What's New in COBOL for z/OS

Tom Ross

Session 8211

SHARE Tampa

Feb, 2007

Enterprise COBOL V3R4 Overview

- Raise 16Mb COBOL data item size limit
- Unicode support stage 2
- Intrinsic function NUMVAL, NUMVAL-C enhancements
- DB2 enhancements
- REDEFINES enhancement
- MDECK compiler option
- SEARCH ALL statement updates

COBOL data item size limits

In prior COBOL releases:

each COBOL data item limited to 16Mb

```
01 X pic X(16777215).
01 G.
   02 V pic s9(9) binary occurs 4194303 times.
01 HV1 usage SQL type is CLOB(16M).
```

- Working-storage section limited to 128Mb
- Linkage section, local-storage section also each limited to 128Mb.

New data item size limits

- Working-storage section: 128Mb total *(unchanged)*
- Linkage section: 128Mb total *(unchanged)*
- Local-storage section: 128Mb total *(unchanged)*
- Individual data items: up to 128Mb
  (limited only by aggregate size limit)
- Picture clause replication:

  01 A PIC X(134217727).
- OCCURS integer:

  05 V PIC X OCCURS 134217727 TIMES.

Examples: large COBOL data items

- DB2 large object host variables:
  - BLOB
  - CLOB
  - ...

```
01 HV1 usage SQL type is CLOB(40M).

01 HV2 usage SQL type is BLOB(100M).
```

Examples: large COBOL data items…

- XML document processing

```
01 XMLdocument PIC N(20000000) national.

XML PARSE XMLdocument
        processing procedure P
XML GENERATE XMLdocument
        from group1
```

Examples: large COBOL data items…

- Large arrays

```
01 MATRIX.
    02 occurs 5000 times.
        03 occurs 5000 times.
            04 X pic S9(8) binary.
* (100 megabytes!)
```

- Existing large group structures may exceed 16Mb when converted to Unicode

Large data items: performance

- Operations on large data items use new z/Architecture instructions:
  - MVCLE (move long extended)
  - CLCLE (compare logical long extended)

Unicode support stage 2

- Rationale:
  - Enable conversion of applications to run entirely on Unicode data
  - Improve usability based on feedback from users of Unicode stage 1
  - Implement more complete subset of COBOL 2002 language for internationalization

Example problems

```
01 G.
   02 G1 pic N(10) national.
   02 G2 pic N(5) national.
01 A pic N(10) national.
01 B pic N(30) national.
```

Inspect G tallying tally for leading spaces

- *looks for EBCDIC spaces not Unicode spaces!*

String G A into B

- *illegal mixture of national and alphanumeric (group) operands!*

Example problems…

```
01 G.
   02 G1 pic N(10) national.
   02 G2 pic N(5) national.
01 H.
   02 H1 pic N(10) national.
   02 H2 pic N(10) national.
```

Move G to H

- *pads with EBCDIC spaces not Unicode spaces!*

Solution: GROUP-USAGE NATIONAL clause

```
01 G group-usage national.
   02 G1 pic N(10).
   02 G2 pic N(5).
```

With GROUP-USAGE NATIONAL clause:

- Group is treated like an elementary national data item
  (with a few exceptions …)
- Subordinate items must all be national
  (implicitly or explicitly)
- USAGE NATIONAL implied for all subordinate items if not explicitly specified

GROUP-USAGE NATIONAL – exception cases

```
01 G group-usage national.
   02 G1 pic N(10).
   02 G2 pic N(5).
```

G is still treated as a group, by:

- INITIALIZE
- MOVE CORRESPONDING
- ADD/SUBTRACT CORRESPONDING
- XML GENERATE … FROM G
- DB2, if used as host variable in EXEC SQL statement

Caution

- GROUP-USAGE NATIONAL changes a group to act as an elementary item
- *In COBOL, groups and elementary items do not behave the same way!*
  - All group moves are legal
  - Group moves/compares are byte-wise
  - No conversions or editing/de-editing, unlike operations on elementary items

Group-usage national vs. usage national

```
01 G usage national.
   02 G1 pic N(10).
   02 G2 pic N(5).
```

- *USAGE clause at group level applies specified USAGE to each subordinate item*
- *G continues to behave as a group, not as an elementary national item*

Problem …

```
01 G.
   02 A pic X(5).
   02 B pic XX/XX/XX.
   02 C pic S99V99.
   02 D pic $$$,$$$,$$9.99CR.
   02 F +9.9999E+99.
```

How to convert to a national group, and process as Unicode? … all items in a national group must be USAGE NATIONAL

Solution: new Unicode data types

- National decimal
- National edited
- National numeric edited
- National floating point

Converted group example …

```
01 G group-usage national.
   02 A pic N(5) usage national.
   02 B pic NN/NN/NN usage national.
   02 C pic S99V99 usage national
          sign leading separate.
   02 D pic $$$,$$$,$$9.99CR national.
   02 F pic +9.9999E+99 usage national.
```

Simpler …

```
01 G group-usage national.
   02 A pic N(5).
   02 B pic NN/NN/NN.
   02 C pic S99V99 sign leading separate.
   02 D pic $$$,$$$,$$9.99CR.
   02 F pic +9.9999E+99.
```

USAGE NATIONAL is implied for each elementary item by the GROUP-USAGE NATIONAL clause

National decimal data type

01 X pic S9(9)V99 national sign leading separate.

- Numeric data, encoded in Unicode
- Use wherever numeric data is supported
  - Arithmetic, MOVE, IF, Perform N times, subscript, INSPECT tallying, …
- Use any numeric PICTURE clause
- Specify USAGE NATIONAL
- If signed, must specify:
  - SIGN LEADING SEPARATE, or
  - SIGN TRAILING SEPARATE

SIGN clause for national group

```
01 NG group-usage national sign is leading separate.
   02 A pic N(5).
   02 X pic S9(29)V99.
   02 Y pic S9(9).
   02 Z pic 9(5).
   02 F pic +9.9999E+99.
```

- SIGN clause can also be specified at group level
  - applies to each numeric item with picture character S
- A, X, Y, Z, F all implicitly USAGE NATIONAL.
- X, Y are implicitly SIGN LEADING SEPARATE.

National-edited data type

01 B pic NN/NN/NN usage national.

- Analogous to alphanumeric-edited data, but encoded in Unicode
- PICTURE symbols:
  - N B 0 /
  - At least one N
- Specify USAGE NATIONAL

National numeric-edited data type

01 E pic Z(9)9V99 usage national.

- Numeric-edited data, encoded in Unicode
- Use wherever numeric-edited data is supported
- Use PICTURE symbols:

  B P V Z 9 0 / , . CR DB * *cs*

- Specify USAGE NATIONAL

Currency sign clause still must specify alphanumeric literals

```
Configuration section.
Special-names.
   Currency sign is '€'
   Currency sign is 'EUR' with picture symbol '@'.
Data Division.
Working-storage section.
01 P  pic €€€9.99CR usage national.
01 T  pic @@@,@@@,@@9.99+ usage national.
```

National floating point

01 F +9.9999E+99 usage national.

- Analogous to external floating point, but encoded in Unicode
- Use wherever external floating point is supported
- Automatically converted to internal floating point for processing
- Specify any external floating point PICTURE
- Specify USAGE NATIONAL

Intrinsic function NUMVAL

- NUMVAL now accepts national argument

```
01 N pic N(50) usage national value N" + 123456.78     ".
01 Y pic S9(20)V99 usage national sign leading separate.

Procedure division.
    Compute Y = function numval(N)
```

Intrinsic function NUMVAL-C

- NUMVAL-C now accepts:
  - national argument-1
  - (optional) national currency sign 2nd argument
  - currency sign with multiple characters
    - such as "EUR"
- If program has multiple CURRENCY SIGN clauses, NUMVAL-C must specify explicit currency sign as 2nd argument

Intrinsic function NUMVAL-C…

```
Configuration section.
Special-names.
    Currency sign is 'EUR' with picture symbol '@'.
Data Division.
Working-storage section.
01 P pic N(50) value N"    EUR  123,456.78   CR     ".
01 X pic S9(20)V99 usage national sign leading separate.
Procedure division.
Compute X = function numval-c(P)
```

Intrinsic function NUMVAL-C…

```
Configuration section.
Special-names.
    Currency sign is '€'
    Currency sign is 'USD' with picture symbol '$'
    Currency sign is 'EUR' with picture symbol '@'.
Data Division.
Working-storage section.
01 P pic N(50) value N"    EUR  123,456.78   CR     ".
01 X pic S9(20)V99 usage national sign leading separate.
Procedure division.
Compute X = function numval-c(P, N"EUR")
```

Performance

- Implementation of national numeric types based on z/Architecture ***extended-translation facility 2***
- New hardware instructions:
  - PKU and UNPKU
  - efficiently convert national decimal to/from packed decimal

Miscellaneous

- Alphanumeric literal supported as value clause on national item

  01 Adresse pic N(30) value '101 Plieninger Straße'.

- FILE STATUS data name may be usage national
- HIGH-VALUE, LOW-VALUE in national context

  Move high-value to Adresse

Caution: do not mix Unicode and EBCDIC versions of HIGH-VALUE

HIGH-VALUE in mixed context

- Caution: do not mix Unicode and EBCDIC versions of HIGH-VALUE
- X'FFFF is not a valid Unicode character
- X'FF' is not a valid EBCDIC character
- Unicode <-> EBCDIC conversion will result in substitution character

```
01 E pic XX  value high-values.          (X'FFFF')
01 U pic NN value high-values.           (X'FFFFFFFF')
If E = U then …
```

Compare will fail, because E is converted to Unicode for comparison,
X'FF' is converted to substitution character, not to X'FFFF'

New DB2 support

- Zoned decimal host variables supported by coprocessor

```
01 HV1 PIC S9(5) USAGE DISPLAY SIGN LEADING SEPARATE.
```

- National decimal items supported as host variables

```
01 HV2 PIC S9(5) USAGE NATIONAL SIGN LEADING SEPARATE.
```

- @,#,$ in EXEC SQL INCLUDE names

New DB2 support …

- A national group can be used as a host variable in EXEC SQL statements
- Note: treated as a *group* by DB2
  - that is, as short hand for a series of host variables
  - *not* treated as an elementary national item

REDEFINES

- Note that REDEFINES is already very flexible
- 01-level item can redefine a smaller item!

```
01 A pic X(5).
01 B redefines A pic X(10).
01 C redefines A pic X(15).
01 D pic 99.
```

compiler ensures that D starts 15 bytes after A

REDEFINES …

- ISO COBOL standard requires multiple REDEFINEs to all reference the original defining item for the storage
- IBM compilers (and others) have extension:

```
01 A1 pic X(5).
01 B1 redefines A1 pic X(10).
01 C1 redefines B1 pic X(15).
```

REDEFINES …

- For non-01 item, redefine of smaller item was not allowed in prior releases:

```
01 G.
    05 A1 pic X(5).
    05 B1 redefines A1 pic X(10).
    05 C1 redefines B1 pic X(15).
```

- S-level diagnostic generated by prior releases of IBM COBOL

REDEFINES enhancement

- New V3R4 compiler now allows this, with warning level diagnostic:

```
01 G.
    05 A1 pic X(5).
    05 B1 redefines A1 pic X(10).
    ...
```

IGYDS1154-W B1 redefined a smaller item. The program was accepted as written.

- Warning message is still generated, to help identify unintended coding errors

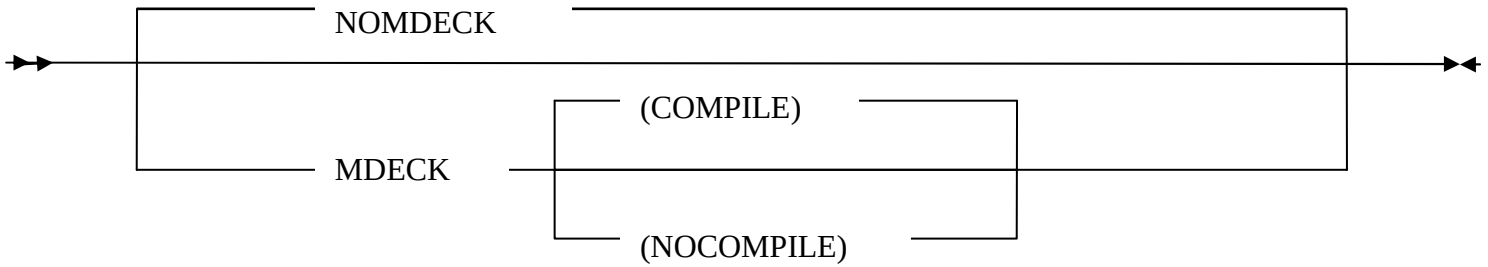
REDEFINES enhancement - rationale

- After converting a complex REDEFINE chain to Unicode, items may not be in decreasing order of size
- Flexible REDEFINE common on various UNIX COBOLs
  - new support eases migration to z/OS

MDECK compiler option

- Expanded COBOL source output from library processing written to file
  - COPY, BASIS, REPLACE,
    EXEC SQL INCLUDE are expanded
- Output written to
  - SYSMDECK DD *(batch JCL or TSO)*
  - *filename*.DEK *(z/Series Unix cob2 command)*

MDECK option syntax



MDECK(COMPILE): continue compile after generating expanded output

MDECK(NOCOMPILE): terminate compile after output

MDECK with no suboption: MDECK(COMPILE) implied

Abbreviations: NOMD, MD, MD(C), MD(NOC)

Default: NOMDECK

MDECK with SQL or CICS coprocessor

- COPY statements expanded in output file
- EXEC CICS statements included as-is
- EXEC SQL INCLUDE statements expanded
  - except for EXEC SQL INCLUDE SQLCA
  - SQLCA is generated rather than included from SYSLIB
  - SQLCA contents vary depending on SQL options specified
- Other EXEC SQL statements included as-is

MDECK and COBOL Service

- When reporting a compiler problem to IBM Service, sending MDECK output will make the problem easier for IBM to reproduce.

Prerequisites

- z/OS 1.4 or later
- z/Architecture processor with extended-translation facility 2 *(for Unicode support)*
- Language Environment:
  PTFs for APAR PQ95214
- Enterprise COBOL V3R4 compiler:
  PTF for APAR PK07977
- DB2 coprocessor:
  PTF for APAR PQ93857

V3R4 Migration – SEARCH ALL statement

- LE APAR PQ95124 corrected errors in the COBOL SEARCH ALL statement
- However, users have found COBOL applications that depend on the erroneous behavior!
- New APARs:

  PK19573/PK15432 (LE/COBOL library)

  PK16765 (V3R4 compiler)

  change the product so the corrected results are only generated for programs compiled with V3R4

SEARCH ALL corrections

With prior COBOL releases:

1. When search argument is longer than table key, in some cases argument matches key even when excess digits/characters are not zeros/blanks:

```
01 Arg1 pic X(8).          "ABCDxxxx".
   ...
03 Key1 pic X(4).          "ABCD"
```

2. Signed search argument with negative value may match unsigned table key:

```
01 Arg2 pic S9(4).         -1234
   ...
03 Key2 pic 9(4).           1234
```

Note these should compare unequal! Just as in:

```
IF Key(IX) = Arg …

SEARCH … WHEN Key(IX) = Arg …          (serial search, no ALL…)
```

V3R4 compiled SEARCH ALL WHEN will compare unequal as well.

SEARCH ALL statement – V3R4 migration

- V3R4 compiler (with PTF for APAR PK16765) generates warning messages for SEARCH ALL statements that might have changed results
  - whether an actual change occurs depends on the contents of the search argument at run time

- LE/COBOL runtime (with PTFs for APAR PK15432) generates warning messages if search argument value can never match a table key
- Goto www.ibm.com/software/awdtools/cobol/zos/support/
  - Search for support item **1243387**