

Attaching ANTS Performance Profiler 6 to a running process

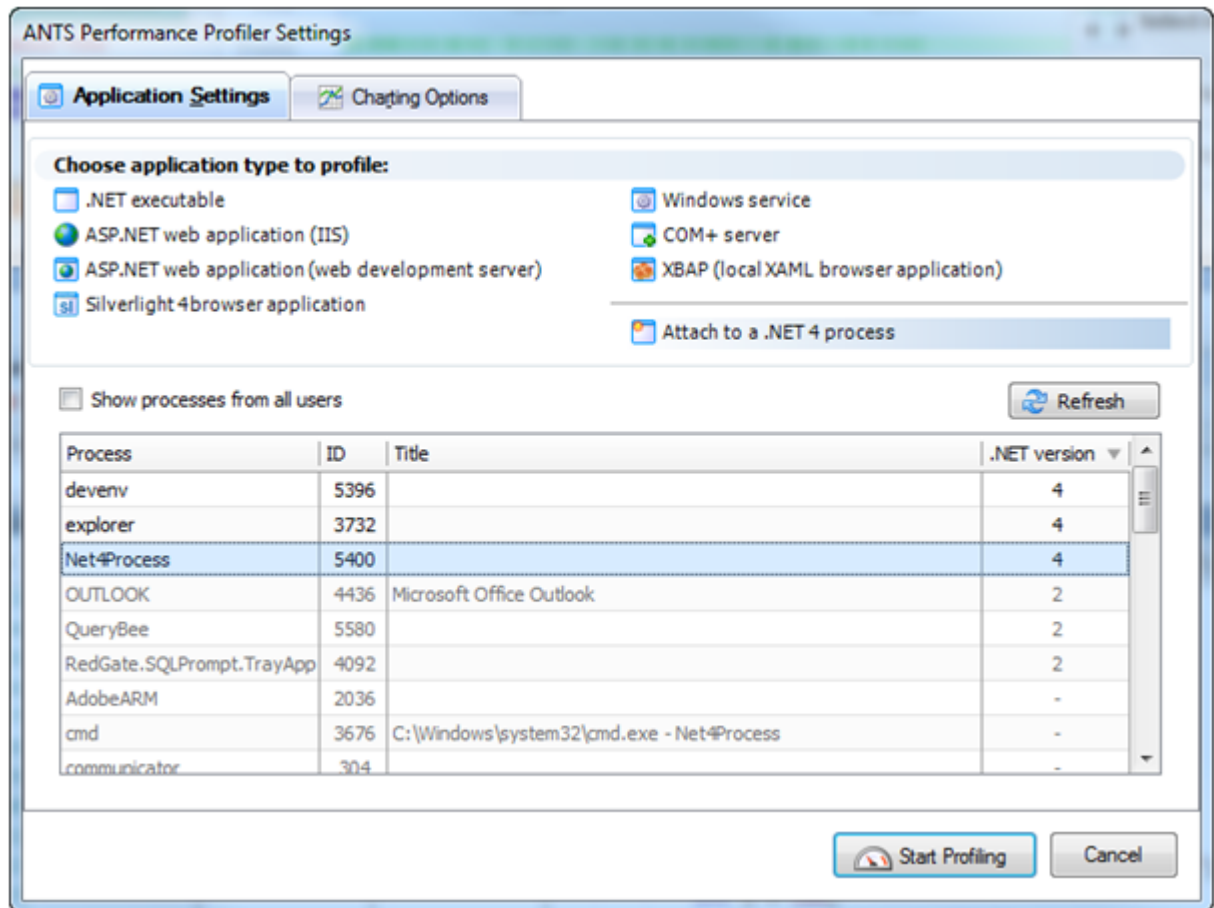
One of the new features in ANTS Performance Profiler 6 is the ability to attach the profiler to a .NET 4 process which is already running. The feature allows you to start profiling your applications when performance problems are noticed, without having to restart the application from scratch. This technical paper explains how to do this, using a simple C# program (called *Net4Process.exe*), which counts down for 300 seconds.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Net4Process
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Countdown test process");
            int i = 300;
            int[] array = new int[301];
            while (i >= 0)
            {
                Console.WriteLine(i + " seconds");
                array[i] = i;
                i--;
                System.Threading.Thread.Sleep(1000);
            }
        }
    }
}
```

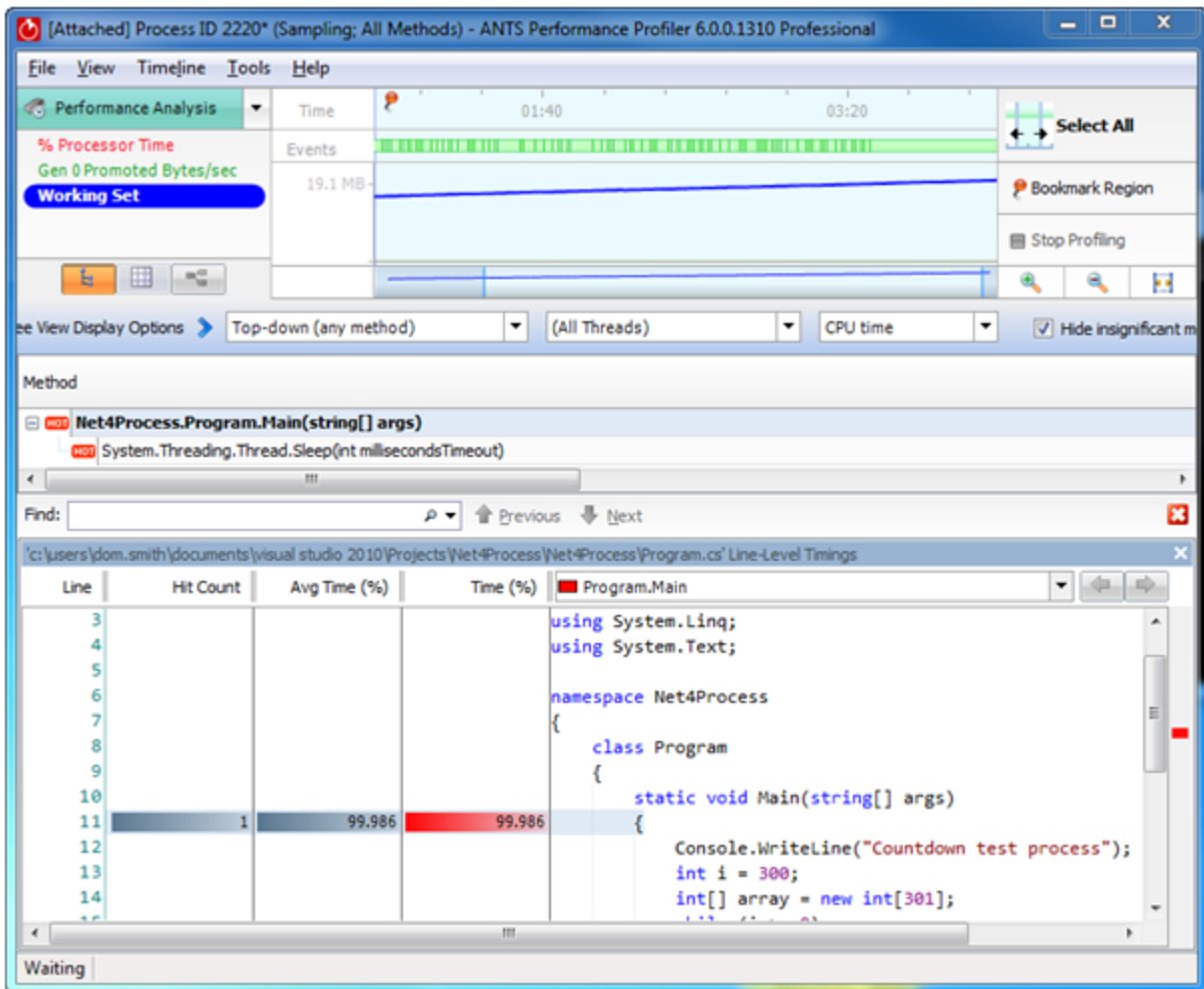
1. Start *Net4Process.exe* from the command prompt.
2. When the countdown starts, open ANTS Performance Profiler.
3. In the ANTS Performance Profiler Settings dialog, under **Application Settings**, select Attach to a .NET 4 process.

A list of the processes currently running is displayed with the .NET framework version that they use. Processes that are not .NET 4 processes are unavailable.



4. Select *Net4Process* and click **Start Profiling**.
5. When the application has finished (or if you press ANTS Performance Profiler's **Stop Profiling** button), the results are shown in the call tree in ANTS Performance Profiler.

As expected, almost 100% of the time is spent in `Main()` and the increasing amount of memory used by the program can be seen in the timeline.



If you need to save results for analysis later, attaching ANTS Performance Profiler to a running process means that you can save just the results you need, helping to reduce the size of the results file. This is particularly useful for line-level timings for all methods.