Proposed disposition for date problems – DRAFT 2

Antonis Christofides, National Technical University of Athens

4 February 2008

This is work in progress. We are aware that the specification for several spreadsheet functions still needs modification, and that some of the modifications already made need some more work, particularly in error conditions. While we are going to finish this work, its spirit is already clear and we present the part done so far.

1. Introduction

This proposal deals with comments AU-0016, BR-0046, CA-0044, CA-0071, CH-0006, CH-0007, CH-0017, CL-0013, CL-0015, CL-0147, CL-00172, CO-0035, CO-0154, CO-0155, CO-0156, CZ-0009, DE-0030, DE-0031, DE-0032, DE-0072, DE-0073, DK-0033, DK-0136, DK-0137, DK-0153, FI-0013, FR-0182, FR-0183, FR-0351, FR-0352, GB-0300, GB-0301, GB-0304, GB-0305, GB-0363, GB-0364, GH-0002, GR-0003, GR-0004, GR-0005, GR-0006, GR-0007, GR-0008, IE-0002, IN-0007, IN-0057, IN-0058, IN-0061, IN-0062, IN-0080, IR-0001, IR-0002, KE-0054, KE-0055, MX-0005, PE-0002, PE-0003, PH-0005, PT-0085, SG-0002, US-0130, US-0131, US-0134, UY-0003, VE-0011, VE-0060, ZA-0014.

Regarding its treatment of dates, the Ecma 376 specification is inconsistent with ISO 8601, specifically with respect to dates before 1900, and with its treatment of 1900 as a leap year. Ecma's proposed disposition does address these problems, but it does not address comment GR-0008 and similar comments submitted by other countries, which mentions that “Having two different date systems with different base dates side-by-side in the same standard document format makes no sense. Rather, it is appropriate to fix a single base date. Applications which use a different base date can convert from the date representation used in the standard document format to the application's preferred date representation, and vice versa”. On the contrary, Ecma's proposed disposition has complicated the problem, because it has been proposed that dates be stored in one of four, rather than two, distinct formats, keeping the old formats for the purpose of compatibility with previous functionality.

We propose an alternative disposition which is much simpler, adequately resolves all related comments including GR-0008, and is also able to reproduce the 1900 bug whenever required, thus also preserving compatibility with previous functionality.

2. Problem description and overview of the solution

The entire problem stems from the fact that legacy applications, namely Microsoft Excel, which in turn reproduces behaviour from Lotus 1-2-3, have no notion of timestamps and intervals as separate data types, and merely store them as real numbers. Excel has no underlying understanding of a date; to Excel, a date is merely a way of displaying a real number.

Despite the fact that, after several decades of usage, users now find it natural, it is important to understand that a timestamp is not a real number, and attempting to make operations such as adding 3.14 to the timestamp of 2008-02-03T12:31 makes no sense. You can add 3.14 days, or 3.14 minutes, or 3.14 years, or any kind of duration, but you can't add a pure real number to a timestamp any more than you can add an apple to an orange.

Therefore, the correct way to handle timestamps and intervals is for the spreadsheet to inherently support them as distinct data types. However, converting legacy spreadsheets

poses a problem: whenever the converter encounters a real number, it does not always know whether a real number per se is intended, or a date.

Ecma has proposed that new spreadsheets also store timestamps as real numbers. Although this achieves compatible conversion from legacy spreadsheets, it perpetuates the problem. Rather than condemn us to carry the errors of the past for decades into the future, it is better to do a little more effort to correct them now. We proceed to propose a way, which actually greatly simplifies Ecma-376.

The way to address the problem is similar to what has been done in OpenOffice and Open Document Format (ODF). ODF dictates that timestamps are stored as timestamps, leaving it to the application to handle legacy conversions. While OpenOffice Calc apparently treats timestamps in the same way as Microsoft Excel, in fact it includes underlying conversions so that it properly stores timestamps as required by ODF.

Similarly, our proposal is that Ecma-376 store timestamps and intervals in ISO 8601 format, including recommendations that applications automatically treat them as numbers whenever required for compatibility reasons. Applications may work in compatibility mode, where such automatic treating of a date as a real number is possible, but should warn users against compatibility mode and prefer strict mode, where an error will occur in such cases.

3. Specific changes to Ecma-376

Part 4, §3.17.4, page 2,522, line 5:

~~3.17.4 Dates and Times~~

~~Each unique instant in SpreadsheetML time is represented as a distinct non-negative numeric serial value, which is made up of an integer date component and a fractional time component. As dates and times are numeric values, they can take part in arithmetic operations.~~

~~Numerous functions take as arguments one or more serial values or strings representing dates and/or times. Functions that care only about the date shall ignore any time information that is provided. Functions that care only about the time shall ignore any date information that is provided.~~

~~3.17.4.1 Date Representation~~

~~Going forward in time, the date component of a serial value increases by 1 each day.~~

~~There are two different bases for serial values:~~

- ~~In the 1900 date base system, the lower limit is January 1, 1900, which has serial value 1. The upper limit is December 31, 9999, which has serial value 2,958,465.~~
- ~~In the 1904 date base system, the lower limit is January 1, 1904, which has serial value 0. The upper limit is December 31, 9999, which has serial value 2,957,003.~~

~~A serial value outside of the range for its date base system is ill-formed.~~

~~As to which date base system an implementation uses by default or whether it allows its users to switch between date base systems, is unspecified. See §3.17.6.7 for XML-related details. [Note: As the XML allows either date base system to be used, an implementation must be able to deal with both systems. end note]~~

~~For legacy reasons, an implementation using the 1900 date base system shall treat 1900 as though it was a leap year. [Note: That is, serial value 59 corresponds to February 28,~~

and serial value 61 corresponds to March 1, the next day, allowing the (non-existent) date February 29 to have the serial value 60. end note] A consequence of this is that for dates between January 1 and February 28, WEEKDAY shall return a value for the day immediately prior to the correct day, so that the (non-existent) date February 29 has a day-of-the-week that immediately follows that of February 28, and immediately precedes that of March 1.

[Example: For the 1900 date base system:

DATEVALUE("01-Jan-1900") results in the serial value 1.0000000...
DATEVALUE("03-Feb-1910") results in the serial value 3687.0000000...
DATEVALUE("01-Feb-2006") results in the serial value 38749.0000000...
DATEVALUE("31-Dec-9999") results in the serial value 2958465.0000000...

For the 1904 date base system:

DATEVALUE("01-Jan-1904") results in the serial value 0.0000000...
DATEVALUE("03-Feb-1910") results in the serial value 2225.0000000...
DATEVALUE("01-Feb-2006") results in the serial value 37287.0000000...
DATEVALUE("31-Dec-9999") results in the serial value 2957003.0000000...

end example]

3.17.4.2 Time Representation

The time component of a serial value ranges in value from 0–0.99999999, and represents times from 0:00:00 (12:00:00 AM) to 23:59:59 (11:59:59 P.M.), respectively.

Going forward in time, the time component of a serial value increases by 1/86,400 each second. [Note: As such, the time 12:00 has a serial value time component of 0.5. end note]

[Example:

TIMEVALUE("00:00:00") results in the serial value 0.0000000...
TIMEVALUE("00:00:01") results in the serial value 0.0000115...
TIMEVALUE("10:05:54") results in the serial value 0.4207639...
TIMEVALUE("12:00:00") results in the serial value 0.5000000...
TIMEVALUE("23:59:59") results in the serial value 0.9999884...

end example]

3.17.4.3 Combined Date and Time Representation

Any date component can be added to any time component to produce a serial value for that date/time combination.

[Example: For the 1900 date base system:

DATE(1910,2,3)+TIME(10,5,54) results in the serial value 3687.4207639...
DATE(1900,1,1)+TIME(12,0,0) results in the serial value 1.5000000...
DATE(9999,12,31)+TIME(23,59,59) results in the serial value 2958465.9999884...

For the 1904 date base system:

DATE(1910,2,3)+TIME(10,5,54) results in the serial value 2225.4207639...
DATE(1904,1,1)+TIME(12,0,0) results in the serial value 0.5000000...
DATE(9999,12,31)+TIME(23,59,59) results in the serial value 2957003.9999884...

~~end example]~~

3.17.4 Dates, Times and Durations

Each unique instant in SpreadsheetML time is represented in the format required by the XSD dateTime data type. Each duration is represented in the format required by the XSD duration data type.

For legacy reasons, the nonexistent date of 29 February 1900 is allowed, but should not be used for new spreadsheets.

[*Note:*

Some legacy spreadsheet applications have been storing instants and intervals as real numbers. Microsoft Excel and Lotus 1-2-3, in particular, have been storing instants as the number of days since the beginning of 1900 or 1904, and intervals as a number of days. In order to be compatible with legacy spreadsheets or familiar to users of legacy applications, an application may work in a compatibility mode that automatically converts timestamps and intervals to numbers when asked to add them to real numbers or perform other operations that make sense with real number operands but not with timestamps and intervals. However, applications should also have a strict mode of operation that prohibits such legacy operations, and should discourage users from working in compatibility mode.

The *dateCompatibility* attribute describes how dates are converted into numbers when applications work in compatibility mode.

end note]

Part 4, §3.2.27, page 1,908, line 26:

- Properties: the workbook has several property collection that store basic workbook settings, such as the ~~date system to use,~~ file protection settings, calculation settings, and smart tag behaviors.
- Names: represent descriptive that represent cells, ranges of cells, formulas, or constant values.

[*Example*:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<workbook
xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/5/main"
mlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships
">
  <fileVersion lastEdited="4" lowestEdited="4" rupBuild="4017"/>
  <workbookPr ~~date1904="1"~~ vbName="ThisWorkbook"
defaultThemeVersion="123820"/>
```

Part 4, §3.2.28, page 1,911, line 1:

[*Example*:

```
 <workbookPr ~~date1904="1"~~ showObjects="none"
 saveExternalLinkValues="0"
   defaultThemeVersion="123820"/>
```

end example]

Part 4, §3.3, page 1,926, line 22:

Worksheet cells can contain text, numbers, dates/times/durations, and formulas.

Page 1,936, check for “date occurring”

Page 2,477 check for “textDates”

Part 4, §3.2.28, page 1,912:

Attributes	Description
~~date1904 (Date 1904)~~	~~Specifies a boolean value that indicates whether the date systems used in the workbook starts in 1904.~~ ~~A value of on, 1, or true indicates the date system starts in 1904.~~ ~~A value of off, 0, or false indicates the workbook uses the 1900 date system, where 1/1/1900 is the first day in the system.~~ ~~The default value for this attribute is false.~~ ~~The possible values for this attribute are defined by the XML Schema boolean datatype.~~
dateCompatibility (Date operations in compatibility mode)	Specifies how timestamps and intervals are converted to real numbers when working in date compatibility mode. A value of 1900 indicates that if the consumer has a date compatibility mode, it should enable it, and that when asked to perform operations with timestamps and/or intervals that do not make sense unless the timestamps and/or intervals are converted to real numbers, then the consumer should attempt to convert intervals to number of days and timestamps to number of days since 1899-12-31, making, in addition, the assumption that 1900 was a leap year. A value of 1904 indicates that if the consumer has a date compatibility mode, it should enable it, and that when asked to perform operations with timestamps and/or intervals that do not make sense unless the timestamps and/or intervals are converted to real numbers, then the consumer should attempt to convert intervals to number of days and timestamps to number of days since 1903-12-31, making, in addition, the assumption that 1900 was a leap year. If the attribute is missing or is an empty string, the consumer should disable the date compatibility mode. Consumers that do not have a date compatibility mode shall ignore the attribute, but should warn the user.

Part 4, §3.2.28, page 1,915, line 3:

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WorkbookPr">
    <attribute name="date1904" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="dateCompatibility" type="xsd:string" use="optional" default=""/>
    <attribute name="showObjects" type="ST_Objects" use="optional" default="all"/>
    <attribute name="showBorderUnselectedTables" type="xsd:boolean" use="optional"
      default="true"/>
    <attribute name="filterPrivacy" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="promptedSolutions" type="xsd:boolean" use="optional" default="false"/>
```

```
    <attribute name="showInkAnnotation" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="backupFile" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="saveExternalLinkValues" type="xsd:boolean" use="optional"
      default="true"/>
    <attribute name="updateLinks" type="ST_UpdateLinks" use="optional" default="userSet"/>
    <attribute name="codeName" type="xsd:string" use="optional"/>
    <attribute name="hidePivotFieldList" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="showPivotChartFilter" type="xsd:boolean" default="false"/>
    <attribute name="allowRefreshQuery" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="publishItems" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="checkCompatibility" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="autoCompressPictures" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="refreshAllConnections" type="xsd:boolean" use="optional"
      default="false"/>
    <attribute name="defaultThemeVersion" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

Part 4, §3.8.31, page 2,140, line 10:

To display	As	Use this code
Years	~~1900~~0000-9999	yyyy

Part 4, §3.8.31, page 2,140, line 12:

~~See §3.17.4.1 for special handling of certain days in the year 1900.~~

Part 4, §3.17.6.7, page 2,529, line 27:

A date and/or time shall be stored in XML as a string in ISO 8601 format.

~~**3.17.6.7 Dates and Times**~~

~~As a date and/or time is represented by a number, a date/time serial value shall be stored in XML as the unformatted text form of that number, as accurately as possible.~~

~~The date base system is recorded in the Workbook part's XML by the presence or absence of the date1904 attribute of the workbookPr element. A value of 1 for this attribute indicates 1904. [*Example*:~~

~~1900: <workbookPr showObjects="all"/>~~
~~1904: <workbookPr date1904="1" showObjects="all"/>~~
~~*end example*~~]

3.17.6.7 Timestamps and durations

Timestamps and durations are stored in the format required by the XSD dateTime and duration types.

Part 4, §3.17.7.2, page 2,534, line 6:

- ~~*issue*, *first interest*, or *settlement* is out of range for the current date base value, #NUM! is returned~~

Part 4, §3.17.7.3, page 2,535, line 16:

- ~~*issue* or *settlement* is out of range for the current date base value, #NUM! is returned~~

Part 4, §3.17.7.7, page 2,539, line 5:

- ~~*date_purchased* or *first_period* is out of range for the current date base value, #NUM! is returned.~~

Part 4, §3.17.7.8, page 2,541, line 1:

- ~~*date_purchased* or *first_period* is out of range for the current date base value, #NUM! is returned.~~

Part 4, §3.17.7.57, page 2,583, line 6:

- ~~*settlement* or *maturity* is out of range for the current date base value, #NUM! is returned.~~

Part 4, §3.17.7.58, page 2,584, line 5:

- ~~*settlement* or *maturity* is out of range for the current date base value, #NUM! is returned.~~

Part 4, §3.17.7.59, page 2,586, line 6:

- ~~*settlement* or *maturity* is out of range for the current date base value, #NUM! is returned.~~

Part 4, §3.17.7.60, page 2,586, line 6:

- ~~*settlement* or *maturity* is out of range for the current date base value, #NUM! is returned.~~

Part 4, §3.17.7.61, page 2,587, line 7:

- ~~*settlement* or *maturity* is out of range for the current date base value, #NUM! is returned.~~

Part 4, §3.17.7.62, page 2,588, line 6:

- ~~*settlement* or *maturity* is out of range for the current date base value, #NUM! is returned.~~

Part 4, §3.17.7.74, page 2,600, line 11:

3.17.7.74 DATE

Syntax:

DATE (year , month , day [, add1900])

Description: ~~Computes the serial value for the given date~~ Constructs a timestamp.

Arguments:

Name	Type	Description
year	number	A year, truncated to integer, that together with month number and day specifies the date value ~~whose serial value is to be computed~~

Name	Type	Description
		to be constructed. ~~For the 1900 date base system:~~ • ~~If *year* is in the range 0–1899, inclusive, the year shall be interpreted as *year* + 1900.~~ • ~~If *year* is in the range 1900–9999, inclusive, the year shall be interpreted as *year*.~~ ~~For the 1904 date base system:~~ • ~~If *year* is in the range 4–1899, inclusive, the year shall be interpreted as *year* + 1900.~~ • ~~If *year* is in the range 1904–9999, inclusive, the year shall be interpreted as *year*.~~
month	number	A month, truncated to integer, that together with year number and day specifies the date ~~whose serial value is to be computed~~ to be constructed. *month* shall be interpreted as the number of months relative to the final month of the year prior to the specified year. ~~If *month* is in the range 1–12, the month shall be interpreted as *month*. If *month* is less than 1 or greater than 12, the month shall be interpreted as the normalized value (see below) of *month*, and the year shall be adjusted accordingly.~~
day	number	A day, truncated to integer, that together with month and number year specifies the date ~~whose serial value is to be computed~~ to be constructed. *day* shall be interpreted as the number of days relative to the last day of the month (and its associated year) prior to the month (and its associated year) as determined from *month* and *year* (see below). ~~If *day* is in the allowable range of days for the month, the day shall be interpreted as *day*. If *day* is less than 1 or greater than the number of days in the given month, the day shall be interpreted as the normalized value (see below) of *day*, and the year and month shall be adjusted~~
add1900	boolean	If true, it specifies that the year is to be interpreted as *year*+1900 if *year* is in less than 1900, and as *year* if it is greater than or equal to 1900. The default value for this parameter is false.

The value of *month* or *day* in a *year-month-day* argument triplet can be out of range. *month* is simply an instance of counting a given number of months, minus one, relative to January of the year specified, using the Gregorian calendar [ISO 8601]. This calendar defines that there are 12 months in a year, and that when counting forward, the month following December of one year is January of the following year, and when counting backward, the month preceding January of one year is December of the previous year. Likewise, *day* is simply an instance of counting a given number of days, minus one, relative to the first day of the adjusted month, using the Gregorian calendar. This calendar defines the number of days in each month, and that when counting forward, the day following the final day of one month is the first day of the following month, and when counting backward, the day preceding the first day of one month is the final day of

the previous month. [*Example*: The *year-month-day* argument triplets (2007,12,32), (2007,13,1), and (2008,1,1) all result in the same date. *end example*]

Return Type and Value: ~~number – The serial value for the given date.~~ dateTime – The timestamp that corresponds to the specified parameters.

If year is less than 0 or is greater than or equal to 10000, #NUM! is returned.

When the dateCompatibility attribute has a value of 1900, a consumer may consider 1900 to be a leap year. In such a case, DATE(1900, 2, 29) will return 1900-02-29, rather than the correct value of 1900-03-01.

~~However, if~~

- ~~year is less than 0 or is greater than or equal to 10000, and the 1900 date base system is being used, #NUM! is returned.~~
- ~~year is less than 4, is greater than or equal to 10000, is in the range 1900–1903, inclusive, and the 1904 date base system is being used, #NUM! is returned.~~

~~[*Example*: For the 1900 date base system:~~

~~DATE(0,1,1) results in a serial value of 1~~
~~DATE(1899,1,1) results in a serial value of 693598~~
~~DATE(1900,1,1) results in a serial value of 1~~
~~DATE(9999,12,31) results in a serial value of 2958465~~

~~For the 1904 date base system:~~

~~DATE(4,1,1) results in a serial value of 0~~
~~DATE(1899,1,1) results in a serial value of 692136~~
~~DATE(1904,1,1) results in a serial value of 0~~
~~DATE(9999,12,31) results in a serial value of 2957003~~

~~*end example*~~]

Part 4, §3.17.7.75, page 2,601, line 29:

3.17.7.75 DATEDIF

Syntax:

`DATEDIF ( start-date , end-date [ , unit ] )`

Description: Calculates the ~~number of days, months, or years~~ difference between two dates.

Arguments:

Name	Type	Description
start-date	dateTime ~~number~~	The first date in the period, truncated to integer.
end-date	dateTime ~~number~~	The last date in the period, truncated to integer.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>unit</td><td>text</td><td>The ~~count~~ type of result to be returned. If omitted, returns the difference as a XSD duration. Otherwise, returns a number that depends on unit, as follows:

<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>"Y"</td><td>The number of complete years in the period.</td></tr>
<tr><td>"M"</td><td>The number of complete months in the period.</td></tr>
<tr><td>"D"</td><td>The number of days in the period.</td></tr>
<tr><td>"MD"</td><td>The difference between the days in start-date and end-date. The months and years of the dates are ignored.</td></tr>
<tr><td>"YM"</td><td>The difference between the months in start-date and end-date. The days and years of the dates are ignored.</td></tr>
<tr><td>"YD"</td><td>The difference between the days of start-date and end-date. The years of the dates are ignored.</td></tr>
</table></td></tr>
</table>

Return Type and Value: number or duration -- The number of days, months, or years between two dates, depending on the value of *unit*, or the XSD duration between the two dates, if *unit* is omitted.

However, if

- ~~start-date or end-date is out of range for the current date base value, #NUM! is returned.~~
- *start-date* ≥ *end-date* `#NUM!` is returned.
- *unit* is any value other than those shown in the table above, `#NUM!` is returned.

[*Example*:

`DATEDIF(DATE(2001,1,1),DATE(2003,1,1),"Y")` results in `2` complete years
`DATEDIF(DATE(2001,6,1),DATE(2002,8,15),"D")` results in `440` days
`DATEDIF(DATE(2001,6,1),DATE(2002,8,15),"YD")` results in `75` days
`DATEDIF(DATE(2001,6,1),DATE(2002,8,15),"MD")` results in `14` days

end example]

Part 4, §3.17.7.75, page 2,601, line 29:

3.17.7.76 DATEVALUE

Syntax:

`DATEVALUE` (date-time-string)

Description: Determines ~~Computes the serial value of~~ the date ~~and/or time~~ represented by a string~~the string date-time-string, taking into account the current date base value~~.

Arguments:

Name	Type	Description
date-time-string	text	The date ~~and/or time~~ whose ~~serial value~~ date is to be computed. *date-time-string* can have any valid date ~~and/or time~~ format. If the year portion of *date-time-string* is omitted, the current year is used. Any time information in *date-time-string* shall be ignored.

Return Type and Value: ~~number~~ dateTime -- The ~~serial value of~~ the date ~~and/or time~~ represented by the string *date-time-string*.

However, if

- ~~date-time-string is out of range for the current date base value, #VALUE! is returned.~~
- *date-time-string* does not represent a date, `#VALUE!` is returned.

[*Example*: When the current year is 2006,

```
DATEVALUE("2/1/2006")
DATEVALUE("01-Feb-2006 10:06 AM")
DATEVALUE("2006/2/1")
DATEVALUE("2006-2-1")
DATEVALUE("1-Feb")
```

all result in 2006-02-01~~38749 for the 1900 date base system, or 37287 for the 1904 date base system~~. *end example*]

Part 4, §3.17.7.78, page 2,605, line 11:

3.17.7.78 DAY

Syntax:

`DAY` (date-value)

Description: Computes the numeric Gregorian day for the date and/or time having the given *date-value*~~, taking into account the current date base value~~.

Arguments:

Name	Type	Description
date-value	~~number~~ dateTime, text	The date and/or time whose day is to be computed. That date and/or time shall be expressed either as a ~~serial~~ dateTime value~~, in which case, its fractional part is ignored,~~ or as a *string-constant* having any valid date and/or time format.~~, in which case, a~~Any time information shall be ignored.

Return Type and Value: number -- The Gregorian day for the date and/or time having the given *date-value*. The returned value shall be in the range 1--31.

~~However, if date-value is out of range for the current date base value, #NUM! is returned.~~

[*Example*:

`DAY(DATE(2006,1,2))` results in `2`
`DAY(DATE(2006,0,2))` results in `31`
`DAY("2006/1/2 10:45 AM")` results in `2`
~~DAY(30000) results in 18 for the 1900 date base system, or 19 for the 1904 date base system~~

end example]

Part 4, §3.17.7.79, page 2,606, line 16:

3.17.7.79 DAYS360

Syntax:

`DAYS360 ( start-date , end-date [ , method-flag ] )`

Description: Computes the signed number of days between two dates based on a 360-day year (twelve 30-day months).

Arguments:

Name	Type	Description
start-date	dateTime ~~number~~	*start-date* and *end-date* are the dates for which the difference is to be computed. *start-date* can be earlier than, the same as, or later than *end-date*.
start-date	dateTime ~~number~~	
method-flag	logical	Specifies whether to use the U.S. or European method in the calculation, as follows:

Value	Meaning
`FALSE` or omitted	U.S. (NASD) method: If the *start-date* is the 31st day of a month, it is changed to the 30th day of that same month. If the *end-date* is the 31st day of a month and the *start-date* is earlier than the 30th day of a month, the *end-date* is changed to the 1st day of the following month; otherwise the *end-date* is changed to the 30th day of the same month.
`TRUE`	European method: *start-date*s and *end-date*s that occur on the 31st day of a month are changed to the 30th day of the same month.

Return Type and Value: number -- The signed number of days between two dates based on a 360-day year (12 30-day months). If *start-date* is later than *end-date*, the return value shall be negative, and the magnitude shall be the difference in days.

~~However, if start-date or end-date is out of range for the current date base value, #NUM! is returned.~~

[*Example*:

`DAYS360(DATE(2002,2,3),DATE(2005,5,31))` results in `1198`
`DAYS360(DATE(2005,5,31),DATE(2002,2,3))` results in `-1197`
`DAYS360(DATE(2002,2,3),DATE(2005,5,31),FALSE)` results in `1198`
`DAYS360(DATE(2002,2,3),DATE(2005,5,31),TRUE)` results in `1197`

| *end example*]

Part 4, §3.17.7.91, page 2,617, line 5:

- ~~*settlement* or *maturity* is out of range for the current date base value, #NUM! is returned.~~

Part 4, §3.17.7.101, page 2,624, line 5:

- ~~*settlement* or *maturity* is out of range for the current date base value, #NUM! is returned.~~

To be continued

This is work in progress. We are aware that the specification for several spreadsheet functions still needs modification, and that some of the modifications already made need some more work, particularly in error conditions. While we are going to finish this work, its spirit is already clear and we have presented the part done so far.