D/a/y

Data representations

Peter Fox author@vulpeculox.net

Contents

28 May 2015 (10:54AM)

1 Introduction

Efficient storage of Days is important as is the ability to sort sensibly out-of-the-box. The native 32-bit encoding supports dates and intervals and is the preferred method of data storage and transmission.

Existing systems may require interfacing with Day and this can cause issues as they have no special values and don't support partial dates such as say 'May 2013'. For these we create or consume hacked versions which use the hours, minutes and seconds elements to give a reasonable coverage of important Day values. Sometimes we want to store and read these bits and in others ignore them. For example to put a 'End-of-time' value into a legacy database 'date-time' field for later retrieval we will need the bits, while with a 'date of last login' field they would be ignored.

In general Days should be inter-operable with existing date-time systems with minimal code changes.

All encodings should be lossless in the Day -> Encoding -> Day sequence.

Intervals are only supported by the native 32-bit encoding.

Conversion functions

For all new data storage use the 32 bit version

```
    myInt32var = myDay.To32Bits();
    myDay = new DAYo(myInt32var);
```

Javascript dates use

```
    myJsDate = myDay.ToDate();   // if h,m,s in myJsDate are 0 then  y,m,d are honest
    myDay = new DAYo(myJsDate);
```

Database strings will probably use

```
    myDateTimeStr = myDay.ToYmdhms('YMDHMS');
    myDay = new DAYo(myDateTimeStr, DAYu.strTypeExtendedUnix);
```

2 Native 32-bit

From most significant to least significant bits:

Table A – Native encoding specification

Bits	Size	Usage	Notes
31 - 29	3	Signature	See table in main document
28	1	Sign	0: –ve 1:+ve
27 - 16	12	Year	0 ... 4095
15 - 12	4	Month	1:Jan ... 12:Dec 0 is legal
11 - 7	5	Day of month	1 ... 31 0 is legal
6 - 4	4	Day of week (Fully specified)	1:Mon ... 7:Sun 0:Undefined
		Not valid reason code (NV)	See table in main document
3	1	Not used	
2	1	Valid flag	1:Is validated
1	1	Fully specified	1:Is fully specified
0	1	Interval	1:Is interval

- The sign is applied to the year for a date or to a whole interval.
- Day of week is for convenience. Should be set to 0 if not applicable or not given
- Note that from an encoding point of view it is possible to have a signature of NV followed by Y,M,D values and a not-valid reason. There is a possible use for this where dates are being supplied in bulk by a process that can't reject the data outright.
- Bits 2 - 0 are for quick-reference convenience
- This will sort according to signature with BC dates before AD dates
- Day and month are zero-based so 8 Feb 2015 will be encoded as 2015,1,7

Table B – Native encoding layout

Day	Description	32 bit
General date layout		`sss±yyyy|yyyyyyyy|mmmmdddd|dwww×vf0`
General interval layout		`001±yyyy|yyyyyyyy|mmmmdddd|d××××v×1`
NVI	Not valid interval	`000×××××|××××××××|××××××××|×rrrr0×1`
NV	Not valid date	`010×××××|××××××××|××××××××|×rrrr0×0`
NK	Not known	`100×××××|××××××××|××××××××|××××××××`
BoT	Beginning of time	`101×××××|××××××××|××××××××|××××××××`
EoT	End of time	`111×××××|××××××××|××××××××|××××××××`
Year only eg 2009		`01100111|11011000|00000000|0000×100`
Year-month only eg Mar 2009		`01100111|11011000|00110000|0000×100`
Fully specified eg 14th Mar '09		`11000111|11011000|00110111|0110×110`

`×` ... not used	`s` ...signature	`w`... day of week
`r` ... reason code	`v` ... validity	`f` ... fully specified
`i` ... signs	`z` ... zero m/d	`q` ... 64-year offset

3 Javascript Date

WARNING Javascript dates are full of tricky trips!

- get/setYear() are faked. Always use get/setFullYear().
- Years less than 100 are assumed to be 1900+
- Javascript date has a year 0! In a real calendar the year after 1BC is 1AD.
- get/set and Date() constructor work on timezoned dates which jiggle about. The following *may* fail *depending on the current date*:

```
d = new Date(2010,1,1);
console.log(d.getDate() == 1);  // day of month _should be_ 1
```

- Always use UTC version of Date methods.
- To create a new date from scratch use d = new Date(Date.UTC(y,m,d,h,m,s));

The full year range is ample for us. We will use the same year range as for the 32-bit native encoding. However we still need to use H and M for Days that are not fully specified calendar dates.

Earliest calendar day : 1st Jan 4095 BC
Latest calendar day : 31st Dec 4095 AD

Note that Day has no year zero but Javascript allows it. We will encode 1BC as -1.

See the external encoding table below

4 Unix timestamp

The object of this is to allow direct plug-in to existing systems. ~~However we don't want to be held hostage by the possibility of the 2038 limit so we hijack the seconds to give us a years offset for the main year value.~~[1]

See the external encoding table below

[1] This project was started in 2007. Now, in 2015, if you're using a system that has the original UNIX timestamp limitations then you should go back to using stone circles to work your dates.

Table C – External encoding

Day signature	**D M Y**	**H M S**	**Comment**
NVI	30 Dec 4096 BC	10 reason 0	
INT	30 Dec 4096 BC	1 0 0	Not supported
NV	30 Dec 4096 BC	2 reason 0	
FLO	d m 0	3 mask 0	JS allows year zero!
NK	31 Dec 4096 BC	4 0 0	
BOT	31 Dec 4096 BC	5 0 0	
Earliest CAL	1 Jan 4095 BC	0 0 0	Fully specified CALs will always have 00:00:00 time
CAL	d m y	0 mask 0	
Latest CAL	31 Dec 4095	0 0 0	
EOT	1 Jan 4096	7 0 0	

- *reason* is the not valid reason
- Hours are used as a signature (Note 10=NVI and 0=CAL)
- *mask* is three bits (y-m-d) set to 1 if element is to be ignored. For example 28 March (no year) would be encoded as 1-3-28 03:01:00
- Date object with zero years are not allowed as inputs to Day conversion routines.
- d and m are 1 for first and 1 for January. (Contrast with Day where they are zero-based.)

Table E – Extended Unix encoding specification

Element	**Used for**	**Notes**
hour	Signature with 0 as fully qualified CAL and 10 as NVI	• NVI isn't supported. • Hour of 0 should be a real date. • Same encoding as Javascript
minute	YMD Zero flags mask	Bits set if missing
	Invalid reason (NV only)	See table in main document

- Floating and partially specified dates cause problems because there is no way to specify zero years, months or days in the standard Ymdhms scheme. This can be handled by using three flags combined to tell us when to ignore the Y,M or D values. For example "March 1988" would be flagged as 001 or 1 minute, "2009" as 011 or 3 minutes.

5 Arrays and strings

For string input details see other documentation.

There might be interfacing cases where it is convenient to use a string to pass data to Day or output an array of y,m,d,h,m,s from Day to be consumed by another application. Both of these are easy to implement as fully qualified dates but we need to be precise if we're trying to communicate Day-ish data.