Why Daylight Savings Time is nonintuitive

devblogs.microsoft.com/oldnewthing/20031024-00

October 24, 2003

Raymond Chen

> *Welcome Knowledge Base article 932955 readers!* Remember, the information on this Web site is not official Microsoft documentation.

Daylight Savings Time ends this weekend in most of North America and Europe, so it seems a good time to discuss the whole problem of Daylight Savings Time and timestamps.

A common complaint is that all the time zone conversion functions like FileTimeToLocalFileTime apply the *current* Daylight Savings Time (DST) bias rather than the bias that was in effect at the time in question.

For example, suppose you have a FILETIME structure that represents "1 January 2000 12:00AM". If you are in Redmond during the summertime, this converts to "31 December 1999 5:00PM", seven hours difference, even though the time difference between Redmond and UTC was eight hours at that time. (I.e., when people in London were celebrating the new year, it was 4pm in Redmond, not 5pm.)

The reason is that the time got converted from "1 January 2000 12:00AM UTC" to "31 December 1999 5:00PM PDT". So, technically, the conversion is correct. Of course, nobody was using PDT on December 31, 1999 in Redmond; everybody was on PST.

Why don't the time zone conversion functions use the time zone appropriate for the time of year?

One reason is that it means that FileTimeToLocalFileTime and LocalFileTimeToFileTime would no longer be inverses of each other. If you had a local time during the "limbo hour" during the cutover from standard time to daylight time, it would have no corresponding UTC time because there was no such thing as 2:30am local time. (The clock jumped from 2am to 3am.) Similarly, a local time of 2:30am during the cutover from daylight time back to standard time would have two corresponding UTC times.

Another reason is that the laws regarding daylight savings time are in constant flux. For example, if the year in the example above was 1977 instead of 2000, the conversion would have been correct because the United States was running on year-round Daylight Savings Time due to the energy crisis. Of course, this information isn't encoded anywhere in the TIME_ZONE_INFORMATION structure. Similarly, during World War 2, the United States went on DST all year round. And between 1945 and 1966, the DST rules varied from region to region.

DST rules are in flux even today. The DST cutover dates in Israel are decided on a year-by-year basis by the Knesset. As a result, there is no deterministic formula for the day, and therefore no way to know it ahead of time.

(Warning: .NET content ahead; two days in a row, what's gotten into me!?)

Compare the output of FileInfo.LastWriteTime.ToString("f") with what you see in the property sheet for a file that was last written to on the other side of the DST transition. For example, suppose the file was last modified on October 17, during DST but DST is not currently in effect. Explorer's file properties reports Thursday, October 17, 2003, 8:45:38 AM, but .NETs FileInfo reports Thursday, October 17, 2003, 9:45 AM.

En gang til for prins Knud: Win32 does not attempt to guess which time zone rules were in effect at that other time. So Win32 says, "Thursday, October 17, 2002 8:45:38 AM PST". Note: Pacific **Standard** Time. Even though October 17 was during Pacific **Daylight** Time, Win32 displays the time as standard time because that's what time it is now.

.NET says, "Well, if the rules in effect now were also in effect on October 17, 2003, then that would be daylight time" so it displays "Thursday, October 17, 2003, 9:45 AM PDT" – **daylight** time.

So .NET gives a value which is more intuitively correct, but is also potentially incorrect, and which is not invertible. Win32 gives a value which is intuitively incorrect, but is strictly correct.

I suspect that the .NET behavior was for compatibility with Visual Basic, but I don't know for sure.

Raymond Chen

Follow