Buzzard Creek Tech, Inc.

Load Library Scanner

The purpose for writing this utility was several fold. I needed a utility to inventory all of the application programs in all of the application load libraries. This occurred during a period when frequent COBOL compiler, CICS and LE upgrades were occurring. Secondly, I wanted to write some test programs to implement as many of the IBM Language Environment (LE) callable services as I could.

So, I formulated a project for this utility, with the added interest of implementing as many LE callable services as possible to accomplish the task. It was going to have to be written in Assembler because I needed to access the directory of a PDS/PDSE in order to know which members to load into main storage. Secondly, it was going to have to be an LE conforming Assembler program because all of the LE program management services that I wanted to use was of the Compiler Writer Interface (CWI) type. Since I had to use LE Assembler Macros to make the program LE conforming it is also re-entrant. You could load this program in the LPA.

The resulting program has undergone a number of enhancements. You can generate one of two possible reports via the PARM statement. Specify RPTOPTS to print the COBOL compiler options report. Specify RPTVERB to print the COBOL verbs and other literals report. You can select specific members within a load library by adding a MBRLIST DD statement in the JCL. The file just needs to have attribute LRECL=80. Comment records contain an asterisk in the first column. Member names can start in any column and only one member name can exist per record. If the MBRLIST DD statement is not in the JCL, the program will process all members (except aliases and non-executable) found in the PDS/PDSE directory. Additionally, you can select members with specific attributes via keyword text, by entering the SELECT(...) parameter on the PARM statement. The select-able items are dependent on the report type specified. For RPTOPTS you can specify text from Table 1 and Table 2. For RPTVERB you can specify text from Table 1 and Table 3. All of the print lines for a given member are built in LE stack storage prior to scanning for the comma delimited text strings contained in the SELECT(...) parameter. A match on any text string declared in the SELECT(...) parameter will cause all of the print lines for a member to be printed. All of the select-able text found in Table 2 and Table 3 are defined in COBOL objects as bit switches. These bit settings can be found in IBM COBOL/II and later Programming Guide manual (search on SIGNATURE information bytes).

This program has been very valuable in helping clients inventory/classify the modules that they are executing. It is also very helpful in the code migration process. With it you can verify before moving programs to production that the COBOL compiler options are correct and ensure that programmers have not coded COBOL programs with SORT or DISPLAY verbs in your CICS programs.

PARM statement format:

```
PARM='/RPTOPTS'
PARM='/RPTVERB'
PARM='/RPTOPTS,SELECT(ƀ31/24,-RN-,DATA(24),ƀƀSSR)'
PARM='/RPTVERB,SELECT(COB3,CALL...ADDRESS OF)'
```

MVS JCL:

```
//SSLEANAL JOB (...),'LOADLIB SCANNER',CLASS=S,
//  MSGCLASS=Q RESTART=
//STEP001  EXEC PGM=SSLEANAL,REGION=4M,
//  PARM='/RPTOPTS,S(NOCMPR2)'
//STEPLIB  DD  DSN=target.loadlib,DISP=SHR
//         DD  DSN=ssleanal.loadlib,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//PRTOUTA  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//APPLLIB  DD  DSN=target.loadlib,DISP=SHR
//
```

```
//SSLEANAL JOB (...),'LOADLIB SCANNER',CLASS=S,
//  MSGCLASS=Q RESTART=
//STEP001  EXEC PGM=SSLEANAL,REGION=4M,
//  PARM='/RPTVERB,S(SEARCH,END-SEARCH)'
//STEPLIB  DD  DSN=target.loadlib,DISP=SHR
//         DD  DSN=ssleanal.loadlib,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//PRTOUTA  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//APPLLIB  DD  DSN=target.loadlib,DISP=SHR
//MBRLIST  DD  DSN=user.parmlib(member),DISP=SHR
//
```

Disclaimers:
Report results obtained by SELECT(...) from Table 3 may be inaccurate. The COBOL compiler does not necessarily set the SIGNATURE bits correctly. (Straight from IBM's Mr. COBOL, Tom Ross)
*Note: In general I find this to be an understatement. I would say that it is at best half right.

If the CICS interface stub is not the first object module in the load module SSLEANAL will not recognize the member to be a CICS program. COBOL and Assembler programs appear to follow this rule, but other languages do not (i.e. C/C++ and PL/I).

The values (OTH, MAP, and ASM) found under column heading LANG are at worst a best guess. The other values should always be correct, because they are retrieved from an LE control block.

Each new release of COBOL may add/change/delete COBOL signature bytes, so coding changes may be required.

Report Description

Column Headings	Length	Descriptions
PROGRAM	8	Program name
MODE	3/3	AMODE and RMODE of the program separated by "/" character • `ANY/ANY`

		• ANY/24 • 31/ANY • 31/24 • 24/24
BINDER OPT(S)	2/2/2/2/2	Binder options separated by "-" character OL – only loadable RF – refreshable RN – re-entrant RU – reuse SC – scatter storage
LE	1	Type of LE routine C – C/370 routine E – LE IEEE routine F – Fast Link routine N – not LE conforming R – CEESTART entry point S – Standard routine X – XPLINK routine Y – unknown LE
LANG	4	Compiler language OTH – other MAP – CICS BMS map ASM – Assembler COB – COBOL CEL – LE library routine C/++ – C or C++ program FOR – FORTRAN DCE – PLI – PL/I program TCP – DBUG – Debug program N/A –
COBOL language version	1	COBOL version 1 – COBOL/VS 2 – COBOL/II 3 – COBOL MVS, OS/370, Enterprise 4 – Enterprise COBOL V5 & V6
compiler VRM	6	COBOL compiler VRM
CICS	3	CICS translator level. The 1st character indicates what language stub was included. A – Assembler C – COBOL I – LE compliant

		P – PL/I X – EXCI stub (batch program) If the program language is LE compliant then the CICS translator stub should be an "I". If not these programs should be re-linked with the appropriate CICS stub. Note: if "*" precedes it then load point address is the same as entry point address. Probably an Assembler program. The load point address should point to the CICS interface stub and the entry point should point to the member. The Assembler program can be modified to specify an ENTRY statement to correct the issue.
API	1	Type of Systematics software API ALFA table 1 – ALFA table starts at location 160 or greater in the executing program (in VS/COBOL this was the beginning of WORKING-STORAGE). Probably an Assembler program. COBOL programs have been converted to option 2. 2 – ALFA table is an external table linked in front of the executing program.
DATE	8	COBOL compile date (format: YYYYMMDD)
TIME	6	COBOL compile time (format: HHMMSS)
SIZE	5	Load module size
text	52	Depending on PARM option this area will contain the COBOL compiler options, or COBOL verbs and other source statement settings. PARM='/RPTOPTS' for compiler options report PARM='/RPTVERB' for verb report

Table 1.

Select Options for Options Report

COBOL Compiler Options	**Descriptions**
DDSTMT(nnnnnnnn)	Data Division statement count
PDSTMT(nnnnnnnn)	Procedure Division statement count
SIGLVL(nnn)	Cobol Signature level.
CP(ccsid|**1140**)	CODEPAGE specifies the coded character set identifier (CCSID) for an EBCDIC code page. (COBOL for MVS, OS/390, Enterprise) (x'00':2)
ARCH(6|7|8|9|10|11|12)	Each higher number generates newer and faster code. 6 – 2084-xxx (z990), 2086-xxx (z890) dropped V5.2 7 – 2094-xxx (IBM S z9 BC), 2096-xxx (IBM S z9 EC)

	8 – 2097-xxx (IBM S z10 EC), 2098-xxx (IBM S z10 BC) 9 – 2817-xxx (IBM zE 196), 2818-xxx (IBM zE 114) 10 – 2827-xxx (IBM zE EC12), 2828-xxx (IBM zE BC12) 11 – 2964-xxx (IBM z13), 2965-xxx (IBM z13) 12 – 3906-xxx (IBM z14) (COBOL V5.1) (x'02':1)
OPT(**0**|1|2)	OPTIMIZE will generate faster code. 0 – limited optimizations, faster compile time, when TEST is specified all debug capabilities are available. Matches NOOPT. 1 – improved run-time, some code removal, when TEST is specified most debug capabilities are available. Matches OPT and OPT(STD) or OPT(FULL) and STGOPT. 2 – most aggressive optimizations, when TEST is specified most debug capabilities are available. If LE CEEHDLR are used then TEST must be specified with OPT(1|2). (COBOL V5.1) (2.6-OPT) (x'03':1)
YW(1900:1999)	YEARWINDOW specifies 1st year in 100 year window. Requires DATEPROC option also be in effect. dropped at COBOL V5.1 (x'06':2)
b̸b̸DTR | NODTR	DIAGTRUNC causes the compiler to issue warning message when a MOVE statement that has numeric receivers, that have fewer integer positions than the sending item. <not in sib>
NS(DBCS|**NAT**)	NSYMBOL controls the compilers interpretation of the N symbol used in literals and PIC clauses. NATIONAL forces on DBCS. <not in sib>

b̸b̸RULES | **NORULES**
RULES(**ENDP**|NOENDP)
RULES(**EVENP**|NOEVENP)
RULES(**LXPRF**|NOLXPRF)
RULES(**SLCKB**|NOSLCKB)
	(COBOL V5.2)
	<not in sib>

b̸b̸DEF | **NODEF**	DEFINE specifies a value for a specific compilation variable.

	(COBOL V6.2) <not in sib>
ƀ ƀZC | **NOZC**	ZONECHECK causes compiler to generate IF NUMERIC class tests for every use of zone decimal data items. Options are ABD or MSG. (COBOL V6.1) <not in sib> (depreciated at Enterprise COBOL V6.1) refer to NUMCHECK
ƀ ƀ**ADV** | NOADV	ADV Adds one byte to the record length for the printer control character. (1,0-ADV) 85 Standard ADV.
APOST | **QUOTE**	Establishes value of figurative constant QUOTE to be either a quote or an apostrophe. The source code can use quotes or apostrophes to enclose literals. (1.1-APOST) 85 Standard QUOTE.
DATA(24|**31**)	Specifies where dynamic data areas WORKING-STORAGE and FD are obtained, above/below the line. LE HEAP run-time option is also involved. LE STACK run-time option controls LOCAL-STORAGE. DATA option has no effect if NORENT is also specified. DATA(24) is required if passing data to AMODE(24) programs. (1.2-31)
ƀ ƀDECK | **NODECK**	DECK directs 80-byte object code to SYSPUNCH. (1.3-D)
ƀ ƀDU | **NODU**	DUMP produces a system dump at compile time when there is an internal compiler error. (1.4-DU)
ƀ ƀDYN | **NODYN**	DYNAM dynamically loads subprograms invoked via CALL literal statement. DYNAM is not supported in the CICS environment. (1.5-DYN) 85 Standard DYNAM.

b̸ b̸FSRT | **NOFSRT**	FASTSRT will improve SORT I/O performance if SORT...USING or GIVING is specified. (1.6-FSRT) 85 Standard NOFSRT.
b̸ b̸FDUMP | **NOFDUMP**	 (COBOL II only) (1,7)
b̸ b̸SQLIMS | **NOSQLIMS**	SQLIMS enables the IMS SQL coprocessor. Suboptions are specified in a quoted string following SQLIMS. (COBOL V5.2) (1.7-SQLIMS)
b̸ b̸**LIB** | NOLIB	LIB is necessary if source program contains COPY or BASIS statement. (2.0-LIB) (Enterprise COBOL V5.1 always on)
b̸ b̸LIST | **NOLIST**	LIST will generate Assembler language listing. LIST and OFFSET are mutually exclusive. (2.1-LIST)
b̸ b̸MAP | **NOMAP**	MAP generates DATA DIVISION map listing. (2.2-MAP)
b̸ b̸MAP(**HEX**|DEC)	MAP generates DATA DIVISION map listing. (COBOL V5.2) <not in sib>
b̸ b̸NUMBER | **NONUMBER**	NUMBER causes source line numbers to be used in error messages instead of compiler generated line numbers. (2.3-NUM)
b̸ b̸**OBJ** | NOOBJ	OBJECT directs object code to SYSLIN DD. Where NOOBJ directs object code to SYSPUNCH DD. NOOBJ conflicts with TEST option. (2.4-OBJ)
b̸ b̸OFF | **NOOFF**	OFFSET will generate a condensed PROCEDURE DIVISION listing. OFF and LIST are mutually exclusive. (2.5-OFF)
b̸ b̸OPTIMIZE | **NOOPTIMIZE**	OPTIMIZE specifies the type of object code optimization that will occur.

	(2.6-OPT)
OPT(FULL|**STD**)	OPTIMIZE specifies the type of object code optimization that will occur. (COBOL for MVS, OS/390, Enterprise) (2,6-OPT) (26.4-FULL)
ƀƀOUT(ddname) | NOOUT	OUTDD directs DISPLAY output to a specific DD. LE uses option MSGFILE(SYSOUT) for its run-time messages. (2.7-OUT)
NUMPROC(MIG)	NUMPROC controls how pack-decimal and zone-decimal non-preferred signs are handled. Process signs like COBOL II. Dropped in COBOL V5.1 (5.0-MIG)
NUMPROC(ƀƀPFD|**NOPFD**)	NUMPROC controls how pack-decimal and zone-decimal non-preferred signs are handled. PFD is the most efficient and states that the compiler doesn't have to generate sign fix-up code. NOPFD informs the compiler to generate sign fix-up code (may be necessary when Assembler sub-programs are providing generated data. (3.0-PFD) 85 Standard NOPFD | MIG
ƀƀ**RENT** | NORENT	RENT generates re-entrant object code. RENT is highly recommended in the CICS environment. (3.1-RENT)
ƀƀRES | NORES	RES (at Enterprise COBOL 5.1 always on) (3.2)
ƀƀ**SEQ** | NOSEQ	SEQUENCE requires that source line numbers be in ascending sequence. Use NOSEQ with LIB. (3.3-SEQ)
SZ(**MAX**|n|nK)	SIZE specifies the amount of storage made available for the compiler front end. Dropped in COBOL V5.2 (3.4-MAX)
SZ(nK|**5000K**)	SIZE specifies the amount of storage made available for the compiler front end.

	Dropped at COBOL V5.2 (COBOL V5.1) <not in sib>
ƀ ƀ**SOURCE** | NOSOURCE	SOURCE print source statements in compiler output. (3.5-S)
ƀ ƀSSR | **NOSSR**	SSRANGE check subscripts, indexes, variable-length group ranges, and reference modification values at run-time. Creates larger object modules. Requires LE run-time option CHECK(ON) to work. May cause lots of run-time overhead. (3.6-SSR) (at Enterprise COBOL 5.1 you can no longer enable/disable the compiled checks at run-time via the LE CHECK(ON|OFF) option)

SSR(ƀ ƀZLEN|**NOZLEN**)
SSR(,**ABD**|MSG)
	(COBOL V6.2) (3.6-SSR) (23.5-ZLEN) (23.6-ABD)

ƀ ƀTERM | **NOTERM**	TERMINAL diagnostic messages are directed to SYSTERM DD. (3.7-TERM)
ƀ ƀTEST | **NOTEST**	TEST required if using IBM DEBUG Tool. TEST forces OBJECT and NODECK. (4.0-TEST)

TEST(STMT)
TEST(PATH)
TEST(BLOCK)
TEST(ƀ ƀSEP|NOSEP)
	TEST required if using IBM DEBUG Tool.
	
	(COBOL V3.1) (4.0-TEST) (26.1-STMT) (26.2-PATH) (26.3-BLOCK) (26.6-SEP)

TEST(**HOOK**)
TEST(ƀ ƀEJPD|**NOEJPD**)
TEST(ƀ ƀSEP|**NOSEP**)
	TEST required if using IBM DEBUG Tool.
	
	(COBOL V4.1) (4.0-TEST) (26.1:3-HOOK) (26.6-SEP) (27.7-EJPD)

TEST(ƀ ƀEJPD|**NOEJPD**)
TEST(ƀ ƀ**S**|NOS)
	TEST required if using IBM DEBUG Tool.
	
	(COBOL V5.1) (4.0-TEST) (27.4-S) (27.7-EJPD)

ƀ ƀTEST(ƀ ƀDWARF|NODWARF) ƀ ƀTEST(ƀ ƀEJPD|NOEJPD) ƀ ƀTEST(SEP(ƀ ƀDSNAME|	DWARF is industry standard debug information/protocol. EJPD provides control enablement of IBM DEBUG commands JUMPTO and GOTO (requires TEST and OPT(1|

NODSNAME)
ƀ ƀTEST(NOSEP)
ƀ ƀTEST(ƀ ƀSO|NOSO)
 	
default(s)
TEST(NOEJPD,DWARF,SO,NOSEP)
NOTEST(NODWARF,NOSO,NOSEP)
	2).
	SOurce includes source data in the DWARF data set. SEParate(NODSNAME) causes the DWARF data to be written to the SYSDEBUG DD. SEParate(DSNAME) causes the SYSDEBUG data set name to be saved in the executable.
	
	(COBOL V6.2)

TEST(ƀ ƀ**EJPD**|NOEJPD)	EJPD required to use IBM DEBUG Tool GOTO and JUMPTO commands. (COBOL V4.1) (4,0-TEST) (27.7-EJPD)
TEST(ƀ ƀ**S**|NOS)	SOURCE is included with the diagnostic information. (COBOL for MVS, OS/290, Enterprise) (4,0-TEST) (27.4-S)
TEST(ƀ ƀ**SEP**|NOSEP)	SEPARATE stores source listing in a separate file (SYSDEBUG DD) instead of executable or in DWARF. (COBOL V3.1) (4,0-TEST) (26.6-SEP)
NOTEST(ƀ ƀ**DWARF**|NODWARF)	DWARF is industry standard debug information/protocol. (COBOL V5.1) (29.7-DWARF)
TRUNC(BIN)	TRUNC specifies how binary fields will be handled. TRUNC processes based on maximum digits the hardware will handle. TRUNC has no effect on COMP-5 data, items are handled as if TRUNC(BIN) is in effect. (5.4-BIN)
TRUNC(**STD**|OPT)	TRUNC specifies how binary fields will be handled. TRUNC processes based on PIC clause. (4.1-STD) 85 Standard TRUNC(STD).
ƀ ƀWD(xxxx) | **NOWD**	WORD specifies a valid reserve word table (IGYCxxxx). WORD forces NOFLAGSTD. The compiler will use the table to verify that the verbs/commands are valid. The user can provide their own table to limit verb usage. (4.2-WD) 85 Standard is NOWORD.
ƀ ƀVBREF | **NOVBREF**	VBREF generates a verb cross-reference report. (4.3-VBREF)

ƀ ƀ**X** | NOX	XREF generates a variable cross-reference report. (4.4-X)
X(**FULL**|SHORT)	XREF generates a variable cross-reference report. <not in sib>
ƀ ƀ**ZWB** | NOZWB	ZWB remove sign from external-decimal (DISPLAY) fields when comparing them to alphanumeric fields. (4.5-ZWB) 85 Standard is ZWB.
ƀ ƀNAME | **NONAME**	NAME generates binder NAME statement in object code. (4.6-NAME)
NAME(ALIAS|**NOALIAS**)	NAME generates binder NAME statement in object code. <not in sib> 85 Standard ALIAS or NOALIAS.
ƀ ƀCMPR2 | NOCMPR2 reserved	COBOL II R2 compatible. (valid prior to Enterprise COBOL) (4.7-CMPR2)
NUMCLS(ALT|**PRIM**)	NUMCLASS specifies what signs are valid in numeric test. (5.1-ALT)
ƀ ƀ**DBCS** | NODBCS	DBCS recognizes shift-out (SO) and shift-in (SI) characters in non-numeric literals. NODBCS conflicts with option NSYMBOL(NATIONAL). (5.2-DBCS)
ƀ ƀAWO | **NOAWO**	Apply-Write-Only clause for all QSAM files with variable length records. (5.3-AWO)
ƀ ƀEVENTS | NOEVENTS	EVENTS (COBOL II) (5.5)
ƀ ƀADATA | **NOADATA**	Create SYSADATA DD compiler file. Can not specify in a PROCESS statement. (COBOL for MVS, OS/390, Enterprise) (5.5-ADATA)
ƀ ƀCURR | **NOCURR**	CURRENCY specify alternate currency symbol. Default is $. (COBOL for MVS, OS/390, Enterprise) (5.6-CURR)

CLASS | PROGRAM	Compilation unit is a CLASS or PROGRAM. (COBOL for MVS, OS/390, Enterprise) (5.7-CLASS)
ƀ ƀWSOPT|NOWSOPT	(COBOL V5) (8.3-WSOPT)
VS(**C**|S)	VSAMOPENFS affects file status reported from successful VSAM OPEN statements that require verified file integrity check. COMPAT returns file status 97 when an OPEN statement is successfully verified. SUCC returns file status 00 when an OPEN statement is successfully verified. (COBOL V6.1) (8.4-C)
ƀ ƀ**SUPP** | NOSUPP	Compiler adheres to SUPPRESS phrase on a COPY statement. (COBOL V6.1) (23.4-SUPP)
ƀ ƀ**INL** | NOINL	INLINE controls whether performed code is in-lined with the PERFORM statement. (COBOL V6.2) (23.7-INL)

RMODE(ANY|24)
RMODE(**AUTO**)
	RMODE specify residency mode. RMODE(AUTO) and NORENT then RMODE(24). RMODE(AUTO) and RENT then RMODE(ANY).
	(COBOL for MVS, OS/390, Enterprise) (26.0-ANY)

INTDATE(**ANSI**|LILIAN)	INTDATE specifies what beginning date is used. ANSI day 1 is 16010101. LILIAN day 1 is 15821015. (COBOL for MVS, OS/390, Enterprise) (26.5-LILIAN)

PGMN(LM|LU)
PGMN(**CO**)
	PGMNAME controls the handling of program names and entry point names. Options are COMPAT, LONGMIXED, or LONGUPPER.
	(COBOL for MVS, OS/390, Enterprise) (27.0-LU) (27.1-LM)
	85 Standard CO | LU and forces NONAME.

ƀ ƀDLL | **NODLL**	DLL generates object code suitable for dynamic link library. Required to support OO-COBOL. Forces NODYNAM and RENT. (COBOL for MVS, OS/390, Enterprise) (27.2-DLL) 85 Standard NODLL.
ƀ ƀEXP | **NOEXP**	EXPORTALL required for DLL. Forces DLL, NODYNAM, and RENT options.

	(COBOL for MVS, OS/390, Enterprise) (27.3-EXP) 85 Standard NOEXP.
ƀ ƀDP | **NODP**	DATEPROC provides support for millennium language extensions. dropped at COBOL V5.1 (COBOL for MVS, OS/390, Enterprise) (27.4-DP) 85 Standard NODATEPROC.
AR(**C**|E)	ARITH specifies the maximum number of digits that can be specified in the pack-decimal PIC clause. 18 for COMPAT. 31 for EXTEND. (COBOL for MVS, OS/390, Enterprise) (27.5-EXTEND)
ƀ ƀTHREAD | **NOTHREAD**	THREAD required for executing in LE enclave with multiple POSIX thread or PL/I tasks. THREAD forces RENT. (COBOL for MVS, OS/390, Enterprise) (27.6-THREAD) 85 Standard NOTHREAD.
ƀ ƀSQL | **NOSQL**	SQL indicates UDB/DB2 integrated translator is used. Requires LIB option. (COBOL for MVS, OS/390, Enterprise) (28.0-SQL)
ƀ ƀCICS | **NOCICS**	Enables the integrated CICS translator. Requires LIB option. (COBOL V3.1) (28.1-CICS) 85 Standard NOCICS.
ƀ ƀMD | **NOMD**	MDECK causes the compiler to write an updated source (SYSMDECK DD) after all library accesses for (COPY, BASIS, REPLACE, and EXEC SQL INCLUDE) are processed. Can not be set using the PROCESS statement. (COBOL V3.4) (28.2-MD)
MD(**C**OMPILE|**NO**COMPILE)	MDECK causes the compiler to write an updated source (SYSMDECK DD) after all library accesses for (COPY, BASIS, REPLACE, and EXEC SQL INCLUDE) are processed. Can not be set using the PROCESS statement. Options are COMPILE and NOCOMPILE. COMPILE specifies that the compile process will continue. NOCOMPILE specifies that the compile process will terminate after all source has been collected and printed. (SYSMDECK DD required at Enterprise COBOL 5.1 along

	with SYSUT1 thru SYSUT15) (COBOL V5.1) <not in sib>
␢␢**SQLC** | NOSQLC	SQLCCSID specifies whether the CODEPAGE compiler option will influence the processing of SQL statements. (COBOL V4.1 (28.3-SQLC)
␢␢OPTFILE | **NOOPTFILE**	OPTFILE specifies that compiler options are specified in the SYSOPTF DD. Must be specified via the PARM= or COBOL PROCESS clause. (COBOL for MVS, OS/390, Enterprise) (28.4-OPTFILE)
XP(C|**X**)	XMLPARSE specifies which XML parser to use (i.e. z/OS XML System Services parser or COBOL's built-in parser, which is compatible with Enterprise COBOL V3). (COBOL for MVS, OS/390, Enterprise) (28.5-XMLSS) (defaults to System Services parser at Enterprise COBOL 5.1)
␢␢BLOCK0 | **NOBLOCK0**	Change compiler QSAM default for QSAM files from unblocked to block (as if BLOCK CONTAINS 0 were specified). (COBOL V4.2) (28.6-BLOCK0) 85 Standard NOBLOCK0.
DS(**C**|S)	DISPSIGN controls output formatting for DISPLAY of signed numeric items. Options are COMPAT and SEP. Suggested setting would be SEP. (COBOL V5.1) (28.7-S)
␢␢STGOPT | **NOSTGOPT**	STGOPT discards non-referenced storage items. Can now use with OPT(2) and not discard W-S constant values (V6.2). (COBOL V5.1) (29.4-SO)
AFP(␢␢**VOLATILE**| NOVOLATILE)	Compiler generates extra code to protect the values of the Additional Floating Point registers (1,3,5,7,8:15). Use AFP(VOLATILE) if < CICS/TS 4.1. (COBOL V5.1) (29.5-VOLATILE)
AFP(␢␢VOLATILE| **NOVOLATILE**)	Compiler generates extra code to protect the values of the Additional Floating Point registers (1,3,5,7,8:15). Use AFP(VOLATILE) if < CICS/TS 4.1.

	(COBOL V6.2) (29.5-VOLATILE)
HGPR(ƀƀ**PRESERVE**| NOPRESERVE)	Controls the compiler usage of the 64-bit registers. (COBOL V5.1) (29.6-PRESERVE)
QUA(**C**|E)	QUALIFY controls whether to extend qualification rules so that some data items that cannot be referenced under COBOL Standard rules can be referenced. Options are COMPAT or EXTEND. (COBOL V5.2) (30.0-E)
VLR(C|**S**)	VLR affects the file status returned from READ statements for variable length records when the length returned is inconsistent with the RDW. Options are COMPAT or STANDARD. (COBOL V5.2) (30.1-C)
ƀƀCPYR | **NOCPYR**	COPYRIGHT (COBOL V5.2) (30.2-CPYR)
ƀƀSERV | **NOSERV**	 (COBOL V5.2) (30.3-SERV)

ZD(MIG)
ZD(**PFD**|NOPFD)
	ZONEDATA causes compiler to generate appropriate code to handle zone decimal data items. PFD assume zone bits are valid. NOPFD generate code to validate the zone bits. MIG generate code to match COBOL V4.
	
	(COBOL V5.2) (30.4-MIG) (30.5-NOPFD)

NONC
NC(ƀƀ**ZON**|NOZON)
NC(ƀƀ**PAC**|NOPAC)
NC(ƀƀ**BIN**|NOBIN)
NC(,ABD|**MSG**)
	NUMCHECK specifies that compiler will generate run-time code to validate sending data items.
	
	(COBOL V6.2) (31.0:2-NC) (31.0-ZON) (31.1-PAC)
	(31.2-BIN) (31.3-ABD)

ƀƀPC | **NOPC** PC(ABD|**MSG**,[n|**100**])	PARMCHECK directs compiler to generate extra data item following the last item (x'AA') in WORKING-STORAGE. (COBOL V6.2) (31.4-PC) (31.5-ABD)
ƀƀIC | **NOIC**	INITCHECK directs compiler to generate run-time checks for uninitialized data items. (COBOL V6.1) <not in sib>

Table 2.

Select Options for VERB Report

COBOL Verbs and Literals	
	Data Division items.
QSAM/SAM	(6.0)
VSAM ESDS	(6.1)
VSAM KSDS	(6.2)
VSAM RRDS	(6.3)
CODE-SET	(ASCII files) (6.4)
SPANNED RECORDS	(6.5)
DBCS PIC G	(COBOL II) (DBCS data item) (6.6)
DBCS PIC G/N	(DBCS data item) (6.6)
ODO	(6.7)
SYNCHRONIZED	(7.0)
JUSTIFIED	(7.1)
USAGE IS POINTER	(7.2)
COMPLEX ODO	(7.3)
EXTERNAL FLOATING POINT	(7.4)
INTERNAL FLOATING POINT	(7.5)
LINE-SEQUENTIAL FILE	(7.6)
USAGE IS PROCEDURE-POINTER	(COBOL for MVS) (7.7)
USAGE IS PROCEDURE/FUNCTION POINTER	(Enterprise COBOL) (7.7)
	Environment Division items.
FILE STATUS	(8.0)
RERUN	(8.1)
UPSI USED	(8.2)
WSOPT	(Enterprise COBOL V6) (8.3)
Used as a compiler option	(8.4)
	Procedure Division verb items.
ACCEPT	(9.0)
ADD	(9.1)
ALTER	(9.2)
CALL	(9.3)
CANCEL	(9.4)
CLOSE	(9.6)
COMPUTE	(10.0)
DELETE	(10.2)
DISPLAY	(10.4)
DIVIDE	(10.5)
END-PERFORM	(11.1)
ENTER	(11.2)
ENTRY	(11.3)

EXIT	(11.4)
EXEC	(Enterprise COBOL) (11.5)
GO TO	(11.6)
IF	(11.7)
INITIALIZE	(12.0)
INVOKE	(COBOL for MVS) (12.1)
INSPECT	(12.2)
MERGE	(12.3)
MOVE	(12.4)
MULTIPLY	(12.5)
OPEN	(12.6)
PERFORM	(12.7)
READ	(13.0)
RELEASE	(13.2)
RETURN	(13.3)
REWRITE	(13.4)
SEARCH	(13.5)
SET	(13.7)
SORT	(14.0)
START	(14.1)
STOP	(14.2)
STRING	(14.3)
SUBTRACT	(14.4)
UNSTRING	(14.7)
USE	(15.0)
WRITE	(15.1)
CONTINUE	(15.2)
END-ADD	(15.3)
END-CALL	(15.4)
END-COMPUTE	(15.5)
END-DELETE	(15.6)
END-DIVIDE	(15.7)
END-EVALUATE	(16.0)
END-IF	(16.1)
END-MULTIPLY	(16.2)
END-READ	(16.3)
END-RETURN	(16.4)
END-REWRITE	(16.5)
END-SEARCH	(16.6)
END-START	(16.7)
END-STRING	(17.0)
END-SUBTRACT	(17.1)
END-UNSTRING	(17.2)

END-WRITE	(17.3)
GOBACK	(17.4)
EVALUATE	(17.5)
SERVICE	(17.7)
END-INVOKE	(COBOL for MVS) (18.0)
END-EXEC	(Enterprise COBOL) (18.1)
XML	(Enterprise COBOL) (18.2)
END-XML	(Enterprise COBOL) (18.3)
ALLOCATE	(COBOL V6.1) (18.4)
FREE	(COBOL V6.1) (18.5)
JASON	(COBOL V6.1) (18.6)
END-JASON	(COBOL V6.1) (18.7)
	(19.0:7)
	(20.0:7)
	Procedure Division other items.
HEXADECIMAL LITERAL	(21.0)
ALTERED GO TO	(21.1)
I/O ERROR DECLARATIVE	(21.2)
LABEL DECLARATIVE	(21.3) obsolete
DEBUGGING DECLARATIVE	(21.4)
PROGRAM SEGMENTATION	(21.5)
OPEN...EXTEND	(21.6)
EXIT PROGRAM	(21.7)
CALL LITERAL	(22.0)
CALL IDENTIFIER	(22.1)
CALL...ON OVERFLOW	(22.2)
CALL...LENGTH OF	(22.3)
CALL...ADDRESS OF	(22.4)
CLOSE...REEL/UNIT	(22.5)
EXPONENTIATION USED	(22.6)
FLOATING POINT USED	(22.7)
COPY statement	(23.0)
BASIS	(23.1)
DBCS NAME USED	(23.2)
DBCS SHIFT-OUT/IN USED	(23,3)
Highest error severity at entry to ASM2 module	(23.4:7) obsolete
DBCS LITERAL	(24.0)
REPLACE	(24.1)
REFERENCE MODIFICATION USED	(24.2)
NESTED PROGRAMS	(24.3)
INITIAL	(24.4)
COMMON	(24.5)
SELECT...OPTIONAL	(24.6)

EXTERNAL	(24.7)
GLOBAL	(25.0)
RECORD IS VARYING	(25.1)
VOLATILE	(25.2) (COBOL V5.1)
ACCEPT FROM SYSIPT	(25.2) (n/a at COBOL V5.1)
DISPLAY UPON SYSLST	(25.3) (n/a at COBOL V5.1)
DISPLAY UPON SYSPUN	(25.4) (n/a at COBOL V5.1)
INTRINSIC FUNCTION USED	(25.5)
Z-LITERAL	(25.6) (COBOL V5.1)
RECURSIVE	(25.7) (COBOL V5.1)
JAVA BASED OO SYNTAX	(Enterprise COBOL) (29.0)
FUNCTION RANDOM USED	(Enterprise COBOL) (29.1)
NATIONAL	(Enterprise COBOL) (29.2)
XML PARSE SCHEMA	(Enterprise COBOL) (29.3)

Table 3.