# If you are going to call Marshal.GetLastWin32Error, the function whose error you're retrieving had better have SetLastError=true

devblogs.microsoft.com/oldnewthing/20150818-00

August 18, 2015

Raymond Chen

A customer reported that their p/invoke to a custom DLL was failing, and the error code made no sense.

```
// C#
using System;
using System.Runtime.InteropServices;
using System.Diagnostics;

class Program
{
  [DllImport("contoso.dll", CallingConvention=CallingConvention.Cdecl)]
  public static extern int Fribble();

  public static void Main()
  {
    Console.WriteLine("About to call Fribble");

    var result = Fribble();
    if (result >= 0) {
      Console.WriteLine("succeeded {0}", result);
    } else {
      Console.WriteLine("failed {0}, last error = {1}",
                        result, Marshal.GetLastWin32Error());
    }
  }
}

// C++

int __cdecl Fribble()
{
 HANDLE hEvent = OpenEvent(EVENT_MODIFY_STATE, FALSE,
                           TEXT("FribbleEvent"));
 if (hEvent == nullptr)
  return -1;
 }

 if (!SetEvent(hEvent)) {
  CloseHandle(hEvent);
  return -2;
 }

 CloseHandle(hEvent);
 return 1;
}
```

The customer reported that their `Fribble` function was returning −1, indicating a failure to open the event, but the error code returned by `Marshal.GetLastWin32Error` is 87, “The parameter is incorrect.” But all of the parameters to `OpenEvent` look correct. Why are we getting this strange error code?

My psychic powers tell me that if the customer had taken the time to troubleshoot their problem by writing a C++ program that calls the `Fribble` function, `GetLastError` would have returned the more reasonable error 2, meaning that the event does not exist.

That's because `GetLastError` is working fine. The last error code is 2.

The problem is with the p/invoke declaration.

The documentation for the `Marshal.GetLastWin32Error` function says as its very first line

> Returns the error code returned by the last unmanaged function that was called using platform invoke *that has the DllImportAttribute.SetLastError flag set*.

(Emphasis mine.)

This reminder about `DllImportAttribute.SetLastError` is repeated in the Remarks.

> You can use this method to obtain error codes only if you apply the System.Runtime.InteropServices.DllImportAttribute to the method signature and set the `SetLastError` field to **true**.

Observe that the `SetLastError` field was not set in the p/invoke declaration. Therefore, what you are actually getting when you call `Marshal.GetLastWin32Error` is whatever error was lying around after the previous call to a p/invoke function that *did* specify `SetLastError = true`.

Changing the p/invoke to

```
[DllImport("contoso.dll", SetLastError=true,
           CallingConvention=CallingConvention.Cdecl)]
public static extern int Fribble();
```

fixed the problem.

Raymond Chen

**Follow**