Solving the Year 2000 problem with HOST COBOL

SHARE session 6005
Author: Tom Ross
Presenter: Laymond Pon

Language Products
IBM Santa Teresa Laboratory

July, 1996

This presentation is meant to explain what the ″YEAR 2000″ problem is and how you can solve it for COBOL applications. There are 3 ways to solve the problem using COBOL/370 and/or LE/370.

There is no way to avoid the pain of changing databases and applications to use 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/370 that allows 2-digit years to be interpreted in a 100-year window since each 2-digit 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.

Note: With APAR PN71337, DFSORT R13 provides century window support.

IBM provides 2 long-term solutions. One is to use COBOL/370 Intrinsic Functions and the other is to use LE/370 callable services. Both of these methods return 4-digit year dates.

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**
- **Depending on the coding technique, you might not be able to tell that a customer had turned 18, or whether a balloon payment is due!**
- **Many businesses are experiencing problems already, as the turn of the century approaches**
- **There is no COBOL language available with OS/VS COBOL or VS COBOL II that returns a 4 digit year**

Long-term solution

- **You must eventually:**

  1. **Re-write applications**

     - **Use COBOL Intrinsic Functions to get 4-digit year dates OR**

     - **Use LE/370 callable services to get 4-digit year dates**

  2. **AND Rebuild databases and files with 4-digit years**

- **The above are the long-term solutions that will take you and your company through to the year 9999**

- **However, there is a short-term solution that is easier**

Short-term solution

Q: What if I can′t change all my applications and data before 2000?

A: Use the LE/370 century window in old applications to interpret 2-digit year dates as 4-digit year dates

- **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

Q: Why can′t I user the century window forever?

A: 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 a 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/370 Intrinsic Functions or LE/370 date services.

Q: Can I call LE/370 services from OS/VS COBOL?

A: No.

Q: Can I call LE/370 services from VS COBOL II programs?

A: Yes, LE/370 date/time services can be called if called dynamically from VS COBOL II programs. Dynamic calls to other LE/370 services are not supported. Static calls from VS COBOL II programs to any LE/370 services are not supported

Q: Can I call LE/370 services from COBOL/370 programs?

A: Yes!!! Either statically or dynamically.

LE/370 century window and old COBOL

Q: How do I solve the YEAR 2000 problem in old applications?

A: Use the century window feature of LE/370:

1. **Run your existing load modules under LE/370**

2. **Upgrade date routines to COBOL/370**

3. **Use century window of LE/370 to interpret 2-digit years:**
   - **CALL ″CEECBLDY″ if compiler option is INTDATE(ANSI)**
   - **CALL ″CEEDAYS″ if compiler option is INTDATE(LILIAN)**
   - **Results in an integer representation**

4. **Use Intrinsic Functions to obtain 4-digit year**
   - **DATE-OF-INTEGER gives YYYYMMDD**
   - **DAY-OF-INTEGER gives YYYYDDD**

5. **Century window can be adjusted by calling CEESCEN. By default, 2 digit years lie within the 100-year range starting 80 years prior to the system date.**

See example MAIN that calls SUBRTN

COBOL example MAIN

COBOL example MAIN

```
  .......... .........

............. ..... ........
           ............................
             ......................
                     ............................
           ....
             ...................
             ............
             .............
           ......

           ........

  .....    .................
             ............
                  ......................
             ........

             ...............................

           ............

  .................
```

COBOL example SUBRTN

COBOL example SUBRTN

COBOL example SUBRTN

COBOL example SUBRTN

COBOL example SUBRTN

Integer Dates

- Problem: LE/370 Lilian and ANSI COBOL Integer dates are different
- Solution: Use LE/370 Lilian dates exclusively
- Problem: What about Intrinsic Functions?
- Solution: Use COBOL for MVS & VM Install option INTDATE
  - INTDATE(LILIAN) or INTDATE(ANSI)
  - Use INTDATE(LILIAN) to get LE/370 Lilian dates from Intrinsic Functions
- Question: What about ANSI Standards?
- Answer: There were too many Integer date formats even before LE!
  - PL/I - no integer date values
  - C - has integer dates that start with Jan 1, 1970
  - COBOL - has integer dates that start with Jan 1, 1601
  - LE/370 - has integer dates that start with Oct 15, 1582
  - CICS - has integer date that starts with Jan 1, 1900
  - MVS - has integer date that starts with Jan 1, 1900
  - DB2 - has integer date that starts with Jan 1, 1900

Why 2 different start dates for integer formats?

LE/370 and the 1989 ANSI Standard Amendment were developed at around the same time. The Language Environment group used a study published by IBM in 1984 that determined that the first reliable date we can count on is Oct 14, 1582. Every day after that date can be accounted for, while prior to that date there are "missing" days and months.

The ANSI COBOL committee X3J4 also decided to use an integer-style date, but they used the starting date of Dec 31, 1600. IBM gave them the results of our published study, but they chose to use their own starting date.

Upon further investigation, IBM found many different integer starting dates in use, and determined that LE should keep using the "best" starting date of Oct 14, 1582. Our default options for the COBOL for MVS & VM compiler must conform to the ANSI Standard, and thus the difference!

Integer Dates

- Recommendations:
  - Only use integer-form dates for performing date calculations in the same program. Convert to integer and back from integer in the same program.
  - Avoid storing integer dates in databases and files
  - Avoid passing integer dates from one program to another
- If you have a requirement to put integer dates in databases and files, then standardize on Lilian dates.