COBOL V5.2 Was Announced! What's New?

Session 16609

Speaker: Tom Ross (Captain COBOL)

March 2, 2015

Insert Custom Session QR if Desired.

COBOL V5.2 Was Announced! What's New?

- Agenda
  - New hardware exploitation: z13
  - Continuous improvement!
  - Announcing access to z/OS JSON services from COBOL
  - New compiler options
  - New COBOL language
  - Easier migration path

Introducing Enterprise COBOL V5.2

- Announced January 13, 2015
- GA February 27
- PID 5655-W32
- The second release of IBM’s groundbreaking new COBOL compiler technology that was first introduced in June, 2013

Enterprise COBOL V5.2

- **New hardware exploitation: z13**
  - The first time COBOL has announced new hardware support on Day 1
  - SIMD (Vector Processing) instructions used for some INSPECT TALLYING or REPLACING statements
    - Single Instruction Multiple Data (SIMD) instructions for vector operations
  - More use of DFP (Dec Floating Point) instructions for PACKED-DECIMAL data
  - New instruction scheduler at OPT(2) and ARCH(11) tuned to new micro architecture

Cool new IBM z13!

(Updated) ARCH quick reference

- ~~ARCH(6)~~
  - ~~2084-xxx models (z990) 2086-xxx models (z890)~~
- ARCH(7)
  - 2094-xxx models (IBM System z9 EC) 2096-xxx models (IBM System z9® BC)
- ARCH(8)
  - 2097-xxx models (IBM System z10 EC) 2098-xxx models (IBM System z10 BC)
- ARCH(9)
  - 2817-xxx models (IBM zEnterprise z196 EC) 2818-xxx models (IBM zEnterprise z114 BC)
- ARCH(10)
  - 2827-xxx models (IBM zEnterprise EC12) 2828-xxx models (IBM zEnterprise BC12)
- **ARCH(11)**
  - **2964-xxx models (IBM z13)**

Z13: SIMD (Vector Facility) Instructions

- For INSPECT TALLYING and INSPECT REPLACING

Sample code:

```
WORKING-STORAGE SECTION.
 01 VARS.
   02 STR PIC X(255).
   02 C PIC 9(5) COMP-5 VALUE 0.
PROCEDURE DIVISION.
   MOVE ALL 'abc def ghi jkl ' TO STR
   PERFORM 100000000 TIMES
     INSPECT STR TALLYING C FOR ALL ' '
   END-PERFORM

   GOBACK
```

Z13: SIMD (Vector Facility) Instructions

V5.1
- ARCH(10)

```
        LHI     R0,0xff
        XR      R1,R1
        LA      R12,152(,R8)    #   STR
L0064:  EQU     *
        CLI     0(,R12),X'40'   #
        JNOP    L0066
        LA      R1,1(,R1)       #
L0066:  EQU     *
        LA      R12,1(,R12)     #
        BRCT    R0,L0064
        A       R1,407(,R8)     #   C
        ST      R1,407(,R8)     #   C
```

V5.2
- *ARCH(11)*

```
        LHI     R0,0xfe
        XR      R1,R1
        LA      R12,152(,R8)        #   STR
        VREPIB  VRF27,0x40
        VGBM    VRF25,0x0
L0066:  EQU     *
        VLL     VRF24,R0,0(,R12)    #
        AHI     R12,0x10
        VCEQB   VRF24,VRF24,VRF27
        AHI     R0,0xfff0
        VLCB    VRF24,VRF24
        VAB     VRF25,VRF25,VRF24
        JNL     L0066
        VGBM    VRF26,0x0
        VSUMB   VRF25,VRF25,VRF26
        VSUMQF  VRF25,VRF25,VRF26
        VLGVG   R1,VRF25,1(,R1)     #
        A       R1,407(,R8)         #   C
        ST      R1,407(,R8)         #   C
```

Timing (100 million times in a loop)

```
V5.1 : 46.63 cpu seconds
V5.2 :  1.54 cpu seconds
```

V5.2 is 30 times faster or 97% less CPU !!!!

z13: DFP (Decimal Floating Point) Instructions

- For Packed-Decimal (COMP-3) data items

Sample code:

```
WORKING-STORAGE SECTION.
      01 VARS.
        02 A PIC S9(25) Packed-Decimal VALUE +1234567890123456789012345.
        02 B PIC S9(25) Packed-Decimal VALUE +2468097531246809753124680.
        02 C PIC S9(25) Packed-Decimal VALUE 0.
      PROCEDURE DIVISION.
          PERFORM 100000000 TIMES
            DIVIDE A BY B GIVING C
          END-PERFORM
```

z13: DFP (Decimal Floating Point) Instructions

V5.1
- ARCH(10)

```
XGR      R0,R0
ICMH     R0,X'1',152(,R8)       #   A
L        R0,153(,R8)            #   A
LG       R1,157(,R8)            #   A
CXSTR    FP0,R0
XGR      R0,R0
ICMH     R0,X'1',165(,R8)       #   B
L        R0,166(,R8)            #   B
LG       R1,170(,R8)            #   B
CXSTR    FP1,R0
DXTR     FP4:FP6,FP0:FP2,FP1:FP3
FIXTR    FP0:FP2,9,FP4:FP6
CSXTR    R0:R1,0,FP0:FP2
STCMH    R0,X'1',178(,R8)       #   C
ST       R0,179(,R8)            #   C
STG      R1,183(,R8)            #   C
ZAP      178(13,R8),178(13,R8)  #   C
```

V5.2
- *ARCH(11)*

```
CXPT     FP0:FP2,152(13,R8),0x8
CXPT     FP1:FP3,165(13,R8),0x8
DXTR     FP4:FP6,FP0:FP2,FP1:FP3
FIXTR    FP0:FP2,9,FP4:FP6
CPXT     FP0:FP2,178(13,R8),0x9
AHI      R2,0xffff
CIJ      R2,L0034,0,HT(mask=0x2),
```

Timing (100 million times in a loop)

```
V5.1 :  2.53 cpu seconds
V5.2 :  1.63 cpu seconds
```

V5.2 uses 36% less CPU

Enterprise COBOL V5.2

- **Continuous improvement!**
- Many improvements were added to V5.1 in response to customer's needs and are included in the base of V5.2:
  - AMODE 24 support
  - XMLPARSE(COMPAT)
  - VLR(COMPAT)
  - MAP(HEX)
  - ZONEDATA(MIG)
  - These features all ease migration from earlier COBOL compilers

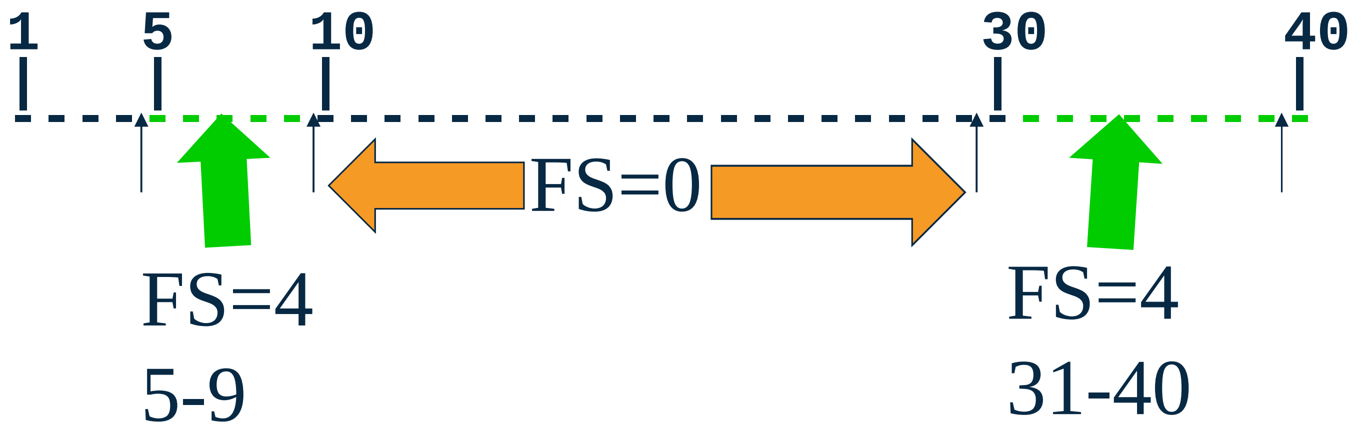
Enterprise COBOL V5.2

- AMODE 24 support
  - AMODE 24 mode execution now allowed in the same cases as V3 and V4
    - Static linking to AMODE 24 assembler programs the big driver here
- XMLPARSE(COMPAT)
  - No need to change XML PARSE statements to move to V5!
- VLR(COMPAT)
  - You can choose safer Standard-conforming or compatible behavior with previous compilers
  - We have added a message to help improve programs in the spirit of the Standard-required FS 04 ‘wrong-length READ cases’

Enterprise COBOL V5.2

- When does the COBOL Standard say that you should get FS=04 for READ statements of variable-length files?

```
FD Record Is Varying From 5 To 40
             Depending on X.
01 Rec1 PIC X(10).
01 Rec2 PIC X(30).
```



Enterprise COBOL V5.2

- MAP(HEX)
  - Customers requested decimal offsets for MAP output, so we changed in V5.1
  - Doing math on V4 and V5 listings at the same time was a problem!
  - Now users can choose MAP(DEC) for decimal offsets or
  - MAP(HEX) for hexadecimal offsets like previous compilers
  - MAP by itself will be interpreted as MAP(HEX)

Enterprise COBOL V5.2

- ZONEDATA(MIG|PFD)
  - Customers moving from NUMPROC(MIG) who also have invalid data got different results when moving to COBOL V5 and NUMPROC(NOPFD)
  - WORKING-STORAGE SECTION.

```
 77  VALUE0    PIC X(4) VALUE '00 0'.         *>  x'F0F040F0'
 77  VALUE1    REDEFINES VALUE0 PIC 9(4).
 77  VALUE3    PIC X(4) VALUE '00A0'.         *>  x'F0F0C1F0'
 77  VALUE4    REDEFINES VALUE3 PIC 9(4).

PROCEDURE DIVISION.
   IF VALUE1 = ZERO
      DISPLAY "VALUE1 = ZERO  " VALUE1
   ELSE
      DISPLAY "VALUE1 NOT = ZERO  " VALUE1
   END-IF

   IF VALUE4 = 10
      DISPLAY "VALUE4 = 10  " VALUE4
   ELSE
      DISPLAY "VALUE4 NOT = 10  " VALUE4
   END-IF
```

Enterprise COBOL V5.2

- Results with
  COBOL V4 and earlier with NUMPROC(MIG) or
  COBOL V5 with ZONEDATA(MIG):

```
VALUE1 = ZERO   00 0
VALUE4 = 10     00A0
```

- Results with
  COBOL V4 and earlier with NUMPROC(PFD or NOPFD) or
  COBOL V5 with ZONEDATA(PFD):

```
VALUE1 NOT = ZERO   00 0
VALUE4 NOT = 10     00A0
```

Enterprise COBOL V5.2

- **Announcing access to z/OS JSON services from COBOL**
  - *What’s the problem here?*
    - Need for generic Web Services client support
  - Why did the z/OS BCPii development team write this?
- *The z/OS Client Web Enablement Toolkit*
  - JSON Parser
  - HTTP/HTTPS z/OS client services

No z/OS Client Generic Web Services

- ***A few z/OS implementations here and there***
  - REXX & Curl in USS
  - Socket from COBOL
  - Apache http client from Java
  - DB2 REST UDF
  - CICS Sockets
  - WOLA & Liberty Profile
- ***No generic web services available to all z/OS clients***
- ***No general usage JSON parser available in all z/OS environments***

Enterprise COBOL V5.2

- **Announcing access to z/OS JSON services from COBOL**
  - The new z/OS Client Toolkit JSON parser enables COBOL programmers to parse or create a JSON document which interfaces with clients such as mobile
- **Announcing access to HTTP/HTTPS services from COBOL**
  - The new z/OS Client Toolkit HTTP services enable COBOL programmers to:
    - Init a connection.
    - Set the desired HTTP connect options.
    - Issue the HTTP connect.
    - Init a request.
    - Set the desired HTTP request options.
    - Issue the HTTP request.
    - Process the response.
    - Term the request or re-use.
    - Term the connection

z/OS Client Web Enablement Toolkit Objectives

- *Externalize http and https client functions in an easy-to-use generic fashion for users in almost any z/OS environment*
- *Implement a native z/OS JSON parser*
- *Make sure the web services and the payload processing of the HTTP request body and response body are independent entities*
  - HTTP/HTTPS functions separate from JSON support
- *Make the interface intuitive for people familiar with programming in this area*
- *Make the solution z/OS-responsible*

Enterprise COBOL V5.2

- **Announcing access to z/OS JSON services from COBOL**
  - Install the PTF for APAR OA46575 to enable z/OS Client Web Enablement Toolkit support on z/OS V2.1.
  - Initially, COBOL development will provide coding examples
    - In updated manuals or Tech Notes and SHARE presentations
  - Later on, COBOL language extensions
    - Similar to XML PARSE and XML GENERATE

Enterprise COBOL V5.2

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
* Parses the sample JSON data.                                    *
*                                                                 *
* Services Used:                                                  *
*                                                                 *
* HWTJPARS: Builds an internal representation of the specified    *
*           JSON string. This allows efficient search, traversal,*
*           and modification of the JSON data.                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
 parse-json-text.
      CALL 'HWTJPARS' USING HWTJ-RETURN-CODE HWTJ-PARSERHANDLE
          json-text-ptr   *> address of JSON text string (input)
          json-text-len   *> length of JSON text string (input)
          HWTJ-DIAG-AREA

     IF (HWTJ-OK)
       DISPLAY 'SUCCESS: JSON data parsed.'
     ELSE
       DISPLAY 'ERROR: Unable to parse JSON data.'
       CALL 'DISPDIAG' USING HWTJ-RETURN-CODE HWTJ-DIAG-AREA
     END-IF
```

Enterprise COBOL V5.2

- **New compiler options**
  - QUALIFY option extends ability to reference ambiguous data items
  - RULES option to help programmers write better code
  - COPYRIGHT and SERVICE options to better manage applications
  - Plus (from V5.1 service stream)
    - XMLPARSE(COMPAT|XMLSS)
    - VLR(COMPAT|STANDARD)
    - MAP(HEX|DEC)
    - ZONEDATA(MIG|PFD)

Enterprise COBOL V5.2

- **Changed compiler options**
  - ARCH
    - ARCH(6) not allowed
    - ARCH(7) is the new default
    - ARCH(11) added for z13
  - SIZE
    - Gone! Improved memory management by COBOL compiler

Enterprise COBOL V5.2

- QUALIFY(EXTEND|COMPAT) compiler option
  - **QUALIFY(COMPAT)**
    - If QUALIFY(COMPAT) is in effect, references to data items must be unique.
  - **QUALIFY(EXTEND)**
    - If QUALIFY(EXTEND) is in effect, qualification rules are extended so that some references that are not unique by COBOL standard rules can be unique.
    - If every level in the containing hierarchy of names is specified, the set of qualifiers is called a *complete set of qualifiers*.
    - If there is only one data item with a specific complete set of qualifiers, the reference resolves to that data item, even if the same set of qualifiers can match with another reference as an incomplete set of qualifiers.

Enterprise COBOL V5.2

- QUALIFY(EXTEND|COMPAT) compiler option
  - QUA(EXTEND) extends ability to reference ambiguous data items

```
01 A.
   02 B.
      03 C PIC X.     *> C of A
      03 A PIC X.
   02 C PIC X.        *> Also C of A

Move Z to C of A      *> Refers to 02 level C (unique only with QUA(EXTEND))
Move Z to A           *> Refers to 01 level A (unique only with QUA(EXTEND))
Move Z to C of B of A *> Refers to 03 level C (unique by COBOL std rules)
Move Z to C of B      *> Refers to 03 level C (unique by COBOL std rules)
```

Enterprise COBOL V5.2

- RULES compiler option
  - Similar to PL/I RULES option
  - Allows users to enforce or recommend ‘proper’ coding
  - Satisfies many user requirements/requests
  - Default is:
    - RULES(ENDPERIOD,EVENPACK,LAXPERF,SLACKBYTES)

Enterprise COBOL V5.2

- Suboptions for new compiler option RULES in V5.2:
  - NOENDPERIOD (NOENDP):
    - Flag conditional statements terminated by a period instead of an explicit scope terminator (END-*)
  - NOEVENPACK (NOEVENP):
    - Flag even number of digits in PACKED-DECIMAL (Packed-Decimal) data descriptions
  - NOLAXPERF (NOLXPRF)
    - Flag COBOL features that are opportunities for performance improvement
  - NOSLACKBYTES (NOSLCKB)
    - Flag data definitions that get slack bytes added when SYNCHRONIZED specified

Enterprise COBOL V5.2

- Here is the list of things that RULES(NOLAXPERF) will flag:
  - Flag inefficient loops

    ```
    Perform varying ix1 from 1 by 1 until ix1 > ix1-max
    ```

    - If ix1 is coded as display value (not packed or binary)
    - Different data types used for different operands in the VARYING clause
  - Accessing a Table item with a subscript defined without binary/packed
  - MOVEs (COMPUTEs, comparisons) with conversion of numeric data because of different storage representation
  - MOVE of character-string to another variable, but with lots of padding (100 bytes or more), like:
    - from-field pic x(10) to-field pic x(32000)
    - move from-field to to-field, which will move 10 bytes and then fill spaces into another 32KB of Memory
  - Slow or non-optimal compiler options:
    - NOAWO, NOBLOCK0, NOFASTSRT, NUMPROC(NOPFD), OPT(0), SSRANGE, TRUNC(STD|BIN), ZONEDATA(MIG)

Enterprise COBOL V5.2

- COPYRIGHT(‘string’) compiler option
  - Places string in object program
  - For vendors who ship COBOL executables
- SERVICE(‘string’) compiler option
  - Places string in object program
  - For users to manage applications
  - When doing ‘Software Archeology’ you can get compile date, compiler version, release, mod, but IBM does not put a service level in the object
  - Now users can set a service string to identify which service level of which compiler was used to compile a program

Enterprise COBOL V5.2

- **New COBOL language**
- Improved XML GENERATE
  - More powerful SUPPRESS capabilities
- New VOLATILE attribute and SERVICE LABEL functionality
  - Enable OPT(1|2) compilation of User-Written condition handlers
- >>CALLINTERFACE directive
- New features from 2002 COBOL Standard
  - Plus a list in the Language Reference Manual of which parts are supported

Enterprise COBOL V5.2

- Improved XML GENERATE: SUPPRESS capabilities
  - The WHEN phrase of the XML GENERATE statement can be omitted to allow unconditional suppression of a named identifier when generating XML output.
    - If the WHEN phrase is omitted, the identifier can be a group data item.
  - A new keyword CONTENT is added to the generic-suppression-phrase to limit suppression to only TYPE IS CONTENT items

Enterprise COBOL V5.2

- New VOLATILE attribute and SERVICE LABEL functionality
  - Enable OPT(1|2) compilation of applications with User-Written condition handlers
  - VOLATILE can also be used in conjunction with STGOPT
    - Specify VOLATILE for unused data items that you don’t want optimized
      - WORKING-STORAGE eye-catcher
      - Code control time stamp
      - Etc

VOLATILE example – using LE condition handlers

*> Main program (causes divide-by-zero exception):

```
identification division.
program-id. main.
data division.
local-storage section.
77 user-handler procedure-pointer.
77 token   pic S9(9) comp.
01 qty     pic 9(8) binary.
01 divisor pic 9(8) binary value 0.
01 answer  pic 9(8) binary.
01 step    pic 9(8) binary value 0 external volatile.
:
Set user-handler to entry 'handler'
Call 'CEEHDLR' using user-handler, token, null
Compute step = 2                  *> optimizer thinks this is a "dead store" and removes it
Compute answer = number / divisor *> divide-by-zero exception occurs here, handler is invoked,
                                  *> and reference to "step" is made but hidden from compiler
Display 'answer = ' answer

Compute step = 3
Display 'step = ' step
Compute answer = qty + 2
```

The solution is to use the "volatile" clause.

*> Condition handler program:

```
identification division.
program-id. handler.
data division.
local-storage.
01 step pic 9(8) external.
procedure division.
:

display 'Error: a problem was encountered in step ' step.
```

The handler program can reference data item "step", defined in main program, but the main program doesn't know this without the "volatile" keyword. Thus, the wrong value of step may be displayed.

Enterprise COBOL V5.2

- >>CALLINTERFACE directive
  - Controls the calling convention of calls (DLL, DYNAM or STATIC) on a **call-by-call** basis.
  - Syntax: >>CALLINTERFACE [DLL | DYNAM | STATIC]
    - Abbreviation: >>CALLINT
  - Overrides settings of DLL and DYNAM compiler options
  - Stays in effect until the next >>CALLINTERFACE directive is encountered, or the end of the program.
- Benefits:
  - Allows easy mixing of call types regardless of the setting of the DLL and DYNAM options.
  - Allows mixing of DLL, Dynamic and Static calls in any programs!
  - With >>CALLINTERFACE, the call type can be controlled very easily.
- e..g,

```
>>CALLINTERFACE DLL
CALL ‘SUB1’   *> DLL call
CALL ‘SUB2’   *> DLL call
>>CALLINTERFACE DYNAM
CALL ‘SUB3’   *> dynamic call
>>CALLINTERFACE
CALL ‘SUB4’   *> Uses current DLL or DYNAM option
              *> setting to determine calling convention
```

Enterprise COBOL V5.2

- Other examples

```
*> Program compiled with DYNAM

CALL ‘SUB5’  *> Dynamic call

>>CALLINTERFACE DLL
CALL ‘SUB6’  *> DLL call
>>CALLINTERFACE

CALL ‘SUB7’  *> Dynamic call

*> All calls will be dynamic until ..
>>CALLINTERFACE Static
*> All calls will be Static calls
 CALL ‘SUB8’  *> Static call
```

Enterprise COBOL V5.2

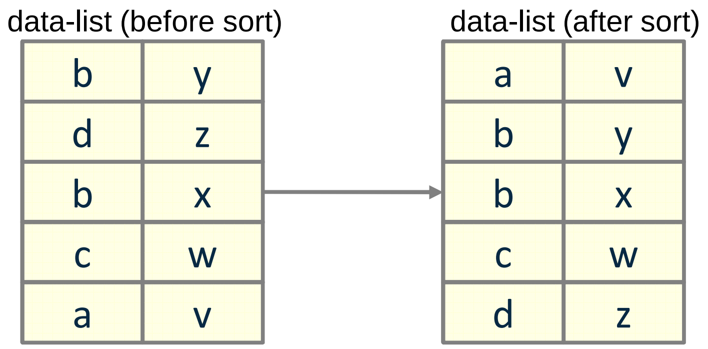
- **New features from 2002 COBOL Standard:**
  - Format 2 of SORT: the table SORT statement
    - Arranges table elements in a user-specified sequence.
  - New formats of EXIT statement:
    - EXIT SECTION
    - EXIT PARAGRAPH
    - EXIT PERFORM
    - EXIT PERFORM CYCLE
  - Improved COPY REPLACING statement
    - LEADING and TRAILING phrases (better partial word replacement)
    - Nested COPY REPLACING now supported (IBM Extension)
  - Improved REPLACE statement
    - LEADING and TRAILING phrases (better partial word replacement)

Format 2 SORT: Table SORT statement

- The table SORT statement causes table elements to be arranged in a user-specified sequence.
- Syntax:
  SORT *data-name-2* [ON { ASCENDING | DESCENDING } KEY [*data-name-1*]...]...[WITH DUPLICATES IN ORDER] [COLLATING SEQUENCE IS *alphabet-name-1*]

```
01 mydata.
   03 data-list occurs 5 times.
      05 data-key pic x.
      05 data-nonkey pic x.
```

```
SORT data-list ON ASCENDING KEY data-key
     WITH DUPLICATES IN ORDER
     COLLATING SEQUENCE Standard-1
```



data-list (before sort)

b	y
d	z
b	x
c	w
a	v

data-list (after sort)

a	v
b	y
b	x
c	w
d	z

EXIT statement enhancements

- EXIT [SECTION | PARAGRAPH]
  - EXIT SECTION – leaves current section
  - EXIT PARAGRAPH – leaves current paragraph
- EXIT PERFORM [CYCLE]
  - EXIT PERFORM leaves an inline PERFORM block
    - Similar to `LEAVE` in PL/I and `break` in 'C'
  - EXIT PERFORM CYCLE skips the remainder of the current iteration of an inline PERFORM block
    - Similar to `ITERATE` in PL/I and `continue` in 'C'

EXIT statement enhancements

```
working-storage section.
01 n pic 99.
procedure division.
s1 section.
p1.
  display 'In p1'
  if n < 5 then
    exit paragraph  *> proceed to p2
  else
    if n < 10 then
      exit section  *> proceed to s2
    end-if
  end-if
  display 'After if-statement in p1'.
p2.
  display 'In p2'.
```

```
s2 section.
  perform 10 times
    if n = 5 then
      exit perform *> exit perform loop
    end-if
    display 'perform1 n=' n
  End-perform
  *> exit perform passes control here
  display 'N=' n
  perform varying n from 1 by 1 until n = 10
    if n = 5 then
      exit perform cycle *> next iteration
    end-if
    display 'perform2 n=' n
    *> exit perform cycle passes control here
  End-perform
  goback.
```

Enterprise COBOL V5.2

- **Improved COPY REPLACING statement**
  - LEADING and TRAILING phrases (better partial word replacement)
    - LEADING and TRAILING phrases of the REPLACE statement and the REPLACING phrase of the COPY statement allow replacement of partial words in source text and library text. This is useful for prefixing and postfixing names.
  - Nested COPY REPLACING now supported (IBM Extension)
    - Old rule:
      - » COPY statements can be nested. However, nested COPY statements cannot contain the REPLACING phrase, and a COPY statement with the REPLACING phrase cannot contain nested COPY statements
    - One COPY REPLACING allowed in a ‘chain’
      - » Pgm A has COPY B, B has COPY C, C has COPY D REPLACING
- Improved REPLACE statement
  - LEADING and TRAILING phrases (better partial word replacement)

New: COPY REPLACING LEADING/TRAILING support

- Syntax for COPY…REPLACING when LEADING/TRAILING specified:

COPY <copy-file-name> [...] REPLACING **{LEADING | TRAILING}**
== *partial-word-1*== BY == *partial-word-2* == … .

```
COPY PAYLIB REPLACING LEADING == DEPT == BY == PAYROLL ==.
```

PAYLIB before replacement

```
01 DEPT.
   02 DEPT-WEEK       PIC S99.
   02 DEPT-GROSS-PAY PIC S9(5)V99.
   02 DEPT-HOURS      PIC S999
      OCCURS 1 TO 52 TIMES
      DEPENDING ON DEPT-WEEK OF
      DEPT.
```

Source after COPY of PAYLIB and replacement

```
01 PAYROLL.
   02 PAYROLL-WEEK       PIC S99.
   02 PAYROLL-GROSS-PAY PIC S9(5)V99.
   02 PAYROLL-HOURS      PIC S999
      OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF
      PAYROLL.
```

Nested COPY REPLACING

- An example of a nested copy replacing operation. Here, a program includes PAYLIB, which itself includes PAYLIB2. The result of the files after replacement is on the right.

```
  COPY PAYLIB REPLACING
       LEADING  == DEPT == BY == PAYROLL ==
       TRAILING == GROSS-PAY == BY == NET-PAY ==.
```

PAYLIB

```
01 PAYROLL.
   02 PAYROLL-WEEK PIC S99.
   02 DEPT-GROSS-PAY PIC S9(5)V99.
   02 PAYROLL-HOURS PIC S999 OCCURS 1 TO 52 TIMES
            DEPENDING ON PAYROLL-WEEK OF PAYROLL.
COPY PAYLIB2.
```

PAYLIB2

```
01 PAYROLL2.
   02 PAYROLL2-WEEK PIC S99.
   02 DEPT2-GROSS-PAY PIC S9(5)V99.
   02 PAYROLL2-HOURS PIC S999 OCCURS 1 TO 52 TIMES
           DEPENDING ON PAYROLL2-WEEK OF PAYROLL2.
```

Resulting source after replacement

```
01 PAYROLL.
   02 PAYROLL-WEEK PIC S99.
   02 PAYROLL-NET-PAY PIC S9(5)V99.
   02 PAYROLL-HOURS PIC S999 OCCURS 1 TO 52 TIMES
           DEPENDING ON PAYROLL-WEEK OF PAYROLL.

01 PAYROLL2.
   02 PAYROLL2-WEEK PIC S99.
   02 PAYROLL2-NET-PAY PIC S9(5)V99.
   02 PAYROLL2-HOURS PIC S999 OCCURS 1 TO 52 TIMES
           DEPENDING ON PAYROLL2-WEEK OF PAYROLL2.
```

Enterprise COBOL V5.2 - Migration

- New minimum hardware requirement
  - Z890, z990 not supported by COBOL V5.2
    - Also not supported by z/OS V2.1
- Otherwise the same prereqs as COBOL V5.1, except:
  - Java V5 went EOS Sept 2013
- New FIXCAT keyword to find PTFs required by other products:
  - SET BDY(GLOBAL)
    REPORT MISSINGFIX ZONES(ZOS13T,ZOS13P)
    FIXCAT(IBM.TargetSystem-RequiredService.Enterprise-COBOL.V5R2)
- Migrating from COBOL V5.1 to V5.2
  - Similar to V4.1 to V4.2
  - Easy, no regression testing required
- Migrating from older compilers to COBOL V5.2
  - Different from V4.1 to V4.2
  - Same as V3 or V4 to V5.1
  - Not so easy, lots of regression testing recommended