Solving the Year 2000 Problem with COBOL/VSE and LE/VSE

Document Number COB2000V

May 1995

Jim Alexander
Clive Nealon

IBM COBOL Family
Australian Programming Centre

Contents

The ″YEAR 2000″ problem

- **What is the †Year 2000† problem?**
- **Can I survive the turn of the century?**
- **What are my options?**

This paper is meant to explain what the ″YEAR 2000″ problem is and how you can solve it for COBOL applications.

If ALL of your applications currently identify the year by using the full four digit number - for example, 1995 - and use that value for all date arithmetic, then you are unlikely to encounter problems. If, however, your applications use only the last two digits of the year - 95, then the switch from 1999 to 2000 is going to cause problems in any date arithmetic calculations. (4-digit year dates are not available using DOS/VS COBOL or VS COBOL II.)

There is no way to avoid the pain of changing databases and applications to use 4-digit year dates, but there are 3 ways to solve the problem using COBOL/VSE and LE/VSE.

IBM provides 2 long-term solutions. One is to use COBOL/VSE Intrinsic Functions and the other is to use LE/VSE callable services. Both of these methods return 4-digit year dates.

IBM does provide a short-term solution but it is inevitable that you must use 4-digit years at some point in the future. Our short term solution is the ′century window′ feature of LE/VSE that allows 2-digit years to be interpreted in a 100-year window since each 2-digit year number can only occur once in any 100-year period. You select the period, give LE/370 a 2-digit year, and LE/370 will give you the date in that 100-year window.

The example shown in this paper using the short-term Century window feature uses only the LE/VSE callable services method, and, as a result, can be adapted for use with PL/I programs.

What is the ″YEAR 2000″ problem?

2-digit year data is the problem

- **On 1/1/2000, the current year will be less than the previous year**
  - **Date field has only the last 2 digits of year**
  - **00 is NOT greater than 99**

There are NO 4-digit year dates with DOS/VS COBOL or VS COBOL II

Depending on the coding technique, you might not be able to tell that a customer had turned 18, or whether a particular payment is due!

Many businesses are experiencing problems already, as the turn of the century approaches.

What will happen when the day arrives?

Suddenly contracts due early in the new century are 100 years late.

How old is a person whose year of birth is ′00′?

There is no way to get 4-digit year dates using DOS/VS COBOL or VS COBOL II

Long-term solution

- **You MUST eventually:**
  1. **Re-write applications**
     - **Use LE/VSE callable services to get 4-digit year dates OR**
     - **Use COBOL Intrinsic Functions to get 4-digit year dates**
  2. **AND Rebuild databases with 4-digit years**

In the long term you will have to rewrite applications using 4-digit year dates, and you will need to respecify databases to use 4-digit years.

The applications can be rewritten to use either Callable Services provided with LE/VSE, or Intrinsic Functions provided as part of the COBOL/VSE compiler.

Once rewritten to use 4-digit year dates, your applications should out-live most people programming today, taking you through till the year 9999.

There is an easier, short-term solution to this problem.

Short-term solution

- **Use the LE/370 century window**
  - **Advantages:**

    Change only a few programs, no database change

    Cheaper, quicker, easier
  - **Disadvantages:**

    This only buys you a few years, depending on the application

    You still must change all date programs and databases eventually

It may not be possible to change all applications and data before the year 2000 rolls around.

To cater for this the LE/VSE century window gives you the ability to interpret 2-digit year dates as 4-digit year dates. This approach gives you the opportunity to change the minimum number of programs and alleviates the need to change the database. This results in a short-term solution that can be implemented easily, quickly, and more cheaply.

Of course, this is only an interim solution. It will provide you with breathing space but only temporarily.

It will not remove the requirement for change to all date programs and databases.

The century window cannot be seen as a permanent solution.

It is a mathematical fact that a 2-digit year can only be unique in a given 100-year period. Over time you will need more than 100 years for your data window - in fact, many companies need more than 100 years now!

The reason that the century window buys you time is that you know in any given application whether you are trying to figure out how old someone is (the date is in the past) or whether a due date has been reached (the date is in the future).

There are limitations. For example, the century window cannot solve the problem of trying to figure out how old a customer is if the customer is older than 100 years and you only have 2-digit year dates! For this problem and others you need COBOL/VSE Intrinsic Functions or LE/VSE date services.

The LE/VSE century window and COBOL

- **Only available through LE/VSE and COBOL/VSE**
  - **Use the century window feature of LE/VSE:**
    - **CALL †CEEDAYS† USING input-date PICSTR**
      LILIAN FC.

      Results in an integer representation of date in days
    - **CALL †CEEDATE† USING input-date PICSTR**
      output-char-date FC.

      Results in a character representation of date with a 4-digit year
- **See example MAIN that calls SUBRTN**

The century window is only available using LE/VSE and COBOL/VSE. It is not possible to use LE/VSE callable services from DOS/VS COBOL or VS COBOL II.

Use of the century window feature provided with LE/VSE allows your existing load modules to be run under LE/VSE. Any date routines should be upgraded, using COBOL/VSE, to use the century window of LE/370 to interpret 2-digit years:

LE/VSE uses Lilian dates for date representation. The base used by LE/VSE for Lilian dates is 14 October 1582.

Then, converting a string representing a date into a Lilian format,

- CALL ″CEEDAYS″ USING input-date PICSTR
  LILIAN FC.

returns an integer representation of the number of days since 14 October 1582.

Converting a Lilian date back to a character string,

- CALL ″CEEDATE″ USING input-date PICSTR
  output-char-date FC.

results in an character representation of the date with a **4-digit** value for the year.

The following program example shows code that accomplishes the aims set out above. The example uses only LE/VSE callable services for date conversions and, as a result, the sample is easily adaptable to PLI/VSE from the presented COBOL.

COBOL example MAIN

```
ID              DIVISION.
PROGRAM-ID. MAIN.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT RUNDATA ASSIGN TO INFILE
           FILE STATUS IS RUNDATA†FS.
DATA DIVISION.
FILE SECTION.
FD RUNDATA.
01 WORK-RECORD.
   03 CUST-NAME.
      05 FIRST-NAME PIC X(10).
      05 LAST-NAME  PIC  X(15).
   03 BIRTH-DATE    PIC  X(8).
   03 ACCOUNT-NUM   PIC  9(8).
   03 CURRENT-AGE   PIC  999.
   03 DUE-DATE      PIC  X(8).
   03 BENNY-FACTOR  PIC  9(3)   COMP-3.
   03 ADJUSTMENT    PIC  9(3)   COMP-3.
   03 PAYOUT        PIC  S9(7) COMP-3.
   03               PIC  X(20).

WORKING-STORAGE SECTION.
01 RUNDATA-FS      PIC 99.
77 EOF-IND         PIC X.
  88 EOF              VALUE †Y†.
  88 NOT-EOF          VALUE †N†.
```

The presented sample code shows that the work records contain both BIRTH-DATE field of the customer and a DUE-DATE field for some provided service.

COBOL example MAIN

```
PROCEDURE DIVISION.

OPENIT. OPEN INPUT RUNDATA.
          IF RUNDATA-FS NOT EQUAL TO 0
            DISPLAY †** ERROR ** †
                    †CAN¢T OPEN RUNDATA FILE **†
          ELSE
            SET NOT-EOF TO TRUE
            PERFORM LOOP
            CLOSE RUNDATA
          END-IF

          STOP RUN.

LOOP. PERFORM UNTIL EOF
          READ RUNDATA
                AT END SET EOF TO TRUE
                NOT AT END
                    CALL †SUBRTN† USING WORK-RECORD
          END-READ

        END-PERFORM.

END PROGRAM MAIN.
```

The main routine of the program simply opens the input file and then reads each record in turn, calling SUBRTN to process the records.

COBOL example SUBRTN

```
ID DIVISION.
 PROGRAM-ID. SUBRTN.
DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 CURR-LILIAN PIC S9(9) COMP.
 01 CURR-SECONDS COMP-2.
 01 CURR-DATE.                                          1
   05 CURR-YEAR      PIC 9(4).
   05 CURR-MONTH     PIC 99.
   05 CURR-DAY       PIC 99.
   05 CURR-TIME      PIC X(9).
 77 YEARS-LEFT       PIC 9(4).

**************************************************
* Working Storage & parameters for CEEDATE       *
**************************************************
 01 4-DIGIT-DATE     PIC X(80).                         2
 01 4-DIGIT-REDEFINED REDEFINES 4-DIGIT-DATE.
   05 YYYY PIC 9(4).
   05 MM   PIC 9(2).
   05 DD   PIC 9(2).
   05 FILLER PIC X(72).

 01 PIC4STR.
   05 PIC4STR-LENGTH PIC 9(2) COMP VALUE 8.
   05 PIC4STR-STRING PIC X(8) VALUE †YYYYMMDD†.
```

In working storage at 1 is the area to hold the returned values from the call to obtain current local date and time, CEELOCT.

CEEDATE returns the date, including a 4-digit value for the year to the storage defined at 2

COBOL example SUBRTN

```
**************************************************
*        Working storage & Parameters for        *
*                CEEDAYS and CEESCEN             *
**************************************************

   01 WORK-DATE.                                      3
      05 WORK-DATE-LEN PIC 9(2) VALUE 8.
      05 WORK-DATE-MM   PIC  X(2) .
      05 WORK-DATE-F1   PIC X VALUE †/†.
      05 WORK-DATE-DD   PIC  X(2).
      05 WORK-DATE-F2   PIC X VALUE †/†.
      05 WORK-DATE-YY   PIC  X(2).

   01 PICSTR.
      05 PICSTR-LENGTH PIC 9(2) COMP VALUE 8.
      05 PICSTR-STRING PIC X(50) VALUE †MM/DD/YY†.
   77 LILIAN            PIC 9(9) COMP.
   77 START-CW          PIC 9(9) COMP.
   77 FC                PIC  X(12).
```

The intermediate date field at 3 is required for the call to CEEDAYS to provide a length prefix field to the date character string.

COBOL example SUBRTN

```
    LINKAGE SECTION.
*************************************************
* Linkage Section description of the record     *
* that is read by MAIN, and shared by SUB.      *
* Note that BIRTH-DATE and DUE-DATE             *
* are expanded to show formats.                 *
*************************************************
    01 WORK-RECORD.
       03 CUST-NAME.
          05 FIRST-NAME    PIC X(10).
          05 LAST-NAME     PIC X(15).
       03 BIRTH-DATE.
          05 BMONTH        PIC 99.
          05               PIC X.
          05 BDAY          PIC 99.
          05               PIC X.
          05 BYEAR         PIC 99.
       03 ACCOUNT-NUM      PIC 9(8).
       03 CURRENT-AGE      PIC 999.
       03 DUE-DATE.
          05 DMONTH        PIC 99.
          05               PIC X.
          05 DDAY          PIC 99.
          05               PIC X.
          05 DYEAR         PIC 99.
       03 BENNY-FACTOR     PIC 9(3)   COMP-3.
       03 ADJUSTMENT       PIC 9(3)   COMP-3.
       03 PAYOUT           PIC S9(7)  COMP-3.
       03                  PIC X(20).
```

COBOL example SUBRTN

```
PROCEDURE DIVISION USING WORK-RECORD.

***************************************************
* get todays date with 4-digit year.              *
***************************************************
     CALL †CEELOCT† USING CURR-LILIAN , CURR-SECONDS ,   4
          CURR-DATE , FC .

***************************************************
* set century window to start with the current    *
* year (current system date).                     *
***************************************************
     MOVE 0 TO START-CW.
     CALL †CEESCEN† USING START-CW FC.                   5
```

The first call, to CEELOCT, 4 , obtains the current local date. The values returned are

CURR-LILIAN
A 32-bit integer representing the current local date in Lilian format

CURR-SECONDS
A 64-bit double-floating point number representing the current local date and time as the number of seconds since 00:00:00 on 14 October 1582.

CURR-DATE
A 17-byte fixed-length character string in the form YYYYMMDDHHMISS999 representing local year, month, day, hour, minute, second, and millisecond.

The call to CEESCEN, 5 , then sets the century window to start with the current year.

COBOL example SUBRTN

```
*************************************************
* convert 2-digit due date year from input file  *
* into integer value using LE/370 service to get *
* COBOL Lilian date                              *
*************************************************
      MOVE DMONTH TO WORK-DATE-MM.
      MOVE DDAY TO WORK-DATE-DD.
      MOVE DYEAR TO WORK-DATE-YY.
      CALL †CEEDAYS† USING WORK-DATE                    6
                          PICSTR LILIAN FC.
*************************************************
* convert COBOL Lilian date into YYYYMMDD format *
*************************************************
      CALL †CEEDATE† USING LILIAN , PIC4STR ,           7
             4-DIGIT-DATE , FC.

*************************************************
* find out how many years until due date         *
*************************************************
      COMPUTE YEARS-LEFT = YYYY - CURR-YEAR.            8
```

The call to CEEDAYS, 6 , then converts the DUE-DATE value from the input record to an integer value Lilian date. The DUE-DATE values are moved to the WORK-DATE field to provide the required parameter format. The input date field to CEEDAYS requires a length prefixed character string.

CEEDATE is then called, 7 , to convert the DUE-DATE value into a character string representation including a 4-digit year value.

As an example, a record containing the DUE-DATE value of -

- DMONTH = 01
- DDAY = 30
- DYEAR = 03

produced a date value of **20030130**.

The YEARS-LEFT field, after the statement at 8 , contained the value **8**.

COBOL example SUBRTN

```
**************************************************
* set century window to start 100 years ago      *
**************************************************
     MOVE 100 TO START-CW.
     CALL †CEESCEN† USING START-CW FC.                  9

**************************************************
* convert 2-digit birthdate year from input file *
* into integer value to get COBOL Lilian date    *
**************************************************
     MOVE BMONTH TO WORK-DATE-MM.
     MOVE BDAY TO WORK-DATE-DD.
     MOVE BYEAR TO WORK-DATE-YY.
     CALL †CEEDAYS† USING WORK-DATE                     10
                         PICSTR LILIAN FC.

**************************************************
* convert COBOL Lilian date into YYYYMMDD format *
**************************************************
     CALL †CEEDATE† USING LILIAN , PIC4STR ,            11
            4-DIGIT-DATE , FC.

**************************************************
* find the age † even after 1999!                *
**************************************************
     COMPUTE CURRENT-AGE = CURR-YEAR - YYYY.            12

     GOBACK.

 END PROGRAM SUBRTN.
```

The call to CEESCEN, 9 , here sets the century window to start 100 years before the current system date. The BIRTH-DATE values are then moved to the WORK-DATE field and a call is made to CEEDAYS, 10 to get the Lilian date integer value.

CEEDATE then converts the date to a 4-digit year date, 11 .

Continuing the example of the previous foil, a record containing the BIRTH-DATE value of -

- DMONTH = 04
- DDAY = 25
- DYEAR = 09

produced a date value of **19090425**.

The CURRENT-AGE field, after the statement at 12 , contained the value **86**.

The example set out above is a simple, but short-term approach towards a solution to the Year 2000 problem using the century window feature of LE/VSE.