# Quest:  Write a Leap Second Aware Application on Windows

We all know that there are 60 seconds in every minute.  Right?  Except that is not always true.  As it turns out the earth does not spin at a constant rate.  Periodically a second is added to a day.  That extra second is called a [Leap Second](https://en.wikipedia.org/wiki/Leap_second).  The last leap second was on December 31, 2016 at 23:59:60.  UTC time.  When this occurred the correct local time in PST was:

December 31, 2016, 15:59:57 PST

December 31, 2016, 15:59:58 PST

December 31, 2016, 15:59:59 PST

December 31, 2016, 15:59:60 PST

December 31, 2016, 16:00:00 PST

December 31, 2016, 16:00:01 PST

December 31, 2016, 16:00:02 PST

Windows Now supports Leap seconds!  How does it work?  Windows will get notified of a leap second through Windows Update or from a sync time operation (over [NTP](https://en.wikipedia.org/wiki/Network_Time_Protocol)).  When a leap second occurs the system clock will add the extra ‘60’ second.  However applications must ‘Opt in’ to receive the ‘60’ second and become leap second aware.  Generally an application gets the time by either calling [GetSystemTime](https://msdn.microsoft.com/en-us/library/windows/desktop/ms724390(v=vs.85).aspx) or [GetLocalTime](https://msdn.microsoft.com/en-us/library/windows/desktop/ms724338(v=vs.85).aspx).  This returns a SYSTEMTIME [structure](https://msdn.microsoft.com/en-us/library/windows/desktop/ms724950(v=vs.85).aspx).  If an application is ‘Opted In’ it will receive a value of 60 in the wSecond field of SYSTEMTIME when a leap second occurs.  If they are not opted in it will instead receive a value of 59.  This has the affect of making the ‘59’ second 2000 milliseconds during a leap second.

This table demonstrates the output of GetSystemTime and GetLocalTime if an application is opted in or not.

|  |  |  |
| --- | --- | --- |
| Official Time | ‘Opted in’ to Leap Seconds | Not ‘Opted in’ to leap seconds |
| December 31, 2016, 15:59:57 PST | December 31, 2016, 15:59:57 PST | December 31, 2016, 15:59:57 PST |
| December 31, 2016, 15:59:58 PST | December 31, 2016, 15:59:58 PST | December 31, 2016, 15:59:58 PST |
| December 31, 2016, 15:59:59 PST | December 31, 2016, 15:59:59 PST | December 31, 2016, 15:59:59 PST |
| December 31, 2016, 15:59:60 PST | December 31, 2016, 15:59:60 PST | December 31, 2016, 15:59:59 PST |
| December 31, 2016, 16:00:00 PST | December 31, 2016, 16:00:00 PST | December 31, 2016, 16:00:00 PST |
| December 31, 2016, 16:00:01 PST | December 31, 2016, 16:00:01 PST | December 31, 2016, 16:00:01 PST |
| December 31, 2016, 16:00:02 PST | December 31, 2016, 16:00:02 PST | December 31, 2016, 16:00:02 PST |

To complete this quest create a program that Opts in to Leap Seconds, and thus becomes Leap Second Aware.  Ideally you should do this to an existing program you have source control over.  To Opt in you need to call [SetProcessInformation](https://msdn.microsoft.com/en-us/library/windows/desktop/hh448389(v=vs.85).aspx).  Here is the sample. The code below will opt in and then print the to screen. If you run this during a leap second event you will get the ‘60’ second.

#include "Windows.h"

#include <iostream>

int wmain() {

std::cout << "Testing Leap Second update actually shows '60' in the seconds...\n";

DWORD ErrorCode;

BOOL Success;

ULONG Flags;

PROCESS\_LEAP\_SECOND\_INFO LeapSecondInfo;

//

// Opt in to Leap Seconds

//

ZeroMemory(&LeapSecondInfo, sizeof(LeapSecondInfo));

Flags |= PROCESS\_LEAP\_SECOND\_INFO\_FLAG\_ENABLE\_SIXTY\_SECOND;

LeapSecondInfo.Flags = Flags;

Success = SetProcessInformation(GetCurrentProcess(),

ProcessLeapSecondInfo,

&LeapSecondInfo,

sizeof(LeapSecondInfo));

if (!Success) {

ErrorCode = GetLastError();

fprintf(stderr, "Set Leap Second Info failed: %d\n", ErrorCode);

}

// print current time 60 times with sleep interval of 950ms

SYSTEMTIME lt;

for (int i = 0; i < 60; i++) {

GetLocalTime(&lt);

printf("the local time is %d/%d/%dT%02d:%02d:%02d\n", lt.wYear, lt.wMonth, lt.wDay, lt.wHour, lt.wMinute, lt.wSecond);

Sleep(950);

}

}

The code above requires the latest SDK for Windows 10. Since you are a early adopter Windows Insider who loves running the latest things you may be running this before the SDK is ready. But fear not! Here is a version of the same code which does not rely on the SDK as it defines everything you need and casts the values in the correct places:

#include "Windows.h"

#include <iostream>

typedef struct \_PROCESS\_LEAP\_SECOND\_INFO {

ULONG Flags;

ULONG Reserved;

} PROCESS\_LEAP\_SECOND\_INFO, \*PPROCESS\_LEAP\_SECOND\_INFO;

#define PROCESS\_LEAP\_SECOND\_INFO\_FLAG\_ENABLE\_SIXTY\_SECOND 0x1

#define ProcessLeapSecondInfo 8

int wmain() {

std::cout << "Testing Leap Second update actually shows '60' in the seconds...\n";

DWORD ErrorCode;

BOOL Success;

ULONG Flags;

PROCESS\_LEAP\_SECOND\_INFO LeapSecondInfo;

//

// Opt in to Leap Seconds

//

ZeroMemory(&LeapSecondInfo, sizeof(LeapSecondInfo));

Flags |= PROCESS\_LEAP\_SECOND\_INFO\_FLAG\_ENABLE\_SIXTY\_SECOND;

LeapSecondInfo.Flags = Flags;

Success = GetProcessInformation(

GetCurrentProcess(), (PROCESS\_INFORMATION\_CLASS) ProcessLeapSecondInfo,

&LeapSecondInfo,

sizeof(LeapSecondInfo));

if (!Success) {

ErrorCode = GetLastError();

fprintf(stderr, "Set Leap Second Info failed: %d\n", ErrorCode);

}

// print current time 60 times with sleep interval of 950ms

SYSTEMTIME lt;

for (int i = 0; i < 60; i++) {

GetLocalTime(&lt);

printf("the local time is %d/%d/%dT%02d:%02d:%02d\n", lt.wYear, lt.wMonth, lt.wDay, lt.wHour, lt.wMinute, lt.wSecond);

Sleep(950);

}

}

After you compile your leap second aware application you need to test it.  You have a few choices:

1. Wait for the next leap second to occur.  (There are non scheduled at the moment so this is a long wait.  You can help this along by trying to slow down the earth’s rotation by running really fast toward the east.
2. Introduce a leap second to the OS using w32tm.exe:

You can get print a report of all Leap Seconds Windows currently knows about by running:

C:\>w32tm /leapseconds /getstatus /verbose

[Leap Seconds]

Enabled: 1 (Local)

Number of Leap Seconds: 4 (Local)

Leap Seconds List (Local):

+2018-06-01T23:59:59

+2018-06-02T23:59:59

+2018-06-03T23:59:59

+2018-06-04T23:59:59

You can Add a leap second using the following (“Note this is in UTC time”).

C:\>w32tm /leapseconds /add:+2018-06-05T23:59:59 /force

The command completed successfully.

Test your application.  Verify that it is getting the ‘60’ second.  Also verify your application knows how to handle the ’60’ second so it doesn’t crash, or render it incorrectly.

Known issues:

Some frameworks are known to calculate time incorrectly after a leap second occurs.  For example, the .NET Framework uses its own internal logic to determine what time it is.  Its logic does not account for leap seconds.  So after a leap second is introduced to the Operating System the output of “System.DateTime.Now.ToString()” will be ahead by one second of the local system time.  (We are working with the .NET framework team on this.)