S16150: What's New in COBOL Version 5 since GA

Tom Ross IBM

Aug 4, 2014

Title: What's new in COBOL v5 since GA

- Refresher about COBOL V5 requirements
- Service updates
- Improved compatibility
- New Function
- Improved performance
- What’s next

Refresher about Enterprise COBOL V5 compiler

- **Enterprise COBOL V5 (5655-W32) GA June 21st 2013**
- COBOL V5 executables are “Program Objects” not “Load Modules” and must be bound into a PDSE dataset
  - Customers using PDS load libraries for COBOL executables must migrate to PDSE load libraries prior to creating COBOL V5 executables. *There is no alternative to converting*.
  - Program Objects and PDSEs have many advantages over PDS datasets
    - Do not have to be taken down for compression or directory reallocation
    - Have a much larger size limit and enable more advanced compiler optimizations and larger source files
    - Allow the compiler to now and in the future take advantage of many modern system features (e.g. NOLOAD, xplink, AMODE 64, better page mapping etc).

Refresher about Enterprise COBOL V5 compiler

- **More working datasets required by COBOL v5**
  - SYSUT8-SYSUT15 are now required
  - SYSMDECK is now required
- **More memory required at compile time**
  - Recommend 200M region size
    - Earlier releases could compile most programs with 10M
    - You may have to change system user id limits
- **More time required to compile**
  - 5 to 15 times as much time, depending on OPTIMIZE level and source program size

Refresher about Enterprise COBOL V5 compiler

- Restrictions against mixing COBOL v5 programs with:
  - VS COBOL II NORES programs. Migrate to Enterprise COBOL.
  - OS/VS COBOL programs. Migrate to Enterprise COBOL.
    To find if you have any OS/VS COBOL programs you can:
    - LMA tool of Debug Tool to scan load libraries for OS/VS COBOL programs
    - Edge Portfolio Analyzer to scan load libraries for OS/VS COBOL programs
    - APAR PM86742 for Language Environment adds warning messages about detected OS/VS COBOL programs at run time:

```
IGZ0268W An invocation was made of OS/VS COBOL program
"program-name"
```

```
IGZ0269W "program-lang" version "program-version" program
"program-name" made a call to OS/VS COBOL program  "program-
name".
```

See COBOL v5 Migration Guide for full details

- http://www-01.ibm.com/support/docview.wss?uid=swg27036733

Refresher about Enterprise COBOL V5 compiler

- There is good news too!
- Performance improvement of 10% or more
- Computation intensive programs can get 20% or more improvement
- New XML GENERATE features
- New XML PARSE feature
- UTF-8 Intrinsic Functions
- Inline comments
- DWARF debugging information in NOLOAD class segments

Refresher about Enterprise COBOL V5 compiler

- There is more good news!
- Raised Working-Storage Section size to 2GB (from 128MB)
- Larger individual data item limit to 999,999,999 bytes
- Support for UNBOUNDED tables in Linkage Section (useful for XML processing)
- SMF 89 records
- Improved listings
- Improved Debug Tool experience with COBOL V5
- And much more!

Service updates

Enterprise COBOL for z/OS v5 – Service updates

- Five Service Refreshes (PTFs) Since GA :
  - September 2013 PTF
  - October 2013 PTF
  - January 2014 PTF
  - March 2014 PTF (5.1.1: AMODE 24)
  - May 2014 PTF
  - July 2014 PTF
- More information available at "**Fix list for Enterprise COBOL for z/OS**":
  - http://www-01.ibm.com/support/docview.wss?uid=swg27041164
- We strongly encourage you to apply all service to the compiler and runtime for the best performance, usability and latest functionality

Enterprise COBOL for z/OS v5 – Service updates

- Latest service news. . . **APAR PI18087**
- Customer in USA discovered a defect in compiler optimization
- Programs with consecutive arithmetic statements that use data from and store into the same data item and are compiled with OPT(1 or 2)
  - The data items are PACKED-DECIMAL or DISPLAY
    (This does not occur with BINARY or FLOATING-POINT types)
  - The second calculation is either:

    ```
    DESTVAR = constant – DESTVAR
    ```
    or
    ```
    DESTVAR = DESTVAR * (-1)
    ```
- Optimizer did not preserve the starting value in some cases
- Incorrect results, possible data integrity issue
- HIPER PTF now available (May 31, 2014): **UI18382**

Enterprise COBOL for z/OS v5 – Service updates

- Latest service news. . . APAR PI18087
- Examples:

```
1   TODAY-YYMMDD                 PIC S9(7)V COMP-3


    COMPUTE TODAY-YYMMDD = TODAY-YYMMDD - 2
    COMPUTE TODAY-YYMMDD = +9999999 - TODAY-YYMMDD


    COMPUTE TODAY-YYMMDD = TODAY-YYMMDD + 3
    COMPUTE TODAY-YYMMDD = TODAY-YYMMDD * -1.
```

Enterprise COBOL for z/OS v5 – Service updates

- Post – GA Performance and Functional Highlights Include:
  - PM92523 : IMS Support Enhancement (EXEC SQLIMS)
  - PM92585 : Collection of various performance and functional improvements identified during the beta program
  - PM93979 : Performance Improvements For Working-Storage Initialization
  - PM95711 : Performance Improvements For Procedure-Pointer Calls
  - PI09629 : Performance Improvements for UNSTRING
  - PI12151 and PM93583 : COBOL Runtime and Compiler Enhancements for improved AMODE 24 support
  - PI13636: Reduced CPU and memory for OPT compiles

Enterprise COBOL for z/OS 5.1.<u>1</u> - Service updates

- PTFs Released March 21 / 2014 (as March PTF on COBOL for z/OS 5.1)
- Key improvements added in 5.1.1
  - Support for AMODE 24 static linking
    http://www-01.ibm.com/support/docview.wss?uid=swg21667752
  - New function code added to vendor interface API routine IGZXAPI to query WORKING-STORAGE area address
  - Various performance improvements
    - Better detection and optimization of DATA(24) read-only data-items
    - More use of Decimal Floating Point (DFP) for COMPUTE statements with many sub-expressions
    - More efficient conversions between internal/external decimal and floating point (common for programs with exponentiation using a fractional exponent)

Customer migration experience update

- Not all incorrect programs are diagnosed as incorrect
- Programs that set the value of an ODO object to outside of the legal range:

```
77      VAR1 COMP-3 PIC 9(3).
01 X.
    02 VAR2 PIC X OCCURS 0 to 1 depending on VAR1.

MOVE 128 to VAR1          *>  This could be legal…
MOVE ALL ’C’ to X         *>  But this is illegal!
```

- Results:
  - For V2, V3, V4: 128 bytes of 'C' were moved
  - For V5R1: 1 byte of 'C' and 127 bytes of junk was moved

Differences in running the new programs

- Parameter length mismatch:

```
WORKING-STORAGE SECTION.
77   GRP1 PIC X(100).
01   FILE-STATUS-DATA-ITEMS.
   02   OTHER-SENSITIVE-DATA-ITEMS …
. . .
    Call ‘SUBP’ Using GRP1.

Program-Id. SUBP.
Linkage Section.
 01  GRP2 PIC X(500).
Procedure Division Using GRP2
    MOVE ‘stuff’ To GRP2(300:320)     *>  This is illegal!
```

Results:

- For V2, V3, V4: Illegal program did not fail
- For V5R1: File-status in CALLER was changed, flow changed

Improved compatibility
AMODE 24

AMODE 24 and COBOL 5.1

Restrictions Before 5.1.1 (March 2014 PTF)

- Restrictions from 5.1 GA to before March 2014 PTF:
  - Dynamic CALLs between COBOL V5 and AMODE 24 programs *were* supported (use DATA(24) option)
  - Before the March 2014 PTF, COBOL 5.1 forced the entry points of COBOL programs to be AMODE 31.
    - This means that 5.1.0 object modules cannot be statically linked with AMODE 24 objects.
    - AMODE 24 execution was *not* supported.
  - To work around this restriction:
    - Convert AMODE 24 programs to AMODE 31 or
    - Convert static to dynamic CALL and use DATA(24)

AMODE 24 and COBOL 5.1

Improvements added in COBOL 5.1.1 (March 2014 PTF)

- Improved compatibility:
- AMODE 24 static call restriction in COBOL 5.1 was one of the biggest migration inhibitors
- See PM93583 (Compiler) and PI12151 (Runtime)
  - http://www-01.ibm.com/support/docview.wss?uid=swg21667752
- IBM has worked to relax this restriction. Entry points in object modules generated by the COBOL 5.1 compiler now have the AMODE MIN attribute by default (unless certain COBOL features are used, see below). The COBOL object modules can then be statically linked with 24-bit object files.
- ***AMODE 24 execution is now supported*** as of the March 2014 PTF.
  - With restrictions, of course ☺

AMODE 24 and COBOL 5.1
Improvements added in COBOL 5.1.1 (March 2014 PTF)

- ***AMODE 24 execution is now supported*** as of the March 2014 PTF.
- Will require linking with RMODE(24) and AMODE(24) binder options
- We now have the same set of AMODE restrictions as COBOL V4.2.
  In the following cases AMODE 31 must be used:
  - Programs containing XML PARSE or GENERATE statements
  - Programs containing the object-oriented language syntax, such as INVOKE or object-oriented class definitions
  - Programs compiled with any of the following compiler options:
    - DLL, PGMNAME(LONGUPPER) or PGMNAME(LONGMIXED)
  - Multithreaded applications
  - Programs run from the z/OS UNIX file system
  - Programs used as COBOL compiler exits (EXIT compiler option)
  - Program objects containing COBOL bound together with C, C++ or PL/I, communicating with static CALL.

New Function
SQLIMS

IMS SQL coprocessor support

- Enterprise COBOL V5.1 plus PTF UK98481 (APAR PM92523)
- IMS V13 plus PTF UK98028 (APAR PM97137)
- IMS SQL support will not be available via separate precompiler nor in COBOL compilers earlier than V5

New SQLIMS option

- Every COBOL program that contains EXEC SQLIMS statement requires the SQLIMS option to be specified
- The IMS SQL coprocessor options could be specified as suboptions of SQLIMS option Example: SQLIMS('APOSTSQL')
- IMS SQL options:
  - APOSTSQL QUOTESQL COMMA PERIOD
- SQLIMS option can be specified in any of the compiler option sources: compiler invocation, PROCESS or CBL statements, or installation default.
- When compiling a batch of programs. The SQLIMS option is in effect only for the first program in the batch.

IMS SQL Language Support

- Support basic binary and character host variables data types
- Support only Code pages 37 and 1140
- EXEC SQLIMS INCLUDE statement is supported
- EXEC SQLIMS statements could be specified in any nested programs
- EXEC SQLIMS statements could be used in copybooks
- REPLACE statement works on SQLIMS statement
- SQLIMS option can not be specified for COBOL Classes

Coprocessor coexistence

- Both SQL and SQLIMS compiler options can be in effect for a given COBOL program
- EXEC SQL and EXEC SQLIMS statement can coexist in the same COBOL program

Coprocessor Interface

- Adopted the DB2 SQL coprocessor interface
- IMS acquired the DB2 coprocessor code
- COBOL compiler uses the DB2 coprocessor code to support the IMS coprocessor
- Advantages:
  - Using a proven, reliable interface
  - Minimize the implementation cost
  - Minimize the cost of future IMS SQL coprocessor support by other languages

IMS SQL Coprocessor APIs

- Same as DB2 coprocessor APIs:
  - DQFHPSRV IMS coprocessor interface module

– SQLIINIT	Initialization
– SQLIFINI	Termination
– SQLIINTP	Option processing
– SQLICMP2	Compile
– SQLIALHV	Host Variable definition

Coprocessor infrastructure

- PLIST structure used as the interface mechanism between the COBOL application and the IMS SQL runtime services.
- The compiler generates the PLIST structure in Working-Storage section.
- The compiler sets fields in the PLIST structure for every EXEC SQLIMS statement and then invokes the IMS SQL runtime service that execute the statement.

Improved performance

COMPILE TIME PERFORMANCE IMPROVEMENTS

Reduced cost of optimization

- PI13636 changed the optimizer to use less memory and CPU
- One customer reported a program that compiled with 1 CPU second using COBOL V4, but used 13 CPU minutes when compiling with COBOL V5 and OPT(2)
- When they tried the same program after installing the fix for PI13636, it took 23 CPU seconds with COBOL V5 and OPT(2)
- We will continue in these efforts!

Improved performance

RUN TIME PERFORMANCE IMPROVEMENTS

Optimization of Packed Decimal To Hex Floating Point Conversions

```
1 p5v2 pic s9(5)v9(2) COMP-3.
1 p3v0 pic s9(3)      COMP-3.
1 p3v2 pic s9(3)v9(2) COMP-3.
...
COMPUTE p5v2 = p3v0 ** p3v2
```

New in 5.1.1

V4

- ***Library Call for packed to float conversion***
- ***Runtime path length is > 100 instructions***

```
L      2,300(0,9)
ZAP    368(3,13),16(3,2)
MVC    288(76,13),53(10)
UNPK   316(4,13),368(3,13)
MVC    304(12,13),231(10)
MVC    351(1,13),276(10)
L      3,92(0,9)
L      15,268(0,3)  V(IGZCFPC )
LA     1,322(0,10)
BASR   14,15
```

V5

- ***Packed to float conversion is inlined using a more efficient algorithm***
- ***OPT(1 or 2), any ARCH level***

```
ZAP     440(8,R13),160(2,R8)
CVB     R0,440(,R13)
CDFR    FP0,R0
...
CXFR    FP4,R0
DXR     FP4,FP1
AXR     FP4,FP0
```

Timing (10 million in a loop)

```
V5 : 3.09 cpu seconds
V4 : 3.60 cpu seconds
```

V5 is 14% faster

Optimization of Large Intermediate Result COMPUTE Statements

```
1 p9v7  pic s9(9)v9(7)  comp-3.
1 p10v7 pic s9(10)v9(7) comp-3.
1 p3v7  pic s9(3)v9(7)  comp-3.
1 p5v2  pic s9(5)v9(2)  comp-3.
...
COMPUTE p5v2 = (100 * p9v7 +
 p10v7) /  (1 + p3v7 * p9v7)
```

New in 5.1.1

V4

- ***Intermediate result is 33 digits***
- ***Library routine for divide***

```
L      2,300(0,9)
ZAP    352(16,13),0(6,2)
MP     360(8,13),50(2,10)
AP     360(8,13),8(7,2)
ZAP    368(16,13),16(6,2)
MP     372(12,13),0(6,2)
AP     373(11,13),44(6,10)
MVC    384(32,13),72(10)
MVC    401(8,13),360(13)
NI     408(13),X'F0'
MVN    415(1,13),367(13)
L      3,92(0,9)
L      15,180(0,3)    V(IGZCXDI )
```

V5

- ***Limit for inlining with DFP raised so large COMPUTE statements can be optimized now***
- **`OPT(1 or 2), ARCH(8)`**

```
ZAP      352(14,R13),152(6,R8)
SRP      359(7,R13),2,0
AP       359(7,R13),160(7,R8)
SRP      352(14,R13),14,0
...
CXSTR    FP1,R0
DXTR     FP0:FP2,FP4:FP6,FP1:FP
FIXTR    FP4:FP6,9,FP0:FP2
SRXT     FP0:FP2,FP4:FP6,5
CSXTR    R0,FP0
ST       R1,176(,R8)
ZAP      176(4,R8),176(4,R8)
```

Timing (100 million in a loop)

```
V5 : 0.65 cpu seconds
V4 : 17.5 cpu seconds
```

V5 is 96% faster

Optimization of Decimal Multiply by 100

```
1 z5v2 pic s9(5)v9(2).
1 z7v2 pic s9(7)v9(2).
...
Compute z7v2 = z5v2 * 100
```

V4

- ***Uses expensive decimal multiply MP***

```
PACK  352(6,13),0(7,2)
MP    352(6,13),46(2,10)
ZAP   353(5,13),353(5,13)
UNPK  8(9,2),353(5,13)
```

V5

- ***Uses less expensive decimal "shift"***
- ***Applies to any power of 10 operand***
- ***Any OPT level, any ARCH level***

```
PACK    312(5,R13),40(7,R8)
SRP     312(5,R13),2,0
UNPK    48(9,R8),312(5,R13)
```

Timing (100 million in a loop)

```
V5 : 0.19 cpu seconds
V4 : 0.93 cpu seconds
```

V5 is 80% faster

What’s next / Improved compatibility

Enterprise COBOL for z/OS 5.1.1

- What's next?
- XMLPARSE(COMPAT) returns!
- We found that many customers had programs that use the COMPAT parser:
  - XML PARSE
  - Enterprise COBOL V3
  - Enterprise COBOL V4 w/XMLPARSE(COMPAT)
- Changing source to migrate to new compiler prohibitive
- XMLPARSE compiler option will be added back to COBOL V5 via service stream, hopefully in 3rd quarter 2014

Enterprise COBOL for z/OS 5.1.1

- Going forward COBOL for z/OS will continue to deliver new features and deliver release to release performance improvements
  - to improve efficiency and capacity
  - to simplify programming and debugging
  - to increase productivity
  - to modernize existing business critical applications

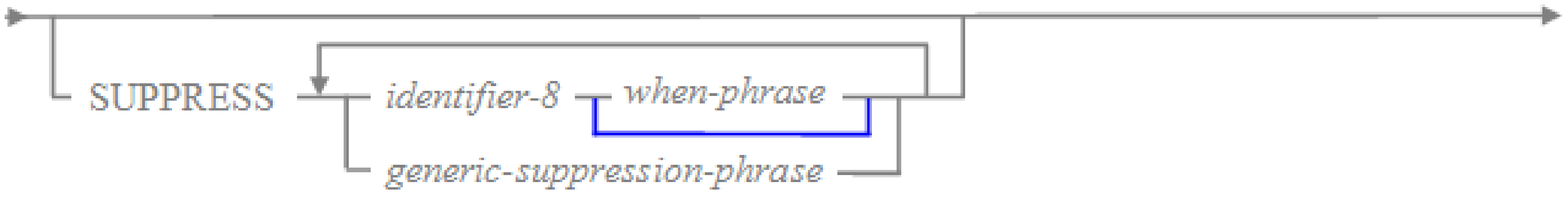
Enterprise COBOL for z/OS Vnext

Speaking of going forward …

- If we added a suboption to MAP to select hexadecimal or decimal offsets in MAP compiler option output, what should the default be for MAP specified by itself?
  - –Currently, in V4 and earlier, MAP gives hex offsets
  - –In current V5, MAP gives decimal offsets
- Many prefer hex, especially when using listings from both V4 and V5 and calculating offsets using MAP output
- Often customers have MAP specified in many places already
- Would it be OK if we changed COBOL V5 so that MAP by itself gave hexadecimal offsets?
- There would be a new compiler option MAP(HEX|DEC)
  - –Any compilation can get either hex or decimal
  - –We would support MAP by itself and interpret it as…
  - –MAP(HEX) ?

Enterprise COBOL for z/OS Vnext

- Speaking of going forward …
- IF we added unconditional suppression for XML GENERATE… how should we do it?
- XML GENERATE … SUPPRESS phrase



- If we allowed SUPPRESS identifier-8 with no WHEN, would that be useful? (Unconditional suppress based on data item name)
- Should we allow identifier-8 to specify a group?
- If we did, should the SUPPRESS
  - –only suppress the group tags or
  - –should it suppress the entire group and its contained data items?
  - –Give the user the choice somehow? Add an ALL keyword?

  - `SUPPRESS identifier-8 ALL`

Enterprise COBOL for z/OS 5.1.1

QUESTIONS?