Enterprise COBOL for z/OS

Customization Guide

Version 4 Release 2

SC23-8526-01

Enterprise COBOL for z/OS

Customization Guide

Version 4 Release 2

SC23-8526-01

Note!
Before using this information and the product it supports, read the information in "Notices," on page 65.

Second Edition (August 2009)

This edition applies to Version 4 Release 2 of IBM Enterprise COBOL for z/OS (program number 5655-S71) 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 order publications online at www.ibm.com/shop/publications/order/, or order by phone or fax. IBM Software Manufacturing Solutions takes publication orders between 8:30 a.m. and 7:00 p.m. Eastern Standard Time (EST). The phone number is (800)879-2755. The fax number is (800)445-9269.

You can also order publications through your IBM representative or the IBM branch office that serves your locality.

Contents

Figures

Tables

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

Use the following description to read the syntax diagrams in this information.

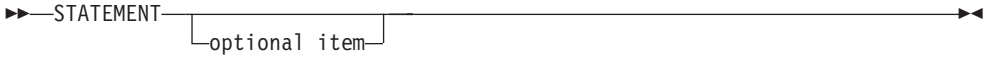
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following table shows the meaning of symbols at the beginning and end of syntax diagram lines.

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).

- Optional items appear below the main path.



- 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.

  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 15.

Using the macro planning worksheets

The planning worksheets in this information ("IGYCDOPT worksheet for compiler options" on page 3 and "IGYCDOPT worksheet for compiler phases" on page 10) 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 major changes that have been made to this edition, and to Enterprise COBOL in Version 4. The latest technical changes are marked by a vertical bar (|) in the left margin in the PDF and BookManager® versions.

Changes to this edition

This edition contains these additions and changes:

- Documentation for new options BLOCK0 and MSGEXIT
- Clarification to the descriptions of the ADEXIT, INEXIT, LIBEXIT, and PRTEXIT options
- Updates to the descriptions of the PGMNAME, TRUNC, WORD, XMLPARSE, and XREFOPT options
- Miscellaneous corrections

Major changes to Enterprise COBOL

Version 4 Release 2 (August 2009)

- New and enhanced XML PARSE capabilities are available when you use the z/OS XML System Services parser:
  - You can parse documents with validation against an XML schema when you use the VALIDATING phrase of the XML PARSE statement.
  - The performance of nonvalidating parsing is improved.
  - Character processing is enhanced for any XML document that contains a reference to a character that is not included in the single-byte EBCDIC code page of the document.

    For further details, see the *Enterprise COBOL Compiler and Runtime Migration Guide*.
- A facility for customizing compiler diagnostic messages and FIPS messages by changing their severity or suppressing them is made possible by a new suboption, MSGEXIT, of the EXIT compiler option.
- A new compiler option, BLOCK0, activates an implicit BLOCK CONTAINS 0 clause for all eligible QSAM files in your program.
- The underscore character (_) is now supported in user-defined words such as data-names and program-names. Underscores are also supported in the literal form of program-names.

  For further details, see the *Enterprise COBOL Language Reference*.
- If you use the integrated CICS translator, the compiler listing will now show the CICS options that are in effect.
- Enterprise COBOL applications that use object-oriented syntax for Java™ interoperability are now supported with Java 5 and Java 6 in addition to the Java SDK 1.4.2.

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

Version 4 Release 1 (December 2007)

- The performance of operations on Unicode (USAGE NATIONAL) data has been significantly improved. The compiler now generates z/Architecture® hardware instructions for most Unicode MOVE operations and comparisons.
- A new compiler option, XMLPARSE, makes it possible to choose between parsing with the parser that is available with the COBOL library (for compatibility with Enterprise COBOL Version 3) or with the z/OS XML System Services parser.
- New XML PARSE capabilities are available when you parse a document with the z/OS XML System Services parser:
  - Namespaces and namespace prefixes are processed using new special registers and new XML events.
  - You can specify the document encoding using the ENCODING phrase.
  - You can parse documents that are encoded in Unicode UTF-8.
  - The RETURNING NATIONAL phrase enables you to receive XML document fragments in Unicode regardless of the original encoding of an XML document.
  - You can parse very large documents a buffer at a time.
- The XML GENERATE statement has been enhanced:
  - You can specify a namespace using the NAMESPACE phrase, and a namespace prefix to be applied to each element using the NAMESPACE-PREFIX phrase.
  - You can specify the code page of the generated document using the ENCODING phrase.
  - XML documents can now be generated in UTF-8 as well as in UTF-16 or various EBCDIC code pages.
  - The WITH ATTRIBUTES phrase causes eligible elementary items to be expressed as attributes rather than as child elements in the generated XML.
  - The WITH XML-DECLARATION phrase causes an XML declaration to be generated.
- A new compiler option, OPTFILE, enables the specifying of COBOL compiler options from within a data set.
- Compiler listings now cross-reference COPY statements and the data sets from which copybooks are obtained.
- Support for new features of DB2® for z/OS V9 is enabled when you use the integrated DB2 coprocessor (SQL compiler option):
  - New SQL data types are supported: XML types, BINARY, VARBINARY, BIGINT, and file reference variables.
  - New SQL syntax for XML manipulation, enhancements to large object manipulation, MERGE, and SELECT FROM MERGE is supported.
  - DB2 processing options STDSQL(YES|NO), NOFOR, and SQL(ALL|DB2) are supported as suboptions to the SQL compiler option.
- Several usability enhancements to COBOL-DB2 applications are available when you use the integrated DB2 coprocessor:
  - The compiler listing is enhanced to show the DB2 options in effect (if you use DB2 for z/OS V9) and to show the expansion of the SQLCA and SQLDA control blocks.
  - You can specify an alternate ddname for DBRMLIB when you invoke the compiler from an assembler language program.

  - An explicitly coded LOCAL-STORAGE SECTION or WORKING-STORAGE SECTION is no longer required.
- Debugging has been enhanced to support Debug Tool V8. A new suboption of the TEST compiler option, EJPD, enables the Debug Tool commands JUMPTO and GOTO for production debugging. The TEST compiler option has been simplified and has restructured suboptions.

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:

- Fill out the Readers' Comment Form and return it by mail or give it to an IBM representative. If there is no Readers' Comments Form at the back, address your comments to:

  IBM Corporation
  Reader Comments
  DTX/E269
  555 Bailey Avenue
  San Jose, CA 95141-1003
  USA
- Use the Online Readers' Comments Form at www.ibm.com/software/awdtools/rcf/.
- Send your comments to the following e-mail address: comments@us.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 System z Enterprise Development Tools & Compilers Information Center at publib.boulder.ibm.com/infocenter/pdthelp/index.jsp 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 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 5
- "Planning to create an additional reserved word table" on page 10

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 55.

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 "Why make 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 5.
- Create additional reserved word tables: see "Planning to create an additional reserved word table" on page 10.

Planning to modify compiler option default values

Compiler option defaults and residency for compiler phases are set in the IGYCDOPT program as shown in Table 1 on page 3 and Table 2 on page 10. The default options module, IGYCDOPT, is link-edited with AMODE (31) and RMODE (ANY) during installation.

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.V4R2M0.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.V4R2M0 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 5.

Why make compiler options fixed?

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 can 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, if you use a JOBLIB or STEPLIB containing the required IGYCDOPT module, you can bypass the fixed option.

For example, if you select the OPT (OPTIMIZE) 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 provides an example of how to change the IBM-supplied defaults for compilers. The other sample job provides an example of how to override compiler options that have been fixed.

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

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.V4R2M0.SIGYSAMP.

Modifying compiler options and phases

If you plan to modify the values for compiler options and compiler phases, use the IGYCOPT syntax format shown in Figure 1. 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

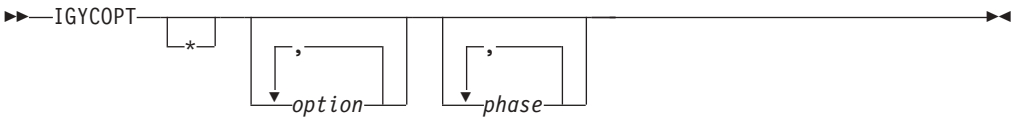


Figure 1. Syntax format for IGYCOPT compiler options and phases macro

IGYCDOPT worksheet for compiler options

The following 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 10, 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 ALOWCBL, 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 PRTEXIT 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=	____	_____	**NO**	“ADATA” on page 15
ADEXIT=	____	_____		“ADEXIT” on page 15

Table 1. IGYCDOPT worksheet for options (continued)

Compiler option	Enter * for fixed	Enter selection	IBM-supplied default	Syntax description
ADV=	____	_____	**YES**	"ADV" on page 16
ALOWCBL=	____	_____	**YES**	"ALOWCBL" on page 16
ARITH=	____	_____	**COMPAT**	"ARITH" on page 17
AWO=	____	_____	**NO**	"AWO" on page 17
BLOCK0=	____	_____	**NO**	"BLOCK0" on page 18
BUF=	____	_____	**4K**	"BUF" on page 18
CICS=	____	_____	**NO**	"CICS" on page 19
CODEPAGE=	____	_____	**1140**	"CODEPAGE" on page 19
COMPILE=	____	_____	**NOC(S)**	"COMPILE" on page 20
CURRENCY=	____	_____	**NO**	"CURRENCY" on page 20
DATA=	____	_____	**31**	"DATA" on page 22
DATEPROC=	____	_____	**NO**	"DATEPROC" on page 22
DBCS=	____	_____	**Yes**	"DBCS" on page 23
DBCSXREF	____	_____	**NO**	"DBCSXREF" on page 24
DECK=	____	_____	**NO**	"DECK" on page 24
DIAGTRUNC=	____	_____	**NO**	"DIAGTRUNC" on page 25
DLL=	____	_____	**NO**	"DLL" on page 25
DYNAM=	____	_____	**NO**	"DYNAM" on page 26
EXPORTALL=	____	_____	**NO**	"EXPORTALL" on page 26
FASTSRT=	____	_____	**NO**	"FASTSRT" on page 27
FLAG=	____	_____	**(I,I)**	"FLAG" on page 27
FLAGSTD=	____	_____	**NO**	"FLAGSTD" on page 28
INTDATE=	____	_____	**ANSI**	"INTDATE" on page 30
INEXIT=	____	_____		"INEXIT" on page 30
LANGUAGE=	____	_____	**ENGLISH**	"LANGUAGE" on page 31
LIB=	____	_____	**NO**	"LIB" on page 32
LIBEXIT=	____	_____		"LIBEXIT" on page 32
LINECNT=	____	_____	**60**	"LINECNT" on page 32
LIST=	____	_____	**NO**	"LIST" on page 33
LITCHAR=	____	_____	**QUOTE**	"LITCHAR" on page 33
LVLINFO=	____	_____		"LVLINFO" on page 34
MAP=	____	_____	**NO**	"MAP" on page 34
MDECK=	____	_____	**NO**	"MDECK" on page 34
MSGEXIT=	____	_____		"MSGEXIT" on page 35
NAME=	____	_____	**NO**	"NAME" on page 35
NSYMBOL=	____	_____	**NATIONAL**	"NSYMBOL" on page 36
NUM=	____	_____	**NO**	"NUM" on page 36
NUMCLS=	____	_____	**PRIM**	"NUMCLS" on page 37
NUMPROC=	____	_____	**NOPFD**	"NUMPROC" on page 37
OBJECT=	____	_____	**YES**	"OBJECT" on page 38
OFFSET=	____	_____	**NO**	"OFFSET" on page 38
OPT=	____	_____	**NO**	"OPTIMIZE" on page 39
OUTDD=	____	_____	**SYSOUT**	"OUTDD" on page 40
PGMNAME=	____	_____	**COMPAT**	"PGMNAME" on page 40
PRTEXIT=	____	_____		"PRTEXIT" on page 41
RENT=	____	_____	**YES**	"RENT" on page 41
RMODE=	____	_____	**AUTO**	"RMODE" on page 42
SEQ=	____	_____	**YES**	"SEQ" on page 43
SIZE=	____	_____	**MAX**	"SIZE" on page 43
SOURCE=	____	_____	**YES**	"SOURCE" on page 44
SPACE=	____	_____	**1**	"SPACE" on page 44

Table 1. IGYCDOPT worksheet for options (continued)

Compiler option	Enter * for fixed	Enter selection	IBM-supplied default	Syntax description
SQL=	____	_____	**NO**	"SQL" on page 45
SQLCCSID=	____	_____	**YES**	"SQLCCSID" on page 45
SSRANGE=	____	_____	**NO**	"SSRANGE" on page 46
TERM=	____	_____	**NO**	"TERM" on page 47
TEST=	____	_____	**NO**	"TEST" on page 47
THREAD=	____	_____	**NO**	"THREAD" on page 48
TRUNC=	____	_____	**STD**	"TRUNC" on page 49
VBREF=	____	_____	**NO**	"VBREF" on page 51
WORD=	____	_____	**NO**	"WORD" on page 51
XMLPARSE=	____	_____	**XMLSS**	"XMLPARSE" on page 52
XREFOPT=	____	_____	**FULL**	"XREFOPT" on page 52
YRWINDOW=	____	_____	**1900**	"YRWINDOW" on page 53
ZWB=	____	_____	**YES**	"ZWB" on page 54

Planning to place compiler phases in shared storage

You might want to make some load modules resident in a link-pack area in order to minimize the search for them when an Enterprise COBOL program is run or when the modules 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?

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 use the same module names; thus, only one set of phases can be placed in the LPA for any given initialization of the operating system.

The IGYCDOPT program indicates where each compiler phase 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 69.

It is recommended that the following four 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.

IGYCPGEN
because it is one of the two largest compiler phases.

IGYCSCAN
because it is the other of the two largest compiler phases.

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 should be given to IGYCRCTL and IGYCSIMD, the two phases that are always resident in the user region during compilation. Also, if sufficient shared storage is not available, priority should be given to IGYCPGEN and IGYCSCAN, the largest compiler phases.

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 IGYCPGEN and IGYCSCAN compiler phases are approximately 250 KB 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

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 5.

IN Indicates that the compiler phase is loaded into the user region from a library available at compile time. The compiler reserves storage for the phase from the value specified in the SIZE option.

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.

IGYCASM1

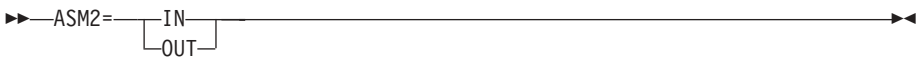
The Assembly 1 phase. It determines the object module storage, allocates the permanent and temporary registers, and optimizes addressability for data and procedure references. It also creates object text for data areas.

Syntax

IGYCASM2

The Assembly 2 phase. It completes preparation of the object program and creates object text, listings, punch data sets, and tables for the debugging feature.

Syntax



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

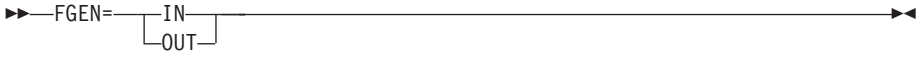
The DMAP phase. It prepares text for output requested by the MAP option.

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

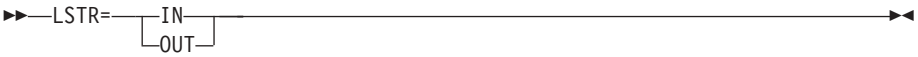
The COPY phase. It processes library source text and does a syntax check of the COPY, BASIS, and REPLACE statements.

Syntax

IGYCLSTR

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

Syntax



IGYCMSGT

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

IGYCOPTM

The optimizer phase. It restructures the PERFORM statements and eliminates duplicate computations.

Syntax

IGYCOSCN

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

Syntax

IGYCPGEN

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

Syntax

IGYCRCTL

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

Syntax

IGYCRWT

The normal reserved word table.

Syntax

IGYCSCAN

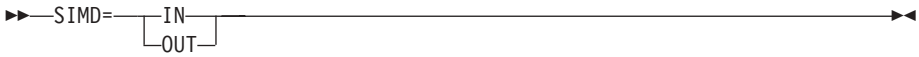
The scanning phase. It does syntax and semantic analysis of the source program and translates the source to intermediate text.

Syntax

IGYCSIMD

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

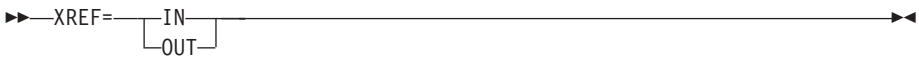
Syntax



IGYCXREF

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

Syntax



IGYCDOPT worksheet for compiler phases

The following worksheet will help you to plan and code the phases portion of the IGYCDOPT program. 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 6.

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

Table 2. IGYCDOPT program worksheet for compiler phases

Phase	Circle selection	Syntax description
ASM1=	**IN** / OUT	7
ASM2=	**IN** / OUT	7
DIAG=	**IN** / OUT	7
DMAP=	**IN** / OUT	7
FGEN=	**IN** / OUT	7
INIT=	**IN** / OUT	8
LIBR=	**IN** / OUT	8
LSTR=	**IN** / OUT	8
MSGT=	**IN** / OUT	8
OPTM=	**IN** / OUT	8
OSCN=	**IN** / OUT	8
PGEN=	**IN** / OUT	9
RCTL=	**IN** / OUT	9
RWT=	**IN** / OUT	9
SCAN=	**IN** / OUT	9
SIMD=	**IN** / OUT	9
XREF=	**IN** / OUT	9

Planning to create an additional reserved word table

When you install Enterprise COBOL, you have access to the following reserved word tables:

- IGYCRWT: the default reserved word table provided for your entire facility
- IGYCCICS: a CICS-specific reserved word table, provided as an alternate reserved word table (see "CICS reserved word table (IGYCCICS)" on page 11)

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?

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 59.

Reserved word tables supplied with Enterprise COBOL

These reserved word tables are on the installation medium:

- Default reserved word table
- CICS reserved word table

Default reserved word table (IGYCRWT)

The default reserved word table is described in the *Enterprise COBOL Language Reference*.

CICS reserved word table (IGYCCICS)

Enterprise COBOL provides an alternate reserved word table specifically for CICS application programs. It is set up so that COBOL words that are not supported under CICS are flagged by the compiler with an error message.

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

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.V4R2M0.SIGYSAMP.

Note: The high-level qualifier IGY.V4R2M0 might have been changed when Enterprise COBOL was installed.

Chapter 2. Enterprise COBOL compiler options

This information describes the compiler options whose default values you can change. 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 using the compiler options, see 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. Table 3 will help you resolve possible conflicts between compiler options.

Table 3. Conflicting compiler options

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

Table 3. Conflicting compiler options (continued)

Compiler option	Conflicts with:
EXPORTALL=YES	DLL=NO DYNAM=YES RENT=NO
FLAGSTD=(other than NO)	WORD=*xxxx*
LIB=NO	CICS=YES MDECK=YES SQL=YES
LIST=YES	OFFSET=YES
MDECK=YES	LIB=NO
NSYMBOL=NATIONAL	DBCS=NO
OBJECT=NO	TEST=(other than NO)
OFFSET=YES	LIST=YES
OPT=STD or OPT=FULL	TEST=(HOOK,...,...)
RENT=NO	CICS=YES DLL=YES EXPORTALL=YES THREAD=YES
SQL=YES	LIB=NO
TEST=(NOHOOK,...,...)	OBJECT=NO
TEST=(HOOK,...,...)	OBJECT=NO OPT=STD or OPT=FULL
THREAD=YES	RENT=NO
WORD=*xxxx*	FLAGSTD=(other than NO)
XREFOPT=NO	DBCSXREF=(other than NO)

Compiler options for standards conformance

For information about specifying compiler options to conform with the COBOL 85 Standard, see the *Enterprise COBOL Programming Guide.*

Compiler options syntax and descriptions

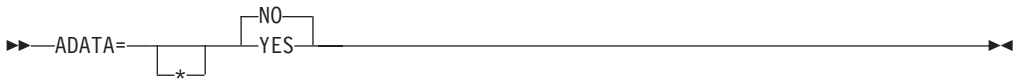
The syntax diagrams in the following topics describe each modifiable compiler option. The text below each diagram describes the effect of selecting a specific parameter.

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

Syntax



Default
ADATA=NO

YES
Produces the Associated Data file with the appropriate records.

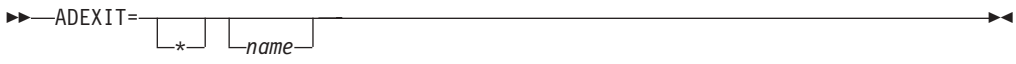
NO
Does not produce the Associated Data file.

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

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. When 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, see the *Enterprise COBOL Programming Guide*.

ADV

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.

ALOWCBL

Syntax

Default
ALOWCBL=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:

- ALOWCBL 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 ALOWCBL=NO.

ARITH

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

Syntax

Default
AWO=NO

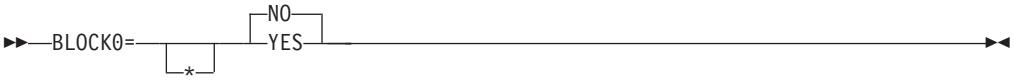
YES
Activates the APPLY-WRITE-ONLY clause for any file within the program that is physical sequential with variable block format regardless of whether or not the APPLY-WRITE-ONLY clause is specified in the program.

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

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.

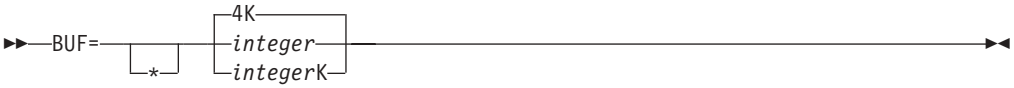
NO
Does not activate the BLOCK CONTAINS 0 clause by default for any file.

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 more information, see the *Enterprise COBOL Programming Guide*.

BUF

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.

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

Note:

- BUF and SIZE values are used by the compiler to determine how much storage to use during compilation. The amount allocated to the buffers is included in the amount of main storage available to the compiler for the SIZE option.
- BUF cannot exceed the track capacity for the device used, nor can it exceed the maximum allowed by data management services.

CICS

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.

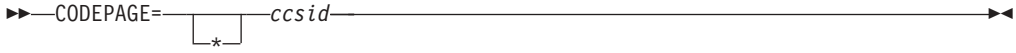
NO
When the NO option is specified, any CICS statements that are found in the source program are diagnosed and discarded.

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

Syntax



Default
CODEPAGE=1140

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

Use CODEPAGE to specify 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.

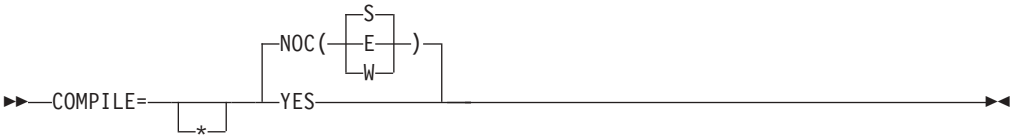
The default CCSID 1140 is the equivalent of CCSID 37 (EBCDIC Latin-1, USA), but additionally includes the euro symbol.

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).

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

COMPILE

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.

CURRENCY

Syntax

The COBOL default currency symbol is the dollar sign ($). The CURRENCY option lets you define an alternate default currency symbol.

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:

- 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 any option setting for literal delimiters.
- 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

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.

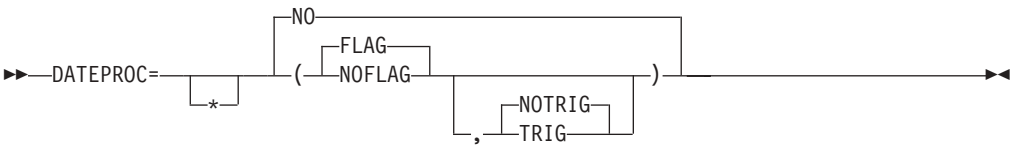
31 Causes allocation of user data areas, such as WORKING-STORAGE and FD record areas, from unrestricted storage or in space acquired by a GETMAIN with the LOC=ANY option. Specifying this option can result in storage being acquired in virtual addresses either above or below the 16-MB line. The operating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

DATEPROC

Syntax



The DATEPROC option determines whether the compiler will perform date processing using the DATE FORMAT clause and other language constructs.

Default
DATEPROC=NO, or DATEPROC(FLAG,NOTRIG) if only DATEPROC is specified.

FLAG
Recognizes the DATE FORMAT clause and performs automatic date processing. In addition, specifying DATEPROC=FLAG flags, either with an information-level message or a warning-level message as appropriate, each statement that uses or is affected by date processing.

NOFLAG
Recognizes the DATE FORMAT clause and performs automatic date processing. Statements that use or are affected by date processing are not flagged with information-level or warning-level messages.

TRIG
Recognizes the DATE FORMAT clause and performs date processing based on automatic windowing that the compiler applies to operations on windowed date fields. The automatic windowed date fields are sensitive to specific trigger or limit values in the date fields and in other nondate fields. These specific values represent dates that are not valid and that can be tested for or used as upper or lower limits.

NOTRIG
Recognizes the DATE FORMAT clause and performs date processing based on automatic windowing that the compiler applies to operations on windowed date fields. The automatic windowed date fields are not sensitive to specific trigger or limit values in the date fields and in other nondate fields. Only the value of the year part of dates is relevant to automatic windowing.

NO
Treats the DATE FORMAT clause as comments and disables automatic date processing. In the case of the new intrinsic functions, specifying DATEPROC=NO generates object code that returns a default value whenever a new intrinsic function is used.

Error-level and severe-level messages are issued regardless of whether DATEPROC=FLAG or DATEPROC=NOFLAG is specified.

DBCS

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.

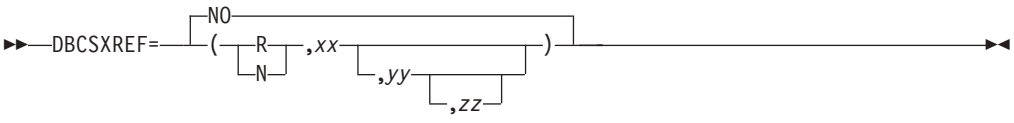
Note:

- 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

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 load module 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 and the SIZE value is anything other than MAX, ensure that the user region is large enough to accommodate both the compiler and 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

Syntax

Default
DECK=NO

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

NO
Sends no object code to SYSPUNCH.

DIAGTRUNC

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.

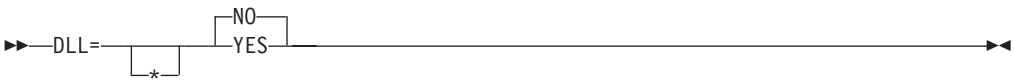
NO
Does not cause the compiler to produce a severity-4 message.

Note:

- The diagnostic 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 for COMP-5 receivers, nor for binary receivers when you specify the TRUNC(BIN) option.

DLL

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.

DYNAM

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 module file.

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.

EXPORTALL

Syntax

Default
EXPORTALL=NO

YES
Automatically exports certain symbols 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

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 the *Enterprise COBOL Programming Guide*.)

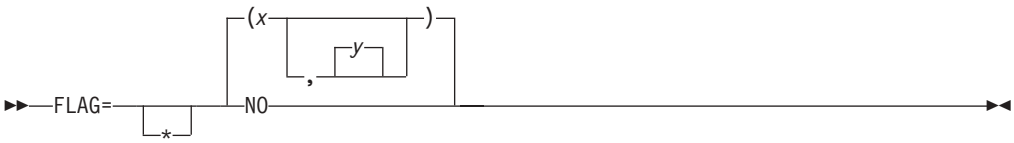
NO
Specifies that Enterprise COBOL does the input and output for the sort and merge.

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

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.

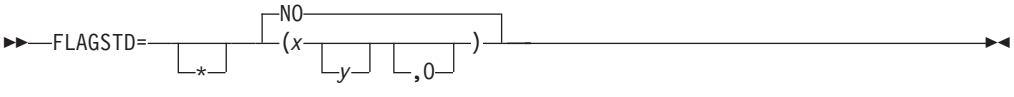
NO
Indicates that no error messages are flagged.

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 Date file. For more information, see the *Enterprise COBOL Programming Guide*.

FLAGSTD

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)

O 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 COBOL 85 Standard:
  - Language syntax, such as the DATE FORMAT clause, used by the COBOL automatic date-processing facilities (as enabled by the DATEPROC compiler option)
  - 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 warning message to be issued:

ADV=NO
DATEPROC=(other than NO)
DBCS=YES
DYNAM=NO
FASTSRT=YES
LIB=NO
LITCHAR=APOST
NUM=YES
NUMPROC=PFD
SEQ=YES
TRUNC=OPT or BIN
WORD=(other than NO or RWT)
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
DATEPROC=(other than NO)
DBCS=YES
DYNAM=NO
LIB=NO
LITCHAR=APOST
NUM=YES
NUMPROC=PFD
SEQ=YES
TRUNC=OPT or BIN
WORD=(other than NO or RWT)
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 35.

INEXIT

Syntax

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. When 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. When the option is supplied, the SYSIN data set is not opened.

For more information, see the *Enterprise COBOL Programming Guide*.

INTDATE

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.

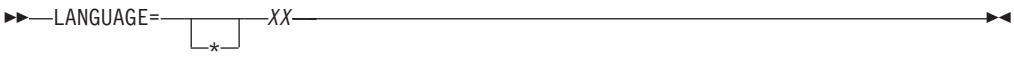
Note:

- 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

Syntax



Default
LANGUAGE=EN

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

Table 4. 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.

LIB

Syntax

Default
LIB=NO

YES
Indicates that the source program contains COPY or BASIS statements.

NO
Indicates that the source program doesn't contain COPY or BASIS statements.

LIBEXIT

Syntax

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. When 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. When the option is supplied, the SYSLIB and library-name data sets are not opened.

For more information, see the *Enterprise COBOL Programming Guide*.

LINECNT

Syntax

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

Syntax

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

Syntax

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 COBOL 85 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

Syntax

Default
No characters are specified.

xxxx
Identifies the one to four alphanumeric characters that are inserted into the listing header following the release number (the last 4 bytes of the signature area). This option might be used to identify “compiler level” information within the listing header.

MAP

Syntax

Default
MAP=NO

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

- DATA DIVISION map
- Global tables
- Literal pools
- Program statistics
- Size of the program’s working storage and its location in the object code if the program is compiled without the RENT compiler option

NO
Mapping is not performed.

MDECK

Syntax

Default
MDECK=NO

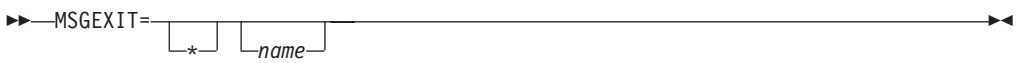
COMPILE
Compilation continues normally after library processing and generation of the MDECK output file.

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

Syntax



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. When 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, see the *Enterprise COBOL Programming Guide*.

NAME

Syntax

Default

NAME=NO

ALIAS
Precedes the NAME statement corresponding to the PROGRAM-ID with a linkage editor ALIAS statement for each ENTRY statement in the program.

NOALIAS
Appends a linkage editor 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 linkage editor 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

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

Syntax

Default
NUM=NO

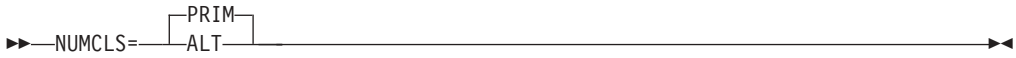
YES
Uses the line numbers from the source program rather than compiler-generated line numbers for error messages and procedure maps.

NO
Uses the compiler-generated line numbers for 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.

NUMCLS

Syntax



Default
NUMCLS=PRIM

ALT
Specifies the sign representations that are recognized as valid by the numeric class test for data items that are defined:

- As signed (with an "S" in the PICTURE clause)
- Using DISPLAY or COMPUTATIONAL-3 (packed-decimal)
- Without the SEPARATE phrase on any SIGN clause

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=MIG or NUMPROC=NOPFD. NUMPROC=PFD specifies more strict rules for valid sign configuration.

NUMPROC

Syntax

Default
NUMPROC=NOPFD

MIG
Aids in migrating OS/VS COBOL application programs to Enterprise COBOL. Processing with MIG entails these actions:

- Using existing signs for comparison and arithmetic operations
- Generating preferred signs for the results of MOVE and arithmetic operations (These results meet the criteria for using NUMPROC=PFD.)
- Performing numeric rather than logical comparisons

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

PFD
Optimizes the generated code, especially when OPT=STD or OPT=FULL. 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=MIG and 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=MIG or 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

Syntax

Default
OBJECT=YES

YES
Places the generated object code in a file defined by SYSLIN.

NO
Places no object code in SYSLIN.

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

OFFSET

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 following are produced:

- Global tables
- Literal pools
- Program statistics
- 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 13.

OPTIMIZE

Syntax

Default
OPT=NO

FULL
Discards any unused data items and does not generate code for any VALUE clauses for these data items. If the OPT(FULL) and MAP option are both specified, discarded data items will have a BL number of XXXX in the MAP output indicating that the number is not used.

STD
Generates optimized object code.

NO
Specifies that your code is not optimized.

Performance consideration: Using OPT=STD or OPT=FULL 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.
- If you want to debug optimized object code using Debug Tool, the only hook-location subparameter allowed for the TEST option is NONE. Any other combination results in a nonzero return code and an error message during installation.
- Optimization is turned off if an S-level error or U-level error occurs.

OUTDD

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

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 the *Enterprise COBOL Programming Guide*.

PRTEXIT

Syntax

Default

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

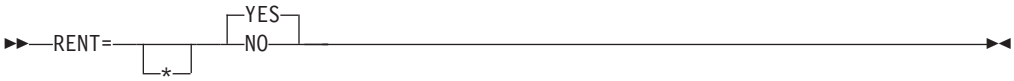
name

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, see the *Enterprise COBOL Programming Guide*.

RENT

Syntax



Default

RENT=YES

YES

Indicates that the object code produced for a COBOL program is 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 the object code produced for a COBOL program is to be nonreentrant.

Note:

- Programs must be compiled with RENT=YES or RMODE(ANY) if they will be run in virtual storage addresses above 16 MB.
- The RENT compiler option is required for programs that are run under CICS.
- Programs compiled with Enterprise COBOL always have AMODE(ANY). The RMODE assigned to a program depends on the RENT|NORENT and RMODE compiler options. Valid combinations are shown in the following table.

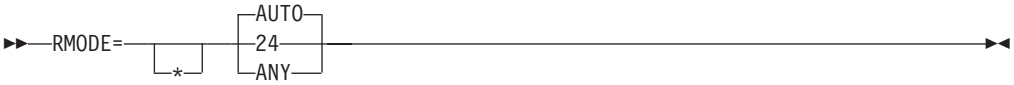
Table 5. Effect of RENT and RMODE on residency mode

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	RMODE ANY
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.

RMODE

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 whether NORENT or RENT 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 6. 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	RMODE ANY
24	RENT	RMODE 24
24	NORENT	RMODE 24

SEQ

Syntax

Default
SEQ=YES

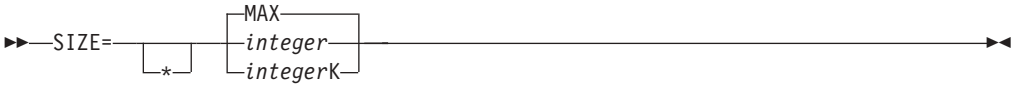
YES
Checks that the source statements are in ascending alphanumeric order by line number.

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.

SIZE

Syntax



Default
SIZE=MAX

integer
Specifies the amount of virtual storage available, in bytes. The minimum acceptable value is 851968.

***integer*K**
Specifies the amount of virtual storage available in 1024-byte (K) increments. The minimum acceptable value is 832K.

MAX
Requests all available space in the user region for use during the compilation. For extended architecture, the compiler obtains the largest contiguous block of free storage above the 16-MB line.

Recommendation: Do not use SIZE(MAX) if you require that the compiler leave a specific amount of unused storage available in the user region. For example, if you are using the CICS or SQL compiler option, use a value such as SIZE(4000K). (This value should work for most programs.) If you compile in 31-bit mode and specify SIZE(MAX), the compiler uses storage as follows:

- Above the 16-MB line: all the storage in the user region
- Below the 16-MB line: storage for:
  - Work-file buffers
  - Compiler modules that must be loaded below the line

SOURCE

Syntax

Default
SOURCE=YES

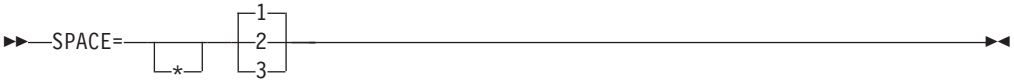
YES
Indicates that you want a listing of the source statements in the compiler-generated output. This listing also includes any statements embedded by COPY.

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

Syntax



Default
SPACE=1

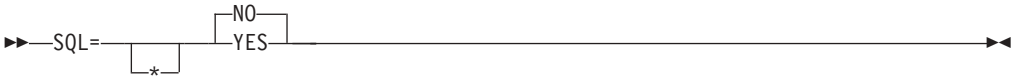
1 Provides single spacing for the source statement listing.

2 Provides double spacing for the source statement listing.

3 Provides triple spacing for the source statement listing.

SQL

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.
- SQL=YES conflicts with LIB=NO.

SQLCCSID

Syntax

Default
SQLSSCID=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 about the SQLCCSID option, see the *Enterprise COBOL Programming Guide*.

SSRANGE

Syntax

Default
SSRANGE=NO

YES
Generates code that checks subscripts, reference modifications, variable-length group ranges, and indexes in the source program 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 a variable-length item does not exceed its defined maximum length as a result of incorrect setting of the OCCURS DEPENDING ON object.

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

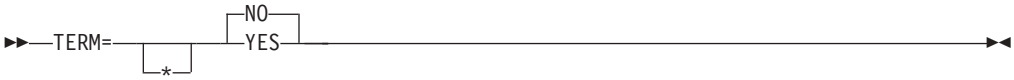
NO
No code is generated to perform subscript or index checking at run time.

Note:

- If the SSRANGE option is in effect at compile time, the range-checking code is generated.
- Range-checking can be inhibited at run time by specifying the Language Environment runtime option CHECK(OFF). However, the range-checking code still requires overhead and is dormant within the object code.
- The range-checking code can be used optionally to aid in resolving any unexpected errors without recompilation.

TERM

Syntax



Default
TERM=NO

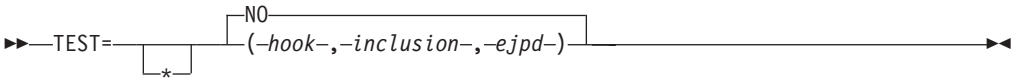
YES
Specifies that the progress and diagnostic messages are sent to the SYSTERM file, which defaults to the user's terminal unless specified otherwise.

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.

TEST

Syntax



Default
TEST=NO

NO Produces object code that cannot be symbolically debugged using Debug Tool.

Other than NO
Produces object code that can be run using Debug Tool. You must specify values for the *hook*, *inclusion*, and *ejpd* suboptions as described below.

hook values:

HOOK
Activates the generation of all compiled-in hooks. Hooks are generated at all statements, all path points, and all program entry and exit points. In addition, if either the DATEPROC=FLAG option or DATEPROC=NOFLAG option is in effect, hooks are generated for all date-processing statements.

NOHOOK
Suppresses the generation of all compiled-in hooks.

inclusion values:

SEPARATE
Generate the debugging information tables in a data set separate from your object program.

NOSEPARATE
Include the debugging information tables in your object program.

ejpd values:

EJPD When the OPT=STD or OPT=FULL option is specified, together with TEST=(NOHOOK,...,EJPD):

- 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.

NOEJPD

When TEST(NOHOOK,...,NOEJPD) and OPTIMIZE are specified:

- The JUMPTO and GOTO commands are not enabled.
- The normal amount of program optimization is done.

Note:

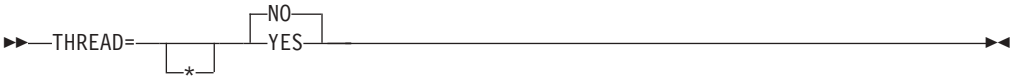
- Because TEST=(HOOK,...,...) generates additional code, it can cause significant performance degradation at run time when used in a production environment.
- If you specify TEST=(HOOK,...,...), the following options are put into effect at compilation time:

  OBJ=YES

  OPT=NO
- A date-processing statement is any statement that references a date field, or any EVALUATE or SEARCH statement WHEN phrase that references a date field.
- For production programs, compile your programs with TEST=(NOHOOK,SEPARATE,...) to get debugging capability but not increase the size of your production load modules.

THREAD

Syntax



Default

THREAD=NO

YES

Use THREAD if you want to indicate that a COBOL program is enabled for execution in a Language Environment enclave with multiple POSIX threads or PL/I tasks.

NO

Use NO if you want to indicate that a COBOL program is not enabled for execution in a Language Environment enclave with 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
  - SORT or MERGE statements
  - STOP literal statement
  - USE FOR DEBUGGING statement
  - WITH DEBUGGING MODE clause
- 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

TRUNC

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. The truncation results are dependent on the particular code sequence generated and might not be the same in OS/VS COBOL and Enterprise COBOL.

STD
Conforms to the COBOL 85 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.

VBREF

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.

WORD

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 11.

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 ($).

ASM1	LIBO	LVL8	RDSC
ASM2	LIBR	OPTM	RWT
DIAG	LSTR	OSCN	SAW
DMAP	LVL0	PGEN	SCAN
DOPT	LVL1	RCTL	SIMD
FGEN	LVL2	RDPR	XREF
INIT	LVL3		

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

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 operational behaviors and results of the XML PARSE statement are compatible with those in Version 3 of Enterprise COBOL.

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® Application Assist Processors (zAAPs)

XREFOPT

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:

- The XREFOPT option sets the default value for the XREF compiler option.
- 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 information, see the description of the XREF compiler option in the *Enterprise COBOL Programming Guide.*

YRWINDOW

Syntax

Default
YRWINDOW=1900

integer
Specifies the first year of the 100-year window used by COBOL windowed date fields, and can be either of the following items:

- An unsigned decimal integer from 1900 through 1999.
- A negative decimal integer from -1 through -99, representing an offset from the current year at run time. The current year is determined for each compilation unit when it is first initialized, or when it is reinitialized after execution of a CANCEL statement that refers to the compilation unit.

Note:

- YRWINDOW has no effect unless DATEPROC=FLAG or DATEPROC=NOFLAG is specified.
- At run time, two conditions must be true:
  - The 100-year window must have its beginning year in the 1900s.
  - The current year must lie within the 100-year window for the compilation unit.
- All windowed dates have a year relative to the base year. For example, if the base year were specified as 1965, all windowed year values would be interpreted as years within the 100-year window 1965-2064, inclusive. So, a windowed year value of 67 would represent the year 1967, whereas a windowed year value of 05 would represent the year 2005.

  If the base year were specified as -30, then the window would depend on when the application were run. Running it during 2009 would represent a

100-year window of 1979 through 2078. So, a windowed year value of 79 would represent the year 1979, whereas a windowed year value of 78 would represent the year 2078.

- The YRWINDOW installation option is analogous to the YEARWINDOW compiler option.

ZWB

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

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.V4R2M0.SIGYSAMP. Table 7 shows the names of the sample modification jobs, which are described in detail in the referenced information.

The sample modification jobs that IBM provides are not customized for your particular system. You must customize them.

Copy members from IGY.V4R2M0.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. Save the modified JCL for future IGYWMLPA reference.

Table 7. Summary of user modification jobs for Enterprise COBOL

Description	Job	See:
Change compiler options default module	IGYWDOPT	“Changing compiler options default module” on page 57
Create an options module to override compiler options specified as fixed	IGYWUOPT	“Creating an options module to override options specified as fixed” on page 57
Create additional reserved word table	IGYWRWD	“Creating or modifying additional reserved word tables” on page 58

Table 7. Summary of user modification jobs for Enterprise COBOL (continued)

Description	Job	See:
Place Enterprise COBOL modules in shared storage	IGYWMLPA	"Placing Enterprise COBOL modules in shared storage" on page 63

Changing the defaults for compiler options

To change the defaults for compiler options or create a compiler-options module to override fixed options:

1. Copy the source of options module IGYCDOPT from IGY.V4R2M0.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 `LVLINFO=(Fix1)` or `LVLINFO=(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 3. For descriptions of the options, see Chapter 2, "Enterprise COBOL compiler options," on page 13.
- 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 compiler options default module

To change the defaults for the Enterprise COBOL compiler options, use the sample job IGYWDOPT. To select your default values, use the information in Chapter 2, "Enterprise COBOL compiler options," on page 13.

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 6 and "Placing Enterprise COBOL modules in shared storage" on page 63.

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.V4R2M0.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 IGY*nnnn* informational messages in your listing to verify the defaults that will be in effect for your installation.

Creating an options module to override options specified as fixed

If you have specified some options as fixed in your compiler default options module, you might occasionally find an application that needs to override a fixed option. 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.V4R2M0.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.V4R2M0.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 DSNAME=YOURLIB in the SYSLMOD DD statement to the name of the partitioned data set that you want your IGYCDOPT module linked 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 IGY*nnnn* 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.

Creating or modifying additional reserved word tables

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 11). 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 that you can use 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 51.

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 IGYC*xxxx*. 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 COBOL 85 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

To create or modify a reserved word table, you must edit a copy of the appropriate source file:

- Member IGY8RWRD in IGY.V4R2M0.SIGYSAMP (the IBM-supplied default reserved word table)
- Member IGY8CICS in IGY.V4R2M0.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 CODASYL 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 "Modifying and running JCL to create a new reserved word table" on page 62. 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 a reserved word table, you must understand the syntax rules for the control statements and for the operands within control statements.

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

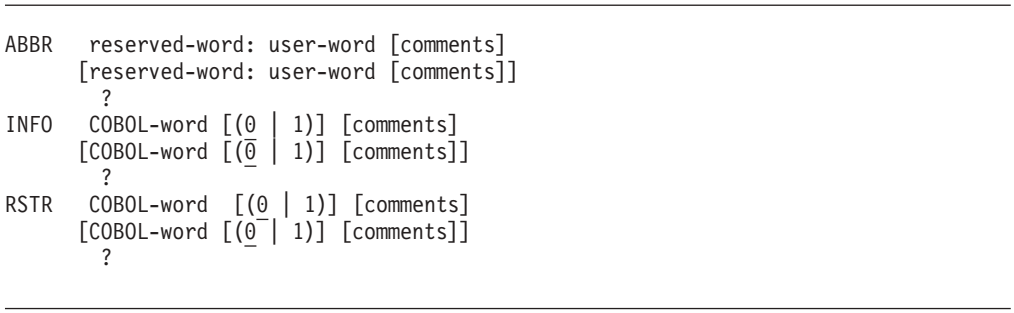


Figure 2. Syntax format for reserved word processor control statements

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

When you code your control statements, follow these rules:

- 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

The following list 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

When you code the control statement operands, follow these rules:

- 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

ABBR reserved-word: user-word [comments]

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

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

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

This statement specifies the COBOL words that are to be flagged by the compiler.

This statement can also be used to control the use of nested programs. 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, the 0086 informational message is issued if the FLAGSTD compiler option is in effect.

1 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

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

This statement specifies COBOL words that cannot be used in a program.

This statement can also be used to control the use of nested programs. 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 JCL to create a new reserved word table

The JCL that you use to create a new reserved word table contains STEP1, STEP2, and STEP3, which do the following, respectively:

Run the reserved word table utility with your modified table.
Assemble your modified reserved word table.
Produce a runtime load module from the object module.

After you run the job, a new 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.

Modifying and running non-SMP/E JCL

Use sample job IGYWRWD in IGY.V4R2M0.SIGYSAMP to create or modify a reserved word table. The sample job uses a member based on IGY8RWRD or IGY8CICS in IGY.V4R2M0.SIGYSAMP as the input to the reserve word utility. It creates load module IGYC*xxxx* (where *xxxx* is the user identification) in IGY.V4R2M0.SIGYCOMP.

Before you run the job, do the steps described below.

To modify the JCL for IGYWRWD, do these steps:

1. Modify the job statement for your site.
2. Add JES ROUTE records if desired.
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.

If, after you run IGYWRWD, 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.

Placing Enterprise COBOL modules in shared storage

All of the modules in IGY.V4R2M0.SIGYCOMP that are reentrant can be included in shared storage by:

- Authorizing the data set IGY.V4R2M0.SIGYCOMP
- (Optional) Including IGY.V4R2M0.SIGYCOMP in the LNKLST*nn* concatenation
- Creating an IEALPA*nn* member in SYS1.PARMLIB that lists the modules to be made resident in the MLPA when the system is IPLed

IGYWMLPA is installed in IGY.V4R2M0.SIGYSAMP for you to use as an example in creating your IEALPA*nn* member.

Under z/OS, you do not need to place IGY.V4R2M0.SIGYCOMP in the LNKLST*nn* concatenation to be able to load modules into the LPA. If you choose not to add it to the LNKLST*nn* 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 LNKLST*nn* concatenation
- Copying the non-LPA modules to a data set that can be used as a STEPLIB or JOBLIB

Using the entire IGY.V4R2M0.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 LNKLST*nn* 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 56.

Tailoring the cataloged procedures to your site

You might want to tailor the cataloged procedures IGYWC, IGYWCL, IGYWCLG, IGYWCG, IGYWCPL, IGYWCPLG, IGYWCPG, or IGYWPL 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.V4R2M0.SIGYCOMP and CEE.SCEERUN in the LNKLST 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

Appendix. 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

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

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

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM
The IBM logo
ibm.com
BookManager
CICS
DB2
IMS
Language Environment
OS/390
System z
z/Architecture
z/OS

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

List of resources

Enterprise COBOL for z/OS

Compiler and Runtime Migration Guide, GC23-8527

Customization Guide, SC23-8526

Language Reference, SC23-8528

Licensed Program Specifications, GI11-7871

Program Directory, GI11-7870

Programming Guide, SC23-8529

z/OS Language Environment

Concepts Guide, SA22-7567

Customization, SA22-7564

Debugging Guide, GA22-7560

Programming Guide, SA22-7561

Programming Reference, SA22-7562

Run-Time Messages, SA22-7566

Run-Time Application Migration Guide, GA22-7565

Writing Interlanguage Communication Applications, SA22-7563

Related 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*

CICS Transaction Server for z/OS Application Programming Guide, SC34-7022

CICS Transaction Server for z/OS Application Programming Reference, SC34-7023

CICS Transaction Server for z/OS Customization Guide, SC34-7001

CICS Transaction Server for z/OS External Interfaces Guide, SC34-7019

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

DB2 for z/OS Installation Guide, GC18-9846

DB2 for z/OS Internationalization Guide (Unicode), SC19-1161

Debug Tool User's Guide, SC23-9536

Debug Tool Reference and Messages, GC23-9535

High Level Assembler for z/OS Language Reference , SC26-4940

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*

ISPF User's Guide Volume I, SC34-4822

z/OS MVS Initialization and Tuning Guide, SA22-7591

z/OS MVS Initialization and Tuning Reference, SA22-7592

SMP/E for z/OS User's Guide, SA22-7773

SMP/E for z/OS Reference, SA22-7772

TSO/E Primer, SA22-7787

TSO/E User's Guide, SA22-7794

Softcopy publications

The following collection kits contain Enterprise COBOL and other product publications:

z/OS Software Products Collection, SK3T-4270

z/OS and Software Products DVD Collection, SK3T-4271

The following collection kit contains z/OS and related product publications:

z/OS Collection, SK3T-4269

Index

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

Readers' Comments — We'd Like to Hear from You

Enterprise COBOL for z/OS
Customization Guide
Version 4 Release 2

Publication No. SC23-8526-01

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Send your comments to the address on the reverse side of this form.

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SC23-8526-01

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Reader Comments
DTX/E269
555 Bailey Avenue
San Jose, CA
U.S.A. 95141-9989

Fold and Tape | **Please do not staple** | Fold and Tape

SC23-8526-01

Cut or Fold
Along Line

Program Number: 5655-S71

Printed in USA

SC23-8526-01