BR_HistoricalDate – Historical Date Services

Robin Lamacraft 2017-04-05

SCOPE

This module must be used to operate with Historical Dates. It provides services to create, edit, retrieve and delete Historical Dates. Historical Dates are dates (and times) recorded from evidence. The possible formats of Historical Dates are controlled by Historical Date Definitions. Gregorian dates are the most common calendar and Gregorian is the usual default calendar. Where possible other calendar dates are converted into Gregorian calendar values for comparison and sorting. The original transcript and other properties are recorded for later retrieval. NOTE: All actions below must be aware of the GUI-Language, the User Data Entry Language and the Report Language.

This module provides 3 types of services:

1. On Historical Date Values:
   - Operations to enter and edit Historical Date values
   - Retrieval of Historical Date values
   - Sorting of a list of Historical Date values and returning the ordered list.
2. On operations between Historical Date Values and Date/Time Interval values, to perform calculations to return a new Historical Date Value result
3. On operations between a Historical Date Value and another Historical Date Value, to return a Date/Time Interval.

HRE manages 2 **Date** and 2 **Time Interval** types:

1. **Default** – (for use in most research disciplines)
   - **Dates:**
     - Bounds: -99,998 years (pre AD) to +99,998 years (post AD)
     - Increment: 1 Minute
     - A non-numerical value (like "First Quarter 1885") if not convertible to numeric form is collated to sort to the first position e.g. as if it was at -99,999 years
     - Internal representation: years, days, hours and minutes as a pair of 64bit integers.
   - **Time Intervals:**
     - Bounds: -98 years to +99 years
     - Increment: 1 millisecond
     - A non-numerical value (like "Sunrise to Sunset") if not convertible to numeric form is collated to sort to the first position e.g. as if it was at -99 years
     - Internal representation: years, days, hours, minutes, seconds and milliseconds as pair of 64bit integers.
2. **Centuries** – (for use in astronomy, geology, etc) *– for later implementation*
   - **Dates:**
     - Bounds: -98 billion years (pre AD) to +99 billion years (post AD)
     - Increment: 100 years
     - A non-numerical value, (like "Big Bang") if not convertible to numeric form is collated to sort to the first position e.g. as if it was at -99 billion years
     - Internal representation: centuries as a pair of 64bit integers.
   - **Time Intervals**
     - Bounds: -98 billion years to +99 billion years
     - Increment: 100 years

- A non-numerical value, if not convertible to numeric form is collated to sort to the first position e.g. as if it was at -99 billion years
- Internal representation: as billion years, millennia, years and 1/1000th of a year as a pair of 64bit integers.

Qualified Normal Historical Dates

In the following, English language examples are used, but the English elements are expected to be replaced by the current GUI Display or GUI User language elements. Normal Dates may be qualified to represent bounded or unbounded date ranges:

1. Unqualified Complete Date has the form where every unit from largest to the last has a value (no missing units), e.g. "28 March 2017"
2. Between Date where the user has entered "between" or "bet." then an Unqualified Complete Date and then a second Unqualified Complete Date (a later value from the first) to define a range of dates to describe a date for an action that is expected to have happened during those dates, e.g. "between 28 March 2017 and 2 Apr 2017"
3. Within Date where the user has entered "within" then an Unqualified Complete Date then a second unqualified date (a later value from the first) to define a range of dates to describe a date for an action that is expected to have happened within those dates, e.g. "within 28 March 2017 to 2 Apr 2017"
4. Unbounded Before Date where the user has entered "before" or "b." then an Unqualified Complete Date to describe a date for an action that is expected to have happened any time before that date, e.g. "before 28 March 2017"
5. Unbounded After Date where the user has entered "after" or "a." then an Unqualified Complete Date to describe a date for an action that is expected to have happened any time after that date, e.g. "after 28 March 2017"
6. Circa Date where the user has entered "circa" or "c." then an Unqualified Complete Date to describe a date for an action that is expected to have happened within a period of plus or minus a number of units (typically years) as set in the Historical Date Settings, e.g. "circa 28 March 2017".

NOTE: The above data have TMG equivalents.

7. Bounded Before Date where the user has entered an Unqualified Complete Date then "--nnnX" to specify an interval before of "nnn" units, "X" is a single unique letter to identify the unit type, e.g. "Y" years, "M" Months, "D" Days", etc. Then the date range commences at "nnnX" units before the date and includes that date in which an action is expected to have happened. This is like a Between Date but has a weighting of the entered date as being the most likely date, e.g. "28 March 2017 --14D" (14 days before)
8. Bounded After Date where the user has entered an Unqualified Complete Date then "++nnnX" to specify an interval after of "nnn" units, "X" is a single unique letter to identify the unit type, e.g. "Y" years, "M" Months, "D" Days", etc. Then the date range commences at that date and then extends "nnnX" units after the date in which an action is expected to have happened. This is like a Between Date but has a weighting of the entered date as being the most likely date, e.g. "28 March 2017 ++3M" (3 months after)
9. Flexible Circa Date. This combines the syntax of both the Bounded Before Date and the Bounded After Date. The user has entered an Unqualified Complete Date to describe a date for an action that is expected to have happened within a period of plus or minus a number of units , but now the user entered "--nnnX" then "++nnnY" to specify the range of uncertainty on either side of the entered date, e.g. "28 March 2017 --5D ++10D" (5 days before, 10days after).

Qualified Normal Historical Time Intervals

In the following, English language examples are used, but the English elements are expected to be replaced by the current GUI Display or GUI User language elements. Normal Dates may be qualified to represent bounded or unbounded date ranges:

1. Unqualified Complete Time Interval has the form where every unit from largest to the last has a value (no missing units), e.g. ”1Y 20D” (1 Year, 20 days)
2. Between Time Interval where the user has entered “between” or “bet.” then an Unqualified Complete Time Interval and then a second Unqualified Complete Time Interval (a larger value from the first) to define bounds for an interval, e.g. “between 6M and 9M” (between 6 and 9 months)
3. Within Time Interval where the user has entered “within” then an Unqualified Complete Time Interval, then a second unqualified Time Interval (a later value from the first) to define an interval, e.g. “within 7D to 14D” (within 7 to14 Days)
4. Before Time Interval where the user has entered “before” or “b.” then an Unqualified Complete Time Interval to describe a Time Interval before the current time, e.g. “before 14D” (14 days before now)
5. After Time Interval where the user has entered “after” or “a.” then an Unqualified Complete Time Interval to describe a date after that Time Interval after the current time, e.g. “after 14 days”(14 days from now)
6. Circa Time Interval where the user has entered “circa” or “c.” then an Unqualified Complete Time Interval to describe a Time Interval for an action that is expected to have happened within a period of plus or minus a number of units as set in the Historical Interval Settings, e.g. “circa 12M” (circa 12 months).
7. Bounded Before Time Interval where the user has entered “Before” then an Unqualified Complete Time Interval then “--nnnX” to specify an interval before of “nnn” units, “X” is a single unique letter to identify the unit type, e.g. “Y” years, “M” Months, “D” Days”, etc. Then the Time Interval commences at “nnnX” units before the Time Interval and includes that Time Interval. This is like a Between Date but has a weighting of the entered Time Interval as being the most likely value, e.g. “Before 14D --2D” (16 to 14 days before now)
8. Bounded After Time Interval where the user has entered “After” then an Unqualified Complete Time Interval then “++nnnX” to specify an interval after of “nnn” units, “X” is a single unique letter to identify the unit type, e.g. “Y” years, “M” Months, “D” Days”, etc. Then the Time Interval commences at the Time Interval and extends “nnnX” units after that Time Interval. This is like a Between Date but has a weighting of the entered Time Interval as being the most likely value, e.g. “After 28D ++2D” (28 to 30 days after now)
9. Flexible Circa Time Interval. This combines the syntax of both the Bounded Before Time Interval and the Bounded After Time Interval. The user has entered an “Before” or “After”, then an Unqualified Complete Time Interval that is before or after now. The user then entered “--nnnX” then “++nnnY” to specify the range of uncertainty on either side of the entered Time Interval end point, e.g. “Before 2Y --1M ++2M” (2 years prior, then in range 1 month before and 2 months after the interval end point).

ACTIONS

For Historical Date Definitions of a particular type of Node or Link Entity focus, on request:

1. Create a new Historical Date Definition
2. View or Edit the Historical Date Definition
3. Check the syntax of a Historical Date Definition
4. Check that data entry field values match required field formats
5. Check before deleting a Historical Date Definition that no Historical Date data uses that Definition.

On request for Historical Date Instances of a particular type and Entity focus:

1. Allocate storage for a new Historical Date and preset it with unset values
2. Allow entry and editing of individual Historical Date values. This includes the conversion from other known calendars when conversion methods are known
3. Retrieve a defined Historical Date converted to another specified known calendar
4. Retrieve a defined subset of the Historical Dates as a chronologically sorted list
5. Given 1 Historical Date:
    a. Return True/False if date is AD or not
    b. Return True/False if date is Gregorian or not
    c. Return True/False if date has no modifiers.
6. Given 2 Historical Dates:
    a. Return the earliest date
    b. Return the latest date
    c. Return True/False if dates are equal or not equal
    d. Return the difference (first minus second) as an Historical Interval.
7. Given an Historical Date and an Historical Interval:
    a. Return the sum as Historical Date
    b. Return the difference (first minus second) as an Historical Date.

On request for <u>Historical Interval Definitions</u>:

1. Create a new Historical Interval Definition
2. View or Edit the Historical Interval Definition
3. Check the syntax of a Historical Interval Definition
4. Check that data entry field values match required field formats
5. Check before deleting a Historical Interval Definition that no Historical Date data uses that Definition.

On request for <u>Historical Interval Instances</u> of a particular type and Entity focus:

1. Allocate storage for a new Historical Interval and preset it with unset values
2. Allow editing of individual Historical Interval values
3. Retrieve a defined subset of the Historical Interval as a chronologically sorted list
4. Given 2 Historical Intervals
    a. Return the shortest interval
    b. Return the largest interval
    c. Return True/False if intervals are equal or not equal
    d. Return the difference (first minus second) as an Historical Interval
    e. Return the sum of the 2 intervals as an Historical Interval
    f. Return the difference (first minus second) as an Historical Interval.

HISTORICAL DATE VALUE ENCODING

This discusses a proposed method of encoding qualified Historical Dates as a triplet of 64bit integers such that the numerical values provide a consistent sorting sequence for date display ordering.

The digit position within an integer will be numbered away from zero, where 1 is the last decimal digit and has value range (0 to 9). Digit 2 has value set (10, 20... 90), etc. These integers are composed of sections with different uses. Encodings can use different number of decimal digits for each element. In some encodings some elements are omitted. Encoding can be performed by multiplication by powers of ten and addition. Decoding can be performed absolute value, integer mod and subtraction.

Encoding Type

Digit 1- the date encoding type:

- If Digit 1 Value = 0, then date values is an **Irregular Date** (a string with no numerical conversional available). The date text is saved in another string field, and the encoded date value is -9,999,999,999,999,999,990
- If Digit 1 Value = 2, the date encoding type is a **Default Date**.
    - If the encoded date value is negative, then it is a Before Christian Era date (BCE). It has a year range of 999,999 BCE to the end of BCE
    - If the encoded date is positive then the encoding has the range of AD 1 to 999,999 years AD, with units of years, months, days, hours, minutes, seconds and milliseconds.
- If Digit 1 Value = 3, the date encoding type is a **Default Interval**. (see next section)
    - If the encoded date value is negative, then it is a Before Christian Era date (BCE). It has a year range of 999,999 BCE to the end of BCE
    - If the encoded date is positive then the encoding has the range of 1 AD to 999,999 years AD, with units of years, months, days, hours, minutes, seconds and milliseconds.

Offset Unit Code

- Digit Value = 0, Milliseconds
- Digit Value = 1, Seconds
- Digit Value = 2, Minutes
- Digit Value = 3, Hours
- Digit Value = 4, Days
- Digit Value = 5, Months
- Digit Value = 6, Years
- Digit Value = 7, Decades
- Digit Value = 8, Centuries
- Digit Value = 9, Millennia.

Qualifier Type

QUALIFIER	Date 1 Qualifier Code	Offset 1 Qualifier Code	Date 2 Qualifier Code	Offset 2 Qualifier Code
“Before”	1	1	7	7
“Before –NN Units”(bounded)	2	2	7	7
“Between”	3	3	3	3
“Within”	4	4	4	4

“Circa”	5	1	5	9
“Circa +- N Units” (bounded)	6	2	6	8
NO Qualifier	7	7	7	7
“After ++ NN Units”(bounded)	7	7	8	8
“After”	7	7	9	9

<u>DEFAULT Historical Date Encoding </u>(Encoding type 2)

The DEFAULT type has a range from -999,999 years to +999,999 years in 1 second increments. It uses 3 64 bit integers, the first 2 for dates and the third for offsets.

Integers 1 and 2:

- <u>Part A</u>: Digits 18-14 (6 digits) is the Year value
- <u>Part B</u>: Digits 13-12 (2 digits) is the Month value
- <u>Part C</u>: Digits 11-10 (2 digits) is the Hour value
- <u>Part D</u>: Digits 9-8(2 digits) is the Minutes value
- <u>Part E</u>: Digits 7-6 (2 digits) is the Seconds Value
- <u>Part F</u>: Digits 5-3 (3 digits) is the Milliseconds Value
- <u>Qualifier Code</u>: Digit 2: Controls the date integer’s qualifier
- <u>Encoding Type</u>: Digit 1: value 2.

Integer 1 and Integer 2 – First Date and Second Date

	Part A Value	**Part B Value**	**Part C Value**	**Part D Value**	**Part E Value**	**Part E Value**	**Qualifier Code**	**Encoding Type**
Digits	5	2	2	2	2	3	1	1
Units	Years	Months	Hours	Minutes	Seconds	Milli-seconds	“Before”	“Normal”

Integer 3:

- <u>Qualifier 1 Code</u>: Digit 17 – Stores the first date’s qualifier
- <u>Offset 1 Unit Code</u>: Digit 16 – Specifies Offset 1 Units
- <u>Offset 1 Value</u>: Digits 15-9 (6 digits) that hold the date offset in the selected units. These are applied in a <u>negative</u> time sense in the first date
- <u>Offset 2 Unit Code</u>: Digit 8 – Specifies Offset 2 Units
- <u>Offset 2 Value</u>: Digits 7-2 (6 digits) that hold the date offset in the selected units. These are applied in a <u>positive</u> sense in the second date integer
- <u>Encoding Type</u>: Digit 1: value 1.

Integer 3 – Modifiers for First Date and Second Date

	Qualifier 1 Code	**Offset 1 Unit Code**	**Offset 1 Value**	**Qualifier 2 Code**	**Offset 2 Unit Code**	**Offset 2 Value**	**Encoding Type**
Digits	1	1	6	1	1	6	1
Units	“Before”	“Years”		“After”	“Days”	10	“Normal”

HISTORICAL INTERVAL VALUE ENCODING

This discusses a proposed method of encoding qualified Historical Intervals as a triplet of 64bit integers such that the numerical values provide a consistent sorting sequence for date display ordering.

<u>DEFAULT Historical Interval Encoding </u>(Encoding type 3)

This encoding has the same layout for all 3 64 bit integers. The positive and negative values represent the difference between two dates, the sum or difference between 2 date intervals. (The Modifiers integer is still as encoding 1).

USED BY

Any GUI screen that operates on:

1. Historical Date Definitions
2. Historical Date Instances
3. Historical Interval Definitions
4. Historical Interval Instances.

AUXILARY DATA USED

1. HistoricalDate DBT
2. HistoricalDateDefn DBT
3. HistoricalInterval DBT
4. HistoricalIntervalDefn DBT.

REQUIRED SERVICES

1. BR_EncodedString
2. BR_EntityLink
3. BR_FieldTranslation.

WARNING CONDITIONS

1. Need details of the condition that raised the warning, example message and possible next steps.

ERROR CONDITIONS

1. Need to record the condition that raised the error, example message and possible next steps.