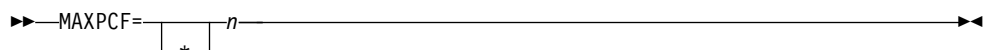
If you specify MAP=DEC, data item offsets within groups will be in decimal notation.

**NO** Mapping is not performed.

## MAXPCF

Use the MAXPCF option to specify a maximum program complexity factor value. The program complexity factor (PCF) is computed by the compiler and the computed value is in the listing file. If the PCF of your program exceeds the maximum value, the compiler will automatically reduce the optimization level to speed up the compilation and use less storage. Therefore, when you compile a suite of programs, you do not have to specify an OPTIMIZE option value for each program.

### Syntax



Default is: MAXPCF=60000

$n$  must be an integer of 0 - 999999.

The aspects of the program taken into consideration when computing the complexity factor include:

- The number of COBOL statements in the PROCEDURE DIVISION, including generated statements from the CICS, SQL or SQLIMS options, and the expansion of COPY and REPLACE statements
- Initialization operations for WORKING-STORAGE or LOCAL-STORAGE data items with value clauses
- Operations for variable-length groups or subgroups in the DATA DIVISION, which compute their size at run time

**Note:** PCF is not a metric to measure how complex a program is. It is merely a count of COBOL items that can cause problems for optimization when there are a lot of them. To measure program complexity, you should use something like the Metrics feature provided by IBM Developer for z Systems®.

For large and complex programs, you can use the MAXPCF option to set a threshold on the program complexity that the compiler attempts optimize. Lower the MAXPCF value to reduce the optimization level, hence the compiler needs less memory and compilation time. Raise the MAXPCF value to attempt to optimize the programs at the cost of longer compilation time.

If you specify MAXPCF=0, no limit is enforced on the complexity of the program, and the MAXPCF option has no effect.

If you specify MAXPCF= $n$  and  $n$  is not zero, when the program complexity factor exceeds  $n$ , any specification of OPTIMIZE(1) or OPTIMIZE(2) is reset to OPTIMIZE(0), and a warning message is generated.

If the COBOL source file contains a sequence of source programs (a batch compile), the MAXPCF limit is applied on a per program basis.

#### Notes:

- If the OPT=1 or OPT=2 option is set at installation time as a fixed, nonoverridable option, then MAXPCF= $n$  with a nonzero  $n$  is an option conflict. In this case, the OPTIMIZE option takes precedence and the MAXPCF=0 option is forced on.
- If you attempt to optimize a program larger than the default threshold by raising the value of MAXPCF to  $n$  where  $n$  is greater than the default, or by specifying MAXPCF(0), the compiler might take excessive time to compile or fail to compile because of insufficient memory.

#### RELATED REFERENCES

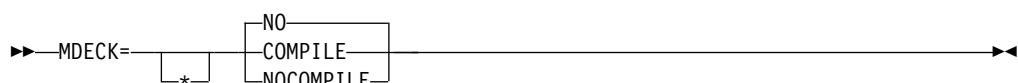
"OPTIMIZE" on page 49

---

## MDECK

MDECK affects whether output from library processing is written to a file.

### Syntax



**Default**

MDECK=NO

**COMPILE**

Compilation continues normally after library processing and generation of the MDECK output file.

**NOCOMPILE**

Compilation ends after library processing is completed and the expanded source program file is written.

**NO** An MDECK output file is not produced.

---

## MSGEXIT

MSGEXIT designates a module to be called to enable customization of compiler messages.

**Syntax**

►► MSGEXIT= \* name ►►

**Default**

No exit is specified. Equivalent to specifying the NOMSGEXIT suboption of the EXIT compiler option. If MSGEXIT=\* is coded without the *name* parameter, NOMSGEXIT cannot be overridden.

***name***

Identifies a module to be used with the EXIT compiler option. If the suboption for this user exit is specified, the compiler loads the named module and calls it to enable customization of compiler messages. The severity of messages can be changed, messages can be suppressed, and FIPS messages resulting from the FLAGSTD compiler option can be converted into diagnostic messages.

For more information about the EXIT compiler option, see *EXIT compiler option* in the *Enterprise COBOL Programming Guide*.

---

## NAME

NAME affects whether a program management binder NAME statement is appended to each object module and whether an ALIAS statement is created for each ENTRY statement.

**Syntax**

►► NAME= \* NO  
NOALIAS  
ALIAS ►►

**Default**

NAME=NO

**ALIAS**

Creates a program management binder ALIAS statement for each ENTRY statement in the program. The ALIAS statement is inserted preceding the NAME statement corresponding to the PROGRAM-ID.

## NOALIAS

Appends a program management binder NAME statement (NAME *modname*(R)) to each object module created in a batch compilation. The module name (*modname*) is derived from the PROGRAM-ID according to the rules for forming external module names.

**NO** Does not append program management binder NAME statements.

The NAME option lets you create multiple modules in a program library with a single batch compilation, which can be useful for dynamic calls.

---

## NSYMBOL

NSYMBOL controls the interpretation of the N symbols used in PICTURE clauses, indicating whether national or DBCS processing is assumed.

### Syntax



### Default

NSYMBOL=NATIONAL

### DBCS

Use DBCS when data items are defined with the PICTURE clause consisting only of the PICTURE symbol N and without the USAGE clause. Such data items are treated as if the USAGE DISPLAY-1 clause were specified. Literals of the form N". . ." or N'. . .' are treated as DBCS literals.

### NATIONAL

Use NATIONAL when data items are defined with the PICTURE clause consisting only of the PICTURE symbol N and without the USAGE clause. Such data items are treated as if the USAGE NATIONAL clause were specified. Literals of the form N". . ." or N'. . .' are treated as national literals.

### Note:

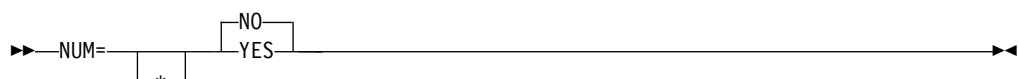
- The NSYMBOL(DBCS) option is compatible with previous releases of IBM COBOL. The NSYMBOL(NATIONAL) option handles the N symbol consistently with the 2002 COBOL Standard.
- NSYMBOL(NATIONAL) forces the DBCS option.

---

## NUM

NUM affects whether source-program line numbers are used in error messages and procedure maps.

### Syntax



### Default

NUM=NO

## YES

Uses the line numbers from the source program rather than compiler-generated line numbers in error messages and procedure maps.

## NO

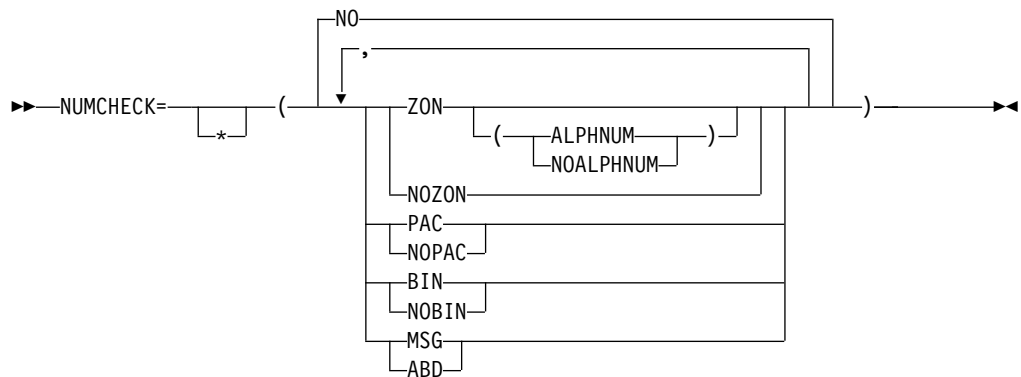
Uses the compiler-generated line numbers in error messages and procedure maps.

If COBOL programmers use COPY statements and NUM=YES is in effect, they must ensure that the source program line numbers and the COPY member line numbers are coordinated.

# NUMCHECK

The NUMCHECK compiler option tells the compiler whether to generate extra code to validate data items when they are used as sending data items. For zoned decimal (numeric USAGE DISPLAY) and packed decimal (COMP-3) data items, the compiler generates implicit numeric class tests for each sending field. For binary data items, the compiler generates SIZE ERROR checking to see whether the data item has more digits than its PICTURE clause allows.

## Syntax



## Default

NUMCHECK=(NO)

Suboption defaults are:

- If no suboption is specified, defaults are NUMCHECK=(ZON(ALPHNUM),PAC,BIN,MSG).
- If no datatype suboption is specified, default datatype suboptions are ZON(ALPHNUM), PAC, and BIN. For example, NUMCHECK=(ABD) has the same effect as NUMCHECK=(ZON(ALPHNUM),PAC,BIN,ABD).
- If only one datatype suboption is specified, defaults are NOZON, NOPAC, NOBIN, and MSG. For example, NUMCHECK=(BIN) has the same effect as NUMCHECK=(NOZON,NOPAC,BIN,MSG).
- If all datatype suboptions are specified with NO, then the listing will show NONUMCHECK. For example, NUMCHECK=(NOZON,NOPAC,NOBIN) has the same effect as NUMCHECK=NO.

## ZON [(ALPHNUM | NOALPHNUM)] | NOZON

Specifying ZON or ZON(ALPHNUM) causes the compiler to generate code for an implicit numeric class test for zoned decimal (numeric USAGE DISPLAY) data items that are used as sending data items in COBOL statements.

Specifying ZON(NOALPHNUM) causes the compiler to generate code for an implicit numeric class test for zoned decimal (numeric USAGE DISPLAY) data items that are used as sending data items in COBOL statements, except when they are used in a comparison with an alphanumeric data item, alphanumeric literal or alphanumeric figurative constant.

Receivers are not checked, unless they are both a sender and a receiver, such as data item B in the following sample statements:

```
ADD A TO B
DIVIDE A INTO B
COMPUTE B = A + B
INITIALIZE B REPLACING ALPHANUMERIC BY B
```

This checking is done before the data is used in each statement:

- If the data is NOT NUMERIC, either a warning message for NUMCHECK=(ZON,MSG) or a terminating message for NUMCHECK=(ZON,ABD) is issued.
- If the data is NUMERIC, the external behavior of the statement is the same as NUMCHECK=(NOZON), other than being slower.

#### **PAC | NOPAC**

Specifying PAC causes the compiler to generate code for an implicit numeric class test for packed decimal (COMP-3) data items that are used as sending data items in COBOL statements. For packed decimal data items that have an even number of digits, the unused bits are checked for ones.

**Restriction:** For CALL statements, NUMCHECK=(ZON) and NUMCHECK=(PAC) check BY CONTENT data items that are zoned decimal or packed decimal, but they do not check BY REFERENCE parameters. (Neither zoned decimal nor packed decimal data items can be specified in a BY VALUE phrase.)

#### **BIN | NOBIN**

When TRUNC(OPT) or TRUNC(STD) are in effect, specifying NUMCHECK=(BIN) causes the compiler to generate code similar to ON SIZE ERROR to test if binary data items contents are bigger than the PICTURE clause. This extra code will be generated only for binary data items that are used as sending data items, and COMP-5 data items will not get this ON SIZE ERROR code generated.

NUMCHECK(BIN) has no effect if TRUNC(BIN) is in effect.

#### **MSG | ABD**

Determines whether the message issued for invalid data is a warning level message to continue processing or a terminating level message to cause an abend:

- If MSG is in effect, a runtime warning message with the line number, data item name, data item content, and program name is issued.
- If ABD is in effect, a terminating message is issued that causes an abend.

**NO** No code is generated to validate data items when they are used as sending data items.

**Performance considerations:** NUMCHECK is much slower than NUMCHECK=NO, depending on how many zoned decimal (numeric USAGE DISPLAY) data items, packed decimal (COMP-3) data items, and binary data items are used in a COBOL program.

In Enterprise COBOL V5.2 with PTF for APAR PI81006 installed, ZONECHECK is deprecated and can no longer be specified in IGYCDOPT.  
NUMCHECK=(ZON(ALPHNUM)) gives the same results as ZONECHECK used to.

#### RELATED REFERENCES

"NUMPROC"

"TRUNC" on page 63

"ZONECHECK" on page 69

"ZONEDATA" on page 69

---

## NUMCLS

NUMCLS, in conjunction with NUMPROC, affects the numeric signs that the compiler treats as valid in numeric class tests.

NUMCLS specifies the sign representations that are recognized as valid by the numeric class test for data items that are defined with all of the following conditions:

- As signed (with an "S" in the PICTURE clause)
- Using DISPLAY or COMPUTATIONAL-3 (packed-decimal)
- No SEPARATE phrase on any SIGN clause

### Syntax

►► NUMCLS= 

PRIM
ALT

 ◄◄

#### Default

NUMCLS=PRIM

#### ALT

Processing with ALT accepts hexadecimal A through F as valid.

#### PRIM

Processing with PRIM accepts hexadecimal C, D, and F as valid.

#### Note:

- The numeric class test is affected by how the NUMPROC and the NUMCLS options are specified.
- The NUMCLS option is effective only for NUMPROC=NOPFD.  
NUMPROC=PFD specifies more strict rules for valid sign configuration.

---

## NUMPROC

NUMPROC affects the treatment and processing of signs in internal decimal and zoned decimal data.

### Syntax

►► NUMPROC= 

*
---

NOPFD
PFD

 ◄◄

#### Default

NUMPROC=NOPFD

## NOPFD

Repairs signs on input. After repair is performed, the signs meet the criteria for NUMPROC=PFD.

## PFD

Optimizes the generated code, especially when a non-zero OPTIMIZE level (OPT=1 or OPT=2) is specified. No explicit sign repair is performed. Note that NUMPROC=PFD has stringent criteria to produce correct results. To use NUMPROC=PFD:

- The sign position of unsigned numeric items must be X'F'.
- The sign position of signed numeric items must be either X'C' if positive or zero, or must be X'D' if negative.
- The sign position of separately signed numeric items must be either '+' if positive or zero, or '-' if otherwise.

Elementary MOVE and arithmetic statements in Enterprise COBOL always generate results with these preferred signs; however, group MOVEs and redefinitions might produce nonconforming results. The numeric class test can be used for verification. With NUMPROC=PFD, a numeric item fails the numeric class test if the signs do not meet the preferred sign criteria.

**Performance consideration:** Using NUMPROC=PFD generates significantly more efficient code for numeric comparisons. For most references to COMP-3 and DISPLAY numeric data items, using NUMPROC=NOPFD generates extra code because of sign “fix-up” processing. This extra code might also inhibit some other types of optimizations. Before setting this option, consult with your application programmers to determine the effect on the application program's output.

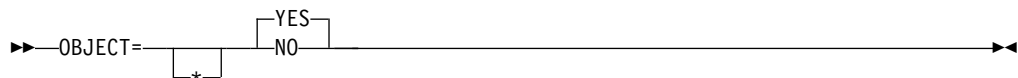
Both the NUMPROC and NUMCLS options affect the numeric class test. With NUMPROC=NOPFD, the results of the numeric class test are controlled by how NUMCLS is set. When NUMPROC=PFD, a data item must meet the preferred sign criteria to be considered numeric.

---

## OBJECT

OBJECT affects whether the generated object code is written to a file.

### Syntax



### Default

OBJECT=YES

### YES

Places the generated object code in a file, defined by the SYSLIN DD statement, to be used as input to the binder.

**NO** Places no object code in SYSLIN.

The OBJECT=NO option conflicts with all values for TEST other than NO.



---

## OFFSET

OFFSET affects whether a condensed PROCEDURE DIVISION listing is produced.

### Syntax



### Default

OFFSET=NO

### YES

Produces a condensed PROCEDURE DIVISION listing. The procedure portion of the listing will contain line numbers, verb references, and the location of the first instruction generated for each verb. In addition, the listing also shows:

- Global tables
- Literal pools
- Size of the program's working storage, and its location in the object code if the program is compiled with the NORENT compiler option

**NO** Does not condense the listing or produce the items listed above.

The LIST and OFFSET compiler options are mutually exclusive. Setting OFFSET=YES and LIST=YES results in a nonzero return code when you attempt to assemble the customization macro. For more information about conflict resolution, see "Conflicting compiler options" on page 15.

---

## OPTIMIZE

OPTIMIZE affects the level of optimization that is made to object code, and can result in performance improvements.

### Syntax



### Default

OPT=0

- 0** Specifies limited optimizations, which result in the shortest compilation time. When the TEST option is specified, full debug capabilities are available.
- 1** Specifies optimizations that improve application runtime performance. Optimizations at this level include basic inlining, strength reduction, simplification of complex operations into equivalent simpler operations, removal of some unreachable code and block rearrangement. Also, OPT=1 includes some intrablock optimizations such as common subexpression elimination and value propagation. When the TEST option is specified, most debug capabilities are available.
- 2** Specifies further optimizations, which include more aggressive simplifications and instruction scheduling. Also, some interblock optimizations such as global

value propagation and loop invariant code motion are included. When the TEST option is specified, some debug capabilities are available.

**Performance consideration:** Using OPT=1 or OPT=2 generally results in more efficient runtime code.

**Note:**

- The OPTIMIZE compiler option is fully supported for programs that use object-oriented syntax for Java™ interoperability.
- Optimization is set to 0 if an S-level error or U-level error occurs, or if the Program Complexity Factor exceeds the MAXPCF integer specified.

For further details, see *OPTIMIZE* in the *Enterprise COBOL Programming Guide*.

---

## OUTDD

OUTDD specifies the ddname to which DISPLAY output should be directed.

**Syntax**



**Default**

OUTDD=SYSOUT

**ddname**

Specifies the ddname of the file used for runtime DISPLAY output.

Change the default for this option if, at run time, you expect there could be a conflict with another product that requires SYSOUT as a ddname.

To understand how OUTDD interacts with the MSGFILE runtime option, see the description of MSGFILE in the *z/OS Language Environment Programming Reference*.

---

## PGMNAME

PGMNAME controls the handling of program-names and entry-point names.

**Syntax**



**Default**

PGMNAME=COMPAT

**COMPAT**

Program names are processed in a manner compatible with COBOL/370 Release 1 and VS COBOL II.

**LONGMIXED**

Program names are processed as is, without truncation, translation, or folding to uppercase.

## LONGUPPER

Program names are folded to uppercase by the compiler but otherwise are processed as is, without truncation or translation.

The PGMNAME option controls the handling of names used in the following contexts:

- Program names defined in the PROGRAM-ID paragraph
- Program entry-point names in the ENTRY statement
- Program-name references in:
  - CALL statements that reference nested programs, statically linked programs, or DLLs
  - SET *procedure-pointer* or *function-pointer* statements that reference statically linked programs or DLLs
  - CANCEL statements that reference nested programs

For further details, see PGMNAME in the *Enterprise COBOL Programming Guide*.

---

## PRTEXTIT

PRTEXTIT designates a module to be called instead of output being written to the SYSPRINT data set.

### Syntax

►► PRTEXTIT= \* name ◀◀

### Default

No exit is specified. Equivalent to specifying the NOPRTEXTIT suboption of the EXIT compiler option. If PRTEXTIT=\* is coded without the *name* parameter, NOPRTEXTIT cannot be overridden.

### *name*

Identifies a module to be used with the EXIT compiler option. When the suboption for this user exit is specified, the compiler loads the named module and calls it instead of writing to the SYSPRINT data set. When the option is supplied, the SYSPRINT data set is not opened.

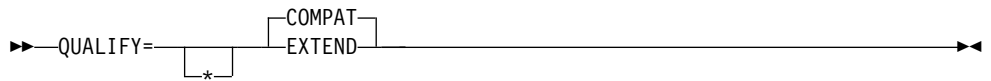
For more information about the EXIT compiler option, see *EXIT compiler option* in the *Enterprise COBOL Programming Guide*.

---

## QUALIFY

QUALIFY affects qualification rules and controls whether to extend qualification rules so that some data items that cannot be referenced under COBOL Standard rules can be referenced.

## Syntax



## Default

QUALIFY=COMPAT

## COMPAT

If QUALIFY=COMPAT is in effect, the behavior will be the same as in previous COBOL compilers. A reference must be unique even if there is only one data item with exactly that complete set of qualifiers.

## EXTEND

If QUALIFY=EXTEND is in effect, qualification rules are extended so that some references that are not unique by COBOL standard rules can be unique. If every level in the containing hierarchy of a group of names is qualified, the set of qualifiers is called a *complete set of qualifiers*. If there is only one data item with a specific complete set of qualifiers, the reference resolves to that data item, even if the same set of qualifiers could match with another reference as an incomplete set of qualifiers.

## Example

```
01 A.  
  02 B.  
    03 C PIC X.  
    02 C PIC X.  
  .  
  .  
  .  
Move space to C of A      *> Refers to 02 level C (unique only with QUALIFY(EXTEND))  
Move space to C of B of A  *> Refers to 03 level C (unique by COBOL standard rules)  
Move space to C of B      *> Refers to 03 level C (unique by COBOL standard rules)
```

# RENT

RENT affects whether generated object code is reentrant.

## Syntax



## Default

RENT=YES

## YES

Indicates that generated object code is to be reentrant. Using RENT=YES enables the program to be placed in shared storage for running above the 16 MB line. However, this option causes the compiler to generate additional code to ensure that the application program is reentrant.

**NO** Indicates that generated object code is not to be reentrant.

## Note:

- Compile programs with RENT if they will be run in virtual storage addresses above 16 MB.
- Execution of nonreentrant programs above 16 MB is not supported. Programs compiled with NORENT must be RMODE 24.
- The RENT compiler option is required for programs that are run under CICS.
- The RMODE assigned to a program depends on the RENT|NORENT and RMODE compiler options. Valid combinations are shown in the following table.

Table 6. Effect of RENT and RMODE on residency mode

RENT NORENT setting	RMODE setting	Residency mode assigned
RENT	AUTO	RMODE ANY
RENT	ANY	RMODE ANY
RENT	24	RMODE 24
NORENT	AUTO	RMODE 24
NORENT	ANY	Compiler option conflict
NORENT	24	RMODE 24

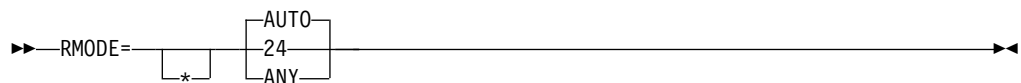
- If the THREAD compiler option is specified, the RENT compiler option must also be specified. If THREAD and NORENT are specified at the same level of precedence, the RENT option is forced on.

For further details, see *RENT* in the *Enterprise COBOL Programming Guide*.

## RMODE

RMODE affects the residency mode of generated object programs.

### Syntax



### Default

RMODE=AUTO

- 24** Specifies that a program will have RMODE 24 whether NORENT or RENT is specified.

### ANY

Specifies that a program will have RMODE ANY if RENT is specified, and will receive an error if NORENT is specified.

### AUTO

Specifies that a program will have RMODE 24 if NORENT is specified, and RMODE ANY if RENT is specified.

### Note:

- Enterprise COBOL NORENT programs that pass data to programs running in AMODE 24 must be either compiled with the RMODE(24) option or link-edited with RMODE 24. The data areas for NORENT programs will be above the 16

MB line or below the 16 MB line depending on the RMODE of the program, even if DATA(24) has been specified. DATA(24) applies to programs compiled with the RENT option only.

- Programs compiled with Enterprise COBOL always have AMODE ANY. The RMODE assigned to a program depends on the RMODE and RENT|NORENT compiler options. Valid combinations are shown in the following table.

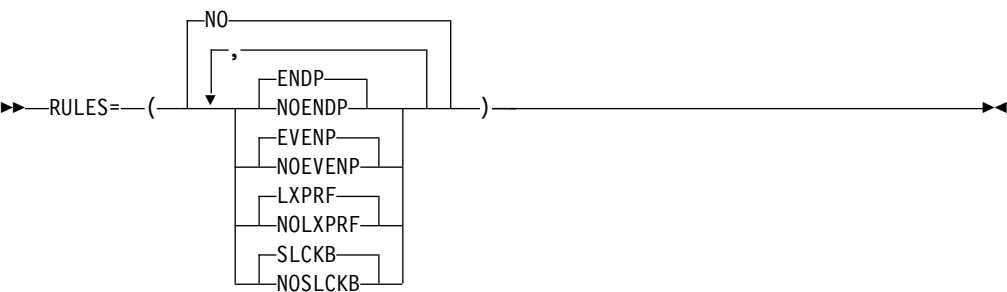
Table 7. Effect of RMODE and RENT | NORENT on residency mode

RMODE setting	RENT NORENT setting	Residency mode assigned
AUTO	RENT	RMODE ANY
AUTO	NORENT	RMODE 24
ANY	RENT	RMODE ANY
ANY	NORENT	Not applicable. A compiler error is issued.
24	RENT	RMODE 24
24	NORENT	RMODE 24

# RULES

You can use the RULES option to request information about your program from the compiler to improve the program by flagging certain types of source code at compile time.

## Syntax



## Default

RULES = (NO)

Has the same effect as RULES=(ENDP,EVENP,LXPRF,SLCKB).

You can specify the following suboptions for RULES other than default ones:

### RULES=(NOENDP)

Causes the compiler to issue warning messages when the scope of a conditional statement is terminated by a period instead of an explicit scope terminator END-\*.

### RULES=(NOEVENP)

Causes the compiler to issue warning messages for any USAGE PACKED-DECIMAL (COMP-3) data items that have an even number of digits because those data items whose unused bits are not zero can lead to an unexpected program behavior.

**Note:** RULES(NO EVENP) helps compilers to identify even-length USAGE PACKED-DECIMAL (COMP-3) data items. However, it is not necessary to change those data items to odd-length items when the related database schema uses even-length packed values.

#### **RULES=(NOLXPRF)**

Causes the compiler to issue warning messages for usage of inefficient COBOL features. These features might include USAGE DISPLAY numeric data items in arithmetic statements, large amounts of space padding in MOVE statements, inefficient compiler options, and other cases.

#### **RULES=(NOSLCKB)**

Causes the compiler to issue warning messages for any SYNCHRONIZED data items that cause the compiler to add slack bytes, either slack bytes within records or slack bytes between records. Each data item that causes slack bytes to be added gets a compiler diagnostic.

#### **Notes:**

- It is not necessary to specify all of the suboptions for RULES. If a suboption is not specified, the default takes effect.
- RULES must be specified with at least one suboption for installation defaults.
- If more than one suboption is specified, separate the suboptions with commas or space, for example, RULES=(NOLXPRF,SLCKB) or RULES=(NOLXPRF SLCKB).

---

## **SERVICE**

Use SERVICE to place a string in the object module if the object module is generated. If the object module is linked into a program object, the string is loaded into memory with this program object. If the Language Environment dump includes a traceback, this string is included in that traceback.

#### **Syntax**

►►—SERVICE= ☐\* ☐NO 'service string' ►►

#### **Default**

SERVICE=NO

The *service string* is limited to 64 characters in length.

---

## **SEQ**

SEQ affects whether the compiler verifies that source statements are in ascending order by sequence number.

#### **Syntax**

►►—SEQ= ☐\* ☐YES ☐NO ►►

#### **Default**

SEQ=YES

**YES**

Checks that the source statements are in ascending alphanumeric order by line number.

**NO** Does not perform sequence checking.

If both SEQ and NUM are in effect at compile time, the sequence is checked according to numeric, rather than alphanumeric, collating sequence.

---

## SOURCE

SOURCE affects whether source statements are included in compiler listings.

**Syntax****Default**

SOURCE=YES

**YES**

Indicates that you want a listing of the source statements in the compiler-generated output. This listing also includes any statements embedded by COPY.

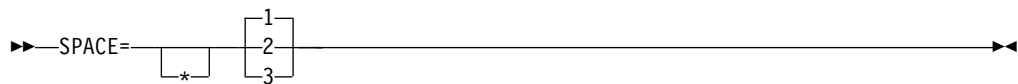
**NO** Source statements do not appear in the output.

The SOURCE compiler option must be in effect at compile time if you want embedded messages in the source listing.

---

## SPACE

SPACE affects whether single-, double-, or triple-spacing is used in source listings.

**Syntax****Default**

SPACE=1

- 1 Provides single spacing for the source statement listing.
- 2 Provides double spacing for the source statement listing.
- 3 Provides triple spacing for the source statement listing.

---

## SQL

SQL affects whether the DB2 coprocessor is enabled and whether DB2 options can be specified.



## Syntax



### Default

SQL=NO

### YES

Use to enable the DB2 coprocessor and to specify DB2 options. You must specify the SQL option if your COBOL source program contains SQL statements and it has not been processed by the DB2 precompiler.

**NO** Specify to have any SQL statements found in the source program diagnosed and discarded.

Use SQL=NO if your COBOL source programs do not contain SQL statements, or if the separate SQL precompiler will be used to process SQL statements before invocation of the COBOL compiler.

### Note:

- You can specify the SQL option in any of the compiler option sources: compiler invocation, PROCESS or CBL statements, or installation defaults.
- Use either quotation marks or apostrophes to delimit the string of DB2 options.
- DB2 options cannot be specified as part of customizing the SQL option. (DB2 options are supported only if the SQL compiler option is specified as an invocation option or in a CBL or PROCESS statement.) However, default DB2 options can be specified when you customize the DB2 product installation defaults.

---

## SQLCCSID

SQLCCSID controls whether the CODEPAGE compiler option influences the processing of SQL statements when the SQL compiler option is in effect.

## Syntax



### Default

SQLCCSID=YES

### YES

Indicates that the CODEPAGE compiler option setting will influence the processing of SQL statements within the source program when the integrated DB2 coprocessor (SQL compiler option) is used.

**NO** Indicates that the CODEPAGE compiler option setting will not influence the processing of SQL statements within the source program when the integrated DB2 coprocessor is used. Only COBOL statements will be sensitive to the CCSID specified in the CODEPAGE option.

### Note:

- The SQLCCSID option has an effect only when you use the integrated DB2 coprocessor (SQL compiler option).
- The NOSQLCCSID option is recommended for applications that require the highest compatibility with the behavior of the DB2 precompiler.
- For further details, see *SQLCCSID* in the *Enterprise COBOL Programming Guide*.

## SQLIMS

SQLIMS affects whether the IMS SQL coprocessor is enabled and whether Information Management System (IMS) suboptions can be specified.

### Syntax



### Default

SQLIMS=NO

### YES

Use to enable the IMS SQL coprocessor and to specify Information Management System (IMS) suboptions. You must specify the SQLIMS option if a COBOL source program contains SQLIMS statements.

**NO** Specify to have any SQLIMS statements found in the source program diagnosed and discarded.

Use SQLIMS=NO if your COBOL source programs do not contain SQLIMS statements.

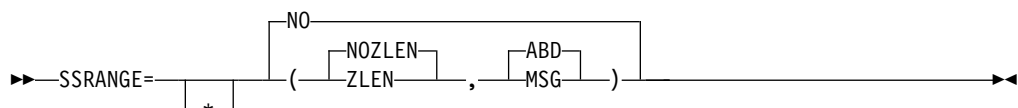
### Notes:

- You can specify the SQLIMS option in any of the compiler option sources: compiler invocation, PROCESS or CBL statements, or installation defaults.
- Use either double quotation marks or single quotation marks to delimit the string of IMS suboptions.
- You can partition a long suboption string into multiple suboption strings in multiple CBL statements.

## SSRANGE

SSRANGE affects whether code is generated to check for out-of-range storage references.

### Syntax



With the PTF for APAR PI53044 installed, new suboptions ZLEN and NOZLEN are added to control how the compiler checks reference modification lengths.

With the PTF for APAR PI86343 installed, new suboptions MSG and ABD are added to control the runtime behavior of the COBOL program when a range check fails.

#### **Default**

SSRANGE=NO

#### **ZLEN|NOZLEN**

Generates code that checks subscripts, reference modifications, variable-length group ranges, and indexes at run time to ensure that they do not refer to storage outside the area assigned. It also verifies that a table with ALL subscripting, specified as a function argument, contains at least one occurrence in the table.

The generated code also checks that variable-length items do not exceed their defined maximum length as a result of incorrect setting of the OCCURS DEPENDING ON object. For unbounded groups or their subordinate items, checking is done only for reference modification expressions. Subscripted or indexed references to tables subordinate to an unbounded group are not checked.

The ZLEN and NOZLEN suboptions control how the compiler checks reference modification lengths:

- If ZLEN is in effect, the compiler generates code to ensure that reference modification lengths are greater than or equal to zero (that is, zero-length reference modification specifications will NOT get an SSRANGE error at runtime).
- If NOZLEN is in effect, the compiler generates code to ensure that reference modification lengths are greater than or equal to 1 (that is, zero-length reference modification specifications will get an SSRANGE error at runtime). This is compatible with how SSRANGE behaved in previous COBOL versions.

**Performance consideration:** If SSRANGE=ZLEN or SSRANGE=NOZLEN at compile time, object code size is increased and there will be an increase in runtime overhead to accomplish the range checking.

#### **MSG|ABD**

The MSG and ABD suboptions control the runtime behavior of the COBOL program when a range check fails.

- If MSG is in effect and a range check fails, a runtime warning message will be issued. Also, the program will continue executing and might potentially identify other out-of-range conditions.
- If ABD is in effect and a range check fails, the first out-of-range condition will result in a runtime error message and the program will abend. You can find the next potential out-of-range condition by fixing the first out-of-range condition and then recompiling and running the program again. To identify all other potential out-of-range conditions, you might need to repeat this process several times.

**Performance consideration:** If anything other than SSRANGE=NO is in effect at compile-time, the object code size will be increased, and the runtime overhead will also be increased to accomplish the range checking.

**NO** No code is generated to perform subscript or index checking at run time.

#### **Notes:**

- If you specify anything other than SSRANGE=NO, range checks are generated by the compiler and the checks are always executed at run time. The

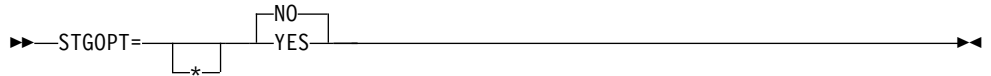
compiled-in range checks cannot be disabled even if you specify the Language Environment runtime option CHECK(OFF).

---

## STGOPT

The STGOPT option controls storage optimization.

### Syntax



Default is: STGOPT=NO

If you specify STGOPT=YES, the compiler might discard any or all of the following data items, and does not allocate storage for them.

- Unreferenced LOCAL-STORAGE and non-external WORKING-STORAGE level-77 and level-01 elementary data items
- Non-external level-01 group items if none of their subordinate items are referenced
- Unreferenced special registers

The compiler will not generate code to initialize discarded data items to the values in their VALUE clauses.

In addition, with STGOPT=YES, data items in the LOCAL-STORAGE SECTION can be reordered in memory to optimize performance.

---

## TERM

TERM affects whether progress and diagnostic messages are sent to the SYSTERM device.

### Syntax



### Default

TERM=NO

### YES

Specifies that the progress and diagnostic messages are sent to the SYSTERM file, which defaults to the user's terminal unless specified otherwise.

**NO** Specifies that no messages are sent to the SYSTERM file.

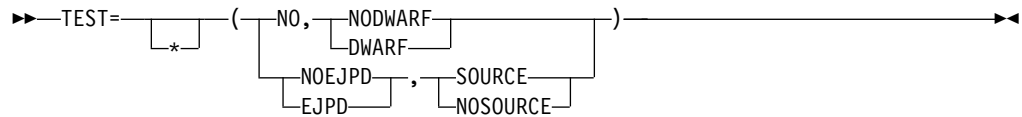
If TERM is specified in the source program, a SYSTERM DD statement must also be specified for each application program.

TERM corresponds to the TERMINAL compiler option.

## TEST

TEST affects the amount of debugging information that is produced in object code, which determines the level of debugging support that is available.

### Syntax



**Note:** Unlike specifying this option in JCL or CBL/PROCESS, all suboptions are required and must be in the order shown in the syntax diagram.

### Option default

TEST=(NO,NODWARF)

### TEST=(NO, DWARF)

If you specify TEST=(NO,DWARF), basic DWARF diagnostic information is included in the application module. This option enables application failure analysis tools, such as CEEDUMP and IBM Fault Analyzer. With TEST=(NO,DWARF), the debugging information is a subset of the DWARF information that is available with TEST.

Debugging information generated by the compiler is in the industry-standard DWARF format. For more information about DWARF, see *About Common Debug Architecture* in the DWARF/ELF Extensions Library Reference.

### TEST=(NO, NODWARF)

If you specify TEST=(NO,NODWARF), DWARF diagnostic information is not included in the application module.

### TEST=(other than NO)

Produces object code that is enabled to be symbolically debugged using Debug Tool. When the TEST option is specified, DWARF debugging information is included in the application module.

### TEST=(EJPD, ...)

If you specify TEST=(EJPD,...) and OPT=(1|2):

- The Debug Tool commands GOTO and JUMPTO are enabled.
- Program optimization will be reduced. Optimization will be done within statements; most optimizations will not cross statement boundaries.

### TEST=(NOEJPD, ...)

If you specify TEST=(NOEJPD,...) and OPT=(1|2):

- The JUMPTO and GOTO commands are not enabled. However, you can still use JUMPTO and GOTO if you use the SET WARNING OFF Debug Tool command. In this scenario, JUMPTO and GOTO will have unpredictable results.
- The normal amount of program optimization is done.

### TEST=(..., SOURCE)

If you specify TEST=(...,SOURCE), the generated DWARF debug

information generated by the compiler includes the expanded source code, and the compiler listing is not needed by IBM Debug Tool.

#### TEST=(..., NOSOURCE)

If you specify TEST=(...,NOSOURCE), the generated DWARF debugging information does not include the expanded source code, and you will not be able to use IBM Debug Tool.

**Note:** If you specify the TEST option, you must set the CODEPAGE option to the CCSID that is used for the COBOL source program. In particular, programs that use Japanese characters in DBCS literals or DBCS user-defined words must be compiled with the CODEPAGE option set to a Japanese codepage CCSID. For more information, see “CODEPAGE” on page 23.

---

## THREAD

THREAD controls whether programs are to be enabled for use in multithreaded applications.

### Syntax



### Default

THREAD=NO

### YES

Use YES to indicate that programs are to be enabled for execution in Language Environment enclaves that have multiple POSIX threads or PL/I tasks.

**NO** Use NO to indicate that programs are not to be enabled for execution in Language Environment enclaves that have multiple POSIX threads or PL/I tasks.

**Performance consideration:** If the THREAD compiler option is specified, runtime performance might be degraded because of the serialization logic that is automatically generated.

### Note:

- If the THREAD compiler option is specified, the program is enabled for use in a threaded application. However, THREAD can be used in nonthreaded applications. For example, you can run a program that was compiled with the THREAD option in the CICS environment if the application does not contain multiple POSIX threads or PL/I tasks at run time.
- If the THREAD compiler option is specified, the RENT compiler option must also be specified. If THREAD and NORENT are specified at the same level of precedence, RENT is forced on.
- For COBOL programs to run in a threaded application, all COBOL programs in the run unit must be compiled with the THREAD compiler option.
- If the THREAD compiler option is specified, the following language elements are not supported. If any of the following language elements are specified, they are diagnosed as errors:

- ALTER statement
- DEBUG-ITEM special register
- GO TO statement without a procedure name
- INITIAL phrase in the PROGRAM-ID paragraph
- Nested programs
- RERUN
- Segmentation module
- Format 1 SORT or MERGE statements
- STOP *literal* statement
- USE FOR DEBUGGING statement
- If you compile programs with the THREAD compiler option, the following special registers are allocated upon each invocation:
  - ADDRESS OF
  - RETURN-CODE
  - SORT-CONTROL
  - SORT-CORE-SIZE
  - SORT-FILE-SIZE
  - SORT-MESSAGE
  - SORT-MODE-SIZE
  - SORT-RETURN
  - TALLY
  - XML-CODE
  - XML-EVENT
  - XML-INFORMATION
  - XML-NAMESPACE
  - XML-NAMESPACE-PREFIX
  - XML-NNAMESPACE
  - XML-NNAMESPACE-PREFIX
  - XML-NTEXT
  - XML-TEXT

---

## TRUNC

TRUNC affects the way that binary data is truncated during MOVE and arithmetic operations.

### Syntax



### Default

TRUNC=STD

### BIN

Should not be used as an installation default. Specifies that:

- Output binary fields are truncated only at halfword, fullword, and doubleword boundaries, rather than at PICTURE-clause limits.

- Sending binary fields are treated as halfwords, fullwords, or doublewords, and no assumption is made that the values are limited to those implied by the PICTURE clause.
- DISPLAY converts and outputs the full content of binary fields with no truncation to the PICTURE description.

#### OPT

The compiler assumes that the data conforms to PICTURE and USAGE specifications. The compiler manipulates the result based on the size of the field in storage (halfword or fullword).

TRUNC=OPT is recommended, but it should be specified only when data being moved into binary areas does not have a value with larger precision than that defined by the binary item PICTURE clause. Otherwise, truncation of high-order digits might occur.

#### STD

Conforms to the 85 COBOL Standard.

Controls the way arithmetic fields are truncated during MOVE and arithmetic operations. The TRUNC option applies only to binary (COMP) receiving fields in MOVE statements and in arithmetic expressions. When TRUNC=STD is in effect, the final intermediate result of an arithmetic expression, or of the sending field in the MOVE statement, truncates to the number of digits in the PICTURE clause of the binary receiving field.

**Performance consideration:** Using TRUNC=OPT does not generate extra code, and generally improves performance. However, both TRUNC=BIN and TRUNC=STD generate extra code whenever a BINARY data item is changed. TRUNC=BIN is usually the slower of these options.

#### Recommendations:

- TRUNC=BIN is the recommended option for programs that use binary values set by other products. Other products, such as IMS, DB2, C/C++, FORTRAN, and PL/I, might place values in COBOL binary data items that do not conform to the PICTURE clause of those data items.
- You can use TRUNC=OPT with CICS programs provided that your data conforms to the PICTURE clause for the binary data items.
- Setting this option affects program runtime logic; that is, the same COBOL source program can give different results depending on the option setting. Verify whether your COBOL source programs assume a particular setting for correct running.

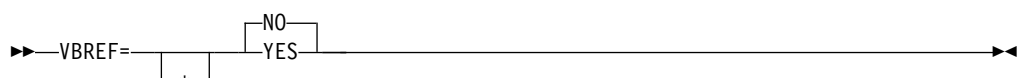
#### RELATED REFERENCES

"NUMCHECK" on page 45

## VBREF

VBREF affects whether a cross-reference of verbs to line numbers and a verb-usage summary is produced.

#### Syntax





**Default**

VBREF=NO

**YES**

Produces a cross-reference of all verb types in a source program to the line numbers where they are found. VBREF=YES also produces a summary of how many times each verb was used in the program.

**NO** Does not produce a cross-reference or verb-summary listing.

---

**VLR**

The VLR option affects the file status returned from READ statements for variable-length records when the length of record returned is inconsistent with the record descriptions. It eases your migration from earlier versions to Enterprise COBOL V5, if your programs have READ statements that result in a record length conflict.

**Syntax****Default**

VLR=STANDARD

After the execution of a READ statement:

- If the number of character positions in the record that is read is less than the minimum size specified by the record description entries for the file, the portion of the record area that is to the right of the last valid character read is undefined.
- If the number of character positions in the record that is read is greater than the maximum size specified by the record description entries for the file, the record is truncated on the right to the maximum size.

In either of these cases, the READ statement is successful, and the file status is set to either 00 (hiding the record length conflict condition) or 04 (indicating that a record length conflict has occurred), depending on the VLR compiler option setting.

**COMPAT**

If you specify VLR=COMPAT, you get the status value of 00 when READ statements encounter a record length conflict.

**Note:** This setting can hide I/O problems that can arise with the wrong length read situation. Use the VLR=COMPAT option with caution, and check for correct READ statements.

**STANDARD**

If you specify VLR=STANDARD, you get the status value of 04 when READ statements encounter a record length conflict.

You can add code to test for FS=04 to avoid accessing undefined data in a record and also avoid getting protection exceptions for attempting to reference a part of the record that was truncated.

---

## VSAMOPENFS

The VSAMOPENFS option affects the user file status reported from successful VSAM OPEN statements that require verified file integrity check.

### Syntax



### Default

VSAMOPENFS=COMPAT

### COMPAT

If you specify VSAMOPENFS=COMPAT, the statement returns the file status 97 when a VSAM OPEN statement is successfully verified. This is compatible with pre-V5 COBOL runtime behavior.

### SUCC

If you specify VSAMOPENFS=SUCC, the statement returns the file status 00 when a VSAM OPEN statement is successfully verified. This allows users to simply check for 0 in the first digit of the returned file status, as they usually do with other successful operations.

---

## WORD

WORD indicates which alternate reserved-word table is to be used during compilation.

### Syntax



### Default

WORD=NO

### CICS

A CICS-specific word table, IGYCCICS, is provided as an alternate reserved word table. For a description, see "CICS reserved word table (IGYCCICS)" on page 13.

### xxxx

Specifies an alternative default reserved word table to be used during compilation. xxxx represents the ending characters (can be 1 to 4 characters in length) of the name of the reserved word table used. The first 4 characters are IGYC. The last 4 characters cannot be any one of the character strings listed below, nor can any of them contain the dollar sign character (\$).

- CBE
- DGEN
- DIAG
- DMAP
- DOPT
- ECWI

- FGEN
- INIT
- LIBO
- LIBR
- LSTR
- LVL0
- LVL1
- LVL2
- LVL3
- LVL8
- OSCN
- PGEN
- RCTL
- RDPR
- RDSC
- RWT
- SAW
- SCAN
- SIMD
- XREF

**NO** Indicates that no alternative reserved word table is to be used as the default.

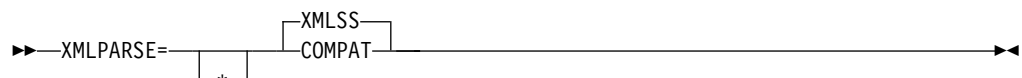
**Note:**

- Specification of WORD affects the interpretation of input reserved words. System names (such as UPSI and SYSPUNCH) and the intrinsic function names should not be used as aliases for reserved words. If a function name is specified as an alias through the reserved word table ABBR control-statement, that function name will be recognized and diagnosed by the compiler as a reserved word and the intrinsic function will not be performed.
- Changing the default value of the WORD=XXXX option conflicts with all values for FLAGSTD other than NO.

## XMLPARSE

XMLPARSE indicates which parser is to be used for processing XML input, and therefore which XML parsing capabilities are available.

### Syntax



### Default

XMLPARSE=XMLSS

### COMPAT

XML PARSE statements are processed using the XML parser that is a built-in component of the COBOL library. The XML PARSE statement results and

operational behaviors are then compatible with those obtained with Enterprise COBOL Version 3, and also with Version 4 when XMLPARSE(COMPAT) was used.

#### XMLSS

XML PARSE statements are processed using the z/OS XML System Services parser. The following XML parsing capabilities are available only when this suboption is in effect:

- Namespace processing enhancements
- The ENCODING, RETURNING NATIONAL, and VALIDATING phrases of the XML PARSE statement
- Support for direct parsing of XML documents encoded in UTF-8
- Support for parsing very large XML documents, a buffer of XML at a time
- Offloading of XML parsing to System z<sup>®</sup> Application Assist Processors (zAAPs)

---

## XREFOPT

XREFOPT sets the default value for the XREF compiler option, which affects the amount of cross-reference information produced in listings.

#### Syntax



#### Default

XREFOPT=FULL

#### FULL

Produces a sorted cross-reference of the names used in the program, and indicates the lines where the names are defined. Also produces a COPY/BASIS cross-reference. If SOURCE=YES is also specified, embedded cross-reference information is printed on the same lines as the source.

#### SHORT

Produces a sorted listing of only the explicitly referenced variables, and produces a COPY/BASIS cross-reference.

**NO** Suppresses the cross-reference listings.

#### Note:

- XREFOPT=NO conflicts with values of DBCSXREF other than NO.
- It is recommended that you not change the default to XREFOPT=NO. If XREFOPT=NO, the COPY/BASIS cross-reference might in some cases be incomplete or missing.

For further details, see XREF in the *Enterprise COBOL Programming Guide*.

---

## ZONECHECK

In Enterprise COBOL V5.2 with PTF for APAR PI81006 installed, ZONECHECK is deprecated and can no longer be specified in IGYCDOPT.

NUMCHECK=(ZON(ALPHNUM)) gives the same results as ZONECHECK used to.

For details, see “NUMCHECK” on page 45.

### RELATED REFERENCES

“NUMCHECK” on page 45

“NUMPROC” on page 47

“ZONEDATA”

---

## ZONEDATA

The ZONEDATA option tells the compiler whether the data in USAGE DISPLAY and PACKED-DECIMAL data items is valid, and if not, what the behavior of the compiler should be.

### Syntax



### Default

ZONEDATA=PFD

### PFD

When ZONEDATA=PFD is in effect, the compiler assumes that all data in USAGE DISPLAY and PACKED-DECIMAL data items is valid, and generates the most efficient code possible to make numeric comparisons. For example, the compiler might generate a string comparison to avoid numeric conversion.

### MIG

When ZONEDATA=MIG is in effect, the compiler generates instructions to do numeric comparisons that ignore the zone bits of each digit in zoned decimal data items. For example, the zoned decimal value is converted to packed-decimal with a pack instruction before the comparison. The compiler will also avoid performing known optimizations that might produce a different result than COBOL V4 (or earlier versions) when a zoned decimal or packed decimal data item has invalid digits or an invalid sign code, or when a zoned decimal data item has invalid zone bits.

### NOPFD

When ZONEDATA=NOPFD is in effect, the compiler generates instructions for numeric comparisons or an alphanumeric comparison of zoned decimal data in the same manner as COBOL V4 (or earlier versions) does when using NUMPROC=NOPFD|PFD with COBOL V4 (or earlier versions):

- In the cases where COBOL V4 (or earlier versions) considered the zone bits, the compiler generates an alphanumeric comparison which will also consider the zone bits of each digit in zoned decimal data items. The zoned decimal value remains as zoned decimal.
- In the cases where COBOL V4 ignored the zone bits, the compiler generates numeric comparisons that ignore the zone bits of each digit in zoned

decimal data items. The zoned decimal value is converted to packed-decimal with a PACK instruction before the comparison.

In order for the compiler to generate comparisons of zoned decimal data in the same way that COBOL V4 (or earlier versions) did, the NUMPROC suboption used in COBOL V5 must match the NUMPROC suboption used in COBOL V4 (or earlier versions):

- To get the COBOL V4 (or earlier versions) NUMPROC=NOPFD behavior in COBOL V5, use ZONEDATA=NOPFD and NUMPROC=NOPFD in COBOL V5.
- To get the COBOL V4 (or earlier versions) NUMPROC=PFD behavior in COBOL V5, use ZONEDATA=NOPFD and NUMPROC=PFD in COBOL V5.

The compiler will also avoid performing known optimizations that might produce a different result than COBOL V4 (or earlier versions) when a zoned decimal or packed decimal data item has invalid digits or an invalid sign code, or when a zoned decimal data item has invalid zone bits.

**Note:** The sign code must be a valid sign code according to the NUMPROC compiler option setting. In addition, the low-order byte must have a valid zone (x'F') for unsigned and signed with either SIGN IS LEADING or SIGN IS SEPARATE.

In all, to ease your migration to COBOL V5:

- If your digits, sign code, and zone bits are valid, use ZONEDATA=PFD and the same NUMPROC setting that you used with COBOL V4 when using COBOL V5.
- If you have invalid digits, invalid sign code, or invalid zone bits:
  - If you used NUMPROC=MIG with COBOL V4, use ZONEDATA=MIG and NUMPROC=NOPFD with COBOL V5.
  - If you used NUMPROC=NOPFD with COBOL V4, use ZONEDATA=NOPFD and NUMPROC=NOPFD with COBOL V5.
  - If you used NUMPROC=PFD with COBOL V4, use ZONEDATA=NOPFD and NUMPROC=PFD with COBOL V5.

**Note:** It is not always possible to entirely match the behaviour of the old compiler even with these options when faced with clearly invalid data. For example, even for compares, ZONEDATA(NOPFD) isn't going to give the same result in all cases as COBOL V4.

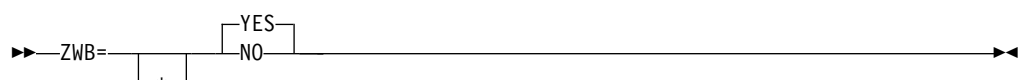
**Performance consideration:** ZONEDATA=PFD gives better runtime performance than ZONEDATA=NOPFD|MIG does. ZONEDATA=NOPFD|MIG disables some of the optimizations that NUMPROC=PFD can give.

---

## ZWB

ZWB determines whether the compiler removes the sign from signed zoned decimal fields before they are compared to alphanumeric fields at run time.

### Syntax



### Default

ZWB=YES

**YES**

Removes the sign from a signed external decimal (DISPLAY) field when comparing this field to an alphanumeric field at run time.

**NO** Does not remove the sign from a signed external decimal (DISPLAY) field when comparing this field to an alphanumeric field at run time.

**Note:**

- Setting this option affects program runtime logic; that is, the same COBOL source program can give different results, depending on the option setting. Verify whether your Enterprise COBOL source programs assume a particular setting to run correctly.
- Application programmers use ZWB=NO to test input numeric fields for SPACES.





---

## Chapter 3. Customizing Enterprise COBOL

The following sections describe the supplied user jobs and modules that you can modify to customize Enterprise COBOL.

You can make modifications to Enterprise COBOL only after installation of the product is complete.

One of the modifications is made using an SMP/E USERMOD. If you do not ACCEPT Enterprise COBOL into the distribution libraries before applying the USERMOD, you will not be able to use the SMP/E RESTORE statement to remove your USERMOD. Do not accept your USERMOD into the distribution libraries. You might want to remove your USERMOD if you find that it does not suit the needs of the programmers at your site.

You will have to remove your USERMOD before applying service to the modules that it changes. In this case, you will probably want to reapply your USERMOD after successful installation of the service.

**Important:** Make sure that Enterprise COBOL serves the needs of the application programmers at your site. Confer with them while you plan the customization of Enterprise COBOL. Doing so will ensure that the modifications you make at install time best support the application programs being developed at your site.

All information for installing Enterprise COBOL is included in the *Program Directory* provided with the product.

---

### Summary of user modifications

Installation of Enterprise COBOL places a number of sample modification jobs in the target data set IGY.V6R2M0.SIGYSAMP. However, these sample modification jobs are not customized for your particular system. You must customize them.

Table 8 shows the names of the sample modification jobs, which are described in detail in the referenced information.

Copy members from IGY.V6R2M0.SIGYSAMP into one of your personal data sets before you modify and submit them so that you have an unmodified backup copy if you make changes that you want to abandon.

Descriptions of possible modifications appear in the comments in the JCL. You can use TSO to modify and submit the job.

*Table 8. Summary of user modification jobs for Enterprise COBOL*

Description	Job	See:
Changing the compiler options default module	IGYWDOPT	"Changing the compiler options default module" on page 75
Overriding options specified as fixed	IGYWUOPT	"Overriding options specified as fixed" on page 75
Changing reserved words	IGYWRWD	"Changing reserved words" on page 76

Table 8. Summary of user modification jobs for Enterprise COBOL (continued)

Description	Job	See:
Place Enterprise COBOL modules in shared storage	No sample job installed	“Placing Enterprise COBOL modules in shared storage” on page 81

## Changing the defaults for compiler options

You can change the defaults for compiler options, and can override fixed compiler options.

To change the defaults for compiler options:

1. Copy the source of options module IGYCDOPT from IGY.V6R2M0.SIGYSAMP into the appropriate job in place of the two-line comment.
2. Change the parameters on the IGYCOPT macro call to match the compiler options that you have selected for your installation.

Observe the following requirements when coding the changed IGYCOPT macro call:

- Place continuation character (X in the source) in column 72 on each line of the IGYCOPT invocation except the last line. The continuation line must start in column 16. You can break the coding after any comma.
- Do not put a comma in front of the first option in the macro.
- Specify options and suboptions in uppercase. Only suboptions that are strings can be specified in mixed case or lowercase. For example, either LVLINF0=(Fix1) or LVLINF0=(FIX1) is acceptable.
- 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 apostrophes ('), not in quotation marks ("). A null string can be specified with either contiguous apostrophes or quotation marks.

To obtain an apostrophe (') or a single ampersand (&) within a string, two contiguous instances of the character must be specified. 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.

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

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

This message bears no spatial relationship to the offending suboption. Furthermore, none of the options is properly parsed if this error is committed.

- Code only those options whose default value you want to change. The IGYCOPT macro generates the default value for any option that you do not code.

For a worksheet to help you plan your default compiler options, see “IGYCDOPT worksheet for compiler options” on page 4. For descriptions of the options, see Chapter 2, “Enterprise COBOL compiler options,” on page 15.

- Place an END statement after the macro instruction.

For additional details about how to code macro calls, see the *High Level Assembler for z/OS Language Reference*.

## Changing the compiler options default module

To change the defaults for the Enterprise COBOL compiler options, modify the sample job IGYWDOPT.

To choose default values, use the information in Chapter 2, “Enterprise COBOL compiler options,” on page 15.

If you coded OUT as the value for any compiler phase options, be sure to place these phases in shared storage before compiling a program by using your new compiler options default module. For more information, see “Compiler phases and their defaults” on page 8 and “Placing Enterprise COBOL modules in shared storage” on page 81.

### To modify the JCL for IGYWDOPT, do these steps:

1. Add a job card appropriate for your site.
2. Add a JES ROUTE card if required for your site.
3. Replace the two comment lines in IGYWDOPT with a copy of the source for IGYCDOPT found in IGY.V6R2M0.SIGYSAMP.
4. Code parameters on the IGYCOPT macro statement in IGYCDOPT to reflect the values you have chosen for your installation-wide default compiler options.
5. Change #GLOBALCSI to the global CSI name.
6. Change #TZONE in the SET BDY statement to the target zone name.

After you modify the IGYWDOPT job, submit it. You will get a condition code of 0 if the job runs correctly. Also check the IGYnnnn informational messages in your listing to verify the defaults that will be in effect for your installation.

## Overriding options specified as fixed

Occasionally, you might have an application that needs to override one or more options that were specified as fixed.

You can provide other options by creating a temporary copy of the options module in a separate data set that can be accessed as a STEPLIB or JOBLIB (ahead of the IGY.V6R2M0.SIGYCOMP data set) when the application is compiled.

Sample job IGYWUOPT creates a default options module that is link-edited into a user-specified data set. The assembly and link-editing take place outside SMP/E control, so the standard default options module is not disturbed.

### To modify the JCL for IGYWUOPT, do these steps:

1. Add a job card appropriate for your site.
2. Add a JES ROUTE card if required for your site.
3. Replace the two comment lines in IGYWUOPT with a copy of the source for IGYCDOPT found in IGY.V6R2M0.SIGYSAMP.
4. Change the parameters on the IGYCOPT macro statement in IGYWUOPT to reflect the values that you have chosen for this fixed option override compiler-options module.
5. If you chose to use a different prefix than the IBM-supplied one for the Enterprise COBOL target data sets, check the SYSLIB DD statement (marked with '<<<<') to ensure that the data set names are correct.

6. Change DSN=YOURLIB in the SYSLMOD DD statement to the name of the data set that you want your IGYCDOPT module bound into. Note that an IGYCDOPT module currently in the chosen data set will be replaced by the new version.

After you modify the IGYWUOPT job, submit it. Both steps return a condition code of 0 if the job runs successfully. Also check the IGYnnnn informational messages in your listing to verify the defaults that are in effect when this module is used in place of the standard default options module.

---

## Changing reserved words

To change the words that Enterprise COBOL treats as reserved, use the reserved word table utility.

The reserved words used by Enterprise COBOL are maintained in a table (IGYCRWT) provided with the product. A CICS-specific reserved word table (IGYCCICS) is provided as an alternate reserved word table (see “CICS reserved word table (IGYCCICS)” on page 13). You can change the reserved words by using the reserved word table utility (IGY8RWTU) to modify IGYCRWT or IGYCCICS, or by creating additional reserved word tables. You can also modify tables that you previously created.

The reserved word table utility accepts control statements to create or modify a reserved word table. The new table then contains the reserved words from the IBM-supplied table with all the changes that you have applied.

You can make the following types of changes to reserved word tables:

- Add words to be flagged with an informational message whenever they are used in a program. To produce these information messages, you must modify the IGYCRWT reserved word table and compile using the FLAGSTD option.
- Add words to be flagged with a severe error message whenever they are used in a program.
- Indicate that words currently flagged with an informational or error message should no longer be flagged.

Each reserved word table that you create must have a unique 1- to 4-character identifier. For a list of character strings that cannot be used, see “WORD” on page 66.

At compile time, the value of the compiler option WORD(xxxx) identifies the reserved word table to be used. xxxx is the unique 1- to 4-character identification that you specified in the member name IGYCxxxx. You can create multiple reserved word tables, but only one can be specified at compile time.

**Note:** The total number of entries in a reserved word table should not exceed 1536 or 1.5 KB.

Because of the following example, when the IBM extension reserved word ENTRY is used in a program, it will be flagged with message 0086.

```
INFO ENTRY
```

The following example restricts the use of Boolean, XD, and PARENT. Use of these will cause errors.

```
RSTR BOOLEAN
      XD
      PARENT
```

The following example restricts the use of GO TO and ALTER. Use of these will cause errors.

```
RSTR GO
      ALTER
```

In the following example, the reserved word table generated allows usage of all the 85 COBOL Standard language except nested programs.

```
RSTR IDENTIFICATION(1)  only allow 1 program per compilation unit
RSTR ID(1)              same for the short form
RSTR PROGRAM-ID(1)      only allow 1 program per compilation unit
RSTR GLOBAL             do not allow this phrase at all
```

## Creating or modifying a reserved word table

You can create a reserved word table, or modify one of the existing reserved word tables.

Edit a copy of the appropriate source file:

- Member IGY8RWRD in IGY.V6R2M0.SIGYSAMP (the IBM-supplied default reserved word table)
- Member IGY8CICS in IGY.V6R2M0.SIGYSAMP (the IBM-supplied CICS reserved word table)
- A user file (user-supplied reserved word table)

You must also modify and invoke the appropriate non-SMP/E JCL.

Your file should have four parts: Parts I, II, III, and IV. Modify the file and non-SMP/E JCL as follows:

1. Make a private copy of the file.
2. Skip Part I (all lines up to and including the line with the keyword MOD). Make *no* alterations in this part of the file!
3. Edit Part II by placing asterisks in column 1 of the lines that contain reserved words for which you do not want informational messages issued.
4. Edit Part III by placing asterisks in column 1 of the lines that contain reserved words for which you do not want severe messages issued.
5. Edit Part IV by coding additional reserved word control statements that create the modifications that you want, as described in “Coding control statements.”
6. Modify and run the JCL, as described in “Running JCL that creates a reserved word table” on page 81. You also must create a unique 1- to 4-character identification for the new reserved word table and supply it in the JCL.

## Coding control statements

To create or modify a reserved word table, you must understand the syntax of the control statements that affect them.

The following figure illustrates the format for coding reserved word control statements.

---

```

ABBR  reserved-word: user-word [comments]
      [reserved-word: user-word [comments]]
      ?
INFO  COBOL-word [(0 | 1)] [comments]
      [COBOL-word [(0 | 1)] [comments]]
      ?
RSTR  COBOL-word [(0 | 1)] [comments]
      [COBOL-word [(0 | 1)] [comments]]
      ?

```

---

*Figure 2. Syntax format for reserved word control statements*

As shown in the preceding figure, the keywords you can use are:

- ABBR** Specifies an alternative form of an existing reserved word
- INFO** Specifies words that are to be flagged with an informational message whenever they are used in a program and the FLAGSTD compiler option is in effect
- RSTR** Specifies words that are to be flagged with an error message whenever they are used in a program

All words that you identify with the control statement keywords INFO and RSTR are flagged with a message in the source listing of the Enterprise COBOL program that uses them. Words that are abbreviated are not flagged in the source listing unless you have also specified them on the INFO or RSTR control statements.

## Rules for coding control statements

You need to follow the rules when you code your control statements.

These rules are:

- Begin the control statement in column 1.
- Place one or more spaces between the keyword and the first operand.
- When specifying a second operand, include a colon (:) and one or more spaces after the first operand.
- Continue a control statement by putting blanks in columns 1 through 5, followed by the operand or operands, to make additional specifications.
- Specify comments by putting an asterisk (\*) in column 1 of the control statements. You can also place comments on the same line as the control statement. In that case, however, there must be at least one space following the operand or operands before a comment begins.
- To specify more than one change within a single control statement, put each additional specification on a separate line.
- Do not add any blank lines.

## Coding operands in control statements

This topic shows the types of operands that you will be coding in the control statements.

### **reserved-word**

An existing reserved word.

### **user-word**

A user-defined COBOL word that is not a reserved word.

**comments**

Any comments that you want to put on the same line with the control statement, or on a separate line that has an asterisk in column 1.

**COBOL-word**

A word of up to 30 characters that can be a system name, a reserved word, or a user-defined word.

## Rules for coding control statement operands

Follow the rules when you code the control statement operands.

These rules are:

- A user-word can be used in only one ABBR statement in any particular reserved word table.
- A reserved-word specified in an ABBR statement can also be specified in either a RSTR or an INFO statement.
- A particular reserved-word can be specified only once in an ABBR statement.
- A particular COBOL-word can be specified only once in either a RSTR or an INFO statement.

## ABBR statement

The ABBR statement defines an alternative symbol for the reserved word specified. The symbol can be used when you code a program.

```
ABBR reserved-word: user-word [comments]
```

**Note:**

- The user-word becomes a reserved word and can be used in place of the reserved-word specified in this statement.
- The reserved-word remains a reserved word with its original definition.
- The source listing shows the original source—the symbol as you coded it.
- The reserved word is used in compiler output—other listings, some messages, and so forth.

In the following example, REDEF and SEP become abbreviations that can be used in source programs. The appropriate reaction to the use of REDEFINES and SEPARATE takes place when the source program is compiled.

```
ABBR REDIFINES: REDEF
SEPARATE: SEP
```

## INFO statement

The INFO statement specifies the COBOL words that are to be flagged by the compiler, and it can also be used to control the use of nested programs.

```
INFO COBOL-word[(0 | 1)] [comments]
```

By selecting either 1 or 0, you can indicate whether a specific COBOL-word can be used only once, or not at all.

- |          |   |
|----------|---|
| <b>0</b> | Indicates that whenever the specified COBOL-word is used, the 0086 informational message is issued if the FLAGSTD compiler option is in effect. |
| <b>1</b> | Indicates that the specified COBOL-word can be used once. If it is used more than once, informational message 0195 is issued.                   |

The messages are handled as information (I) messages. The compilation condition is not changed.

## RSTR statement

The RSTR statement specifies COBOL words that cannot be used in a program, and it can also be used to control the use of nested programs.

**RSTR COBOL-word[(0 | 1)] [comments]**

By selecting either 1 or 0, you can indicate whether a specific COBOL-word can be used only once, or not at all.

**0** Indicates that whenever the specified COBOL-word is used, message 0084 is issued.

**1** Indicates that the specified COBOL-word can be used once. If it is used more than once, severe message 0194 is issued.

The following reserved words can be restricted only by using option 1:

IDENTIFICATION  
FD  
ENVIRONMENT  
DATA  
WORKING-STORAGE  
PROCEDURE  
DIVISION  
SECTION  
PROGRAM-ID

## Modifying and running non-SMP/E JCL

Use sample job IGYWRWD in IGY.V6R2M0.SIGYSAMP to create or modify a reserved word table.

The sample job uses a member based on IGY8RWRD or IGY8CICS in IGY.V6R2M0.SIGYSAMP as the input to the reserve word utility. It creates program object IGYCxxxx (where xxxx is the user identification) in IGY.V6R2M0.SIGYCOMP.

To modify the JCL for IGYWRWD, do these steps:

1. Modify the job statement for your site.
2. Add JES ROUTE records if required.
3. Change the data set name on the STEPLIB DD statement in STEP1 to match the compiler target data set name you used during installation.
4. To point to your modified reserved word table, do *only* one of the following steps:
  - Change the data set name in //RSWDREAD DD DSN=... to the data set name and member name of your modified reserved word table.
  - Replace the RSWDREAD DD with //RSWDREAD DD \* and insert your modified reserved table immediately following that line.

For specific instructions, see the comments in job IGYWRWD.

5. Change the name of the data set in the SYSLMOD DD statement in STEP3 to match the name of the data set to which you are adding your modified reserved word table. (The data set name in the SYSLMOD DD statement should be the name of the compiler target data set.) Also, you must specify the name of your modified reserved word table in the parentheses that follow the data set name in the SYSLMOD DD statement.



After you run IGYWRWD, if you receive a nonzero return code from the table utility, use the error messages in the output data set specified on the RSWDPRNT DD statement to correct any mistakes and rerun the job.

## Running JCL that creates a reserved word table

The JCL that creates a reserved word table contains STEP1, STEP2, and STEP3.

The three steps do the following tasks, respectively:

1. Run the reserved word table utility with your modified table.
2. Assemble your modified reserved word table.
3. Produce a runtime program object from the object module.

After you run the job, a reserved word table is created, the library that you specified contains the new table, and the table has IGYC plus the 1- to 4-character identification that you specified.

---

## Placing Enterprise COBOL modules in shared storage

All of the modules in IGY.V6R2M0.SIGYCOMP that are reentrant can be included in shared storage.

To include the modules in shared storage, you can:

- Authorize the data set IGY.V6R2M0.SIGYCOMP
- (Optional) Include IGY.V6R2M0.SIGYCOMP in the LNKSTnn concatenation
- Add compiler modules to LPA dynamically after the system is IPLed

Under z/OS, you do not need to place IGY.V6R2M0.SIGYCOMP in the LNKSTnn concatenation to be able to load program objects into the LPA. If you choose not to add it to the LNKSTnn concatenation, you must make the modules that are not included in the LPA available to steps that compile Enterprise COBOL applications by doing *one* of these steps:

- Copying the non-LPA modules to a data set that is in the LNKSTnn concatenation
- Copying the non-LPA modules to a data set that can be used as a STEPLIB or JOBLIB

Using the entire IGY.V6R2M0.SIGYCOMP data set as a STEPLIB or JOBLIB defeats the purpose of placing the modules in the LPA because modules are loaded from a STEPLIB or JOBLIB before the LPA is searched.

Modules that you copy into another data set are not serviced automatically by SMP/E in that data set. You must rerun your copy job after you apply service to Enterprise COBOL to make the updated modules available in the LNKSTnn data set or in the STEPLIB.

For more information about including modules in the LPA, see these documents:

- *z/OS MVS Initialization and Tuning Guide*
- *z/OS MVS Initialization and Tuning Reference*

If you are placing compiler phases in shared storage, code the corresponding phase options with the value OUT when you run the sample job IGYWDOPT to change the compiler options defaults. For more information, see “Changing the defaults for compiler options” on page 74.

---

## Tailoring the cataloged procedures to your site

You might want to tailor the cataloged procedures IGYWC, IGYWCL, or IGYWCLG for use at your site.

Consider these changes:

- Modifying the data set name prefixes if you used a different prefix than the IBM-supplied ones for Enterprise COBOL or Language Environment target data sets
- Removing the STEPLIB DD statements if you placed IGY.V6R2M0.SIGYCOMP, CEE.SCEERUN, and CEE.SCEERUN2 in the LNKST concatenation
- Changing the default region size for the GO steps if most of the programs at your site require a larger region for successful execution
- Changing the UNIT=parameter

---

## 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 grant 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  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

**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  
J46A/G4  
555 Bailey Avenue  
San Jose, CA 95141-1003  
U.S.A.

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 measurements 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 or withdrawal without notice, and represent goals and objectives only.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. \_enter the year or years\_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Programming interface information

IBM Enterprise COBOL for z/OS provides no macros that allow a customer installation to write programs that use the services of IBM Enterprise COBOL for z/OS.

**Attention:** Do not use as programming interfaces any IBM Enterprise COBOL for z/OS macros.

---

## Trademarks

IBM, the IBM logo, and [ibm.com](http://ibm.com)<sup>®</sup> are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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



---

## List of resources

---

### Enterprise COBOL for z/OS COBOL for z/OS publications

You can find the following publications in the Enterprise COBOL for z/OS library:

- *Customization Guide*, SC14-7380
- *Language Reference*, SC14-7381
- *Programming Guide*, SC14-7382
- *Migration Guide*, GC14-7383
- *Program Directory*, GI11-9180
- *Licensed Program Specifications*, GI11-9181

### Softcopy publications

The following collection kits contain Enterprise COBOL and other product publications. You can find them at <http://www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss>.

- *z/OS Software Products Collection*
- *z/OS and Software Products DVD Collection*

---

### Related publications

#### z/OS library publications

You can find the following publications in the z/OS Internet Library.

#### Run-Time Library Extensions

- *DWARF/ELF Extensions Library Reference*
- *Common Debug Architecture Library Reference*
- *Common Debug Architecture User's Guide*

#### z/OS ISPF

- *User's Guide Vol I*

#### z/OS Language Environment

- *Concepts Guide*
- *Customization*
- *Debugging Guide*
- *Programming Guide*
- *Programming Reference*
- *Run-Time Messages*
- *Run-Time Application Migration Guide*
- *Writing Interlanguage Communication Applications*

#### z/OS MVS

- *Initialization and Tuning Guide*
- *Initialization and Tuning Reference*

#### z/OS SMP/E

- *User's Guide*
- *Reference*

#### z/OS TSO/E

- *Primer*
- *User's Guide*

### CICS Transaction Server for z/OS

You can find the following publications in the CICS Library:

- *Application Programming Guide*
- *Application Programming Reference*
- *Customization Guide*
- *External Interfaces Guide*

### DB2 for z/OS

You can find the following publications in the DB2 Library:

- *Installation and Migration Guide*
- *Internationalization Guide*

### Debug Tool

You can find the following publications in the Debug Tool Library:

- *Reference and Messages*
- *User's Guide*

### Other publications

- *American National Standard ANSI INCITS 23-1985, Programming Languages - COBOL*, as amended by *ANSI INCITS 23a-1989, Programming Languages - Intrinsic Function Module for COBOL*, and *ANSI INCITS 23b-1993, Programming Language - Correction Amendment for COBOL*
- *International Standard ISO 1989:1985, Programming languages - COBOL*, as amended by *ISO/IEC 1989/AMD1:1992, Programming languages - COBOL: Intrinsic function module*,

and ISO/IEC 1989/AMD2:1994, *Programming languages - Correction and clarification amendment for COBOL*

- *DBCS Ordering Support Program (DBCSOS)*, SH88-0171 (N:SH18-0144) (in Japanese)



---

# Index

## A

accessibility  
  of Enterprise COBOL xii  
  of this information xiii  
  using z/OS xii  
ADATA compiler option 17  
ADEXIT compiler option 17  
ADV compiler option 18  
AFP compiler option 18  
ALOWCBL compiler option 19  
ARCH compiler option 19  
ARITH compiler option 21  
assistive technologies xiii  
asterisk (\*) for indicating fixed compiler options 17  
AWO compiler option 21

## B

BLOCK0 compiler option 22  
BUF compiler option 22

## C

CBL statement 19  
CICS reserved word table 13  
COMPILE compiler option 24  
compiler options  
  conflicting options 15  
  default values 1  
  description of  
    ADATA 17  
    ADEXIT 17  
    ADV 18  
    AFP 18  
    ALOWCBL 19  
    ARCH 19  
    ARITH 21  
    AWO 21  
    BLOCK0 22  
    BUF 22  
    COMPILE 24  
    COPYRIGHT 25  
    CURRENCY 25  
    DATA 26  
    DBCS 27  
    DBCSXREF 27  
    DECK 28  
    DIAGTRUNC 29  
    DISPSIGN 29  
    DLL 30  
    DYNAM 31  
    EXPORT 31  
    FASTSRT 32  
    FLAG 32  
    FLAGSTD 33  
    HGPR 35  
    INEXIT 36  
    INITCHECK 36  
    INTDATE 37

compiler options (*continued*)  
  description of (*continued*)  
    LANGUAGE 38  
    LIBEXIT 39  
    LINECNT 39  
    LIST 39  
    LITCHAR 40  
    LVINFO 40  
    MAP 41  
    MAXPCF 41  
    MDECK 42  
    MSGEXIT 43  
    NAME 43  
    NUM 44  
    NUMCHECK 45  
    NUMCLS 47  
    NUMPROC 47  
    OBJECT 48  
    OFFSET 49  
    OPTIMIZE 49  
    OUTDD 50  
    PGMNAME 50  
    PRTEXIT 51  
    QUALIFY 51  
    RENT 52  
    RMODE 53  
    RULES 54  
    SEQ 55  
    SERVICE 55  
    SOURCE 56  
    SPACE 56  
    SQL 57  
    SQLIMS 58  
    SQLSSCID 57  
    SSRANGE 58  
    STGOPT 60  
    TERM 60  
    TEST 61  
    THREAD 62  
    TRUNC 63  
    VBREF 64  
    VLR 65  
    VSAMOPENFS 66  
    WORD 66  
    XMLPARSE 67  
    XREFOPT 68  
    ZONECHECK 69  
    ZONEDATA 69  
    ZWB 70  
  fixed  
    indicate with asterisk (\*) 17  
    making compiler options fixed 2  
  modifying 4, 74  
  planning worksheet 4  
  setting defaults for 1, 74  
compiler phases  
  defaults for 8  
  description of  
    DGEN 8  
    DIAG 8  
    DMAP 9

compiler phases (*continued*)  
  description of (*continued*)  
    FGEN 9  
    INIT 9  
    LIBR 9  
    LSTR 9  
    MSGT 9  
    OPTM 10  
    PGEN 10  
    RCTL 10  
    RWT 10  
    SCAN 10  
    SIMD 10  
    XREF 11  
  fixed phases 7  
  INOUT parameters 8  
  macro worksheet 11  
  modifying 4  
  placing in shared storage 7  
COPYRIGHT compiler option 25  
CURRENCY compiler option 25  
customization  
  compiler options 15, 74  
  installation jobs  
    Enterprise COBOL 73  
  planning for 1

## D

DATA compiler option 26  
DBCS compiler option 27  
DBCSXREF compiler option 27  
Debug Tool 61  
DECK compiler option 28  
default reserved word table 13  
default values  
  compiler options 1  
  compiler phases 8  
DGEN phase 8  
DIAG phase 8  
DIAGTRUNC compiler option 29  
DISPSIGN compiler option 29  
DLL compiler option 30  
DMAP phase 9  
DYNAM compiler option 31

## E

Enterprise COBOL  
  disable 14  
  enable 14  
  job modification 73  
error messages  
  flagging 33  
EXPORT compiler option 31

## F

FASTSRT option 32  
FGEN phase 9

fixed compiler options  
    indicate with asterisk (\*) 17  
    making compiler options fixed 2  
FLAG compiler option 32  
FLAGSTD compiler option 33

## H

HGPR compiler option 35

## I

IGYCCICS (CICS reserved word table) 13  
IGYCDOPT  
    link AMODE 31 and RMODE ANY 1  
    planning worksheet 4  
IGYCOPT  
    syntax format 4  
IGYCRWT (default reserved word table) 13  
index checking 59  
INEXIT compiler option 36  
INIT phase 9  
INITCHECK compiler option 36  
INTDATE compiler option 37

## K

keyboard navigation xiii  
keywords x

## L

LANGUAGE compiler option 38  
LIBEXIT compiler option 39  
LIBR phase 9  
LINECNT compiler option 39  
LIST compiler option 39  
LITCHAR compiler option 40  
LSTR phase 9  
LVINFO compiler option 40

## M

macros  
    IGYCDOPT (compiler options)  
        planning worksheet 4  
        syntax format 4  
    IGYCDOPT (compiler phases)  
        planning worksheet 11  
        syntax format 4  
MAP compiler option 41  
MAXPCF compiler option 41  
MDECK compiler option 42  
messages, flagging 33  
MSGEXIT compiler option 43  
MSGT phase 9

## N

NAME compiler option 43  
nested programs 13

nonoverridable compiler options, indicate  
    with asterisk (\*) 17  
NUM compiler option 44  
NUMCHECK compiler option 45  
NUMCLS compiler option 47  
NUMPROC compiler option 47

## O

object code, reentrant 52  
OBJECT compiler option 48  
OFFSET compiler option 49  
OPTIMIZE compiler option 49  
optional words ix  
OPTM phase 10  
OUTDD compiler option 50

## P

PGEN phase 10  
PGMNAME compiler option 50  
phases, compiler  
    defaults for 8  
    macro worksheet 11  
    modifying 4  
    placing in shared storage 7  
planning worksheets  
    description of x  
    IGYCDOPT (compiler options) 4  
    IGYCDOPT (compiler phases) 11  
preface ix  
PROCESS (CBL) statement 19  
product registration 14  
PRTEXTIT compiler option 51  
publications 87

## Q

QUALIFY compiler option 51

## R

RCTL phase 10  
reentrant object code 52  
RENT compiler option 52  
required words ix  
reserved word table  
    contents of 13  
    creating or modifying 76  
    nested programs 13  
    planning for 12  
    specifying an alternative table 66  
    supplied with Enterprise COBOL  
        IGYCCICS (CICS) 13  
        IGYCRWT (default) 13  
residency mode 53  
RMODE compiler option 53  
RULES compiler option 54  
RWT phase 10

## S

sample installation jobs 3  
SCAN phase 10  
sequence checking of line numbers 55

SEQUENCE compiler option 55  
SERVICE compiler option 55  
shared storage  
    compiler phases in 6, 8  
    placing Enterprise COBOL modules  
        in 81, 82  
    planning for 6  
SIMD phase 10  
SOURCE compiler option 56  
SPACE compiler option 56  
SQL compiler option 57  
SQLCCSID compiler option 57  
SQLIMS compiler option 58  
SSRANGE compiler option 58  
stacked words ix  
STGOPT compiler option 60  
subscript checking 59  
syntax checking 24  
syntax diagrams, how to read ix  
syntax notation  
    asterisk (\*) 17  
    COBOL keywords x  
    repeat arrows x  
SYSLIN 48  
SYSOUT 50  
SYSPUNCH 28  
SYSTEM 60

## T

TERM compiler option 60  
TEST compiler option 61  
THREAD compiler option 62  
TRUNC compiler option 63

## U

user exit routine  
    ADEXIT compiler option 17  
    INEXIT compiler option 36  
    LIBEXIT compiler option 39  
    MSGEXIT compiler option 43  
    PRTEXTIT compiler option 51

## V

VBREF compiler option 64  
VLR compiler option 65  
VSAMOPENFS compiler option 66

## W

WORD compiler option 66

## X

XMLPARSE compiler option 67  
XREF compiler option 68  
XREF phase 11  
XREFOPT option 68

## **Z**

ZONECHECK compiler option 69

ZONEDATA compiler option 69

ZWB compiler option 70







Product Number: 5655-W32

Printed in USA

SC14-7380-03

