How to recognize different types of timestamps from quite a long way away

devblogs.microsoft.com/oldnewthing/20030905-02

September 5, 2003

Raymond Chen

The great thing about timestamps is that there are so many to choose from. Sometimes, while debugging (or reading incomplete documentation) you'll find a timestamp and wonder how to convert it into something readable. Here are some tips.

We will use November 26, 2002 at 7:25p PST as our sample time.

UNIX timestamps are in seconds since January 1, 1970 UTC. It is a 32-bit number, the only 32-bit number in common use as a timestamp.

November 26, 2002 at 7:25p PST = 0x3DE43B0C.

If it's a 32-bit value starting with "3", it's probably a UNIX time. (The "3" era began in 1995 and ends in 2004.)

To convert these values to something readable, you have several choices.

The C runtime time_t value is the same as a UNIX timestamp, so you can use the `ctime()` function, for example.

This is the time format used by the C runtime and by the Windows NT event log.

Number two: The Win32 FILETIME

Win32 FILETIME values count 100-nanosecond intervals since January 1, 1600 UTC. It is a 64-bit number.

November 26, 2002 at 7:25p PST = 0x01C295C4:91150E00.

If it's a 64-bit value starting with "01" and a letter, it's probably a Win32 FILETIME. The "01A" era began in 1972 and the "01F" era ends in 2057.

To convert these values to something readable, you can use the `FileTimeToSystemTime()` function followed by `GetDateFormat()` and `GetTimeFormat()` .

Number three: The CLR System.DateTime

Warning: Actual .NET content (I'm sorry). CLR System.DateTime values count 100-nanosecond intervals since January 1, 1 UTC. It is a 64-bit number. These aren't used much yet.

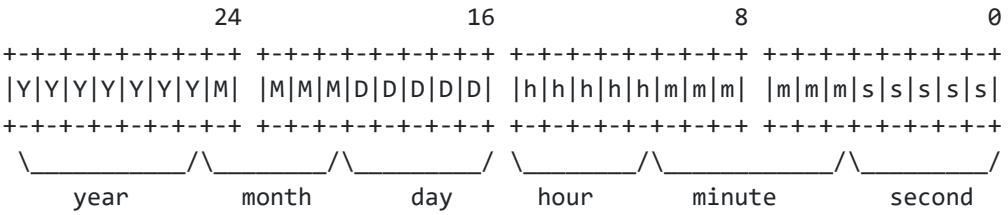
November 26, 2002 at 7:25p PST = 0x08C462CB:FCED3800. (? somebody check my math)

If it's a 64-bit value starting with "08" and a letter, it's probably a CLR System.DateTime. The "08A" began in 1970 and the "08F" era ends in 2056.

To convert these values to something readable, construct a System.DateTime object passing the 64-bit time value as the constructor parameter.

Number four: The DOS date/time format

The DOS date/time format is a bitmask:



The year is stored as an offset from 1980. Seconds are stored in two-second increments. (So if the "second" value is 15, it actually represents 30 seconds.)

These values are recorded in *local time*.

November 26, 2002 at 7:25p PST = 0x2D7A9B20.

To convert these values to something readable, convert it to a FILETIME via DosDateTimeToFileTime, then convert the FILETIME to something readable.

Number five: OLE Automation date format

The OLE automation date format is a floating point value, counting days since midnight 30 December 1899. Hours and minutes are represented as fractional days.

Converting among these formats

Often there is no direct conversion between two formats; you will have to go through some intermediary formats.

UNIX timestamp to/from Win32 FILETIME

Converting a UNIX timestamp to a WIn32 FILETIME is described in KB article Q167297 and a scaled-down version of the article is also available in the Platform SDK. Some high school algebra will get you the reverse conversion.

FILETIME to/from SYSTEMTIME
Use FileTimeToSystemTime() and SystemTimeToFileTime().

FILETIME to/from System.DateTime
Use System.DateTime.FromFileTime() and System.DateTime.ToFileTime().

OLE date to/from System.DateTime
Use System.DateTime.FromOADate() and System.DateTime.ToOADate().

DOS date/time to/from FILETIME
Use DosDateTimeToFileTime() and FileTimeToDosDateTime().

DOS date/time to/from SYSTEMTIME
Parse it yourself.

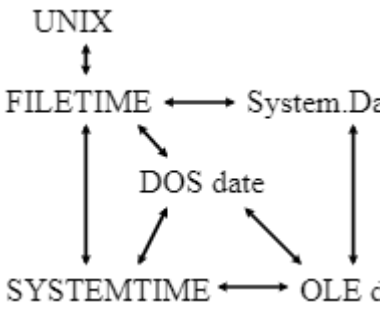
SYSTEMTIME to/from OLE date.
Use SystemTimeToVariantTime() and VariantTimeToSystemTime(), or use VarDateFromUdate() and VarUdateFromDate().

DOS date/time to/from OLE date.
Use DosDateTimeToVariantTime() and VariantTimeToDosDateTime().

Let's see if I can draw a little chart.

I'm not sure that chart actually cleared up anything.

If you allow yourself to use MFC, then there are some more conversions available.



UNIX time, FILETIME, SYSTEMTIME, or DOS date/time to OLE date format.
Use the MFC COleDateTime helper object.

I won't bother trying to add these (unidirectional) arrows to the chart above.

Brad Abrams' blog followed some of these arrows and produced a cute little formula to convert UNIX time_t directly to System.DateTime.

Other time formats

JScript's Date object constructor can construct from an integer representing milliseconds since 1970. This is the same as UNIX time, just multiplied by 1000.

Raymond Chen

Follow