

Re: Math Error in the .NET Framework 1.1.4322 SP1

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.general/2005-04/msg00480.html>

- From: dan.c.roth@xxxxxxxxxx (Daniel Roth)
 - Date: 5 Apr 2005 19:40:47 -0700
-

Hi

Your problem is the way .NET processes the ==

Try this

```
double x = 8;
double y = 2;
double log = System.Math.Log(x,y);

if( Convert.ToDecimal(log) == Convert.ToDecimal(3) )
{
    Console.WriteLine("true");
}
else
{
    Console.WriteLine("false");
}
/// Output is true
Daniel Roth
MCS.D.NET
```

limelight <limelight@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:<A4308AC5-1069-459E-9A6D-CEAF5F9CD4B3@xxxxxxxxxxxxxxxx>...
> I have discovered a math error in the .NET framework's Log function. It
> returns incorrect results for varying powers of 2 that depend on whether the
> program is run from within the IDE or from the command line. The amount by
> which the calculation is off is very small; even though the double data type
> holds the errant value, it seems to round off when printed (via ToString())
> and shows the correct one. The problem is that the errant value is still used
> in further calculations, which can throw off some functions.
>
> Specific example:
>
> The function System.Math.Log(8,2) yields a value of $3 - 4 \times 10^{-16}$
> (2.9999999999999996) instead of 3 when run from a command line (this will not
> occur if run inside the IDE). You can store the result of this computation in

> a double precision variable, but if you print the variable's value to the
> console by calling its ToString() method, the output will be 3. You can
> verify, however, that the value is indeed off by printing the result of some
> calculations using it. To see this, use Visual Studio to create a new C#
> console application and paste the following code sample in its "Main" method:
>
> ----- Begin Code Sample -----
> // try log function using the overloaded method that specifies the base
> Console.WriteLine("Using System.Math.Log overload to specify desired
> base...");
> double log = System.Math.Log(8,2); // compute the log of 8 to the
> base of 2 (should be 3, but will be $3 - 4 \times 10^{-16}$ when run from a command
> prompt - note will print out as 3 regardless)
> double floor = System.Math.Floor(log); // get the floor of the result
> (should be 3, but will be 2 when run from a command prompt)
> Console.WriteLine("Log2 8 = " + log.ToString()); // print the log of 8 to
> the base of 2
> Console.WriteLine("Floor(Log2 8) = " + floor.ToString()); // print the
> floor of the result
> Console.WriteLine("Log2 8 == 3? : " + ((bool) (log == 3)).ToString()); //
> print whether the log of 8 to the base of 2 is equal to 3 (should be true,
> but will be false when run from a command prompt)
>
> Console.WriteLine();
>
> // try the log function while doing things the "manual" way
> Console.WriteLine("Using Log x / Log y method of computing the same
> logarithm...");
> log = System.Math.Log(8) / System.Math.Log(2); // compute the log of 8 to
> the base of 2 (will be 3)
> floor = System.Math.Floor(log); // get the floor of the result (will
> be 3)
> Console.WriteLine("Log2 8 = " + log.ToString()); // print the log of 8 to
> the base of 2 computed via Log x / Log y
> Console.WriteLine("Floor(Log2 8) = " + floor.ToString()); // print the
> floor of the result
> Console.WriteLine("Log2 8 == 3? : " + ((bool) (log == 3)).ToString()); //
> print whether the log of 8 to the base of 2 is equal to 3 (will be true)
>
> Console.Read();
> ----- End Code Sample -----
>
>
> Note that if you calculate via $\log x / \log y$ then the calculation is
> correct. Using ILDASM to examine mscorlib.dll reveals that this is what is
> actually happening internally. That is, the overloaded method
> System.Math.Log(double a,double newBase) actually calls
> System.Math.Log(double d) and divides the result by another call to
> System.Math.Log(double d). That there is a difference between the results
> returned by the framework when it does this internally as opposed to when you
> do it is certainly interesting.

Re: Math Error in the .NET Framework 1.1.4322 SP1

>
> You can see the actual value of the log function by putting a statement
> such as "Console.Read()" as your first line so you have a way to get the
> program to stop. Compile the program and run it from the command line. Open a
> new instance of Visual Studio and click on Tools > Debug Processes and select
> the application. Once you have attached the debugger, break the application
> and start stepping through the code. Add the variable "log" to your watch
> window and you will see its value is $3 - 4 \times 10^{-16}$, not 3 when computed using
> System.Math.Log(8,2). These steps must be taken, as the problem will not
> occur if you initially start the program within the IDE.
>
> Additionally, the problem can be verified with one line of code. Create a
> new C# console application and run the following code:
>
> Console.WriteLine(System.Math.Log(8,2) == 3);
>
> which will print "true" when run from within the IDE and "false" when
run
> from the command line.
>
> The interesting thing here is that the issue occurs on some powers of 2 if
> the program is run from the command line and different powers of two if run
> from within the IDE. I have verified this by computing the first 64
> logarithms of 2 from both IDE and command line. You can do so by running the
> following code:
>
> for(int pow = 1;pow <= 64;pow++)
> {
> if(System.Math.Log(System.Math.Pow(2,pow),2) != pow)
> Console.WriteLine(pow.ToString());
> }
>
> Additionally, you can compute the same logarithms in VB.NET and the answers
> will still be off. This rules out the problem being isolated to a single
> language and shows the issue is within the framework itself. As an
> interesting side check, I computed these logarithms from VB 6.0 and all
> passed the test, so the problem is isolated to .NET.
>
> Lastly, computation using $\log x / \log y$ is also off, but for different
> powers of 2 than what $\log(\text{power}, \text{newBase})$ is and it is more consistent; it
> computes the incorrect value for the same powers when run from either the
> command line or the IDE.
>
> I contacted Microsoft support regarding this and they verified that they
> have reproduced the problem. The last correspondence with them said, "[I am]
> creating a problem report to send to the dev team now".
>
> The most concerning thing about all of this is that programs, at least in
> this instance, behave differently when run from the command line than they do
> when run inside the IDE. I find the fact that the value of calculations is
> different depending on how you start your application to be alarming.

- **References:**

- ◆ **[Math Error in the .NET Framework 1.1.4322 SP1](#)**

- ◇ From: limelight

- Prev by Date: **[RE: A very persistant Access To Path Denied Error](#)**

- Next by Date: **[Re: Where to find Components](#)**

- Previous by thread: **[Math Error in the .NET Framework 1.1.4322 SP1](#)**

- Next by thread: **[Datagrid and Databinding](#)**

- Index(es):

- ◆ **[Date](#)**

- ◆ **[Thread](#)**