YEARFRAC Incompatibilities between Excel 2007 and OOXML (OXML), and the Definitions Actually Used by Excel 2007

David A. Wheeler
2008-05-16

This document identifies incompatibilities between the YEARFRAC function, as implemented by Microsoft Excel 2007, compared to how it is defined in the Office Open Extensible Mark-up Language (OOXML), final draft ISO/IEC 29500-1:2008(E) as of 2008-05-01 (aka OXML). It also identifies the apparent definitions used by Excel 2007 for YEARFRAC, which to the author's knowledge have never been fully documented anywhere. They are not defined in the OOXML specification, because OOXML's definitions are incompatible with the apparent definition used by Excel 2007.

These incompatibilities are important. The YEARFRAC function implements the "Basis" type, which in turn is the foundation for many functions used in large-value financial calculations. Errors in these functions can have large financial stakes. This incompatibility means that, given OOXML's current definition, **OOXML cannot represent any Excel spreadsheet that uses financial functions using "basis" date calculations, such as YEARFRAC, if they use common "basis" values (omitted, 0, 1, or 4)**. Excel functions that depend upon "basis" date calculations include: ACCRINT, ACCRINTM, AMORDEGRC, AMORLINC, COUPDAYBS, COUPDAYS, COUPDAYSNC, COUPNCD, COUPNUM, COUPPCD, DISC, DURATION, INTRATE, MDURATION, ODDFPRICE, ODDFYIELD, ODDLPRICE, ODDLYIELD, PRICE, PRICEDISC, PRICEMAT, RECEIVED, YEARFRAC, YIELD, YIELDDISC, and YIELDMAT (26 functions).

Members of the OpenDocument committee developed a test suite of 32.9 million test cases, including a complete cross-product of dates in a number of consecutive years (both leap and non-leap years). The year 1900 was ignored; Excel 2007 and OOXML have a known leap-year defect in 1900. The test suite enabled the derivation of definitions that are compatible with observed Excel 2007 behavior, which are different than the definitions given in OOOXML.

Other functions that use a "basis" parameter have not yet been cross-compared but they are likely to have similar problems. It is known that DAYS360 of Excel, which uses a "Method" parameter and not a "Basis" parameter, does *not* use the same definitions as below (e.g., in Excel, reversing date parameters has a different effect in DAYS360 compared to YEARFRAC).

Currently, it is unclear what will happen, i.e., if OOXML will be changed to match Excel 2007, Excel 2007 will be changed to match OOXML (which could be harmful to many users), neither will change, both will change, or something else. Addressing this problem is an interesting open problem, a chance to make the world a better place... or at least, a more precise and openly-documented place.

This document has three sections: a quotation from the OOXML specification, examples of incompatibilities between Excel 2007 and OOXML, and the actual behavior of Excel 2007 as represented by derived definitions.

OOXML specification

The draft OOXML document of 2008-05-01, section 18.17.7.352, defines the basis values this way:

- "Basis 0 or omitted: US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
  - If the date is 28 or 29 February, it is adjusted to 30 February.

- For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.

- Basis 1: Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days. (Later, in the "Return type and value" it adds): If the Actual/actual basis is used, the year length used is the average length of the years that the range crosses, regardless of where start-date and end- date fall in their respective years.
  Note: These statements appear to be fundamentally contradictory, and thus un-implementable. Does crossing February 29 matter? The first part says it does; the second part says it does not.
- Basis 2: Actual/360. Similar to Basis 1, but always has 360 days per year.
  Note: This implies that true day difference is in the numerator for YEARFRAC.
- Basis 3: Actual/365. Similar to Basis 1, but always has 365 days per year.
- Basis 4: European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
  - If the date is 28 or 29 February, it is adjusted to 30 February.
  - For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February."

It does clarify that " All arguments are truncated to integers".

The OOXML text clarifies that start-date may be before, the same, or after end-date, but does not say what happens if start-date is after end-date (are their positions reversed? Does it return a negative number?).

Note that there actually many "U.S." and "European" systems, but discussion of all of them is beyond the scope of this paper.

Incompatibilities between Excel 2007 and OOXML

Excel 2007 and OOXML are incompatible in basis 0 (the default), basis 1, and basis 4; this could have grave consequences for some users. There are ***many*** incompatibilities, but the existence of ***any*** incompatibilities is a serious concern. Here are just a few examples of these incompatibilities between Excel 2007 and OOXML, which are sufficient to prove that there *are* incompatibilities::

- Basis 0. The actual rule followed by Excel 2007 is different than the OOXML definition. The OOXML rules for changing a date from 31 to 30, for example, are quite different from Excel:
  - Given 2000-01-01 and 2000-01-31, Excel 2007 returns 0.083333333, which is 30/360. In contrast, the OOXML rules produce 0.080555555555555561 (29/360). The OOXML rules claim that if the second day has a day value of 31 and the first date is not 28 or 29 February, it is changed to a day value of 30, and 30 minus 1 is 29.
  - Given 2000-01-01 and 2000-02-28, Excel 2007 returns 0.158333333, which is 57/360. In contrast, the OOXML rules produce 0.16388888888888889, which is 59/360. That is because by the OOXML rules, 2000-02-28 is converted to the notional 2000-02-30, producing a 59-day difference if every month is 30 days long.
- Basis 1. OOXML's definition of basis 1 is self-contradictory; it cannot be correct nor

implemented. But some "likely" interpretations clearly contradict what Excel 2007 does. As discussed below, Excel 2007's basis=1 actual algorithm first determines if the two dates "appear to be less than or equal to a year"; if they are, it uses one algorithm, else it uses a different one. The two different sub-algorithms are somewhat similar to the two algorithms that Microsoft described in OOXML. But the OOXML spec does not document that there are two different algorithms in use, it does not document the algorithm used to select between the them, and one of the algorithm's description is clearly different (at least, if two dates are in the same year, then you always use that year's leap-year-length even if the dates don't cross Feb. 29; that contradicts their description):

- Given 2000-01-01 and 2001-01-01, Excel 2007 returns 1.0000000000, which is 366/366. In contrast, OOXML's later statement suggests that the OOXML spec produce 1.0013679890560876. The dates clearly cross Feb. 29 (2000-02-29), and the end-date is in a different year, so it would appear that OOXML would demand the computation 366/((366+365)/2) . In other words, in this case only the first year's length (a leap-year) is used; here Excel 2007 clearly does *not* follow the OOXML statement "the year length used is the average length of the years that the range crosses, regardless of where start-date and end- date fall in their respective years."

  Excel actually applies this rule in some cases and not in others, in a way not at all defined in OOXML. E.G., given 2000-01-01 and 2002-01-01, Excel 2007 returns 2.0009124090, which is =(731)/((366+365+365)/3). If Jan 1. were ignored as an end-date, it would be (731)/((366+365)/2) = 2 exactly. In this case Excel *does* follow the rule "the year length used is the average length of the years that the range crosses, regardless of where start-date and end- date fall in their respective years", even though it does *not* follow the rule in others. Perhaps OOXML intended to define two algorithms, but if so, it does not do so adequately, and it fails to define the rule for selecting between the algorithms.
- Given 2000-01-01 and 2000-01-02, Excel returns 0.0027322400, which is 1/366, and NOT 1/365. Note that Feb. 29 is *not* crossed in this year, so by the first OOXML rule the value should be 1/365 = 0.0027397260.
- Given dates 2000-01-01 and 2004-01-31, Excel returns 4.080459770, which is ((366+365+365+365+366)/5)... Note that there are two, not one, 366's in there. The OOXML rule replaces a 366 with 365, because it does not cross Feb. 29, again an inconsistent answer.

- Basis 2 and 3. Excel 2007 matches the published specification (these are trivial algorithms and hard to get wrong)
- Basis 4: Again, Excel and OOXML disagree. In particular, Excel *never* adjusts a date to "Feb 30" if it started at Feb 28 or Feb 29:
  - Given 1999-01-05 and 2000-02-28, Excel returns 1.147222222. The OOXML spec produces ((2000*360 + 2*30 + 30) - (1999*360 + 1*30 + 5))/360. = 1.1527777777777777.
  - Given 1999-01-05 and 2000-02-29, Excel returns 1.15. The OOXML spec produces ((2000*360 + 2*30 + 30) - (1999*360 + 1*30 + 5))/360. = 1.1527777777777777.

Actual Excel 2007 Algorithms

Using the test suite described above, the **definitions in use** by Excel 2007 were determined to be as follows:

YEARFRAC takes two dates, determines the "number of days different", and divides that by the "number of days in the year". Its interpretation of these two terms depends on the "basis" value, which is defined as follows:

- Basis 0 or omitted: 30/360. Truncates date values and swaps them if date1 is after date2. If the dates are equal, the difference of days is 0. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
  - If both day-of-months are 31, they are changed to 30
  - Otherwise, if date1's day-of-month is 31, it is changed to 30
  - Otherwise, if date1's day-of-month is 30 and date2's day-of-month is 31, date2's day-of-month is changed to 30 (note that date2's day-of-month will stay 31 if date1's day < 30)
  - Otherwise, if both dates are the last day of February in their respective years, both day-of-month is changed to 30
  - Otherwise, if date1 is the last day of February, its day-of-month is changed to 30
- Basis 1: Actual/actual. Truncates date values and swaps them if date1 is after date2. If the dates are equal, the difference in days is 0. If date1 and date2 not "less than or equal to a year apart" (as defined below), then the days in the years between the dates is the average number of days in the years between date1 and date2, inclusive. Otherwise, the days in the years between the dates is 365, except for these cases (where it is 366): the dates are in the same year and it is a leap-year, a February 29 occurs between the two dates, or date2 is February 29.

  To determine if date1 and date2 are "less than or equal to a year apart" for purposes of this algorithm, one of these conditions much be true:
  - The two dates have the same year
  - Date2's year is exactly one more than date1's year, and ((date1.month > date2.month) or ((date1.month == date2.month) and (date1.day >= date2.day)))
- Basis 2: Actual/360. Computes the actual difference in days, and presumes there are always 360 days per year.
- Basis 3: Actual/365. Computes the actual difference in days, and presumes there are always 365 days per year.
- Basis 4: 30/360. Truncates date values and swaps them if date1 is after date2. If the dates are equal, the difference of days is 0. Assumes that each month has 30 days and the total number of days in the year is 360; any day-of-month (in date1, date2, or both) with a value of 31 is changed to 30. Note that February dates are *never* changed, because there is no February 31.

To compute a date difference between two dates that use an "actual date" system (basis 1, 2, and 3), perform the swapping as discussed above (so date2 is always after date1), consider both dates as the number of days after the same fixed date (the "epoch"), and compute date2-date1.

To compute a date difference between two dates that use an "30 month day" system (basis 0 and 4), first perform modifications described above (so date2 is always after date1), and then compute the difference as (date2.year*360+date2.month*30+date2.day) - (date1.year*360 + date1.month*30 + date1.day).

Here is pseudocode that implements the derived definitions for YEARFRAC (if these differ from the text above, the pseudocode is correct; please contact the author of any differences):

```
def appears_le_year(date1, date2):
```

```
  # Returns True if date1 and date2 "appear" to be 1 year or less apart.
  # This compares the values of year, month, and day directly to each other.
  # Requires date1 <= date2; returns boolean.  Used by basis 1.
  if date1.year == date2.year:
    return True
  if (((date1.year + 1) == date2.year) and
        ((date1.month > date2.month) or
        ((date1.month == date2.month) and (date1.day >= date2.day)))):
    return True
  return False

def basis0(date1,date2):
  # Swap so date1 <= date2 in all cases:
  if date1 > date2:
    date1, date2 = date2, date1
  if date1 == date2:
    return 0.0
  # Change day-of-month for purposes of calculation.
  date1day, date1month, date1year = date1.day, date1.month, date1.year
  date2day, date2month, date2year = date2.day, date2.month, date2.year
  if (date1day == 31 and date2day == 31):
    date1day = 30
    date2day = 30
  elif date1day == 31: date1day = 30
  elif (date1day == 30 and date2day == 31): date2day = 30
  # Note: If date2day==31, it STAYS 31 if date1day < 30.
  # Special fixes for February:
  elif (date1month == 2 and date2month == 2 and last_day_of_month(date1)
      and last_day_of_month(date2)):
     date1day = 30  # Set the day values to be equal
     date2day = 30
  elif date1month == 2 and last_day_of_month(date1):
     date1day = 30 # "Illegal" Feb 30 date.
  daydiff360 = ( (date2day + date2month * 30 + date2year * 360) -
                 (date1day + date1month * 30 + date1year * 360))
  return daydiff360 / 360.

def basis1(date1,date2):
  # Swap so date1 <= date2 in all cases:
```

```
  if date1 > date2:
      date1, date2 = date2, date1
  if date1 == date2:
      return 0.0
  if appears_le_year(date1, date2):
    if (date1.year == date2.year and is_leap_year(date1.year)):
      year_length = 366.
    elif (feb29_between(date1, date2) or
          (date2.month == 2 and date2.day == 29)): # fixed, 2008-04-18
      year_length = 366.
    else:
      year_length = 365.
    return diffdays(date1, date2) / year_length
  else:
    num_years = (date2.year - date1.year) + 1
    days_in_years = diffdays(date(date1.year, 1, 1), date(date2.year+1, 1, 1))
    average_year_length = days_in_years / num_years
    return diffdays(date1, date2) / average_year_length

def basis2(date1,date2):
  # Swap so date1 <= date2 in all cases:
  if date1 > date2:
      date1, date2 = date2, date1
  return diffdays(date1, date2) / 360.

def basis3(date1,date2):
  # Swap so date1 <= date2 in all cases:
  if date1 > date2:
      date1, date2 = date2, date1
  return diffdays(date1, date2) / 365.

def basis4(date1,date2):
  # Swap so date1 <= date2 in all cases:
  if date1 > date2:
    date1, date2 = date2, date1
  if date1 == date2:
    return 0.0
  # Change day-of-month for purposes of calculation.
  date1day, date1month, date1year = date1.day, date1.month, date1.year
```

```
  date2day, date2month, date2year = date2.day, date2.month, date2.year
  if date1day == 31: date1day = 30
  if date2day == 31: date2day = 30
  # Remarkably, do NOT change Feb. 28 or 29 at ALL.
  daydiff360 = ( (date2day + date2month * 30 + date2year * 360) -
                 (date1day + date1month * 30 + date1year * 360))
  return daydiff360 / 360.
```

More information, including the test suite with over 30 million test case results, is at:

http://www.dwheeler.com/yearfrac

Comments and corrections on this technical material would be greatly appreciated!