

MCR10185

Interpret Two-Digit Years in a Sliding 100-Year Window

Revision 1.0

Eric Swenson

October 10, 2026

1 Problem Description

`convert_date_to_binary_` accepts years of one or two digits (12/15/79, 79-12-15, and the year in a request id such as 841013124523.982222). The 1999 Y2K change converts such a year to four digits with a fixed rule in the internal procedure `CONVERT_TO_4_DIGIT_YEAR`:

```
/* Implement the cutoff for 2 digit year strings.
```

```
    00..29 = 2000..2029
```

```
    30..99 = 1930..1999
```

```
*/
```

```
    if p_yy < 30 then
        yyyy = 2000;
    else yyyy = 1900;
    yyyy = yyyy + p_yy;
```

[Ticket #243](#) points out that this is already close to failing (the title says 2129; the break is at 2029):

```
clock iso_date [clock ^yc-^my-^dm 30-10-12]
1930-10-12
```

From 2030 on, a date written with a two-digit year for the current year is taken to be a century earlier. Every caller is affected: commands and active functions that take a time string (`date`, `clock`, `list -from`, `memo`, `enter_abs_request -time`, and others), `exec.coms` that build dates with the `year` active function (which returns two digits; `time_strings.gi.info` even shows `date [month]/1/[year]`), and anything parsing request ids.

The ticket and its comments discuss two remedies: moving the fixed pivot to 1960 (00-59 = 2000-2059), which only postpones the problem to 2059; and a window defined relative to a “context year”, ideally moving each year and configurable per process, project, user, and through a new active function.

2 Proposed Fix

Interpret a one- or two-digit year as the year in a 100-year window that moves with the reference time: it begins 80 years before the reference year and ends 19 years after it.

Reference year	yy meaning 19yy	yy meaning 20yy
2026	46-99	00-45
2030	50-99	00-49
2060	80-99	00-79

The window always contains the current year and the 80 years before it, so dates in existing records and in recent request ids keep their meaning, and the rule never needs to be changed again. 80 years before and 19 after is the window used by several other systems (for example Java's `SimpleDateFormat`); favoring the past is appropriate because two-digit years are mostly found in existing data and in dates typed for the present or near future.

The reference time is `clock_now`, the time `convert_date_to_binary_` already uses for defaults: the current time, or the clock value given to the `$relative` and `$analyze` entries. This is consistent with how the rest of a time string is interpreted relative to that time, and it means `convert_date_to_binary_$set` (the test entry that fixes "now") can be used to test the window in any year. The reference year is the year of `clock_now` in the process's default time zone, computed once per call when a two-digit year is first seen. (The zone given in the time string itself is not known yet when the year is parsed; using the default zone can only matter for a year exactly at the window edge within hours of New Year.)

2.1 Code

The automatic variable `reference_year` is set to 0 at the start of every call (label `COMMON`), and `CONVERT_TO_4_DIGIT_YEAR` becomes:

```
CONVERT_TO_4_DIGIT_YEAR:
    procedure (p_yy)
    returns (fixed binary);

dcl p_yy fixed binary parameter;

dcl YEARS_BEFORE fixed binary int static options (constant) init (80);

dcl first_year fixed binary;          /* first year of the window */
dcl rcode fixed binary (35);
dcl 1 rtime_value aligned like time_value;
dcl yyyy fixed binary;

    if reference_year = 0 then do;
        call date_time_$from_clock (clock_now, "", addr (rtime_value), rcode);
        if rcode = 0
        then reference_year = rtime_value.yc;
        else reference_year = -1;
    end;

    if reference_year > 0
    then first_year = reference_year - YEARS_BEFORE;
```

```

else first_year = 1930;

yyyy = first_year - mod (first_year, 100) + p_yy;
if yyyy < first_year then
    yyyy = yyyy + 100;

return (yyyy);

end CONVERT_TO_4_DIGIT_YEAR;

```

If `clock_now` cannot be decoded (it always can in practice), the old 1930–2029 window is used. The procedure is called from the same two places as before: `Ayear_and_LEX` (a year token shorter than three characters) and `Arequest_id`. A block comment above the procedure explains the rule.

The actual `compare_ascii` output for the change is below:

```

cpa [lpn convert_date_to_binary_.rd] ==

Inserted in B:
B277          reference_year          fixed bin,          /* year of clock_now for
      2-digit years; 0 until needed */
Preceding:
A277          number                  fixed bin(35),        /* a temp for numeric token
      values.          */

A434          COMMON: tcode, lcode, errloc = 0;              /* no errors found during
      semantic analysis. */
Changed by B to:
B435          COMMON: reference_year = 0;                    /* see CONVERT_TO_4_DIGIT_YEAR
      */
B436          tcode, lcode, errloc = 0;                      /* no errors found during
      semantic analysis. */

A1791          /***** Implement the cutoff for 2 digit year strings.
A1792
A1793          00..29 = 2000..2029
A1794          30..99 = 1930..1999
Changed by B to:
B1793          /***** Implement the window for 2 digit year strings.
B1794
B1795          A 2 digit year is taken to be in the 100 year window that begins
B1796          YEARS_BEFORE (80) years before the year of the reference time and ends
B1797          19 years after it. The reference time is clock_now: the current time,
B1798          or the clock value given to $relative or $analyze. In 2026, say,
B1799
B1800          46..99 = 1946..1999
B1801          00..45 = 2000..2045
B1802
B1803          The window moves with the reference time, so it never needs changing.
B1804          The fixed window used before (00..29 = 2000..2029, 30..99 =

```

```

B1805      1930..1999) would have put 30 in 1930. The past is favored because 2
B1806      digit years mostly come from existing records.
B1807
B1808      The reference year is the year of clock_now in the default time zone;
B1809      it is found once per call, when first needed. If clock_now cannot be
B1810      decoded, the old window is used.

```

Inserted in B:

```

B1818      dcl YEARS_BEFORE fixed binary int static options (constant) init (80);
B1819
B1820      dcl first_year fixed binary;                      /* first year of the window */
B1821      dcl rcode fixed binary (35);
B1822      dcl 1 rtime_value aligned like time_value;
Preceding:
A1802      dcl yyyy fixed binary;

```

```

A1804      if p_yy < 30 then
A1805          yyyy = 2000;
A1806      else yyyy = 1900;
A1807          yyyy = yyyy + p_yy;
Changed by B to:
B1825      if reference_year = 0 then do;
B1826          call date_time_$from_clock (clock_now, "", addr (rtime_value),
          rcode);
B1827          if rcode = 0
B1828              then reference_year = rtime_value.yc;
B1829              else reference_year = -1;
B1830      end;
B1831
B1832      if reference_year > 0
B1833          then first_year = reference_year - YEARS_BEFORE;
B1834          else first_year = 1930;
B1835
B1836          yyyy = first_year - mod (first_year, 100) + p_yy;
B1837          if yyyy < first_year then
B1838              yyyy = yyyy + 100;

```

Comparison finished: 5 differences, 49 lines.

2.2 Not included

- **Configurable windows.** The per-process, per-project and per-user settings and the active function proposed in the ticket's comments are not implemented. A per-process setting would add a value to `time_defaults_`, entries to `date_time_`, and support in `set_time_default` and `print_time_defaults`; it can be added later on top of this change, with `YEARS_BEFORE` as the default. A moving window removes the need that motivated most of that proposal.
- **Other programs.** Some programs interpret two-digit years themselves rather than through `convert_date_to_binary_` (for example tape label dates in the `mtape_` per-format modules).

They are not changed here.

3 Documentation

`time_strings.gi.info`: a paragraph added to the notes describing how a one- or two-digit year, including the year in a request id, is interpreted, with the 2026 example, and that a year outside the window must be given with four digits. No other documentation described the old rule.

The `compare_ascii` of the old and new version is below:

```
cpa [wsp info time_strings.gi.info] ==
```

```
A1      :Info: time_strings.gi: time_strings:
```

```
A2      2025-07-27 Time Strings
```

```
Changed by B to:
```

```
B1      :Info: time_strings.gi: time_strings: 2026-10-10 Time Strings
```

```
B2
```

```
Inserted in B:
```

```
B127    A year given with one or two digits (yy), including the year in a
B128    request id, is taken to be in the 100-year window that begins 80 years
B129    before the current year and ends 19 years after it. In 2026, for
B130    example, 46 through 99 mean 1946 through 1999, and 00 through 45 mean
B131    2000 through 2045. The window moves each year. For
```

```
B132    convert_date_to_binary_$relative, the window is based on the year of
B133    the given time instead of the current year. To give a year outside
B134    the window, give all four digits.
```

```
B135
```

```
Preceding:
```

```
A127    Day, month, and zone names are shown here in English, but they may be
```

```
Comparison finished: 2 differences, 13 lines.
```

4 Test

`convert_date_to_binary_$set TIME_STRING` makes “now” a fixed time for the process, and `convert_date_to_binary_$unset` restores it, so the window can be checked for any year. Always give the `$set` argument with a four-digit year.

1. Current year (2026).

<code>clock iso_date 30-10-12</code>	2030-10-12 (was 1930-10-12)
<code>clock iso_date 45-01-01</code>	2045-01-01
<code>clock iso_date 46-01-01</code>	1946-01-01
<code>clock iso_date 99-12-31</code>	1999-12-31
<code>clock iso_date 00-01-01</code>	2000-01-01
<code>clock iso_date 12/15/79</code>	1979-12-15
<code>clock iso_date 841013124523.982222</code>	1984-10-13
<code>date [month]/1/[year]</code>	first of this month, this year

2. **Other reference years.** After `convert_date_to_binary_$set 2060-06-01:` 30-10-12 is 2030, 79-01-01 is 2079, 80-01-01 is 1980. After `convert_date_to_binary_$set 2001-06-01:` 21-01-01 is 1921, 20-01-01 is 2020. Then `convert_date_to_binary_$unset`.
3. **\$relative.** A small test program calls `convert_date_to_binary_$relative ("30-10-12", c, R, code)` with R in 1990 and in 2060: the result is in 1930 and 2030 respectively.
4. **Four-digit and other forms unchanged.** 1930-10-12, 0030-10-12, December 15, 1979, today, yesterday, this_month/1/this_year and offsets give the same results as before.
5. **Request ids.** `list_requests / enter_abs_request` and `memo` work as before.

5 Files Changed

<code>bound_date_time..s.archive</code>	<code>convert_date_to_binary..rd:</code> sliding two-digit-year window
<code>bound_date_time..archive</code>	<code>convert_date_to_binary_</code> (reduction compiled and recompiled)
<code>>s11</code>	<code>bound_date_time_</code> (rebound). It is on the hard-core list, so it is installed with a new system tape.
<code>>doc>info</code>	<code>time_strings.gi.info</code>

6 Version History

2026-10-10	1.0	Initial version of the MCR.
------------	-----	-----------------------------