

1. ANTS Performance Profiler 7 documentation	3
1.1 Requirements	4
1.2 Installing	5
1.3 Licensing	6
1.3.1 Activating	7
1.3.2 Deactivating	13
1.3.3 Troubleshooting licensing and activation	16
1.4 Upgrading	19
1.4.1 Using Check for Updates	20
1.4.2 Troubleshooting Check for Updates errors	22
1.5 Setting up and running a profiling session	24
1.5.1 Working with Application Settings	26
1.5.2 Setting up Charting Options	30
1.5.3 ANTS Performance Profiler Options	32
1.5.4 Profiling .NET executables	34
1.5.5 Profiling managed code add-ins	35
1.5.6 Profiling ASP.NET applications running on IIS	36
1.5.7 Profiling ASP.NET applications running on the web development server	39
1.5.8 Profiling ASP.NET applications running on IIS Express	41
1.5.9 Profiling SharePoint	43
1.5.10 Profiling Silverlight applications	45
1.5.11 Profiling Windows services	47
1.5.12 Profiling a COMplus server application	49
1.5.13 Profiling XBAP applications	52
1.5.14 Attaching to a running .NET 4 process	53
1.5.15 Profiling SQL queries in ANTS Performance Profiler 7.2 and later	55
1.5.16 Profiling SQL queries in ANTS Performance Profiler 7.0	56
1.5.17 Profiling File IO	60
1.5.18 Profiling tests in MSTest	64
1.5.19 Profiling from the command line (API)	65
1.5.20 Integrating ANTS Performance Profiler in a test procedure	70
1.5.21 Using the Visual Studio add-in	71
1.5.22 Setting up continuous profiling	72
1.6 Working with profiling results	74
1.6.1 Working with the timeline	77
1.6.2 Working with the call tree	86
1.6.3 Tips on using the call tree	91
1.6.4 Working with integrated decompilation	92
1.6.5 Working with the methods grid	95
1.6.6 Working with the call graph	97
1.6.7 Working with the database calls view	103
1.6.8 Working with source code	105
1.6.9 Filtering the call tree and methods grid	106
1.6.10 Changing results display options	107
1.6.11 Working with continuous profiling	108
1.7 Worked examples	110
1.7.1 Worked example - Profiling performance of an algorithm	111
1.7.2 Worked example - Profiling network overheads	113
1.7.3 Worked example - Profiling an ASP.NET application - NerdDinner	114
1.7.4 Worked example - Profiling an ASP.NET application - TheBeerHouse	119
1.7.5 Worked example - Profiling from the command line	123
1.8 Troubleshooting	126
1.8.1 Common issues	127
1.8.1.1 Troubleshooting licensing and activation errors	128
1.8.1.2 Troubleshooting application crashes	131
1.8.1.3 Troubleshooting missing results	133
1.8.1.4 Troubleshooting PDB problems	134
1.8.1.4.1 Creating a global debugging symbols (PDB) directory	136
1.8.1.5 Troubleshooting SharePoint Profiling	137
1.8.1.6 Troubleshooting IIS profiling	143
1.8.1.7 Troubleshooting SQL and HTTP call profiling	145
1.8.2 Error messages	146
1.8.2.1 "No Disk" error occurring while profiling application	147
1.8.2.2 Couldn't open metabase	148
1.8.2.3 Error stopping IISAdmin profiling IIS web application on Windows XP	149
1.8.2.4 Failed to CoCreate Profiler	150
1.8.2.5 IIS ceases to work after profiling web applications	151
1.8.2.6 Method not found: 'UInt32 <Module>._ANTS_Begin_Sql(System.String)'	152
1.8.2.7 No .NET methods were profiled on web application	153
1.8.2.8 Operation could destabilize the runtime error profiling ASP.NET	155
1.8.2.9 Please specify a valid URL message profiling ASP.NET	156
1.8.2.10 The system cannot find the file specified	157

1.8.3 3rd party components	158
1.8.3.1 Profiling assemblies protected with DeployLX	159
1.8.4 Unexpected behavior / technical questions	160
1.8.4.1 ANTS Performance Profiler menu items not showing in Visual Studio 2010	161
1.8.4.2 Attach to process unavailable with some anti-virus software	162
1.8.4.3 Call graph percentages do not add up exactly	163
1.8.4.4 Can I profile Compact Framework applications?	164
1.8.4.5 Double hit counts occurring on one line	165
1.8.4.6 Enabling line-level timings for SecurityTransparent code	166
1.8.4.7 Failed to coCreate Profiler on ASP .NET web application	167
1.8.4.8 Forcing your application to use .NET 4	168
1.8.4.9 HTTP request timings in IIS	169
1.8.4.10 Isolating single ASP .NET pages in ANTS Profiler results	170
1.8.4.11 Log files	171
1.8.4.12 Memory leaks observed when profiling Windows Presentation Foundation (WPF) applications	172
1.8.4.13 Missing hits for lines in the source code view	173
1.8.4.14 Problems synchronizing results	174
1.8.4.15 Profiler prompts for location of source code which is not your own source code	175
1.8.4.16 Profiler stopping while profiling an in-browser Silverlight application	176
1.8.4.17 Profiling an assembly in the Global Assembly Cache (GAC)	177
1.8.4.18 Profiling ClickOnce applications deployed to IIS	178
1.8.4.19 Profiling Microsoft Office managed-code add-ins	179
1.8.4.20 Profiling unit tests using Nunit	180
1.8.4.21 Profiling web services in IIS Express	181
1.8.4.22 Setting file IO and child process profiling in high DPI modes	182
1.8.4.23 Showing the amount of time taken for a method in one particular thread	184
1.8.4.24 Times in source code window are greater than the times showing in the method grid or tree view	185
1.8.4.25 Times on individual lines do not add up to method time	186
1.8.4.26 Windows service profiling fails if the service uses a system account	187
1.9 Release notes and other versions	188
1.9.1 ANTS Performance Profiler 7.4 release notes	189
1.9.1.1 What's new in version 7.4	190
1.9.2 ANTS Performance Profiler 7.3 release notes	191
1.9.3 ANTS Performance Profiler 7.2 release notes	192
1.9.4 ANTS Performance Profiler 7.0 release notes	193
1.9.5 ANTS Performance Profiler 6.3 release notes	195
1.9.6 ANTS Performance Profiler 6.2 release notes	196
1.9.7 ANTS Performance Profiler 6.1 release notes	197
1.9.8 ANTS Performance Profiler 6.0 release notes	198
1.9.9 ANTS Performance Profiler 5.2 release notes	200
1.9.10 ANTS Performance Profiler 5.1 release notes	201
1.9.11 ANTS Performance Profiler 5.0 release notes	202

# ANTS Performance Profiler 7 documentation

## About ANTS Performance Profiler

ANTS Performance Profiler enables you to profile the code of applications written in any of the languages available for the .NET Framework, including Visual Basic .NET, C#, and Managed C++. This is useful, for example, to identify inefficient areas of your application by recording the time spent in each line of your code or method as you run your application.

You can use ANTS Performance Profiler to profile .NET desktop applications, ASP.NET web applications hosted in Internet Information Services (IIS) or the ASP.NET Development Server, .NET Windows services, COM+ server applications, Silverlight 4 or later applications, and XBAPs. In addition, you can profile applications that host the .NET Runtime, for example Visual Studio .NET plug-ins.

For more information, see the [ANTS Performance Profiler product page](#).

For information about new features, see [What's new in version 7.4](#).

## Quick start guide

1. [Set up a new profiling session, and start profiling.](#)
2. [Optionally, select a region on the timeline to restrict the profiling results to a specific period.](#)
3. [Review the profiling results.](#)

## Community content

[Using ANTS Performance Profiler to guide optimization](#)

[ANTS Performance Profiler integration with .NET Reflector](#)

[5 minute wonders: Finding lazy loading nasties with ANTS Profiler](#)

[Remove unused View Engines \(ASP.NET MVC\)](#)

[Speeding up your application with the IIS Auto-Start feature](#)

# Requirements

You can use ANTS Performance Profiler with the following versions of the .NET Framework:

- 1.1 (32-bit applications only)
- 2.0 (32-bit or 64-bit applications)
- 3.0 (32-bit or 64-bit applications)
- 3.5 (32-bit or 64-bit applications)
- 4.0 (32-bit or 64-bit applications)
- 4.5 (32-bit or 64-bit applications)

## Technical requirements

These are the minimum technical requirements for ANTS Performance Profiler:

- Windows XP, Windows Vista, Windows 7, Windows 8, Windows 2003 Server, Windows 2008 Server, Windows 2012 Server
- Microsoft .NET Framework version 3.5
- 512 MB RAM
- Internet Explorer 6+ (to profile applications)
- Internet Explorer 7+ (to view HTML profiling results)
- 1 GB free hard disk space

## Advanced features

The following features in ANTS Performance Profiler require Windows Vista or later (or Windows Server 2008 or later):

- File I/O performance counter
- SQL queries performance counter

The following feature in ANTS Performance Profiler requires Windows Vista or later (or Windows Server 2008 or later) and .NET 4:

- Attach to process

If you have any questions regarding the technical specification, please [contact us](#).

# Installing

Most Redgate products are available as part of a bundle. You can select which individual products to install when you run the installer.

When you install a non-free product, you have 14 days to evaluate the product. For the DLM Automation Suite, DLM Automation Suite for Oracle, SQL Source Control, Schema Compare for Oracle, Data Compare for Oracle, and Source Control for Oracle, you have 28 days. For more information, see [Licensing](#).

To install a Redgate product:

1. Download the product from the [website](#).
2. Run the installer and follow the instructions.

The product is listed on the **Start** menu under **Red Gate**.

# Licensing

When you install most Redgate products (apart from free ones), you have **14 days** to evaluate them without purchase.

For a few products, you have 28 days: DLM Automation Suite, DLM Automation Suite for Oracle, SQL Prompt, SQL Source Control, Source Control for Oracle.

If you need more time to evaluate a product, email [licensing@red-gate.com](mailto:licensing@red-gate.com).

## Finding your serial number

When you buy a license for a product, we'll send you an invoice that contains your serial number to activate the product. Your invoice shows how many instances of a product the serial number can be used to activate. For information about how to activate, see [Activating](#).

If you can't find your invoice, you can view your serial numbers at [red-gate.com/myserialnumbers](https://red-gate.com/myserialnumbers). You'll need to log in to your Redgate account with the email address and password you provided when you bought the product.

If you need to reinstall products on the same computer (eg after installing a new operating system), you can reactivate them using the same serial number. This doesn't affect the number of distinct activations for the serial number. For information about moving a serial number to a different computer, see below.

## Serial numbers for bundles and suites

If you've bought a bundle or suite of products, your serial number activates all the products in the bundle or suite. For bundles containing both server and client tools (such as the SQL DBA Bundle) you will have two serial numbers.

If you deactivate a bundle or suite serial number, all products using that serial number will be deactivated.

For information on which products are included in a bundle, see [Bundle history](#).

## Changing the serial number used to activate a product

To change the serial number used to activate a product, on the **Help** menu, select **Enter Serial Number**. For some products, you will need to deactivate the old serial number first.

## Moving a serial number to a different computer

To move a serial number to a different computer, deactivate the serial number on the old computer, then use it to activate the product on the new computer.

To deactivate a serial number, on the **Help** menu, select **Deactivate Serial Number**. If the Deactivate Serial Number menu item isn't available, use the [deactivation tool](#).

If you can't deactivate a serial number, use the [Request Extra Activations](#) page to request more activations for your serial number. You'll need to provide your serial number and the reason for the additional activations.

## Activating

This page applies to a number of Redgate products, so the screenshots below may not match your product.

When you activate a product with your serial number, the licensing and activation program sends an activation request to the Redgate activation server, using checksums of attributes from your computer. The checksums sent to the activation server do not contain any details that might pose a security risk. The activation server returns an activation response and an encrypted key to unlock the software. The licensing and activation program should activate your product within a few seconds.

If you experience problems with activating your products, you'll be directed to [activate manually](#).

- [Activating using the GUI](#)
- [Activating using the command line](#)
- [Manual activation](#)

### Activating using the GUI

These instructions apply to a number of Redgate products, so the screenshots below may not match your product.

To activate your products:

1. On the **Help** menu, click **Enter Serial Number**.

The product activation dialog box is displayed, for example:

**Activate SQL Compare**

**Enter your SQL Compare serial number**

**Serial number**

Your serial number is on your invoice or you can [find it online](#)


☒ **Track this activation**  
Sends information about this activation (including your machine name) to Red Gate.  
This is useful if you contact support about your activations. [More information](#)

If you purchased SQL Compare as part of a bundle, other products may be activated by this process. The products activated are listed when activation is completed.

**E-mail (optional)**  
Please provide the email address you would like us to send update notifications to:

☒ **I'd also like to receive the Red Gate Newsletter.** [Read our privacy policy](#)

redgate Activate Cancel

2. Enter your serial number.  
When you have entered a valid serial number,  
  
is displayed next to the serial number box:

**Activate SQL Compare**

### Enter your SQL Compare serial number

**Serial number**  
000-000-123456-0000 ✓

Your serial number is on your invoice or you can [find it online](#)

☒ **Track this activation**  
Sends information about this activation (including your machine name) to Red Gate.  
This is useful if you contact support about your activations. [More information](#)

If you purchased SQL Compare as part of a bundle, other products may be activated by this process. The products activated are listed when activation is completed.

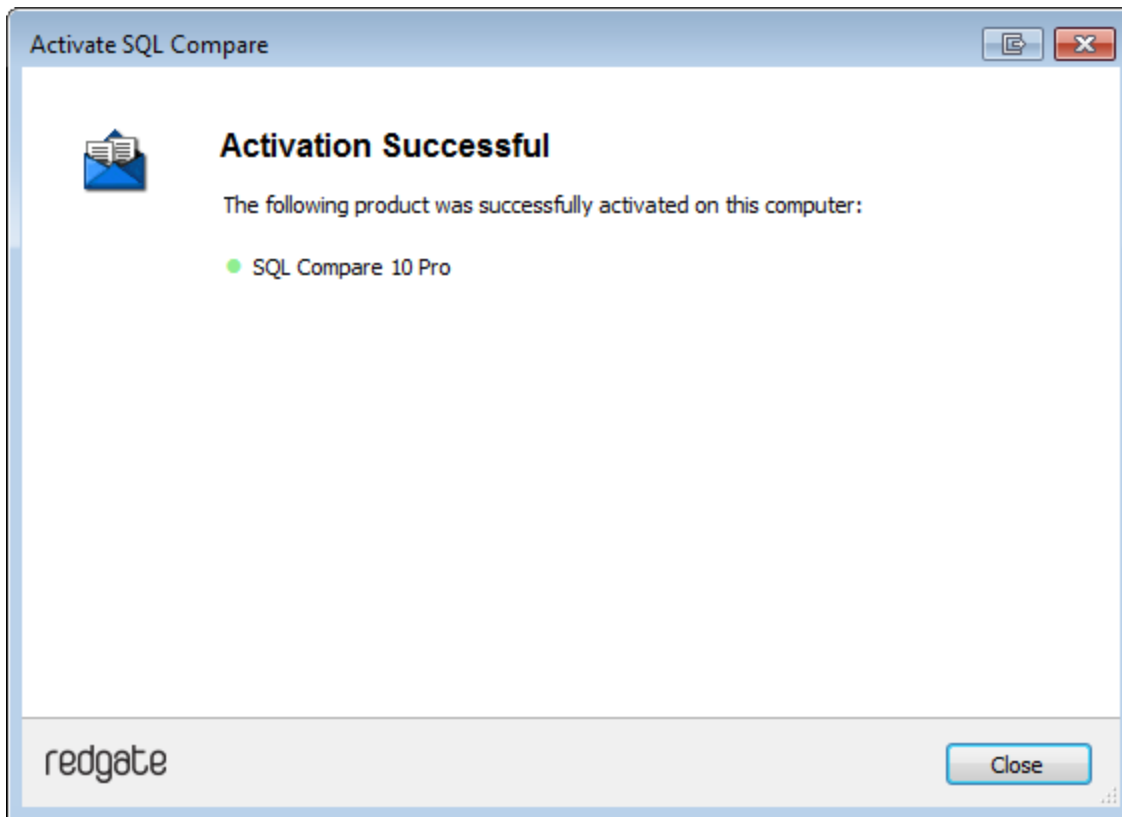
**E-mail (optional)**  
Please provide the email address you would like us to send update notifications to:  
user@example.com

☒ **I'd also like to receive the Red Gate Newsletter.** [Read our privacy policy](#)

redgate Activate Cancel

3. If you want to receive email updates from Redgate, enter your email address.  
The list of identifiers and your email address may already be populated using information available to the licensing client from the Windows installation on your computer. No information is sent back to Redgate when the fields are populated.  
When you activate your product, the optional information you entered is recorded by Redgate with your serial number. Your email address is not linked to the data collected should you consent to participate in the Quality Improvement Program provided with some Red Gate products.
4. Click **Activate**.  
Your activation request is sent to the Red Gate activation server.  
When your activation has been confirmed, the **Activation successful** page is displayed, for example:





If there is a problem with your activation request, an error dialog box is displayed. For information about activation errors and what you can do to resolve them, see [Troubleshooting licensing and activation errors](#). Depending on the error, you may want to try [manual activation](#).

5. Click **Close**.  
You can now continue to use your product.

## Activating using the command line

Open a command prompt, navigate to the folder where your product executable file is located and run a command with the following syntax:

```
<name of productEXE> /activateSerial:<serialNumber>
```

For example:

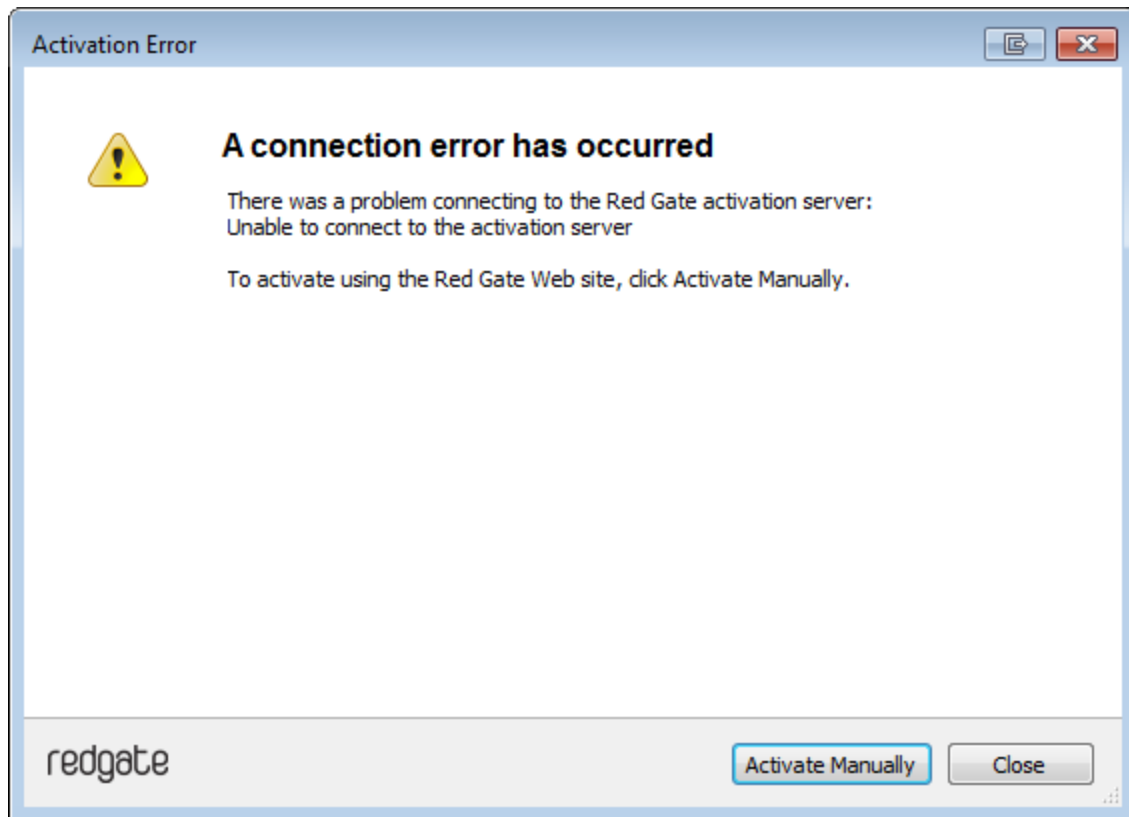
```
sqlcompare /activateSerial:123-456-789012-ABCD
```

The product activation dialog box is displayed. Follow the instructions below.

## Manual activation

Manual activation enables you to activate products when your computer does not have an internet connection or your internet connection does not allow SOAP requests. You will need access to another computer that does have an internet connection.

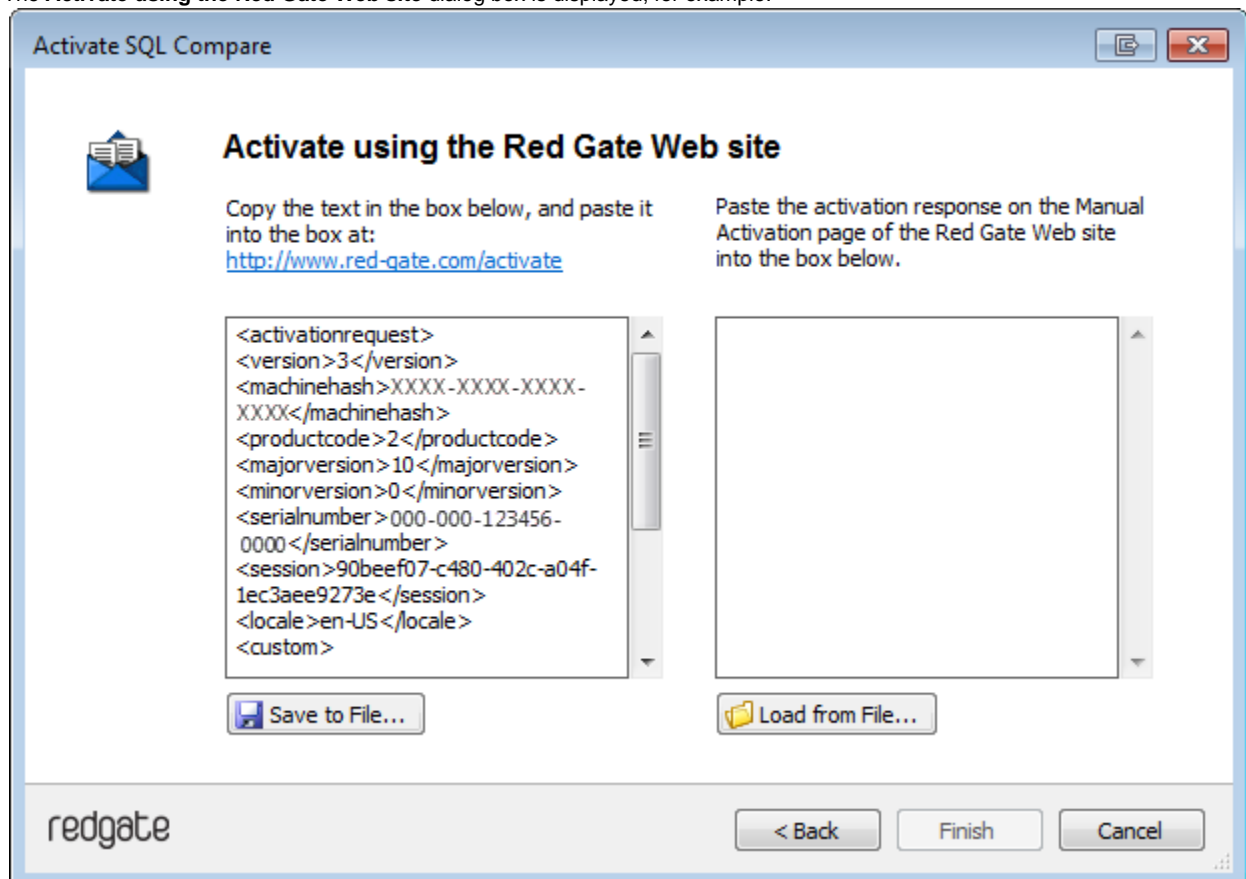
You can use manual activation whenever the **Activation Error** dialog box is displayed and the **Activate Manually** button is available, for example:



To activate manually:

1. On the error dialog box, click **Activate Manually**.

The **Activate using the Red Gate Web site** dialog box is displayed, for example:



2. Copy all of the activation request, and **leave this dialog box open** (if you close the dialog box, you may have to start again). Alternatively you can save the activation request, for example to a location on your network or to a USB device.
3. On a computer that has an Internet connection, go to the **Manual Activation** page at <http://www.red-gate.com/activate> and paste the activation request into the box under **Step 1**.

Account ▾ Quotes Shopping Cart

redgate  
ingeniously simple tools

Home Products Store Community Support Our Company

I'm looking for... 🔍

## Manual Activation

Use the activation request from the licensing program to generate an activation response so that you can activate products on your computer.

### Step 1

Paste the activation request into the box below. Make sure you paste all of the text.

```
<activationrequest>
<version>3</version>
<machinehash>XXXX-XXXX-XXXX-XXXX</machinehash>
<productcode>2</productcode>
<majorversion>10</majorversion>
<minorversion>0</minorversion>
<serialnumber>000-000-123456-0000</serialnumber>
<session>90beef07-c480-402c-a04f-1ec3aee9273e</session>
<locale>en-US</locale>
<custom>
```

Get Activation Response

### Step 2

Copy the contents of this box into your product activation dialog box.


Save to File...

### Got a question?

0800 169 7433  
shop@red-gate.com

4. Click **Get Activation Response**.
5. When the activation response is displayed under **Step 2**, copy all of it. Alternatively you can save the activation response to a .txt file.
6. On the computer where the licensing and activation program is running, paste the activation response or if you saved it, load it from the file.


Activate SQL Compare

 **Activate using the Red Gate Web site**


Copy the text in the box below, and paste it into the box at:  
<http://www.red-gate.com/activate>

Paste the activation response on the Manual Activation page of the Red Gate Web site into the box below.

```
<activationrequest>
<version>3</version>
<machinehash>XXXX-XXXX-XXXX-XXXX</machinehash>
<productcode>2</productcode>
<majorversion>10</majorversion>
<minorversion>0</minorversion>
<serialnumber>000-000-123456-0000</serialnumber>
<session>90beef07-c480-402c-a04f-1ec3aee9273e</session>
<locale>en-US</locale>
<custom>
```

 Save to File...

```
<activationresponse>
<data>
<machinehash>XXXX-XXXX-XXXX-XXXX</machinehash>
<version>3</version>
<productcode>2</productcode>
<majorversion>10</majorversion>
<minorversion>0</minorversion>
<edition>professional</edition>
<userspurchased>1</userspurchased>
<serialnumber>000-000-123456-0000</serialnumber>
```

 Load from File...

redgate

< Back Finish Cancel

7. Click **Finish**.  
The **Activation successful** page is displayed.
8. Click **Close**.  
You can now continue to use your product.

## Deactivating

This page applies to several Redgate products, so the screenshots below may not match your product.



Download deactivation tool

You can use the deactivation tool to deactivate a serial number so you can reuse it on another computer. You can also use it to deactivate serial numbers for products you've uninstalled.

When you deactivate a serial number for a bundle of products, all the products in the bundle are deactivated. For information about what products are in your bundle, see [Bundle history](#).

To deactivate a serial number, your computer must have an internet connection. If you can't deactivate a serial number, you can [request additional activations](#) for that serial number. You may need to do this if:

- your computer doesn't have an internet connection
- your network uses a proxy server that interrupts contact between the product and the Redgate activation server
- your serial numbers aren't displayed in the deactivation tool (eg if the product installation is corrupted)

### Deactivating using the command line

Open a command prompt, navigate to the folder where your product executable file is located and run a command with the following syntax:

```
<productEXE> /deactivateSerial
```

For example:

```
sqlcompare /deactivateSerial
```

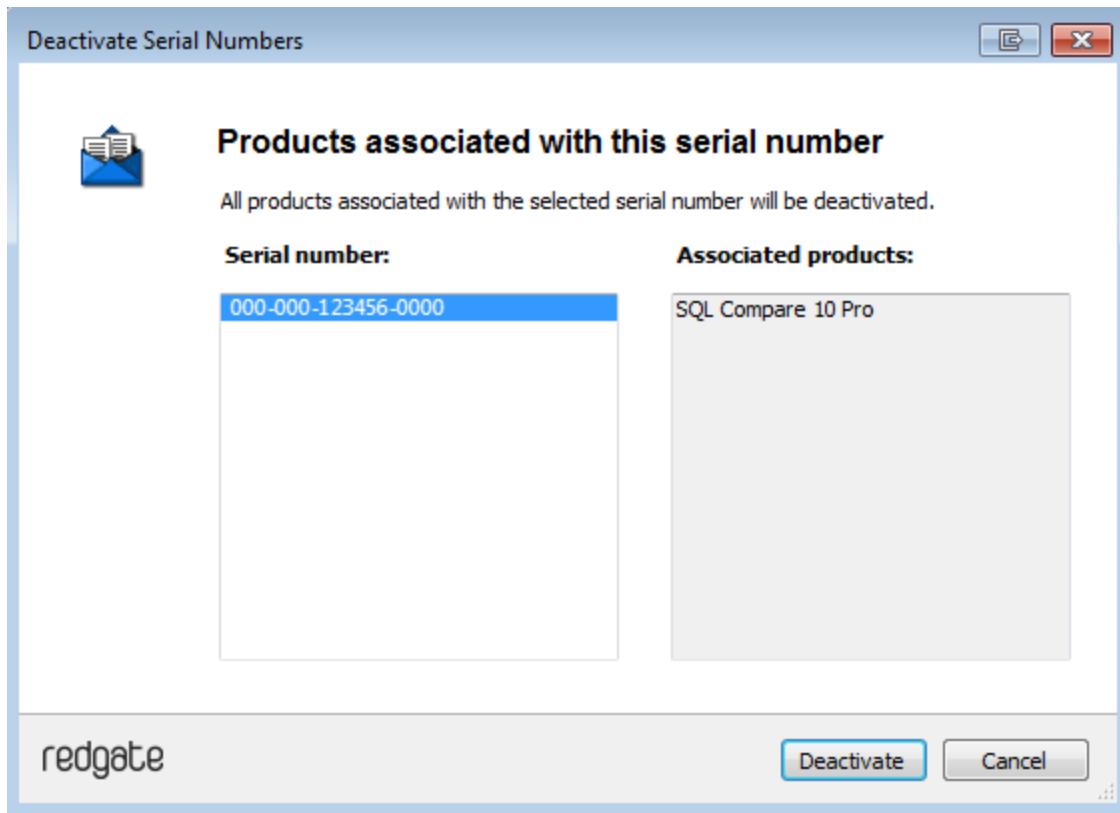
The **Deactivate Serial Numbers** dialog box is displayed. Follow the instructions below.

### Deactivating using the GUI

To deactivate your products:

1. Start the deactivation tool. To do this, either [download](#) the tool and run the executable file, or on the **Help** menu of the product, click **Deactivate Serial Number**.

The **Deactivate Serial Numbers** dialog box is displayed. For example:



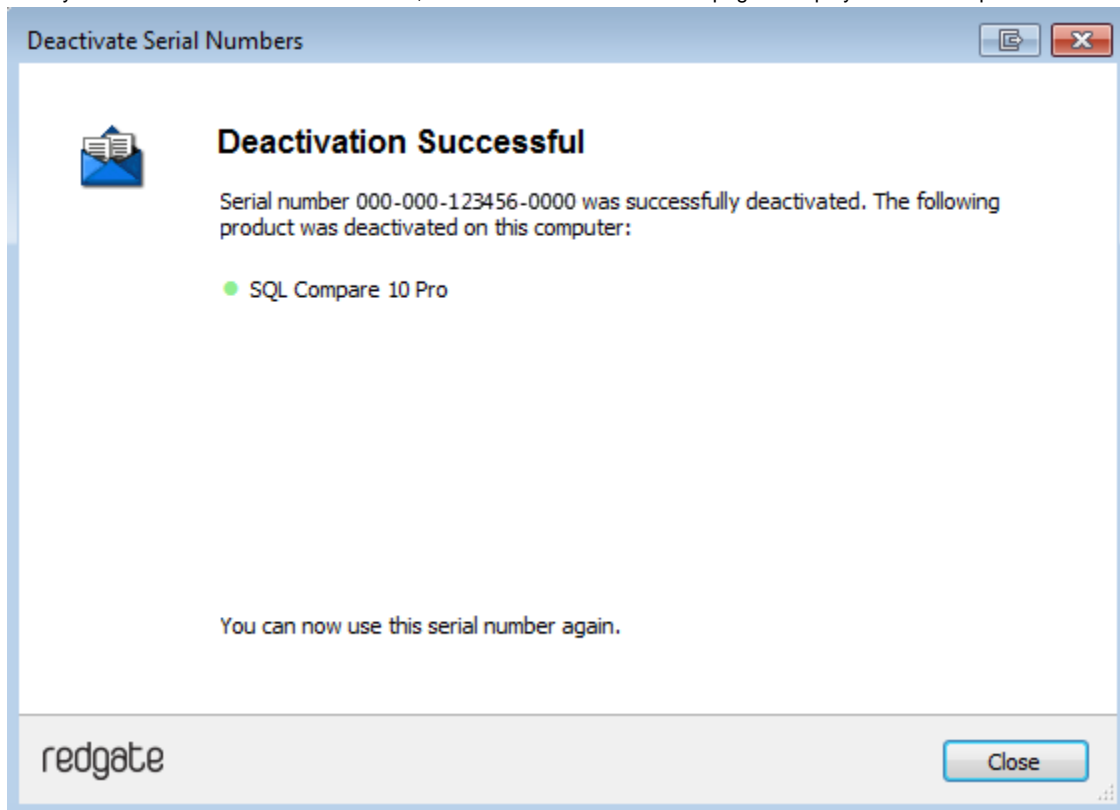
If you're running the executable file, the dialog box displays all the serial numbers for Red Gate products that have been activated on your computer.

If the serial number is for a bundle, all the products in the bundle are displayed under **Associated products**.

2. Select the serial number you want to deactivate and click **Deactivate**.

Your deactivation request is sent to the Red Gate activation server.

3. When your deactivation has been confirmed, the **Deactivation successful** page is displayed. For example:



If there's a problem with your deactivation request, an error dialog box is displayed. For information about deactivation errors and how to resolve them, see [Troubleshooting licensing and activation errors](#).

4. Click **Close**. You can now use this serial number on a different computer.

## Troubleshooting licensing and activation

This page provides information about errors you may encounter when you activate Redgate products:

- The number of activations for this serial number has been exceeded
- This serial number has been disabled
- This serial number was for a trial extension
- This serial number is not registered with the activation server
- This serial number is not for <product name>
- This serial number is not for this version
- The activation request is in the wrong format
- The activation request contains an invalid machine hash
- The activation request contains an invalid session
- The activation request contains an invalid serial number
- The activation request contains an invalid product code or version number
- There's a problem deactivating your serial number
- This serial number is not activated on this computer
- Products not activated on this computer

### The number of activations for this serial number has been exceeded

This error message is displayed when a serial number is activated on more computers than the number of licenses that were purchased for that serial number.

When you purchase products from Redgate, we send you an invoice that includes your serial numbers. The serial numbers enable you to activate the software a number of times, depending on how many licenses you purchased and the terms in the [license agreement](#). When this limit is reached, you will see this error message.

To fix the problem, you can:

- [deactivate](#) the product on another computer to free up a license
- [purchase](#) more licenses
- [request additional activations](#) for your serial number

### This serial number has been disabled

This error message is displayed when you try to activate a product using a serial number that Redgate has disabled.

When you upgrade a product, your existing serial numbers will be disabled and we will issue new ones with your invoice. If you cannot find your new serial numbers, you can review them at <http://www.red-gate.com/myserialnumbers>

Redgate will also disable serial numbers for non-payment of invoices or breach of the terms in the [license agreement](#). If you think we have disabled your serial numbers in error, email [licensing@red-gate.com](mailto:licensing@red-gate.com)

### This serial number was for a trial extension

This error message is displayed when you have requested a trial extension and you try to reuse the serial number that was provided for the trial extension; trial extensions can be used one time only.

To continue using the product, you need to [purchase](#) it.

### This serial number is not registered with the activation server

This error message is displayed when the serial number you entered does not exist on the Redgate activation server.

To find out your serial numbers, check your invoice or go to <http://www.red-gate.com/myserialnumbers>

### This serial number is not for <product name>

This error message is displayed when the serial number you entered is not for the product you are trying to activate.

To find out your serial numbers, check your invoice or go to <http://www.red-gate.com/myserialnumbers>



## This serial number is not for this version

This error message is displayed when the serial number you entered is for a different version of the product you are trying to activate.

If the serial number is for an older version of the product, and you don't have that version installed on your computer, you can download it from the [Release notes and other versions page](#).

If you want to upgrade to the latest version of the product, go to the [Upgrade center](#) to get a quote or purchase an upgrade, or email [sales@red-gate.com](mailto:sales@red-gate.com).

## The activation request is in the wrong format

This error message is displayed:

- if your internet connection does not allow SOAP requests.  
Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed.
- if you are activating by email and there is a problem with the format of the activation request.  
Check that you copied and pasted all of the activation request.  
Alternatively, try using manual activation. Go to <http://www.red-gate.com/activate> and paste your activation request under **Step 1**.
- when you are using manual activation and there is a problem with the format of the activation request. If the format is incorrect, for example part of the request is missing, the Redgate activation server cannot process the request.  
Check that you copied and pasted all of the activation request.

For more information about activating manually, see [Manual activation](#).

## The activation request contains an invalid machine hash

This error message is displayed:

- if your internet connection does not allow SOAP requests.  
Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see [Manual activation](#).
- when you are using manual activation and there is a problem with the format of the *machinehash* element in the activation request. The *machinehash* is a checksum of attributes from your computer. We use the *machinehash* to identify computers on which our products have been activated. If the format of the *machinehash* element is incorrect, the Redgate activation server cannot process the request.  
Check that you copied and pasted the activation request correctly.

## The activation request contains an invalid session

This error message is displayed:

- if your internet connection does not allow SOAP requests.  
Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see [Manual activation](#).
- when you are using manual activation and there is a problem with the format of the activation request. If the format of the *session* element is incorrect, the Redgate activation server cannot process the request.  
Check that you copied and pasted the activation request correctly.

## The activation request contains an invalid serial number

This error message is displayed:

- if your internet connection does not allow SOAP requests.  
Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see [Manual activation](#).
- when you are using manual activation and there is a problem with the format of the activation request. If the format of the serial number is incorrect, the Redgate activation server cannot process the request.  
Check that you copied and pasted the activation request correctly.

## The activation request contains an invalid product code or version number

This error message is displayed:

- if your internet connection does not allow SOAP requests.  
Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see [Manual activation](#).
- when you are using manual activation and there is a problem with the format of the activation request. If the product code or version numbers are incorrect, the Redgate activation server cannot process the request.  
Check that you copied and pasted the activation request correctly.

## There's a problem deactivating your serial number

This error message is displayed if your computer is not connected to the internet or your internet connection does not allow SOAP requests. You cannot deactivate a serial number if your computer does not have an internet connection.

Try deactivating again later. If the problem persists, contact your system administrator.

If you require more activations because you cannot deactivate your serial number, you can request them on the [Request Extra Activations](#) page.

## This serial number is not activated on this computer

This error message is displayed when you try to deactivate a serial number that has not been activated on your computer.

If you think the product installation on your computer is corrupt, you can try re-activating the product, and then deactivating the product again.

If you require more activations because you cannot deactivate your serial number, you can request them on the [Request Extra Activations](#) page.

## Products not activated on this computer

This error message is displayed when you try to deactivate a serial number for a bundle of Redgate products and those products were not activated on your computer.

If you think the product installation on your computer is corrupt, you can try re-activating the product, and then deactivating the product again.

If you require more activations because you cannot deactivate your serial number, you can request them on the [Request Extra Activations](#) page.

# Upgrading

**Minor releases** are free for all users. For example, if you have a license for version 7.0 of a product, you can upgrade to version 7.1 at no cost. When you download and install a minor release, the product is licensed with your existing serial number automatically.

**Major releases** are free for users with a current Support and Upgrades contract. For example, if you have a license for version 7 of a product, you can upgrade to version 8 at no cost. When you download and install a major release, the product is licensed with your existing serial number automatically.

If you don't have a current Support and Upgrades contract, installing a major release will start a free 14-day trial. You'll need to buy a new license and activate the product with your new serial number.

To check whether you have a current Support and Upgrades contract or see the cost of upgrading to the latest major version of a product:

- visit the [Upgrade Center](#)
- email [sales@red-gate.com](mailto:sales@red-gate.com)
- call:
  - 1 866 733 4283 (toll free USA and Canada)
  - 0800 169 7433 (UK freephone)
  - +44 (0)870 160 0037 (rest of world)

To check the latest version of a product, see [Current versions](#).

## How to upgrade

You can download the latest version of a product using [Check for Updates](#), the [Upgrade Center](#), or the [Redgate website](#).

- If you download the latest version from the Upgrade Center or our website, you need to run the installer to upgrade the product.

Some Redgate products are available as part of bundle. You can select which products you want to upgrade when you run the installer.

- If you use Check for Updates, the installer runs automatically.

You can install the latest *major* version of any product (other than SQL Backup Pro) on the same machine as the previous version. For example, you can run version 9 and version 10 in parallel. However, installing a *minor* release will upgrade the existing installation.

To revert to an earlier version, uninstall the later version, then download and install the version you want from the Release notes and other versions page. You can use a serial number for a later version to activate an earlier version.

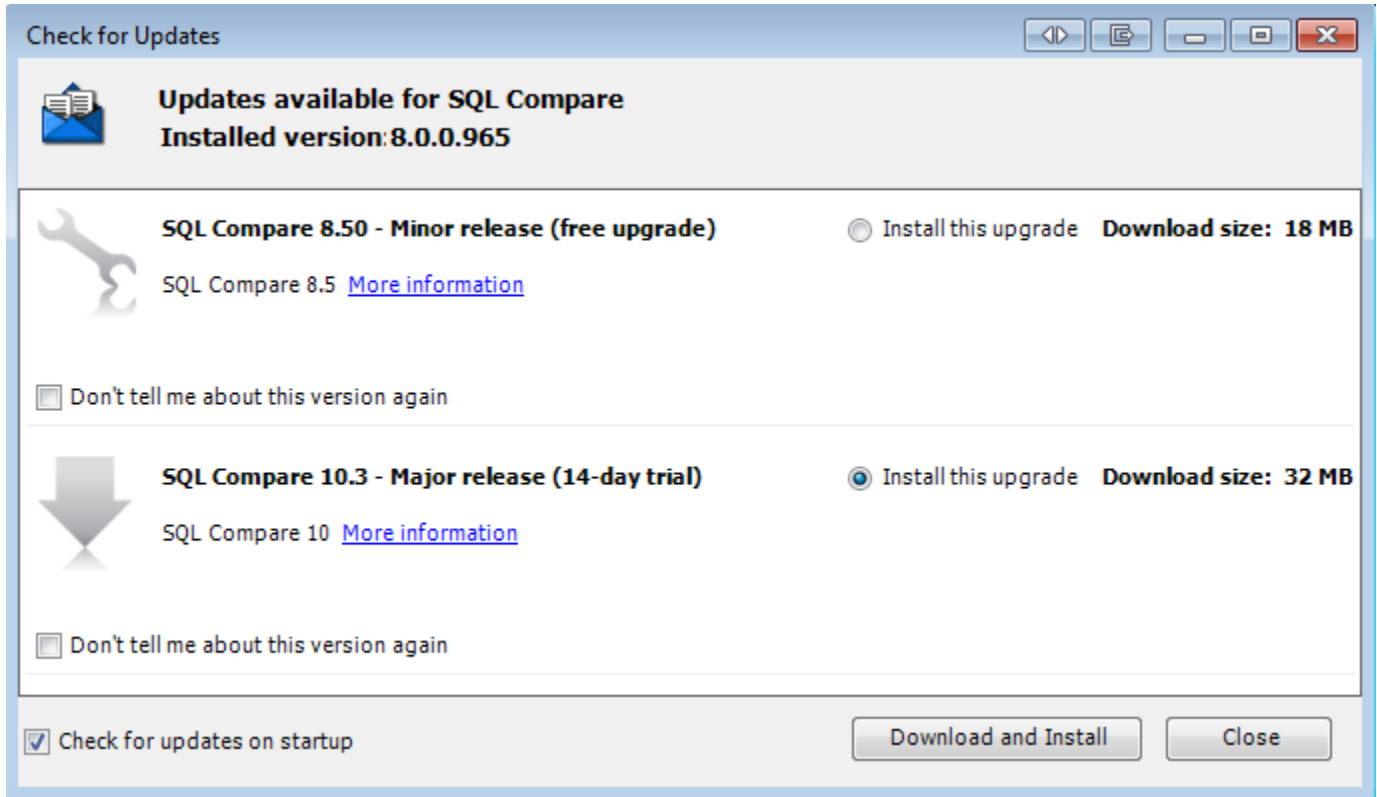
## Using Check for Updates

This page applies to several Redgate products, so the screenshots below may not match your product.

The Check for Updates service checks whether a more recent version of the product is available to download. To use the service, your computer must have a connection to the internet. If your internet connection uses a proxy server, make sure your web browser connection settings are configured correctly.

The Check for Updates service doesn't work with automatic configuration scripts.

To check for updates for a Redgate product, on the **Help** menu, click **Check for Updates**. Any available updates are listed:



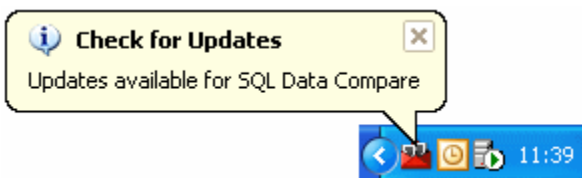
To view the full release details in your default web browser, click **More information**.

To get the update, click **Download and Install**. If you have a choice of updates, choose by selecting **Install this upgrade**, and then click **Download and Install**.

The installer will ask you to close the program. If you're upgrading an add-in, you'll also be asked to close the host program (SQL Server Management Studio, Visual Studio or Query Analyzer).

## About the Check for Updates service

When you start the application, the Check for Updates service informs you automatically when there are updates available:



If you don't want to receive these notifications for the product, clear the **Check for updates on startup** check box.

If you don't want the Check for Updates service to inform you about a particular update again, select the **Don't tell me about this version again** check box. The Check for Updates service will still inform you of new updates when they become available.

## Troubleshooting Check for Updates errors

For details about how to use the Check for Updates service, see [Using Check for Updates](#).

### Error: There is a problem saving the download file to your computer

This error message is displayed if:

#### You don't have enough disk space

The Check for Updates service downloads the updates to the location defined by the *RGTEMP* environment variable, or the *TMP* variable if the *RGTEMP* variable doesn't exist.

If you don't have enough disk space, you can change the environment variable to a location with more space.

Changing the *RGTEMP* or the *TMP* variables will affect other programs that use those variables. The *RGTEMP* variable affects only Redgate programs. For information about environment variables, see your Windows documentation.

#### There's a problem with permissions on your computer

The Check for Updates service downloads the updates to the location defined by the *RGTEMP* environment variable, or the *TMP* variable if the *RGTEMP* variable does not exist. If your user account doesn't have permissions to write to the location specified by these environment variables, contact your system administrator.

#### There's a problem with the download file on the Redgate web server

Contact [Redgate support](#).

### Error: There is a problem with the network connection

This error message is displayed if:

#### Your internet connection dropped while the Check for Updates service was downloading the updates

Try checking for updates again later.

#### Proxy authentication failed

Check your user name and password.

#### Your computer can't connect to the Check for Updates service.

Contact your system administrator. If you're using a proxy server, check it's configured correctly (see Control Panel > Internet Options > Connections).

The Check for Updates service doesn't work with automatic configuration scripts.

#### There's a problem with the download file on the Redgate web server

Contact [Redgate support](#).

## Check for updates may fail when used through proxies

The Red Gate Check for Updates service, available through the Help menu of most Red Gate products, may fail to connect to the Red Gate update service if internet access goes through a proxy server that requires authentication.

The Check for Updates service uses SOAP to communicate with a web service hosted at [update.red-gate.com](http://update.red-gate.com) on TCP port 80. The service supports connections through an HTTP proxy as long as the proxy server integrates with Internet Explorer and has been properly configured in **Internet Options > Connections** (available from Internet Explorer or Windows Control Panel). In other words, Check for Updates uses Windows WinInet API to send and receive requests for updates.

If the proxy server doesn't integrate with Windows and doesn't support transparent NTLM authentication, Check for Updates will prompt you for a proxy username and password and try to authenticate using Basic Authentication. Finally, the request will fail with the message "could not connect to the update service".

Even when WinInet is properly configured for your proxy server, there are two known circumstances that may cause Check for Updates to fail:

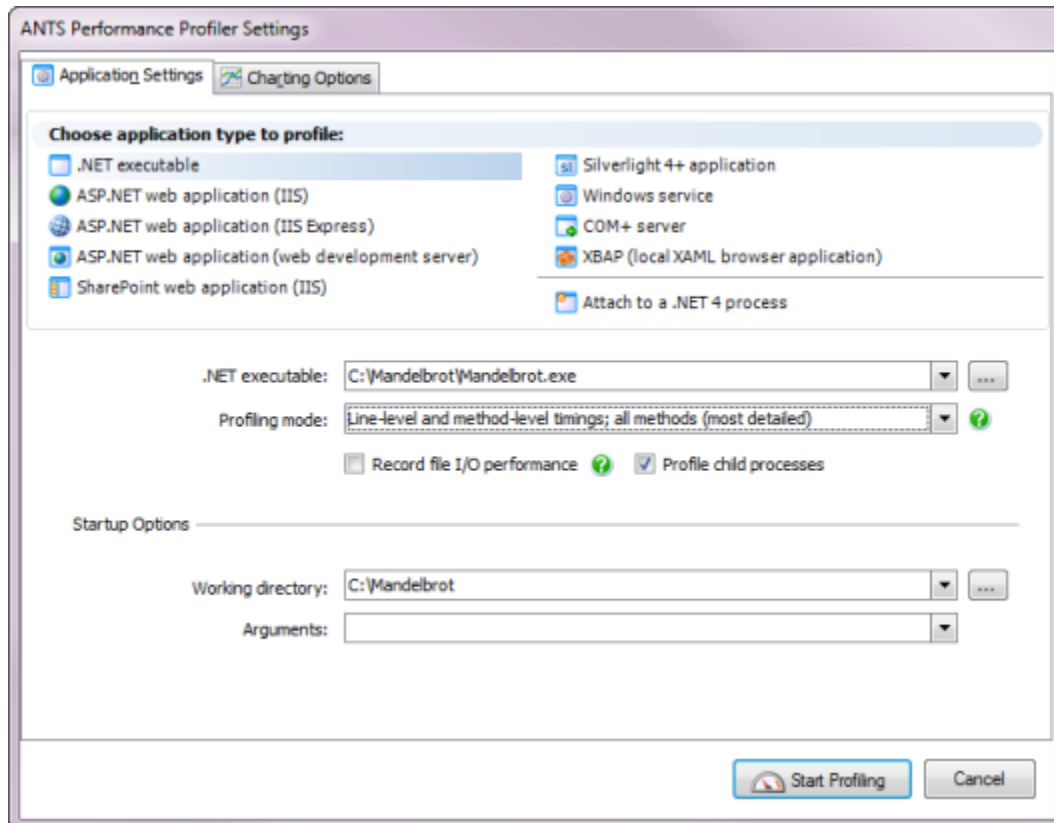
- An automatic configuration script is used to configure the Internet Explorer proxy connection. In this case, the connection properties can be configured differently for every connection. This can cause Check for Updates to behave erratically, sometimes being able to connect and sometimes not.
- A bug in some versions of Check for Updates whereby any proxy server that doesn't have "realm" configured will cause Check for Updates to crash. Contact your proxy administrator to make sure a realm is configured in the authentication settings.

# Setting up and running a profiling session

To profile an application, you must first set up a profiling session. A session specifies:

- The application type, location, and options for the application you want to profile.
- The profiling mode, which determines the level of detail gathered by the profiler while your application is running.
- The method used to calculate timing values (CPU time or wall-clock time).
- The performance counters to display on the timeline.

When you start ANTS Performance Profiler, the **ANTS Performance Profiler Settings** dialog box is automatically displayed; if ANTS Performance Profiler is already running, click **New Profiling Session** on the **File** menu.

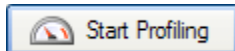


The **Application Settings** tab displays the settings for the last profiling session you ran. The settings available depend on the selected application type, and may differ from those illustrated above.

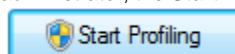
The **Charting Options** tab enables you to choose which performance counter values to display on the timeline for the new profiling session.

## To set up and run a profiling session:

1. On the **ANTS Performance Profiler Settings** dialog box, complete the details on the [Application Settings](#) tab.
2. Choose which performance counters to monitor during profiling using the [Charting Options](#) tab.
3. Click



On Windows Vista, Windows Server 2008, and Windows 7, if you are not running ANTS Performance Profiler as an Elevated administrator, the **Start Profiling** button has a User Account Control (UAC) shield:




The UAC shield indicates that ANTS Performance Profiler will request elevation when you start profiling.

The timeline is displayed at the top of the main ANTS Performance Profiler window, and the application you want to profile is automatically started. Status text at the bottom-left of the main window indicates what ANTS Performance Profiler is doing during the profiling session.

The timeline starts displaying performance-counter data and events in near-real time. There may be a slight delay between starting a profiling session and seeing the first performance-counter data appear on the timeline.

4. To display profiling results, do one of the following:



- Drag a region on the timeline.  
Profiling data is summarized and displayed for the selected time period only. Your application will continue running and profiling will continue.
- Click   
**Stop Profiling.**  
Your application is closed. Profiling data is summarized and displayed for the entire profiling period.
- Close your application.  
Profiling data is summarized and displayed for the entire profiling period.

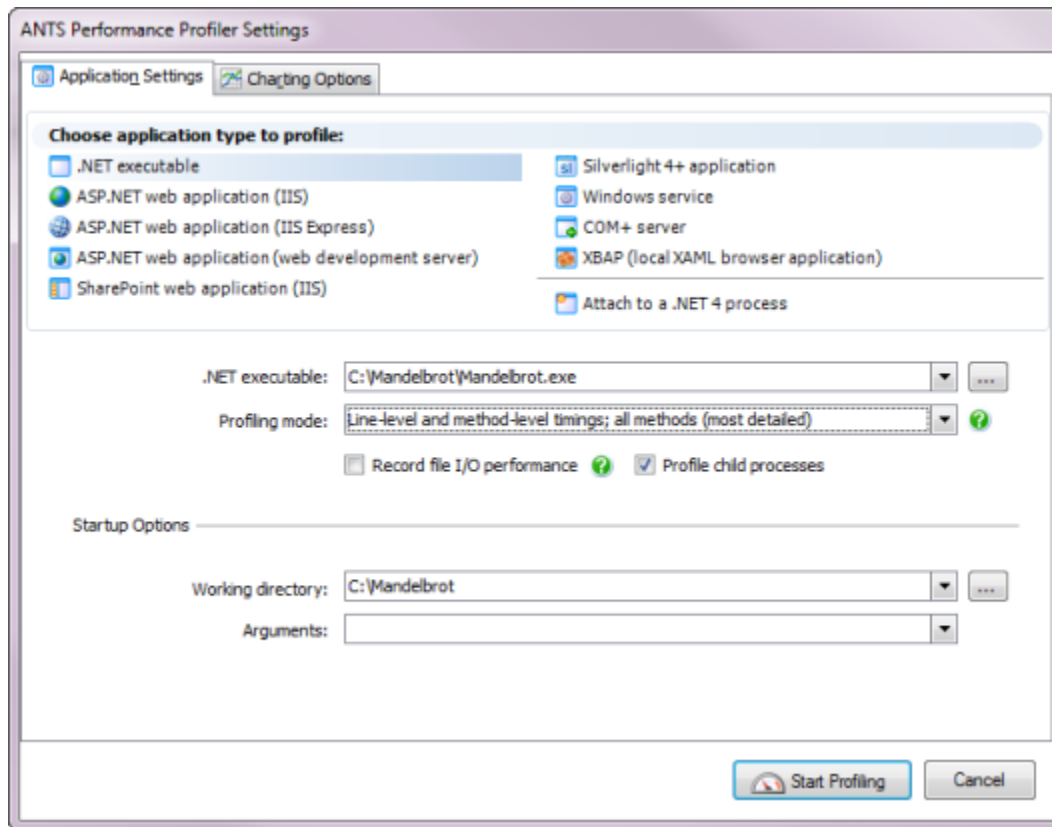
You can continue [working with the timeline](#) to locate periods of interest during the execution of your application, and to display the associated profiling results.

Once you have displayed some profiling data, you can view and analyze it. For more information about the different ways you can do this, see [Working with profiling results](#).

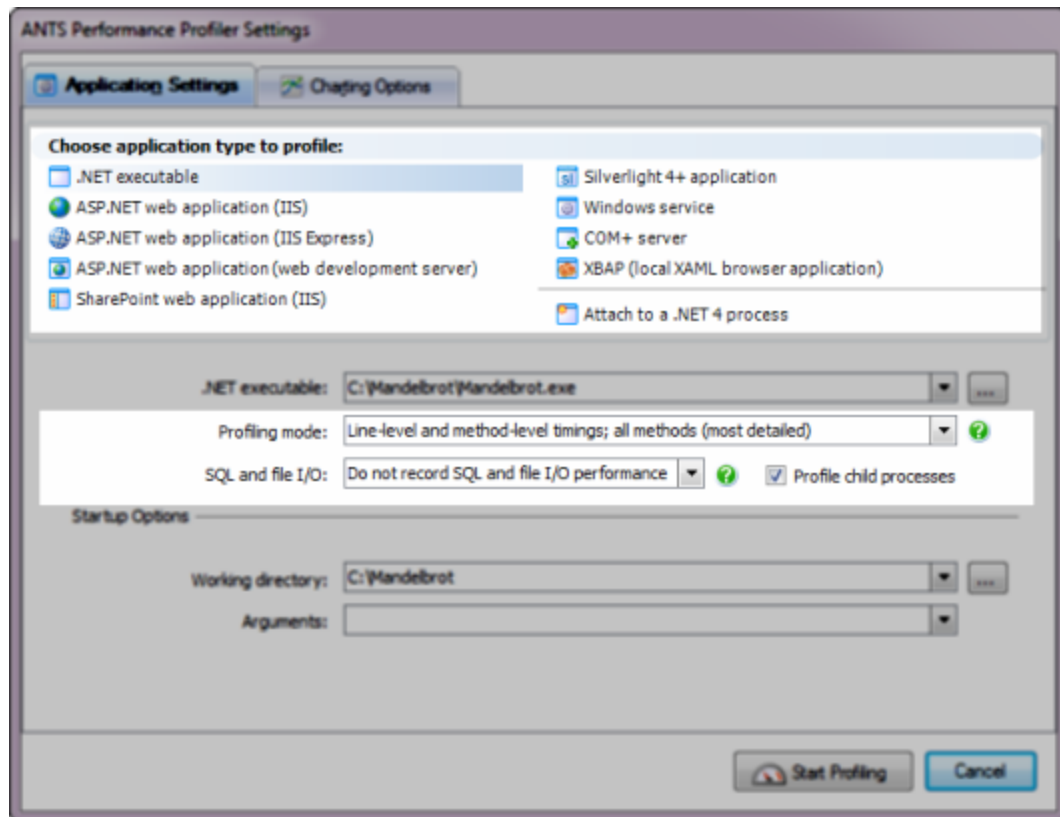
## Working with Application Settings

When you set up an application, you specify which application type you want to profile.

This page explains which application type to choose. It also explains settings that are common to all application types: **Profiling mode**, **File I/O (SQL and file I/O in ANTS Performance Profiler 7.0)** and **Profile child processes**.



✓ Screenshot for ANTS Performance Profiler 7.0...



Other settings are specific to individual application types; to continue setting up your application, follow the appropriate link for your application type.

## Application types

Select the application type from the **Choose application type to profile** list.

- **.NET executable**  
 Select this option to profile .NET executables, managed code add-ins, or remotely-hosted XBAP applications.  
 For additional settings, see [Profiling .NET executables](#), [Profiling managed code add-ins](#), or [Profiling XBAP applications](#).
- **ASP.NET web application (IIS)**  
 Select this option to profile ASP.NET web applications running on IIS, WCF services running on IIS, or SharePoint.  
 For additional settings, see [Profiling ASP.NET applications running on IIS](#), [Profiling WCF services running on IIS](#), or [Profiling SharePoint](#).  
 Web applications are profiled in Microsoft Internet Explorer, even if it is not your preferred browser. This is because ANTS Performance Profiler uses the low-level data exposed by Internet Explorer.  
 Note that you must profile ASP.NET web applications on the same computer as the one on which IIS is running.
- **ASP.NET web application (IISE)**  
 Select this option to profile ASP.NET web applications running on IIS Express.  
 For additional settings, see [Profiling ASP.NET applications running on IIS Express](#).  
 Web applications are profiled in Microsoft Internet Explorer, even if it is not your preferred browser. This is because ANTS Performance Profiler uses the low-level data exposed by Internet Explorer.  
 Note that you must profile ASP.NET web applications on the same computer as the one on which IIS is running.
- **ASP.NET web application (web development server)**  
 Select this option to profile ASP.NET web applications running on the web development server (WebDev, also known as Cassini).  
 For additional settings, see [Profiling ASP.NET applications running on the web development server](#).  
 Web applications are profiled in Microsoft Internet Explorer, even if it is not your preferred browser. This is because ANTS Performance Profiler uses the low-level data exposed by Internet Explorer.
- **SharePoint web application (IIS)**  
 Select this option to profile SharePoint sites and collections running in IIS.  
 For additional settings, see [Profiling SharePoint](#).
- **Silverlight 4+ application**  
 Select this option to profile Silverlight 4 and 5 applications.  
 For additional settings, see [Profiling Silverlight 4+ applications](#).  
 This feature requires the Silverlight 4 plug-in to be installed in Internet Explorer.
- **Windows service**  
 Select this option to profile Windows services, or WCF services.  
 For additional settings, see [Profiling Windows services](#).
- **COM+ server**

Select this option to profile a COM+ server application.  
For additional settings, see [Profiling COM+ server applications](#).

- **XBAP (XAML Browser Application)**

Use this option to profile a locally-hosted XBAP application.  
For additional settings, see [Profiling XBAP applications](#).

- **Attach to .NET 4 process**

Choose the .NET process you want to attach to.

This feature requires Windows Vista or later and .NET 4.

When you attach to a .NET 4 process, the Sample method-level timings profiling mode is used, SQL and file I/O performance profiling is enabled, and Profile child processes is disabled.

## Profiling mode

The **Profiling mode** determines the level of detail gathered by the profiler while your application is running. The level of detail that you choose may affect the profiling speed and the overall accuracy of the results.

Profiling Mode	Speed	Accuracy	Detail	SQL / HTTP data (ANTS Performance Profiler 7.2 and later)	Profiling Data
<b>Line-level and method-level timings; all methods*</b>	*	**	****	Yes	All methods. This includes methods without source code, such as those in the .NET Framework class libraries.
<b>Method-level timings; all methods</b>	***	***	**	Yes	
<b>Line-level and method-level timings; only methods with source*</b>	**	*	***	Yes	Only methods for which source code is available, for example, timings will not be measured for .NET Framework methods.
<b>Method-level timings; only methods with source</b>	****	****	*	Yes	
<b>Sample method-level timings</b>	**** *	*	*	No	

\*Profiling data is also collected for individual lines of code.

When you attach to a .NET 4 process, **Sample method-level timings** are always used.

## Record File I/O (ANTS Performance Profiler 7.2 and later)

File I/O profiling is only available in ANTS Performance Profiler Professional edition.

If you have Windows Vista or later, you can record file I/O operations. For more information, see [Profiling File IO](#).

To enable I/O profiling, select **Record file I/O performance**. Note that enabling this option will make profiling slower.

When you attach to a .NET 4 process, **Record file I/O performance** is always enabled.

## Profile SQL calls (ANTS Performance Profiler 7.2 and later)

SQL call profiling is only available in ANTS Performance Profiler Professional edition.

In ANTS Performance Profiler Professional edition, performance data for SQL calls is automatically recorded in all modes except sampling. SQL call profiling works with all versions of SQL Server including SQL Server Compact, and with Oracle databases, and including calls to databases hosted in the cloud on SQL Services (SQL Azure) and Amazon RDS.

## SQL and File I/O (ANTS Performance Profiler 7.0)

SQL and file I/O profiling are only available in ANTS Performance Profiler Professional edition.

---

If you have Windows Vista or later, you can record file I/O operations. For more information, see [Profiling File IO](#).

If you have Windows Vista or later and a SQL server (except Express editions) installed on the local computer, you can also record SQL queries. For more information, see [Profiling SQL queries](#).

To enable SQL and/or file I/O profiling, select **Record SQL and file I/O performance**. Note that enabling this option will make profiling slower.

When you attach to a .NET 4 process, **Record SQL and file I/O performance** is always enabled.

## Profile child processes

To simultaneously record performance information from your application and all other processes started by your application, select **Profile child processes**.

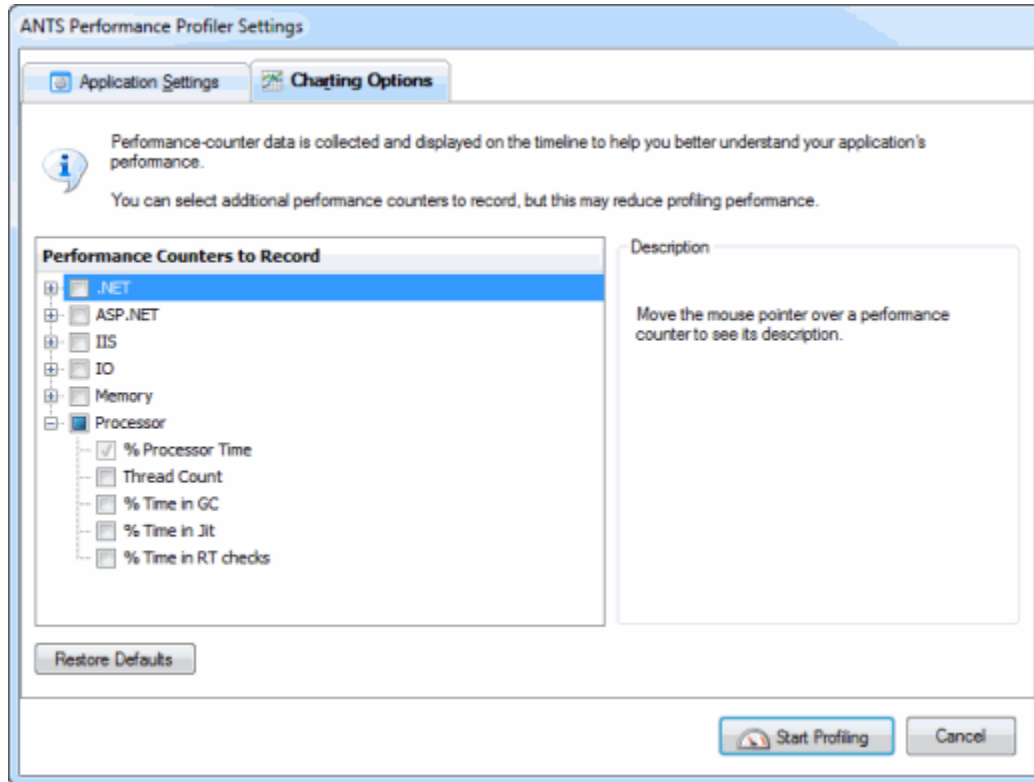
When you attach to a .NET 4 process, **Profile child processes** is disabled.

## Setting up Charting Options

ANTS Performance Profiler can monitor the values of a number of built-in Windows performance counters while the application you are profiling is executing. The values of these counters are constantly updated on the timeline as profiling proceeds.

You choose the performance counters you want to monitor using the **Charting Options** tab on the **ANTS Performance Profiler Settings** dialog box.

Note that this feature requires ANTS Performance Profiler Professional.



Not all performance counters are appropriate to all application types that you may be profiling. You can find more information about individual performance counters under the **Description** group box, including details about a counter's relevance to particular application types.

Your choice will depend on your own requirements but, as an example, you might choose the following:

From the **.NET** group:

- The *Gen 0 Promoted Bytes/sec* counter  
This will give the rate at which the garbage collector promotes objects from Generation 0 to Generation 1.

From the **Memory** group:

- The *Working Set* counter  
This shows the total amount of physical memory used by your service (including memory used by shared DLLs and the .NET runtime itself)

From the **Processor** group

- The *% Processor Time* counter (selected by default)  
This shows the percentage of time which all running threads use on the CPU.
- The *% Time in GC* counter  
This shows the percentage of time which the process was suspended to allow the last garbage collection to take place.

We recommend you avoid adding more performance counters than you need, as each additional counter that is recorded adds to the overhead introduced by the profiler. Adding too many counters may slow your application substantially.

### Adding custom performance counters

You can add custom performance counters to the list of available counters in the **Charting Options** tab. To do this:

1. Close ANTS Performance Profiler.

2. Expose your performance counter to the Windows Performance Counter API using the *PerformanceCounter* and *PerformanceCounterCategory* classes of the *System.Diagnostics* namespace. (An example describing how to do this is given at <http://msdn.microsoft.com/en-us/library/system.diagnostics.performancecounter.aspx>)
3. Create a new XML file as follows:

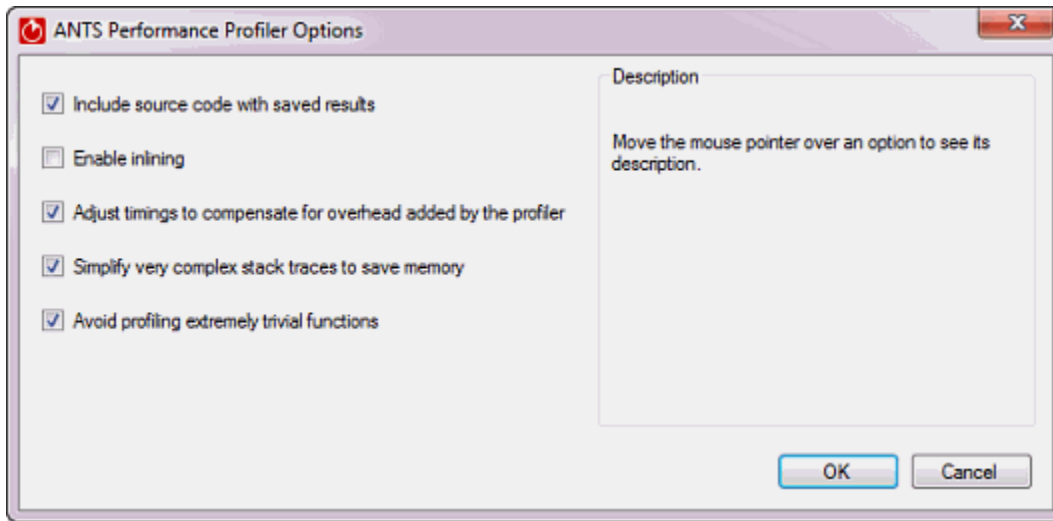
```
<Counters>
  <Category Name="CategoryName">
    <Counter Category="CategoryName" Name="CounterName" Units="Measurement
Units">
      <Instanced />
    </Counter>
  </Category>
</Counters>
```

Ensure that *CategoryName* and *CounterName* are the same as the names used for the *PerformanceCounterCategory* and *PerformanceCounter*. Remove the `<Instanced />` node if your counter collects data about the computer, not only the individual process. You can add multiple categories and counters in the same XML file.

4. Save the XML file as *UserCounters.xml* in `%LOCALAPPDATA%\Red Gate\ANTS Performance Profiler 7\`.
5. Restart ANTS Performance Profiler.
6. The counters that you defined are shown in the list on the **Charting Options** tab.

## ANTS Performance Profiler Options

ANTS Performance Profiler includes a number of options that are applied to all profiling sessions. To access these options, on the **Tools** menu, click **Options**.



Unless you have a particular need to adjust the options, leave them at their default settings. Changing the default setting for certain options may cause problems during profiling.

[Include source code with saved results](#)

[Enable inlining](#)

[Adjust timings to compensate for overhead added by the profiler](#)

[Simplify very complex stack traces to save memory](#)

[Avoid profiling extremely trivial functions](#)

### Include source code with saved results

Includes the contents of source files when you save profiling results. This means that you can review line-level performance data in saved results, without having to restore your source files to their original state.

Note that code generated using integrated decompilation is not saved with results even if this option is selected. For more information, see [Working with integrated decompilation](#).

You may want to clear this option if, for example, you need to distribute performance profiling results for an application that has confidential source code.

By default, this option is selected.

### Enable inlining

Enables inlining of methods by the .NET JIT compiler, for the process being profiled.

If you are profiling the release build of an application, selecting this option will produce a profile that is closer to the "real-world" performance. However, the accuracy of the results will be reduced. In particular, line-level timings will be distorted, hit counts will not be recorded for inlined methods, and time spent in inlined methods will be reported as part of the calling method.

By default, this option is not selected.

### Adjust timings to compensate for overhead added by the profiler

Adjusts timings by estimating the influence the profiler has had on the process being profiled, and subtracts this from the profiling results. This estimate is most accurate when you use a profiling mode that does not collect line-level timings.

The design of modern processors means that this estimate may not always be accurate, especially for short function calls.



By default, this option is selected.

### **Simplify very complex stack traces to save memory**

Summarizes complex stack traces in profiling results. This conserves resources on the machine you are using for profiling. The stack traces that are summarized are unlikely to be important to your profiling results. However, if you wish to see these summarized results, you can clear this option.

Clearing this option can significantly increase the memory required by the profiler. Depending on the application you are profiling, the profiler may become unstable if you clear this option.

By default, this option is selected.

### **Avoid profiling extremely trivial functions**

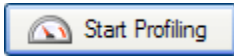
Prevents profiling of methods that have a running time measured in tens of nanoseconds, and which contribute to less than one-billionth of the run time in total. Typically, these methods do not produce very relevant performance data. Ignoring these methods reduces the amount of memory required to store and process profiling results.

By default, this option is selected.

## Profiling .NET executables

To profile .NET executables, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

1. Under **Choose application type to profile**, click **.NET executable**.
2. Browse to the **.NET executable** that you want to profile.  
Use the dropdown list to select a recently-profiled application.
3. Select the required **Profiling mode**, **file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).
4. If required, change the **Working directory**.  
The working directory is the path where the application will start. By default, this is the directory where the executable is located.  
Use the dropdown list to select a recently-used working directory.
5. If required, specify any command line **Arguments** that are to be used when running the application.
6. If required, change the performance counters to record; see [Setting up Charting Options](#).
7. Click



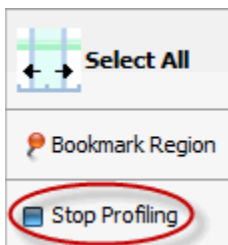
The .NET executable starts; interact with the application normally.

During a profiling session you can interact with the profiler while your application is still being profiled, and obtain results by selecting areas of the timeline.

When you have finished interacting with your application, click the



**Stop Profiling** button in ANTS Performance Profiler.



See also [Worked example: Profiling the performance of an algorithm](#).

## Profiling managed code add-ins

Microsoft Visual Studio, Microsoft Word, Microsoft Excel and Microsoft SQL Server Management Studio are all examples of desktop applications which host the .NET Common Language Runtime. You can create .NET add-ins for these programs.

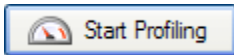
The procedure for using ANTS Performance Profiler to profile a native desktop application that hosts the .NET runtime is similar to that for profiling other .NET applications.

This page uses a SQL Server Management Studio add-in as an example. SQL Server Management Studio is a native application which hosts the CLR which, in turn, allows it to execute managed code. If you are profiling an add-in for a different program (Word or Excel, for example), you must use that program instead of SQL Server Management Studio in all of the following instructions.

### Setting up the performance profiler

To profile managed code add-ins, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

1. Under **Choose application type to profile**, click **.NET executable**.
2. Browse to the **.NET executable** that you want to profile.  
Note: this is the native application that contains your add-in; for example, SQL Server Management Studio.  
Use the dropdown list to select a recently-profiled application.
3. Select the required **Profiling mode**, **file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).
4. If required, change the **Working directory**.  
The working directory is the path where the application will start. By default, this is the directory where the native application is located.  
Use the dropdown list to select a recently-used working directory.
5. If required, specify any command line **Arguments** that are to be used when running the application.
6. If required, change the performance counters to record; see [Setting up Charting Options](#).
7. Click



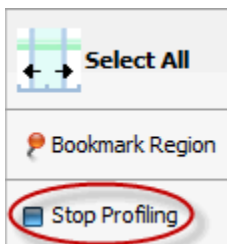
SQL Server Management Studio starts. Connect to a server and interact with your add-in.

During a profiling session you can interact with the profiler while your application is still being profiled, and obtain results by selecting areas of the timeline.

When you have finished interacting with your web application, click the



**Stop Profiling** button in ANTS Performance Profiler.



### Other applications

The same approach will work with all other hosting applications. For example, to profile a Visual Studio .NET add-in, enter the full path to *devenv.exe* as the .NET application that you want to profile.

You can also profile a managed code Microsoft Management Console (MMC) snap-in in the same way; select *mmc.exe* as the .NET executable, add the snap-in, then interact with it.

## Profiling ASP.NET applications running on IIS

To profile ASP.NET applications running on IIS, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

1. Under **Choose application type to profile**, click **ASP.NET web application (IIS)**.
2. Select the **ASP.NET web application (URL)** for the root directory of the web application that you want to profile.  
To load a list of currently-running sites from IIS into the dropdown list, click



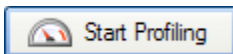
Note that the port specified in this URL is the port where the application usually runs under IIS, which is not necessarily the same as the port where the application is to be profiled (see step 4).

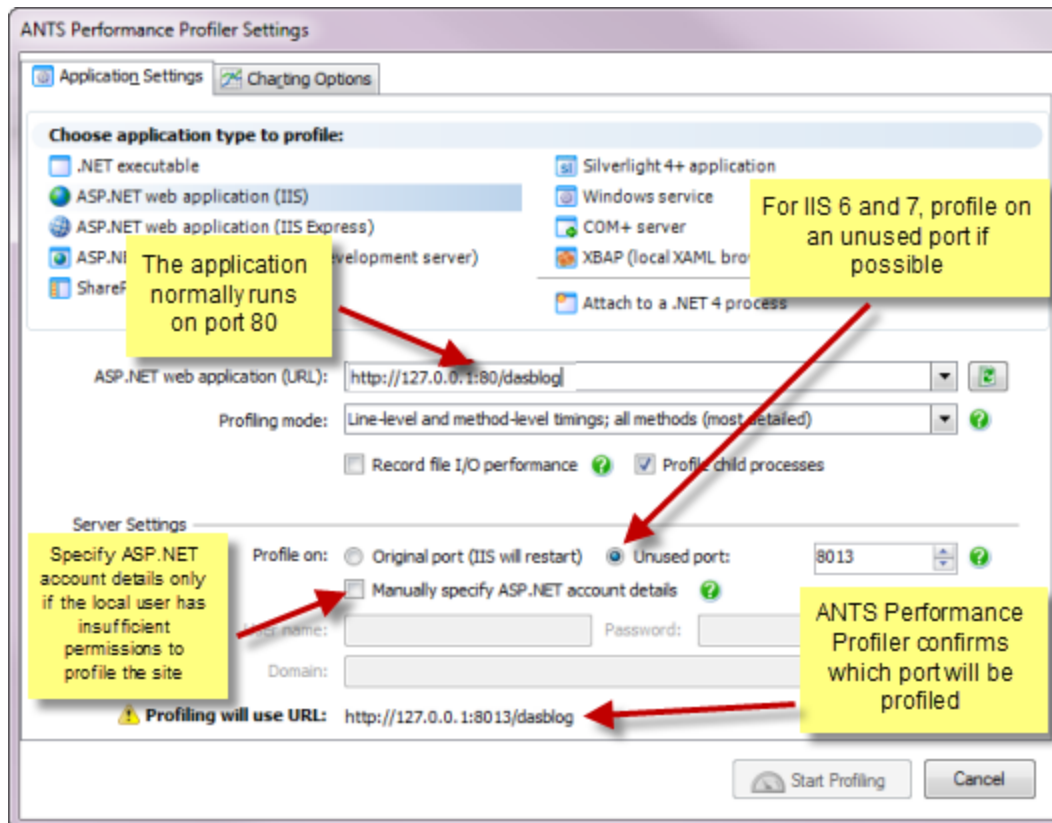
3. Select the required **Profiling mode**, **file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).  
In ANTS Performance Profiler 7.3 and later, SQL queries and incoming HTTP calls will be profiled automatically.
4. ANTS Performance Profiler will detect the version of IIS installed on your computer:
  - If you are using IIS 5, ANTS Performance Profiler will profile your application on its original port. This will automatically restart IIS.
  - If you are using IIS 6 or IIS 7, you can choose to profile on the original port or on an unused port:  
Select **Unused port** if your application does not bind to a specified port. ANTS Performance Profiler will start a new instance of your application on the specified port, avoiding a restart of IIS. The port where the application will be profiled is displayed at the bottom of the ANTS Performance Profiler Settings dialog box.  
Select **Original port** if your application's code binds to a specific port. ANTS Performance Profiler will stop your application, attach to it, and restart it on the original port.

Note that restarting IIS stops IIS and only restarts the application that you are profiling. Other sites running on the same IIS instance (including any other sites on which your web app depends) will not be present when IIS restarts.

If your application takes too long to start, IIS might not restart correctly. Use IIS Manager to stop the website manually until you have finished profiling.

5. If required, select **Manually specify ASP.NET account details** and enter the **User name**, **Password** and (in IIS6 and 7) **Domain**.  
With IIS 5, ANTS Performance Profiler can only run as the ASPNET user: ensure that this user has permission to read from *%ProgramFiles%\Red Gate\ANTS Performance Profiler 7\ProfilerCore.dll*  
If your Application Host configuration file is set to protect the credentials of the ASPNET user, ANTS Performance Profiler may be unable to detect them. In this case, select **Manually specify ASP.NET account details** and enter the ASPNET username and password.  
With IIS 6 or IIS 7, ANTS Performance Profiler profiles your web application as the Windows Local System user by default. This is appropriate for most websites.  
However, if your web application connects to a remote server (such as a database server), the Windows Local System user might not have appropriate permissions to make the remote connection. In this case, enter the credentials of a user who does have the required permissions. Note that the user you specify must be an administrator, and must have permission to read from *%ProgramFiles%\Red Gate\ANTS Performance Profiler 7\ProfilerCore.dll*
6. If required, change the performance counters to record; see [Setting up Charting Options](#).
7. Click





Note that port selection options are not displayed when profiling with IIS 5.

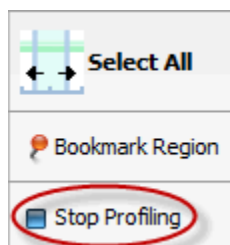
ANTS Performance Profiler launches the IIS user mode worker process (*w3wp.exe*), using a cut-down configuration file based on the site's *applicationHost.config* configuration file. Internet Explorer then starts and displays your web application.

During a profiling session you can interact with the profiler while your application is still being profiled, and obtain results by selecting areas of the timeline.

When you have finished interacting with your web application, click the



**Stop Profiling** button in ANTS Performance Profiler.



Note that you can only profile ASP.NET applications running on the computer where the profiler is installed. You cannot profile web applications that use SSL, because *w3wp.exe* does not support running SSL websites outside of IIS.

See also [Profiling an ASP.NET application \(worked example\)](#).

## Troubleshooting

If you encounter problems while trying to profile an application in IIS, see [Troubleshooting IIS profiling](#).

### Troubleshooting in ANTS Performance Profiler 7.0

In ANTS Performance Profiler 7.0, the continuous profiling tool runs as a separate tool from the main ANTS Performance Profiler product. If you installed an early access build of the ANTS Performance Profiler continuous profiler IIS module, other profilers - including the desktop ANTS Performance Profiler product - will be unable to profile applications running in IIS on this computer. To re-enable other profilers with IIS, uninstall

the IIS Profiler Module:

1. From your computer's Start menu, launch the **Continuous Profiling Configuration Tool**
2. Click **Uninstall**.

For more information on configuring continuous profiling, see [Setting up continuous profiling](#).

## Profiling WCF services running on IIS

The procedure for profiling Windows Communication Foundation (WCF) services running in IIS is similar to the procedure used to profile other types of web application in IIS. It may help to think of the service as a server in a server-client relationship.

Please note the following:

- Before you start, change the WCF client contract to communicate on the unused port that you select in ANTS Performance Profiler (by default, 8013).  
Changing the port is necessary because otherwise the client will communicate with the copy of the server hosted in IIS, not the copy in the worker process started by ANTS Performance Profiler.
- Set the **ASP.NET web application (URL)** to the path to the web application on the server.
- When you start profiling, Internet Explorer will launch.  
Minimize this window and interact with your client application instead. Do not close the Internet Explorer window during profiling; this will stop ANTS Performance Profiler from collecting the performance data.

## Profiling ASP.NET applications running on the web development server

The ASP.NET web development server (also known as 'WebDev' or 'Cassini') is the built-in web server for your development environment. It is a good place to start debugging web applications because, unlike in IIS, you do not have to worry about configuration and security settings. You also do not have to worry about stopping and restarting the web server. The web development server does have limitations, however, so eventually you will want to test your web application under IIS. This is especially true if your web application has pages accessible only by users with the appropriate security settings.

You can profile a debug build of your ASP.NET web application while it runs in the web development server by following the instructions below.

### Setting up the Performance Profiler

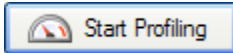
To profile ASP.NET applications running on web development server, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

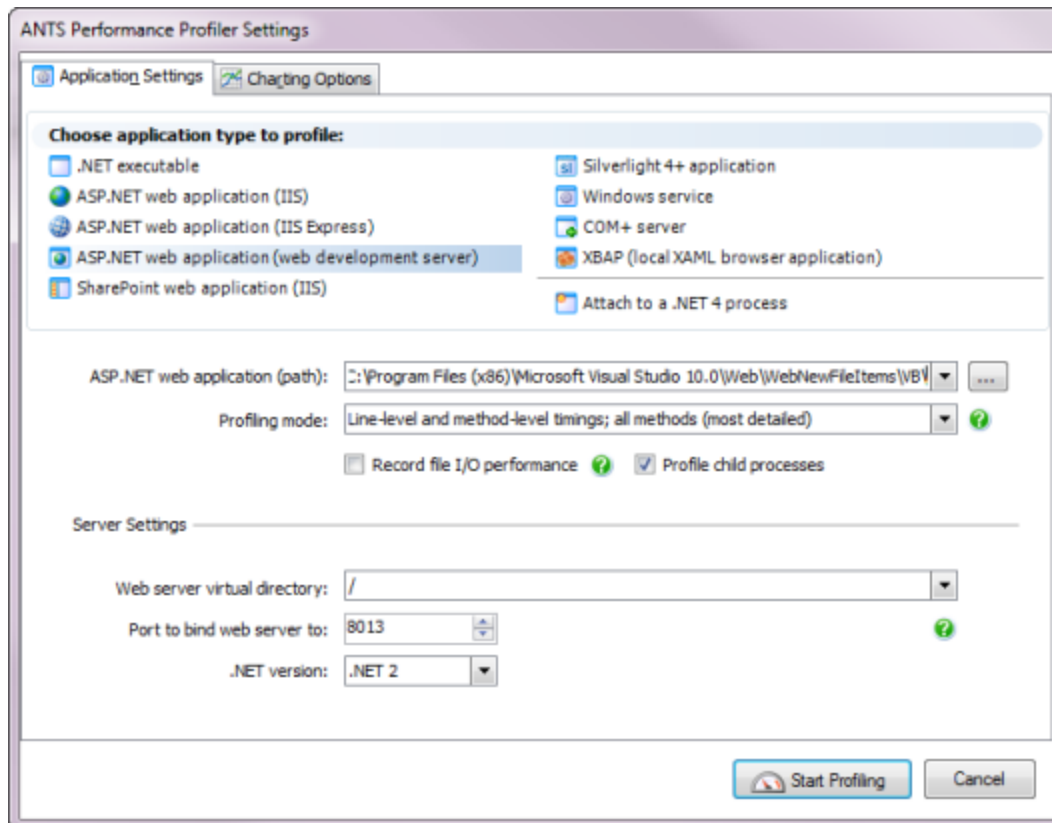
1. Under **Choose application type to profile**, click **ASP.NET web application (web development server)**.
2. Set the **ASP.NET web application (path)** for the web application that you want to profile.  
The path is where the file with the file extension *.aspx* is located.  
Note: You may find that the *.aspx* file is saved in the */WebSites/* directory, not in the */Projects/* directory where you would normally expect to find it.

#### ANTS Performance Profiler 7.3 and later

For MVC3 applications that do not have a *.aspx* file, enter the path to the folder containing the highest-level *Web.config* file for the application.

3. Select the required **Profiling mode**, **file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).  
In ANTS Performance Profiler 7.3 and later, SQL queries and inbound HTTP requests are profiled automatically.
4. Under **Server Settings**, set **Web server virtual directory** to the application's virtual path on the web server.
5. In the **Port to bind web server to** box, set the port on which ANTS Performance Profiler should listen.  
For example, if you specify *staging* for the virtual directory and *8013* for the port number, your web application starts on URL *http://localhost:8013/staging/*.
6. Set the **.NET version** used by your web application.
7. If required, change the performance counters to record; see [Setting up Charting Options](#).
8. Click





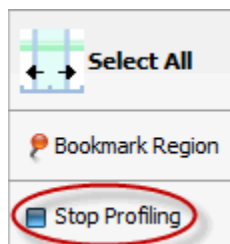
The web development server starts and the web application is shown in Internet Explorer. Note that, although you can interact with the application using any web browser, closing the Internet Explorer instance opened by ANTS Performance Profiler will end your profiling session.

During a profiling session, while your application runs, you can obtain results in the profiler by selecting areas of the timeline.

When you have finished interacting with your web application, click the



**Stop Profiling** button in ANTS Performance Profiler.



This closes both the web development server and the ASP.NET application. ANTS Performance Profiler shows all of the profiling data collected for the application.



## Profiling ASP.NET applications running on IIS Express

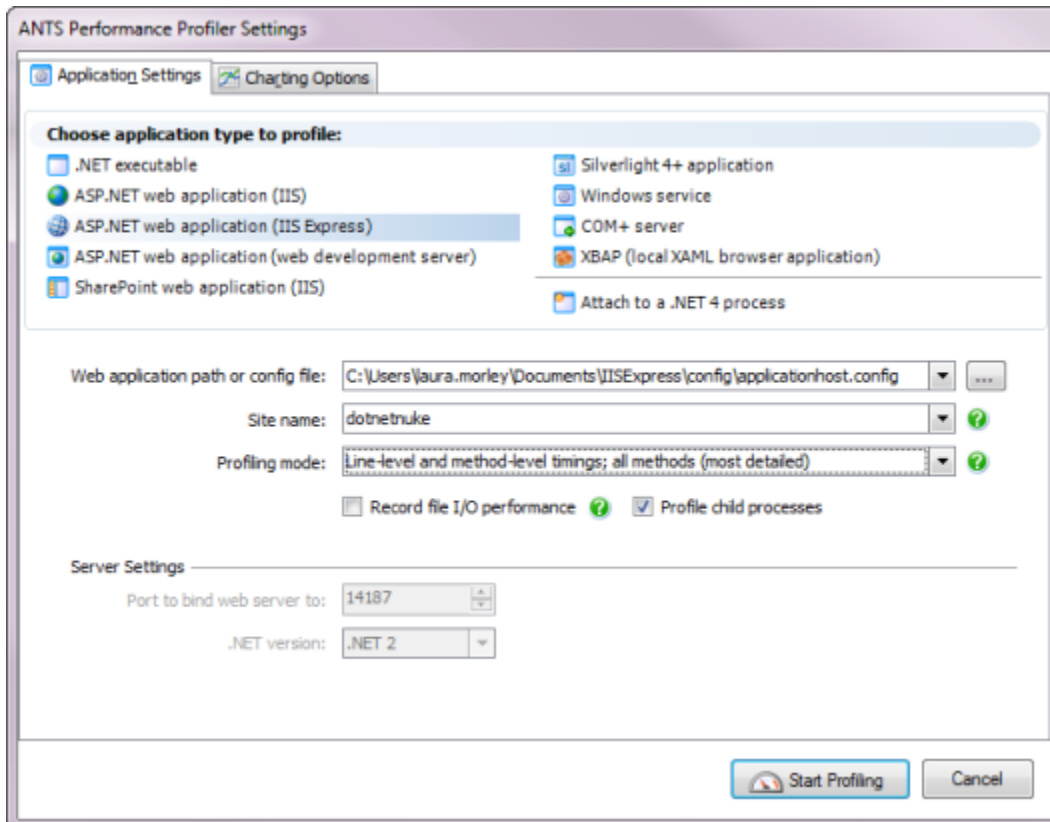
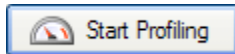
IIS Express is a free version of IIS, commonly used in development environments. Like the ASP.NET built-in development server, Cassini, IIS Express can run on development machines without requiring the installation of a full web server, and does not require an administrator account to run. This makes it convenient for testing debug builds of web applications.

You can profile sites running on IIS Express by following the instructions below.

### Setting up the Performance Profiler

To profile ASP.NET applications running on IIS Express, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

1. Under **Choose application type to profile**, click **ASP.NET web application (IIS Express)**.
2. Set the **Web application path or config file** for the web application that you want to profile.  
You can specify the path directly to the web application, or to the site's *ApplicationHost.config* file. If you select the *ApplicationHost.config* file, a list of available applications configured in that file is displayed: select the application you want to profile.
3. Select the required **Profiling mode**, **file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).  
In ANTS Performance Profiler 7.3 and later, SQL queries and inbound HTTP requests are profiled automatically.
4. Under **Server Settings**, set **Web server virtual directory** to the application's virtual path on the web server.
5. In the **Port to bind web server to** box, set the port on which ANTS Performance Profiler should listen.  
For example, if you specify *staging* for the virtual directory and *8013* for the port number, your web application starts on URL *http://localhost:8013/staging/*.
6. Set the **.NET version** used by your web application.
7. If required, change the performance counters to record; see [Setting up Charting Options](#).
8. Click



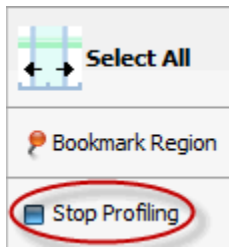
IIS Express starts and the web application is shown in Internet Explorer. Note that, although you can interact with the application using any web browser, closing the Internet Explorer instance opened by ANTS Performance Profiler will end your profiling session.

During a profiling session, while your application runs, you can obtain results in the profiler by selecting areas of the timeline.

When you have finished interacting with your web application, click the



**Stop Profiling** button in ANTS Performance Profiler.



This closes the ASP.NET application. ANTS Performance Profiler shows all of the profiling data collected for the application.

## Profiling SharePoint

ANTS Performance Profiler can profile managed code that runs on a Microsoft SharePoint 2007 or 2010 server.

### To profile SharePoint:

In ANTS Performance Profiler, on the **ANTS Performance Profiler Settings** dialog box, under **Choose application type to profile**, click **SharePoint web application (IIS)**.

1. Select the **ASP.NET web application (URL)** for the SharePoint site that you want to profile.  
To load a list of currently-running SharePoint sites into the dropdown list, click



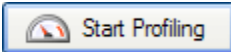
Note that the port specified in this URL is the port where the application usually runs, which is not necessarily the same as the port where the application is to be profiled (see step 4).

2. Select the required **Profiling mode**, **file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).
3. Select the port on which to profile your application:
  - If your application does not bind to a specific port, select **Unused port** and choose a port that is not used by IIS.
  - If your application must use the port on which it currently runs, select **Original port (IIS will restart)**.

If IIS does not restart correctly, use IIS Manager to stop the website until you have finished profiling.

The port where the application will be profiled is displayed at the bottom of the ANTS Performance Profiler Settings dialog box.

4. If required, select **Manually specify ASP.NET account details** and enter the **User name**, **Password** and **Domain**.  
ANTS Performance Profiler profiles your web application as the Windows Local System user. If the Windows Local System user might not have appropriate permissions to use your SharePoint site collection, enter the credentials of a user who does have the required permissions. Note that the user you specify must also be an administrator on the computer on which ANTS Performance Profiler runs.
5. If required, change the performance counters to record; see [Setting up Charting Options](#).
6. Click



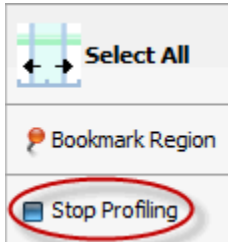
During the profiling session, use your SharePoint site collection as normal. Any additions that you have coded, such as web parts and lists, will be reflected in the ANTS Performance Profiler results if these objects have been accessed. You can interact with the profiler while your site is still running, selecting regions of the timeline to view results.

Calls to sandboxed web parts run in a separate process under partial trust, and cannot be profiled along with the rest of a SharePoint collection. To profile sandboxed web parts, instead profile **SPUserCodeV4** and its child processes as a Windows service: for instructions, see [Profiling Windows services](#).

When you have finished interacting with your web application, click the



**Stop Profiling** button in ANTS Performance Profiler.



Note that you can only profile SharePoint site collections running on the computer where the profiler is installed.

## Troubleshooting

If you encounter problems while trying to profile a SharePoint application, see [Troubleshooting SharePoint profiling](#).

### Troubleshooting in ANTS Performance Profiler 7.0

In ANTS Performance Profiler 7.0, the continuous profiling tool runs as a separate tool from the main ANTS Performance Profiler product. If you installed an early access build of the ANTS Performance Profiler continuous profiler IIS module, other profilers - including the desktop ANTS Performance Profiler product - will be unable to profile applications running in IIS on this computer. To re-enable other profilers with IIS, uninstall the IIS Profiler Module:

1. From your computer's Start menu, launch the **Continuous Profiling Configuration Tool**
2. Click **Uninstall**.

For more information on configuring continuous profiling, see [Setting up continuous profiling](#).

## Profiling Silverlight applications

You can use ANTS Performance Profiler to profile Silverlight 4 and 5 applications.

Note that line-level timings are not available when profiling Silverlight applications, and that some performance counters are not shown on the timeline.

### Introducing Silverlight applications

Microsoft Silverlight applications can run either in a web browser, or in an Out-Of-Browser mode, depending on the setting chosen at compile-time.

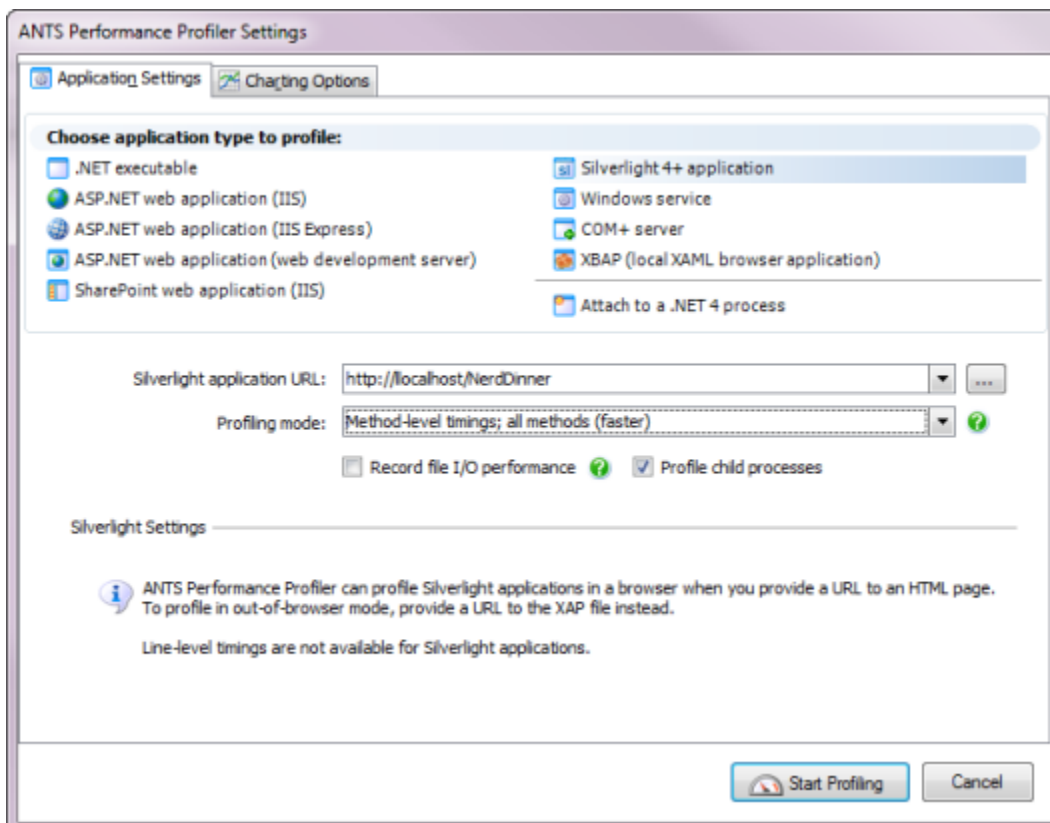
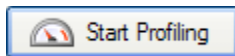
Silverlight applications that run in a web browser are contained in an HTML or ASPX file which may be stored:

- in the local file system.
- on an HTTP server on the local computer.
- on an HTTP server on a remote computer.

### Setting up the Performance Profiler (Silverlight applications running in a web browser)

To profile Silverlight browser applications running in a web browser, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

1. Close all instances of Internet Explorer that are currently running on your computer.  
This ensures that ANTS Performance Profiler connects to the correct instance of *iexplore.exe* when you start profiling.
2. Under **Choose application type to profile**, click **Silverlight 4+ browser application**.
3. For the **Silverlight application URL**, enter either the local file path (prefixed by *file:///*) to the HTML file that embeds the Silverlight application, or the URL of the HTML or ASPX file that embeds the application.  
Note that the URL must be a server running on the same computer as the computer where ANTS Performance Profiler is installed.
4. Select the required **Profiling mode**, **file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).  
Note that line-level timings are not available for Silverlight applications.
5. If required, change the performance counters to record; see [Setting up Charting Options](#).
6. Click



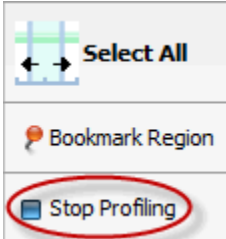
Microsoft Internet Explorer launches, and the Silverlight application is shown.

During a profiling session you can interact with the profiler while your application is still being profiled, and obtain results by selecting areas of the timeline.

When you have finished interacting with your web application, click the



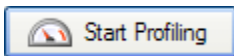
**Stop Profiling** button in ANTS Performance Profiler.



### Setting up the Performance Profiler (Silverlight applications running in Out-Of-Browser mode)

To profile Silverlight 4 or 5 browser applications running in Out-Of-Browser mode, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

1. Under **Choose application type to profile**, click **Silverlight 4+ browser application**.
2. For the **Silverlight application URL**, enter the local file path (prefixed by *file:///*) to the XAP file.  
This is normally found in `%LOCALAPPDATA%\Microsoft\Silverlight\OutOfBrowser\`.
3. Select the required **Profiling mode**, **file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).  
Note that line-level timings are not available for Silverlight applications.
4. If required, change the performance counters to record; see [Setting up Charting Options](#).
5. Click



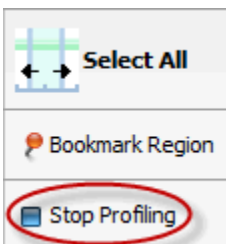
The Microsoft Silverlight Out-Of-Browser launcher starts, and the Silverlight application is shown.

During a profiling session you can interact with the profiler while your application is still being profiled, and obtain results by selecting areas of the timeline.

When you have finished interacting with your web application, click the



**Stop Profiling** button in ANTS Performance Profiler.



### Troubleshooting

If you experience problems:

- Ensure that the Silverlight application is compiled against Silverlight 4 or 5. Older versions of Silverlight may appear to work but will give inaccurate results.
- Check that the Silverlight application runs correctly on the computer which you are profiling it on, without the profiler attached.
- Note that you may need to run ANTS Performance Profiler as an Administrator for some types of Silverlight application and computer configuration.

## Profiling Windows services

A Windows service is a long-running executable that performs specific functions and which is designed not to require user intervention.

To profile a Windows service, it must be installed on the computer on which the profiling will take place. You can install your service using the *installutil.exe* utility which is supplied with Microsoft Visual Studio.

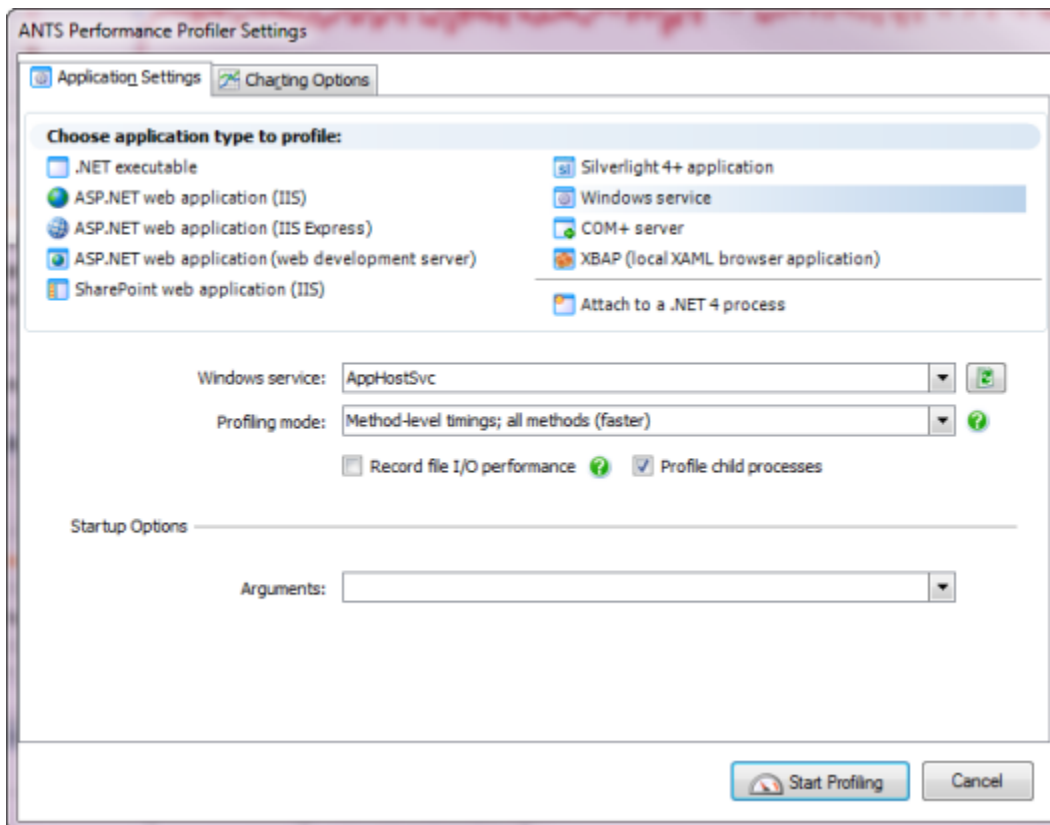
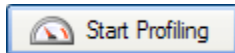
### In ANTS Performance Profiler 7.0

To obtain results with source code, you must use a debug build of your .NET Windows service.

### Setting up the performance profiler

To profile Windows services, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

1. Under **Choose application type to profile**, click **Windows service**.
2. Select your .NET **Windows Service** from the drop down list. The service here is named *Sample Service*.
3. Select the required **Profiling mode**, **file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).
4. If required, specify any **Arguments** that are to be used when running the service.
5. If required, change the performance counters to record; see [Setting up Charting Options](#).
6. Click



If the service is not already started, ANTS Performance Profiler will start the service.

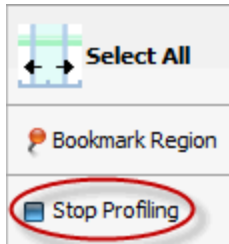
If the service is already started, ANTS Performance Profiler will restart the service.

During a profiling session you can obtain results by selecting areas of the timeline.

When you have finished profiling the service, click the



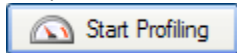
**Stop Profiling** button in ANTS Performance Profiler.



### Profiling WCF services

If the service that you want to profile is implemented using the Windows Communication Foundation (WCF), think of the service as a server in a server-client relationship.

Set up ANTS Performance Profiler as described above but after you have clicked



, start interacting with the client program to call the service.

The service's communications with the client are included in the results.



## Profiling a COMplus server application

To profile a COM+ server application correctly, there are two main steps:

1. Change the COM+ server so that it can be profiled.
2. Set up ANTS Performance Profiler.

### Change the COM+ server so that it can be profiled

ANTS Performance Profiler can only profile COM+ server applications which are activated in a process provided by the system, not by the client. To do this, set the `ApplicationActivation` attribute as follows:

```
[assembly: ApplicationActivation(ActivationOption.Server)]
```

If you cannot set this attribute (for example, if you do not have access to the source code), you may still be able to profile the COM+ application by profiling the client application. You should note, however, that:

- the resulting server application will not be a true COM+ application and will run in the client process
- ANTS Performance Profiler will profile the client application, and will treat the COM+ server as a DLL invoked by the client.

### Set up ANTS Performance Profiler

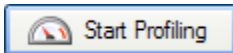
To profile COM+ server applications, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

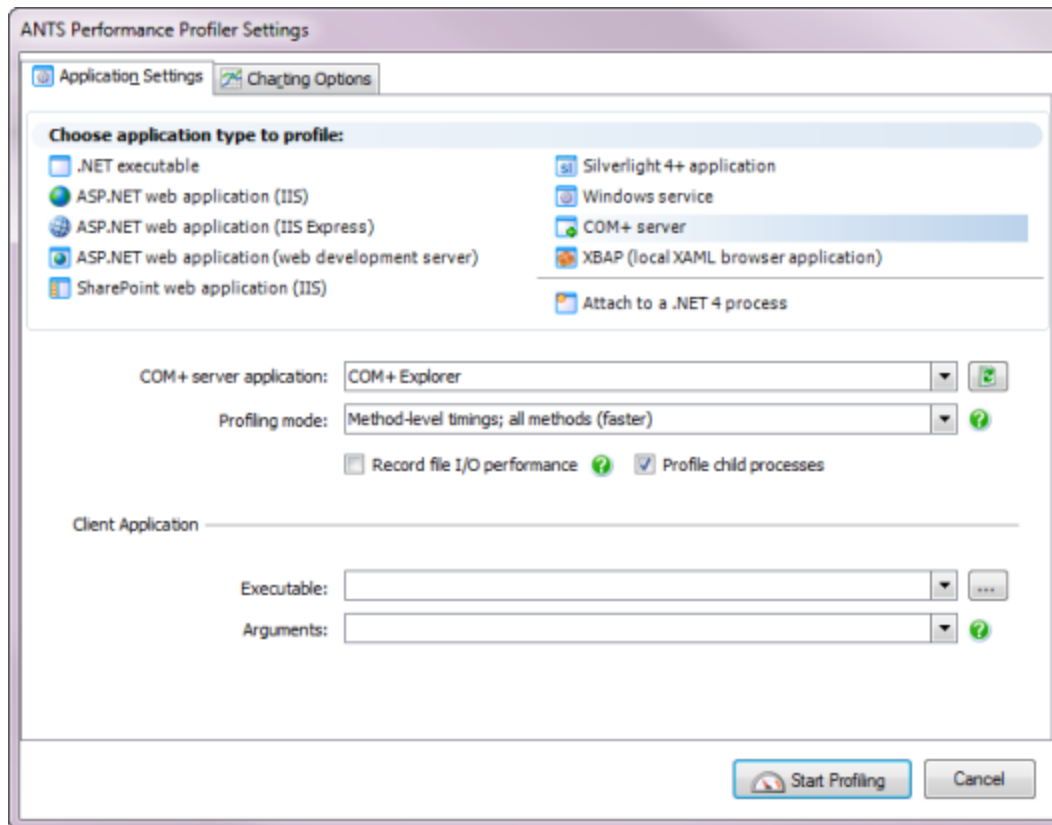
1. Under **Choose application type to profile**, click **COM+ server**.
2. Use the dropdown list to select the **COM+ server application** that you want to profile. Click



to update the list of COM+ server applications.

3. Select the required **Profiling mode**, **file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).
4. If you want to profile a client application's communications with the COM+ server:
  - a. Browse to the client application **Executable**.
  - b. If required, specify any command line **Arguments** that are to be used when running the client application.
5. If required, change the performance counters to record; see [Setting up Charting Options](#).
6. Click





If the client application was specified, the client application starts.

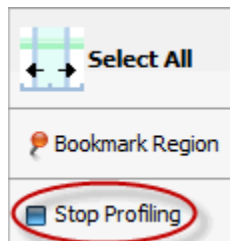
If the client application was not specified, start it manually.

During a profiling session you can interact with the client application while your COM+ server is still being profiled. You obtain results by selecting areas of the timeline.

When you have finished interacting with your client application, and you are ready to finish profiling the COM+ server, click the



**Stop Profiling** button in ANTS Performance Profiler.



## Profiling remote COM+ applications

The security architecture of COM+ does not allow COM+ applications to be started from a Remote Desktop session with a GUI, or a terminal services session.

To profile COM+ applications remotely, type `mstsc /console` at a command prompt to start a Remote Desktop session in console mode.

Note that all users who are logged on to the computer that you are connecting to will be logged off when you connect.

## Troubleshooting

Some COM+ server applications need to be fully trusted before being profiled. If profiling the COM+ server does not work, you may need to make the application's code fully trusted by setting the `ApplicationAccessControl` attribute as follows:

```
[assembly: ApplicationAccessControl(false)]
```

Note that you should not normally release the COM+ application in this trusted state.

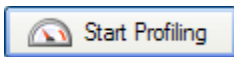
## Profiling XBAP applications

The procedure for profiling XBAP applications depends on whether the XBAP is locally-hosted or remotely-hosted.

### To profile locally-hosted XBAP applications

To profile locally-hosted XBAP applications, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

1. Under **Choose application type to profile**, click **XBAP (local XAML browser application)**.
2. Navigate to the **XBAP application** that you want to profile.
3. If required, change the performance counters to record; see [Setting up Charting Options](#).
4. Click



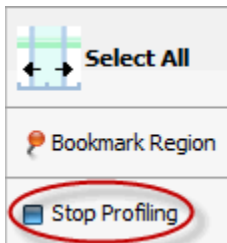
Internet Explorer starts, and displays the XBAP application. Use the XBAP application normally.

During a profiling session you can interact with the profiler while your application is still being profiled, and obtain results by selecting areas of the timeline.

When you have finished interacting with your XBAP application, click the



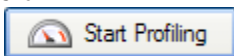
**Stop Profiling** button in ANTS Performance Profiler.



### To profile remotely-hosted XBAP applications

To profile remotely-hosted XBAP applications, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

1. Under **Choose application type to profile**, click **.NET executable**.
2. Choose Internet Explorer as the **.NET executable** that you want to profile.  
Internet Explorer is normally located in *%ProgramFiles%/Internet Explorer/*
3. Select the required **Profiling mode**, **SQL and file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).  
Note that line-level timings are not available when profiling XBAP applications.
4. If required, change the performance counters to record; see [Setting up Charting Options](#).
5. Click



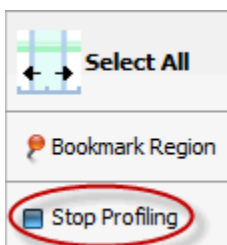
Internet Explorer starts. Navigate to the page which embeds the XBAP application, and use the XBAP application normally.

During a profiling session you can interact with the profiler while your application is still being profiled, and obtain results by selecting areas of the timeline.

When you have finished interacting with your XBAP application, click the



**Stop Profiling** button in ANTS Performance Profiler.



## Attaching to a running .NET 4 process

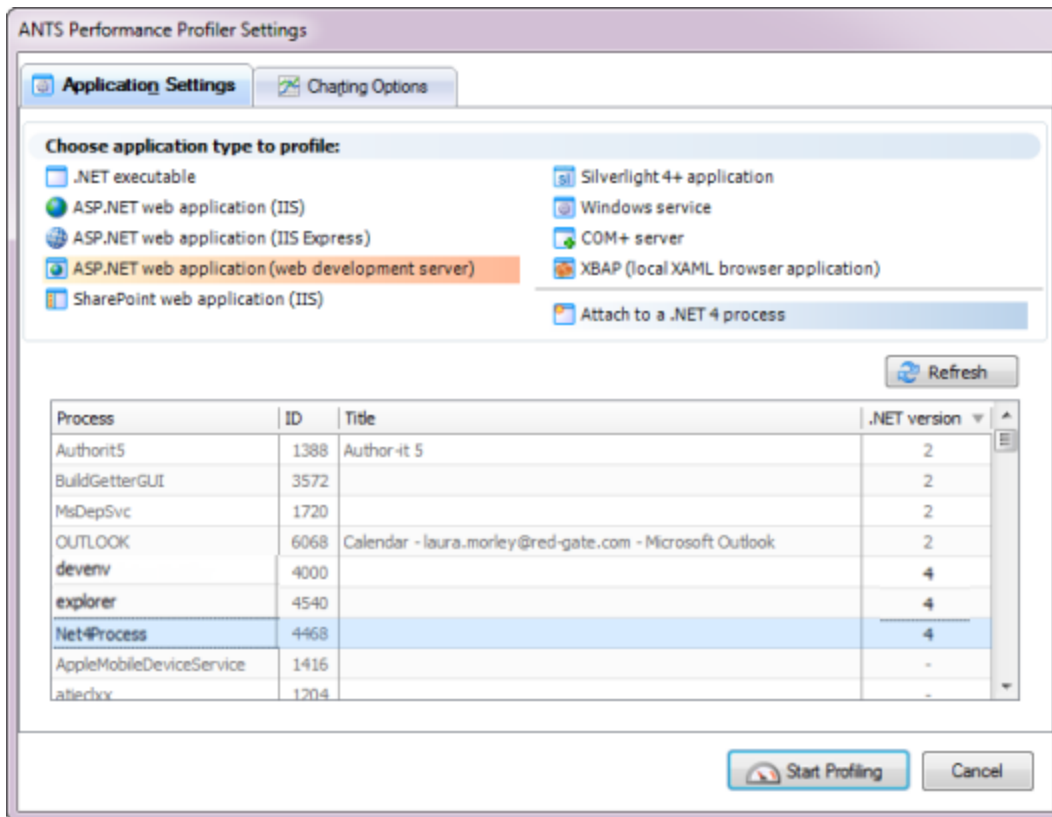
ANTS Performance Profiler can attach to a .NET 4 process which is already running. This feature allows you to start profiling your applications when performance problems are noticed, without having to restart the application from scratch.

This page 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;

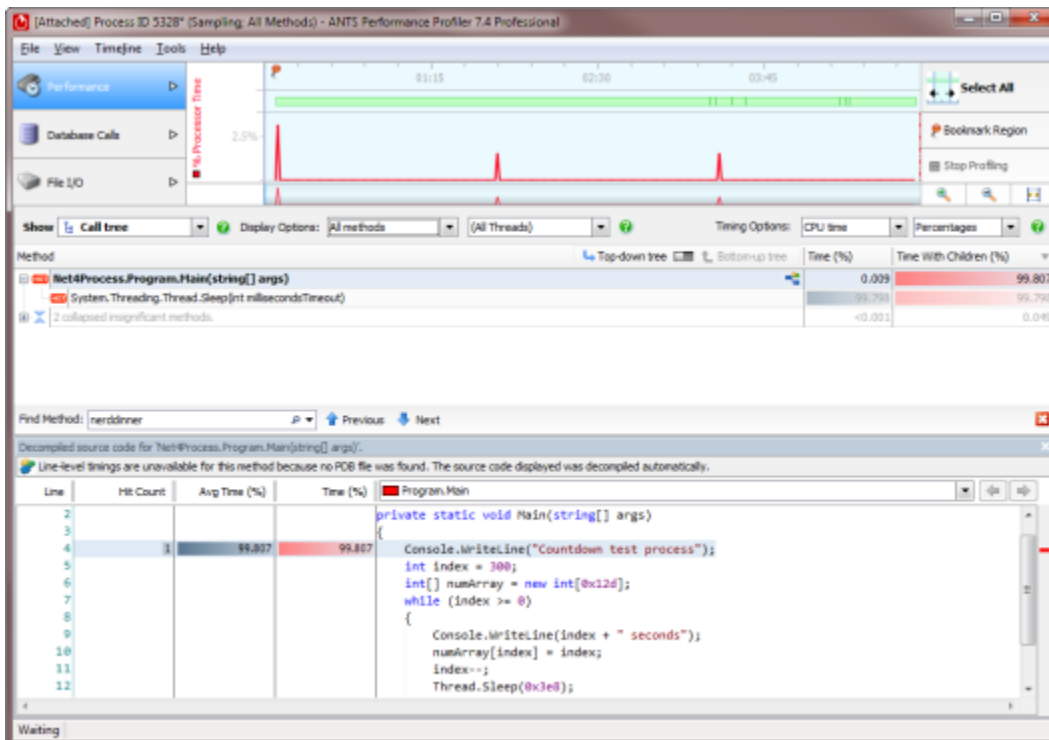
namespace Net4Process
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Countdown test process");
            int i = 300;
            int[] array = new int[301];
            while (i >= 0)
            {
                Console.WriteLine(i + "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



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.

## Profiling SQL queries in ANTS Performance Profiler 7.2 and later

ANTS Performance Profiler 7.4 Professional automatically profiles the calls your application makes to any Microsoft SQL server or Oracle database instance. From ANTS Performance Profiler 7.3, this includes applications running in Amazon RDS and SQL Services (SQL Azure). Database call profiling works with calls to both local and remote databases, and for any version of SQL server.

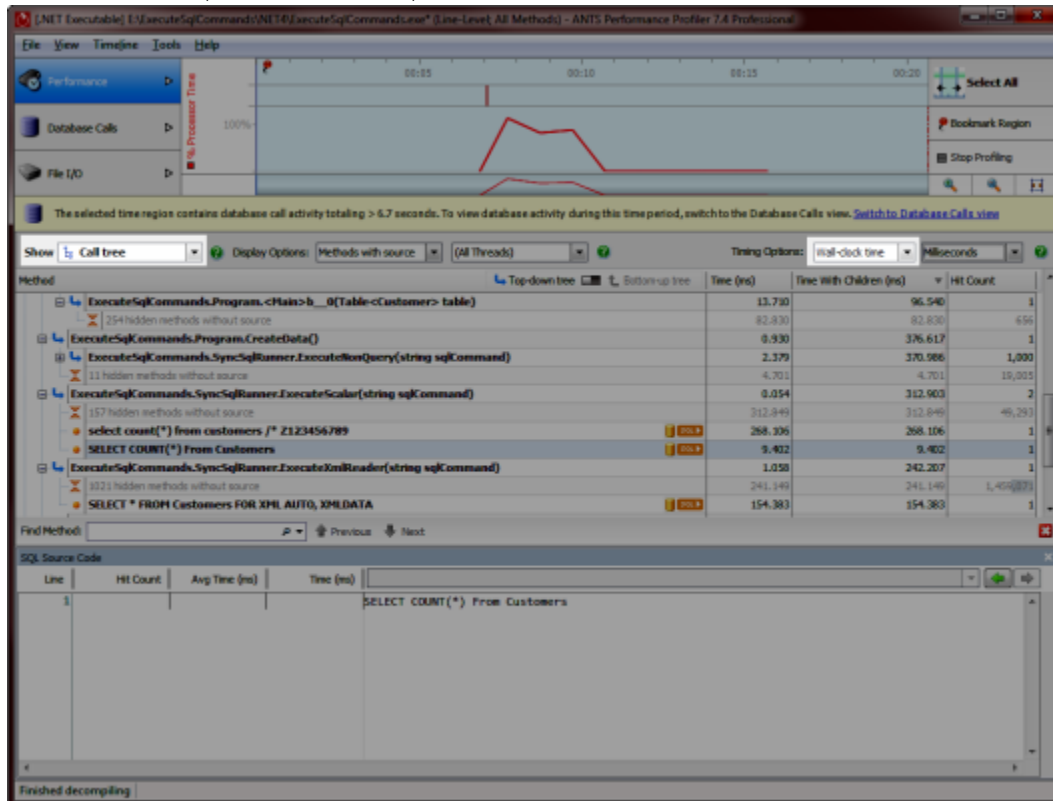
This feature is available only in ANTS Performance Profiler Professional edition. To upgrade from Standard edition, see [Upgrading](#).

Timing data for SQL calls is captured in all profiling modes except for sampling mode.

## Viewing SQL profiling results

Database calls appear in the call tree as children of the .NET methods that generate them.

1. On the results toolbar, select **Call tree** view, and **Wall-clock time**.



2. Drag to select the portion of the timeline that you are interested in.
3. In the call tree, calls to your database during the selected time period are identified with the



icon.

The call tree shows the first line of the database query (excluding any initial blank lines), the time the query took to return its first result, and the query's hit count.

4. To view the full text of the query, click the



icon to switch to **Database Calls** view.

For more details on using Database Calls view to see all the queries your application made during the selected time period, see [Working with Database Calls view](#).

## Troubleshooting

See [Troubleshooting SQL and HTTP call profiling](#).

## Profiling SQL queries in ANTS Performance Profiler 7.0

You can use ANTS Performance Profiler Professional to profile database queries sent by an application to a Microsoft SQL server instance.

You might want to profile SQL queries if performance timings have revealed that a line of code involving a database query is particularly slow.

Note that:

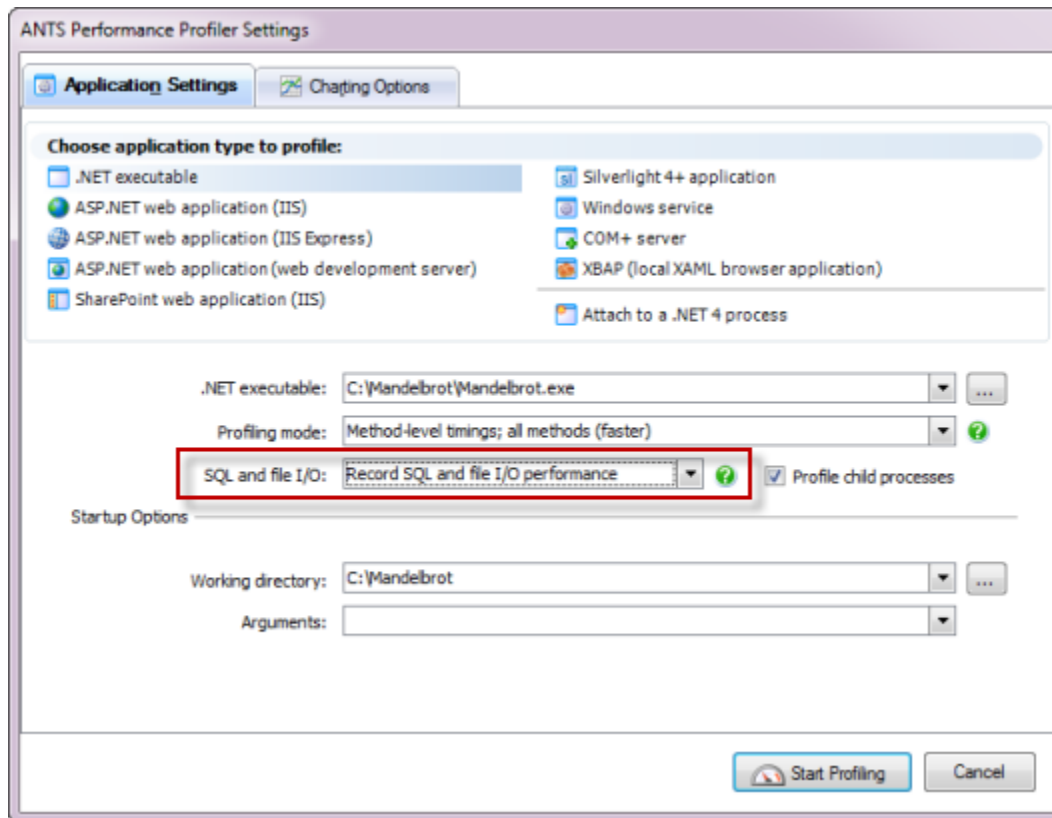
- profiling SQL queries is not possible with SQL Server Express: Because SQL Server Express does not expose performance counters, ANTS Performance Profiler cannot obtain results.
- the SQL Server instance must be on the same computer that the profiler is running on.
- you can only profile SQL queries on Windows Vista or later, or Microsoft Server 2008.

## Viewing SQL profiling results

Before profiling SQL, we recommend that you check for performance issues in your code (and any third-party code). Profile SQL when you have identified a slow line of code that involves a SQL query.

Set up a new profiling session using the [Application Settings dialog](#).

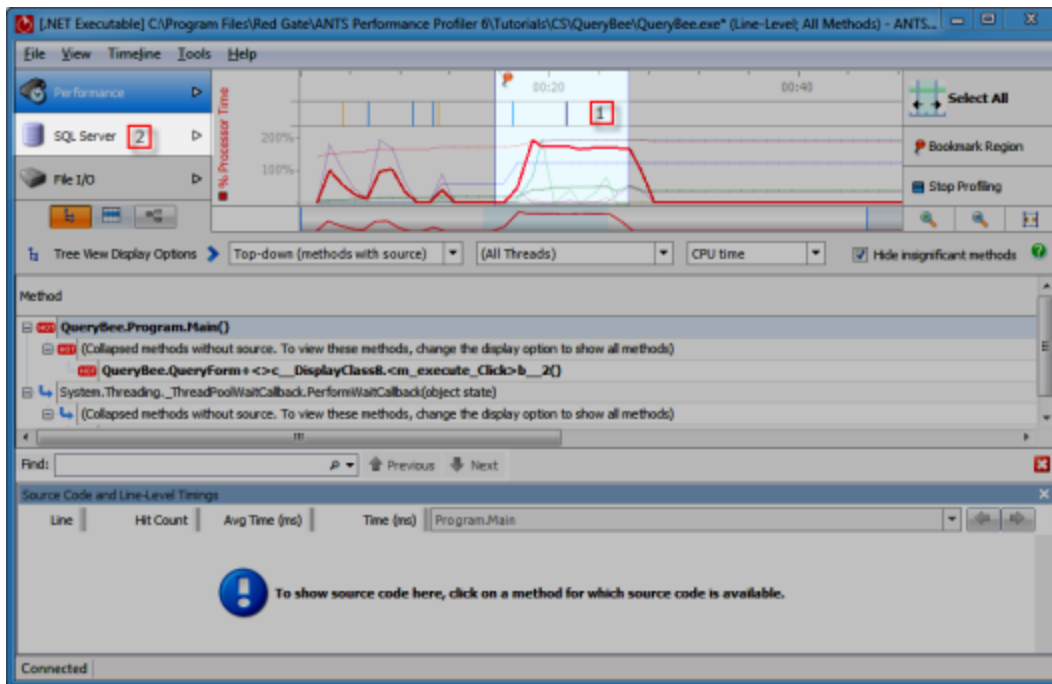
Ensure that **SQL and file I/O** is set to **Record SQL and file I/O performance**.



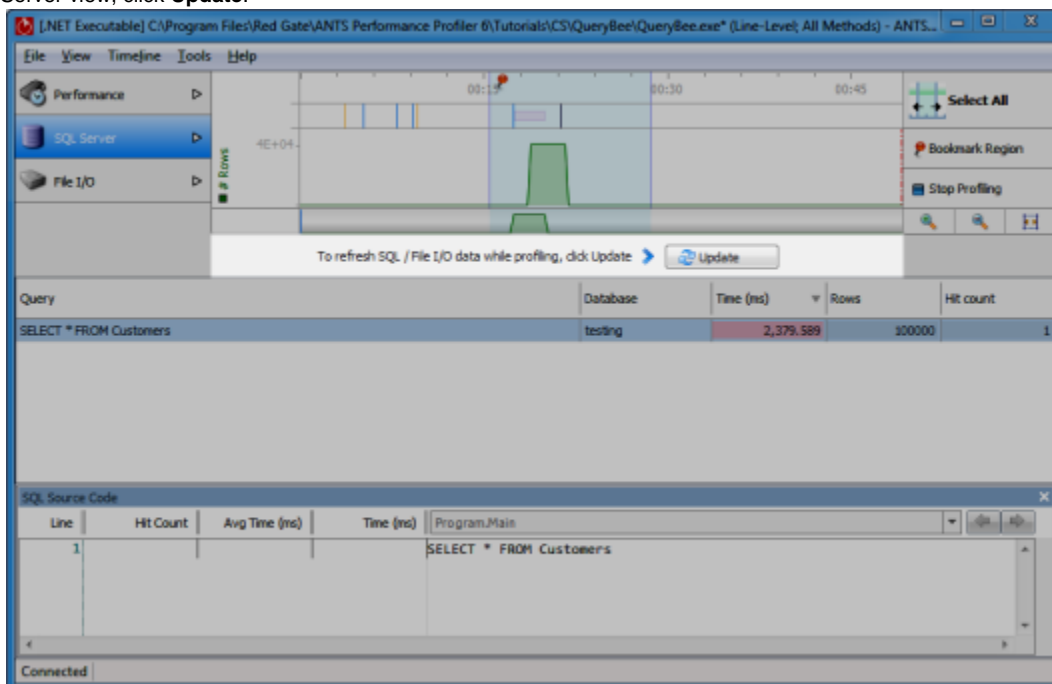
## Viewing SQL results while profiling

1. Drag to select the portion of the timeline that you are interested in.
2. Click **SQL Server**.





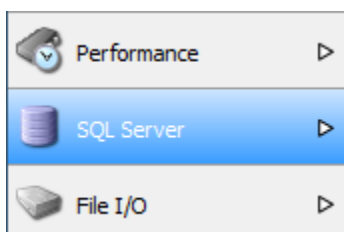
- Note that the timeline is not automatically updated while profiling SQL queries. To display queries performed since you switched to SQL Server view, click **Update**.



- The timeline and the Query panel update to show the latest data.

## Viewing SQL results after profiling

If profiling is not currently in progress, click **SQL Server**.



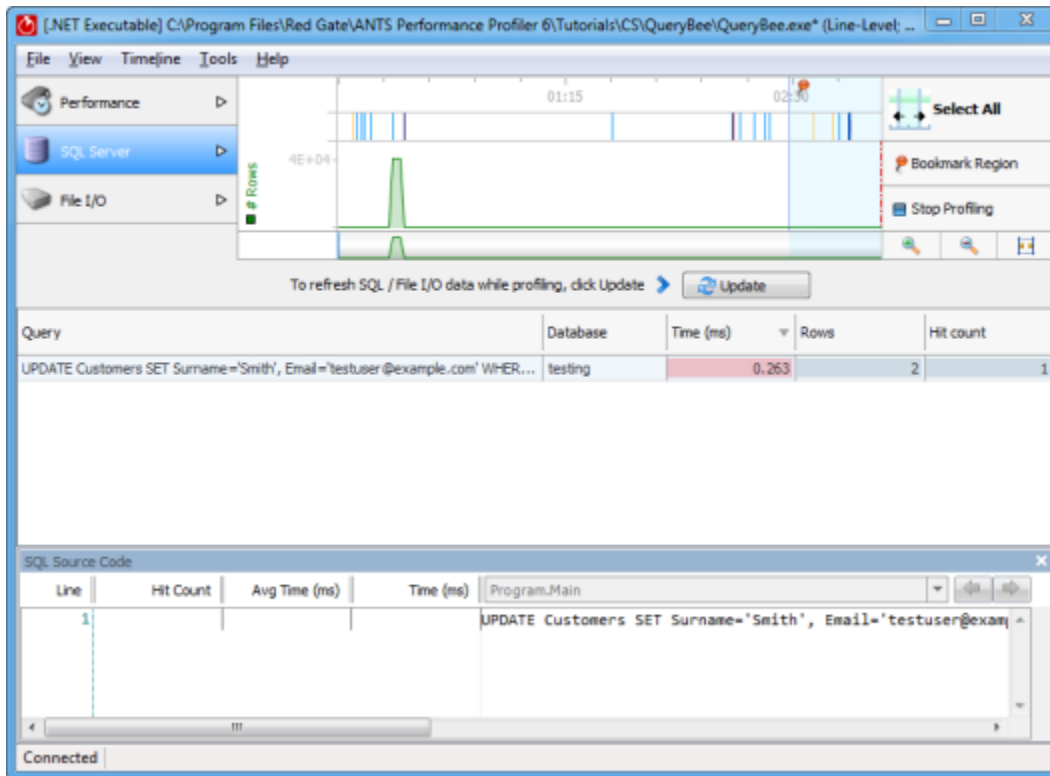
SQL results for the entire complete profiling session are displayed on the timeline.

## Tips

### Long-length queries

If the SQL query is a multi-line query, or is just very long, it may be truncated in the Query list. To display the full query, select it. The query is shown in the scrollable SQL Source Code panel.

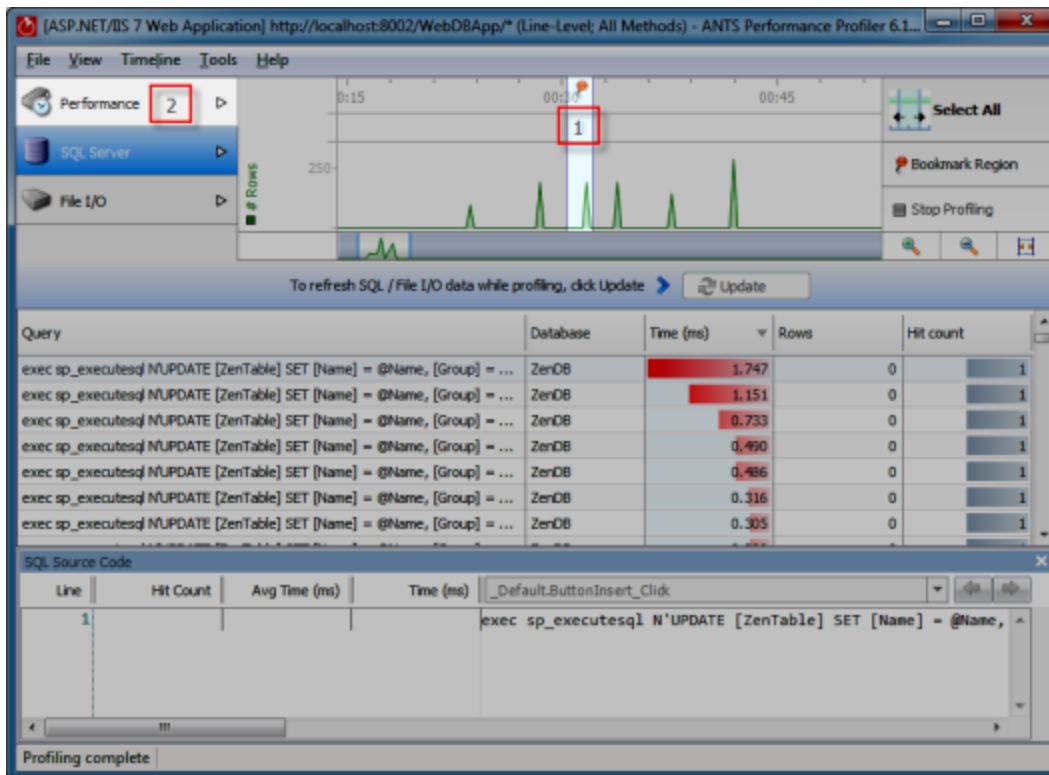
Note that the Hit Count, Avg Time and Time columns are empty in the SQL Source Code panel, because line-level timings are unavailable for SQL Server. See the Hit Count and Time columns in the Query list to view the time taken by the entire query.



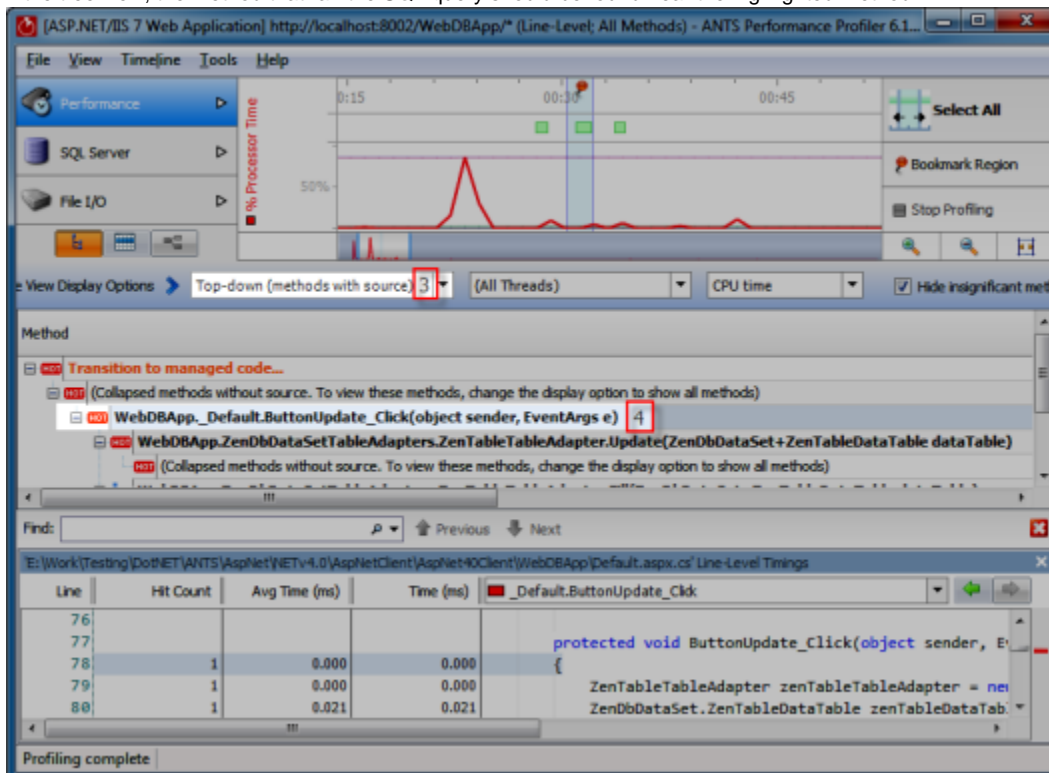
### Linking back to your code

To find out which of your code's methods ran a particular SQL query:

1. Select the time when the query ran on the timeline. Include some time just before the SQL query ran, and the server load increased: this will ensure your selected range includes the time when the method that runs the query was called.
2. Switch back to **Performance** view.



- Under Tree View Display Options, select **Top-down (methods with source)**.
- In the tree view, the method that ran the SQL query should be found near the highlighted method.



- Browse the line-level timings to find code that could be optimized.

## Profiling File I/O

You can use ANTS Performance Profiler Professional to profile when the application that you are profiling reads from discs or writes to discs (including network drives).

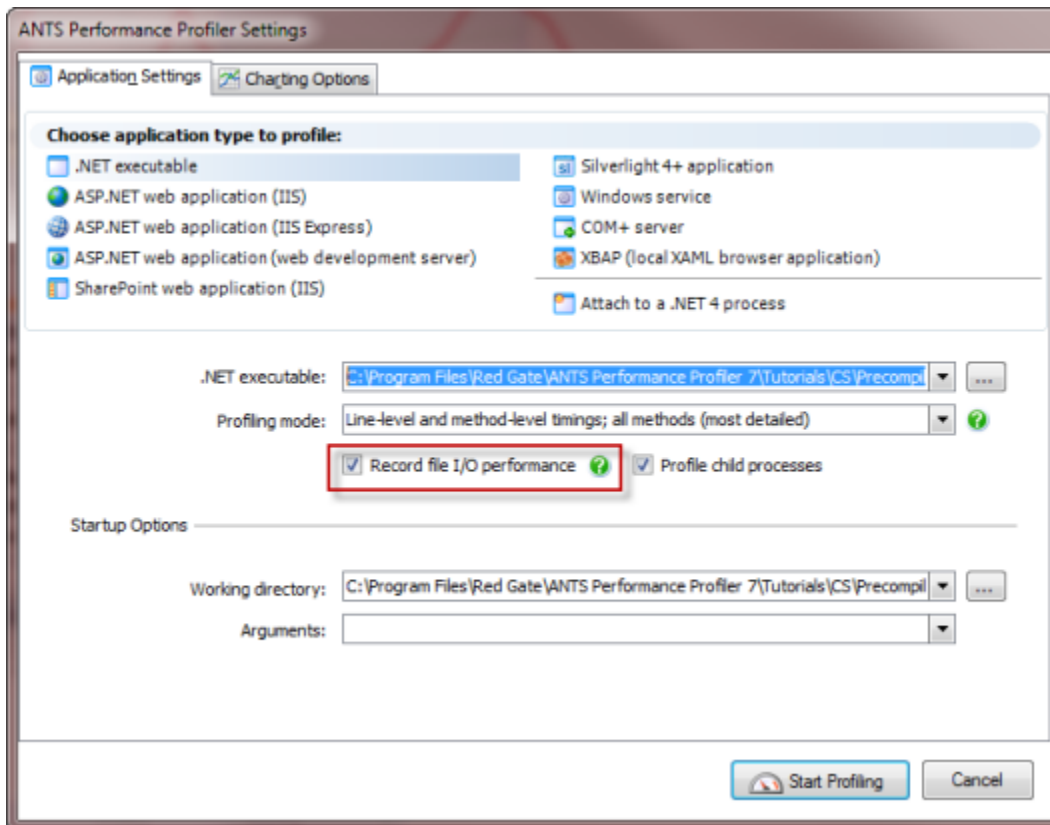
Note that you can only profile File I/O on Windows Vista or later, or Microsoft Server 2008.

### Setting up File I/O profiling

Before profiling File I/O, we recommend that you check for performance issues in your code (and any third-party code).

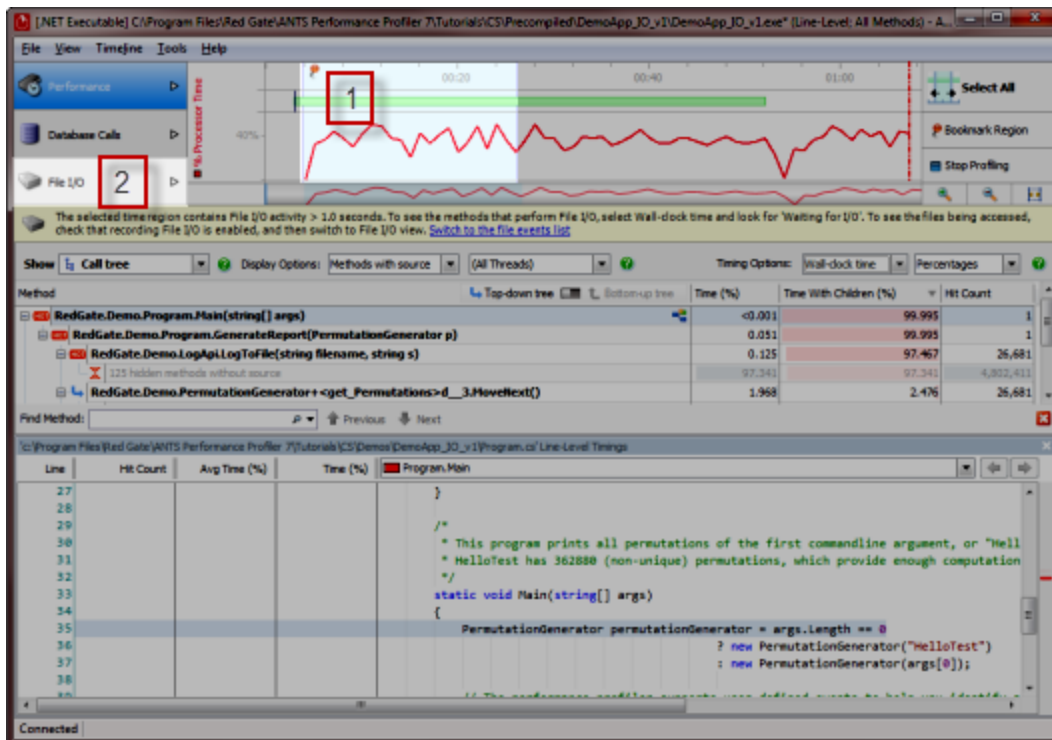
To enable File I/O profiling, set up the profiler in the same way that you would [configure the settings for performance profiling](#). Ensure that **Record file I/O performance** is selected (in ANTS Performance Profiler 7.0, that **SQL and file I/O** is set to **Record SQL and file I/O performance**).

While profiling in Performance view, set the Tree View Display Options to show **Wall-clock time**, because CPU time does not include time spent blocked waiting for File I/O. When you have identified a slow line of code, which involves a reading from or writing to a disc, profile File I/O.

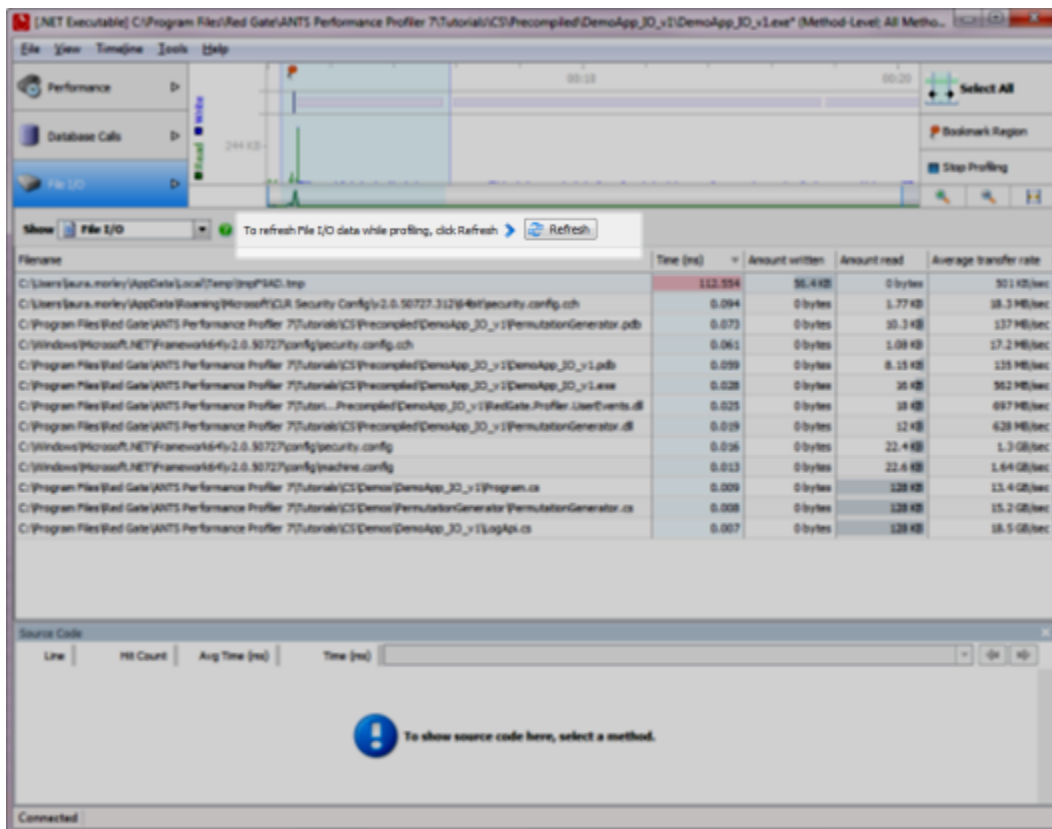


### Viewing File I/O results while profiling

1. Drag to select the portion of the timeline that you are interested in.
2. Click **File I/O**.

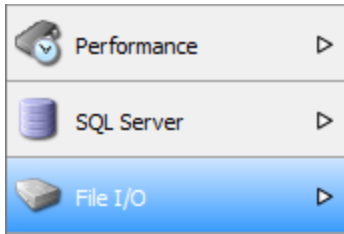


- Note that the timeline is not automatically updated while profiling File I/O. To display queries performed since you switched to File I/O view, click **Refresh**.



## Viewing File I/O results after profiling

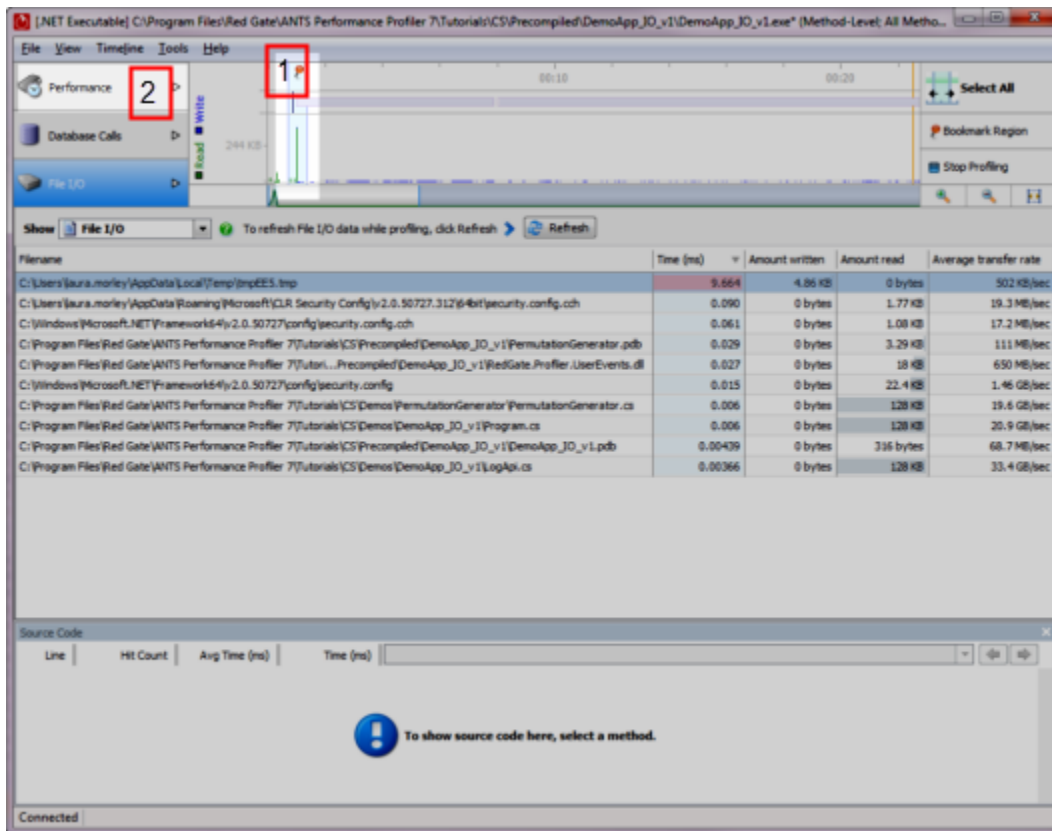
If profiling is not currently in progress, click **File I/O**.



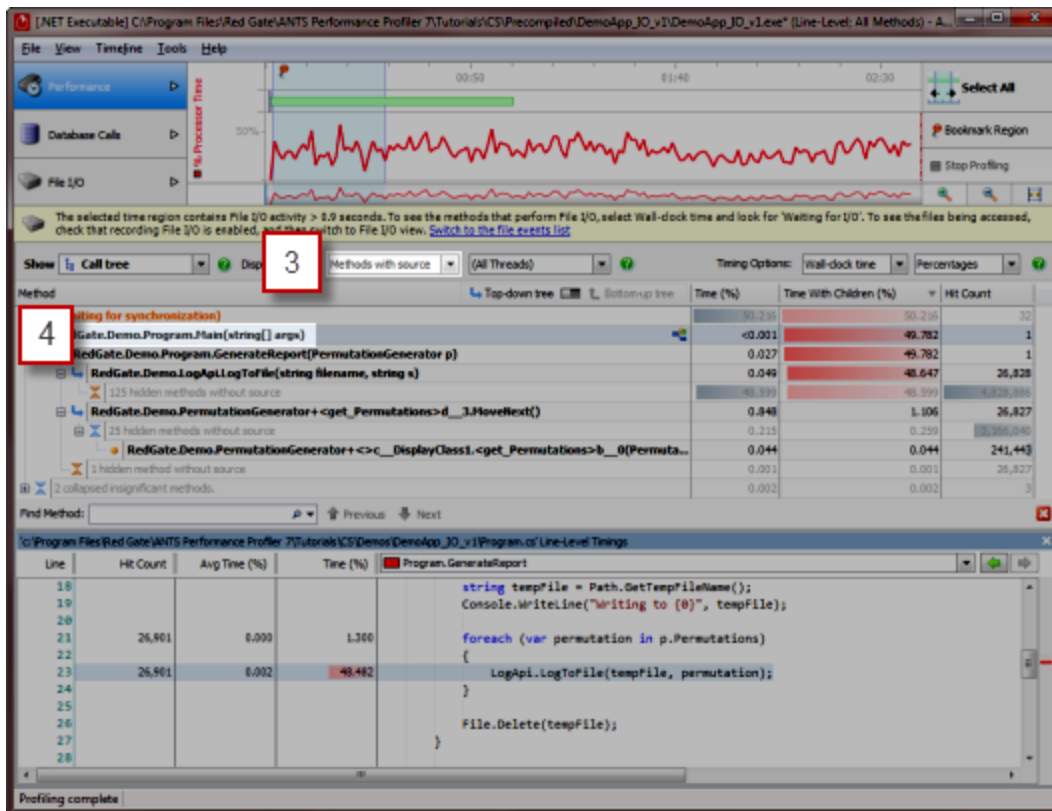
## Linking back to your code

To find which of your code's methods caused File I/O to occur:

1. Select the time when the I/O occurred on the timeline. Include some time just before the I/O because the method which caused the I/O will be called before the I/O takes place.
2. Switch back to **Performance** view.



3. Under Tree View Display Options, select **Methods with source**.
4. In the tree view, the method which caused the File I/O should be found near the highlighted method.



Use the line-level timings to look for code that could be optimized.

## Profiling tests in MSTest

MSTest is a software unit testing framework developed by Microsoft, which lets you create, manage, and run unit tests from within the Visual Studio IDE, as well as from the command line.

You can profile tests running in MSTest. Profiling your tests ensures that you're quickly alerted to any bottlenecks in your tests.

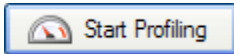
This page assumes that you're familiar with MSTest, and have already built an assembly full of tests that you want to profile.

Before you start profiling, you'll need a debug build of your test assembly.

### Setting up the performance profiler

To profile .NET executables, on the **ANTS Performance Profiler Settings** dialog box, perform the following steps:

1. Under **Choose application type to profile**, click **.NET executable**.
2. Set the **.NET executable** box to the path where *MSTest.exe* is installed:
  - For .NET 2.0 applications, this is often: `%ProgramFiles(x86)%\Microsoft Visual Studio 9.0\Common7\IDE\MSTest.exe`
  - For .NET 4.0 applications, this is often: `%ProgramFiles(x86)%\Microsoft Visual Studio 10.0\Common7\IDE\MSTest.exe`
3. Select the required **Profiling mode**, **file I/O**, and **Profile child processes** options; see [Working with Application Settings](#).
4. Specify the command line **Arguments** that are to be used when running the application.  
Set a `/testcontainer` argument to tell MSTest the path to the assembly that contains all of the tests. For example:  
`/testcontainer:"C:\Documents and Settings\<USER NAME>\My Documents\Visual Studio 2008\Projects\LoginForm\MyTests\bin\Debug\MyTests.dll"`
5. If required, change the performance counters to record; see [Setting up Charting Options](#).
6. Click



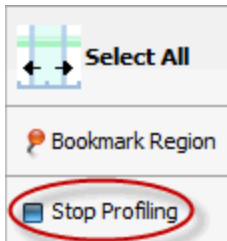
MSTest starts and executes all of the tests contained within the assembly.

During a profiling session you can interact with the profiler while your tests are still being profiled, and obtain results by selecting areas of the timeline.

When you have finished interacting with your application, click the



**Stop Profiling** button in ANTS Performance Profiler.





## Profiling from the command line (API)

You can profile applications from the command prompt in ANTS Performance Profiler Professional.

To run a profiling session from the command prompt, run *Profile.exe* with the appropriate options.

For example, to profile an executable called SimpleApp.exe using line-level timings and saving results as a CSV, use:

```
Profile.exe /e:"C:\testing\SimpleApp.exe" /ll /csv:"C:\testing\results.csv"
```

Profiling applications from the command prompt is useful if you want to integrate performance profiling in an automated test procedure; see [Integrating ANTS Performance Profiler in a test procedure](#).

## Profiling IIS from the command line

You cannot currently profile applications which run on IIS 5, 6, 7, or 7.5 from the command line. To work around this limitation:

1. Profile IIS from the ANTS Performance Profiler graphical user interface.
2. When you have finished profiling, on the **File** menu, click **Save Project**.
3. Use the `/project` argument on the command line to load the saved project.
4. Interact with your application (e.g. through a web browser or client application - this can be manually or via automated calls, if your application can be run from the command line). This is necessary to execute the managed code that the profiler will profile.

Applications in IIS Express, however, can be profiled directly from the command line. Arguments for IIS Express profiling are included below.

## List of command line arguments

### ***/help*** (Alias: */?*)

Displays this help message. Use in conjunction with `/verbose` for more detailed information.

If this switch is used with any switches other than `/verbose`, `/html`, `/out`, `/force` or `/outputwidth` then those switches will be ignored, the help message will be printed, and 0 will be returned as the process exit code.

### ***/html***

Causes help to be output as HTML.

Must be used with the `/help` switch.

### ***/quiet*** (Alias: */q*)

Quiet mode - no output.

### ***/verbose*** (Alias: */v*)

Verbose mode.

### ***/force*** (Alias: */f*)

Forces overwriting of output files that already exist. If this flag is not set and a file already exists then program will exit with an exit code indicating an I/O error.

### ***/argfile:<argfile>***

File containing the XML argument specification.

### ***/out:<fileName>***

Redirects console output to the specified file.

### ***/project:<project>*** (Alias: */p*)

A performance profiler project that should be used to begin profiling.

### ***/executable:<executable>*** (Alias: */e*)

An executable to run in the profiler.

***/arguments:<arguments> (Alias: /args)***

The arguments to pass to the executable.

***/workingDirectory:<workingDirectory> (Alias: /wd)***

The working directory to use when profiling the application.

***/port:<port>***

The port that web applications should be profiled on. (Default: 8013)

***/cassini (Alias: /webdev)***

Profiling should be performed on the ASP.NET web development server (Cassini).

***/cassiniPath:<cassiniPath> (Alias: /cpath)***

The location of the ASP.NET project to use with the ASP.NET web development server (Cassini).

***/cassiniVirtualDirectory:<cassiniVirtualDirectory> (Alias: /cvd)***

The virtual directory to use for the ASP.NET web development server (Cassini).

***/cassiniNetVersion:<cassiniNetVersion> (Alias: /cnv)***

If Visual Studio 2010 is installed, this is the version of .NET to use when running the web development server (Cassini). Default: 0

***/express***

Profiling should be performed on the IIS Express webserver.

***/expressSitePath alias="epath"***

The path where the website can be located on the disk.

***/expressConfigPath alias="config"***

The path to an "applicationhost.config" file to load.

***/expressSiteName alias="site"***

The site name to load from an applicationhost.config file. By default the first site found will be used.

***/expressNetVersion alias="env"***

The version of .NET (2.0 or 4.0) to use when running IIS Express.

***/service:<service>***

The name of a Windows service to profile.

***/complus:<complus>***

The name of a COM+ server to profile.

***/silverlight:<silverlight>***

The URL of a site containing a Silverlight application to profile.

***/RecordSqllo (Alias: /rs)***

The profiler should try to record SQL and File I/O events.

***/profileSubprocesses (Alias: /sp)***

The profiler should profile both the target and any child processes it spawns.

***/timeout:<timeout> (Alias: /t)***

The number of seconds to wait before terminating the target process. Set to 0 to indicate that no timeout should be used. The default is 120 seconds.

***/lineLevel (Alias: /ll)***

The profiler should record line-level timings as well as method-level timings.

***/methodLevel (Alias: /ml)***

The profiler should record only method-level timings (the default).

***/onlyWithSource (Alias: /ows)***

The profiler should only record values for methods which have source code files specified in their debugging data (pdb) files.

***/sampling (Alias: /sm)***

The profiler should use sampling to produce approximate results quickly.

***/includeSource:<includeSource> (Alias: /is)***

Whether or not the results should include the source code. Permitted values are *on* and *off*. The default is *on*.

***/inlining:<inlining> (Alias: /in)***

Default: *on*

Whether or not the profiler should allow .NET to inline functions. Turning this off will produce results for more methods, at the expense of a less accurate reflection of the processes performance. Permitted values are *on* and *off*. The default is *on*.

***/compensate:<compensate> (Alias: /comp)***

Whether or not the profiler should adjust results to account for its own overhead. Permitted values are *on* and *off*. The default is *on*.

***/simplify:<simplify> (Alias: /simp)***

Whether or not the profiler should simplify certain complicated stack traces to reduce resource requirements. Permitted values are *on* and *off*. The default is *on*.

***/avoidTrivial:<avoidTrivial> (Alias: /notriv)***

Whether or not the profiler should avoid extremely trivial functions to reduce resource requirements. These functions have a low hit count and a running time of only a few processor cycles. Permitted values are *on* and *off*. The default is *on*.

***/aspxPages:<aspxPages> (Alias: /aspx)***

Whether or not the profiler should profile the compiled contents of ASPX pages as well as the code that lies behind them. Turning this option on may considerably increase the amount of time the application spends in the JIT. Permitted values are *on* and *off*. The default is *off*.

***/threshold:<threshold>***

The threshold time in percent that a method must have used in order to be included in the report. Set to 0 to include all results. The default is 0.1.

***/csv:<csv>***

The name of a file to write a summary of the profiler results as CSV data to.

***/xml:<xml>***

The name of a file to write a summary of the profiler results as XML data to.

***/htmlreport:<htmlreport> (Alias: /h)***

The name of a file to write a summary of the profiler results as an HTML report.

**/calltree:<calltree>**

The name of a file to write a summary of the call tree profiler results as XML data to.

**/calltreehtml:<calltreehtml> (Alias: /cth)**

The name of a file to write a summary of the call tree profiler results as an HTML report.

**/data:<data>**

The name of a file to save the profiler results to. The contents of this file can be inspected using the ANTS Performance Profiler desktop application.

## Exit Codes

If an error occurs, the following exit codes may be displayed:

0 Success.

1 General error code.

3 Illegal argument duplication. Some arguments may not appear more than once in a command-line. If such arguments appear more than once this exit code will be returned.

8 Unsatisfied argument dependency or violated exclusion when user runs command line.

For example, /arg2 depends on /arg1 but you have specified /arg2 without specifying /arg1, or alternatively /arg2 cannot be used with /arg1 but you have tried to use them both.

32 Value out of range. Numeric value supplied for an argument that is outside the range of valid values for that argument.

33 Value overflow. The magnitude of a value supplied for an argument is too large and causes an overflow.

34 Invalid value. The value supplied for an argument is invalid.

35 No / invalid software license or trial period has expired.

64 General command-line usage error.

65 Data error. Some input data required by the tool is invalid or corrupt.

69 A resource or service required to run the tool is unavailable.

73 Failed to create report

74 IO error occurred. Generally returned if the program attempts to write to a file that already exists without the user having specified the /force option.

77 Action cannot be completed because the user does not have permission.

126 Execution failed because of an error.

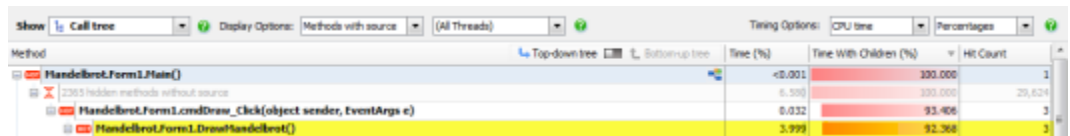
130 Execution stopped because Ctrl+Break.

## Examples of CSV and XML results files

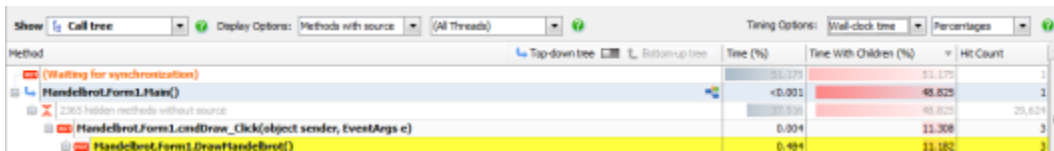
This section exemplifies the format of the CSV and XML results files that can be saved when profiling from the command line.

The examples are for the `Mandelbrot.Form1.DrawMandelbrot()` method. For comparison, the following screenshots show the data for the same method when displayed in the GUI.

With **CPU time** selected (i.e. excluding time elapsed while a thread was blocked) :



With **Wall-clock time** selected (i.e. showing total time elapsed, including blocking):



In **XML** format:

```
<Method class="Mandelbrot.Form1" name="DrawMandelbrot" PID="7728" has-source="yes">
  <HitCount>3</HitCount>
  <CPU ticks="3893693658" millisecs="1298.0397" percent="92.3678" />
  <Wallclock ticks="3893772648" millisecs="1298.0660" percent="11.1822" />
  <WithSelf ticks="168573329" millisecs="56.1907" percent-cpu="3.9990"
percent-wallclock="0.4841" />
</Method>
```

In **CSV** format (with headings row):

(Note that spaces have been added between each field to ensure that the lines in this example wrap.)

```
Method type, Class, Method, Hit count, CPU %, CPU milliseconds, CPU ticks, Wallclock
%, Wallclock milliseconds, Wallclock ticks, CPU % time with self, Wallclock % time
with self, Milliseconds with self, Ticks with self
, Mandelbrot.Form1,DrawMandelbrot(), 3, 92.3678149753427, 1298.039706707, 3893693658,
11.1821681679019, 1298.06604003906, 3893772648, 3.99896638783019, 0.484110266291296,
56.1907373464783, 168573329
```

## Integrating ANTS Performance Profiler in a test procedure

Integrating ANTS Performance Profiler in your existing automated test framework ensures that you are alerted whenever a change is made that would adversely affect your application's performance.

To integrate ANTS Performance Profiler in automated tests, you perform three general steps:

1. Profile your application from the command line, saving the results to a CSV or XML file.
2. Read the results into the test harness.
3. Make assertions about the data you have read.

### 1. Profiling your application from the command line

For automated tests, you do not need much detail in the results. We recommend that you either profile using method-level timing only, or that you profile in sampling mode.

To profile your application from the command line:

1. Choose whether you only need method-level timings, or whether you want to run the tests in sampling mode.
2. Choose whether you prefer results in CSV or XML format.  
To assist your decision, see [Examples of CSV and XML results files](#).
3. Set the command line arguments to reflect your choices.

For example, to profile *SimpleApp.exe*, using method-level timings (the default) and saving results to a CSV file, run:

```
Profile.exe /e:"C:\testing\SimpleApp.exe" /csv:"C:\testing\results.csv" /data:"C:\testing\results.app7results"
```

To profile *SimpleApp.exe*, using sampling and saving results to a XML file, run:

```
Profile.exe /e:"C:\testing\SimpleApp.exe" /sm /xml:"C:\testing\results.xml" /data:"C:\testing\results.app7results"
```

Note that in these examples, the .app7results file is also saved. The .app6results file is not used for integrating with the test procedure, but saving it ensures that you can investigate any problems without needing to profile your application again.

For a full list of arguments you can use when profiling at the command line, see [Profiling from the command line \(API\)](#).

### 2. Read the results into the test harness

Write a simple command line application to read the CSV or XML data.

For information on how to read XML data in C# using the `XMLTextReader` class, see [How to read XML from a file by using Visual C#](#) (Microsoft documentation)

### 3. Make assertions about the data you have read

Using your automated test framework, ensure that the performance data is within a reasonable bound for each method that you are interested in.

For example, if you know from experience that `exampleMethod()` normally takes about 0.02 seconds of CPU time, you assert that the time taken by `exampleMethod()` must not be longer than 0.02 seconds + 20% (0.024 seconds).

In the NUnit test framework:

```
Assert.Greater(0.024, double exampleMethodTime);
```

## Using the Visual Studio add-in

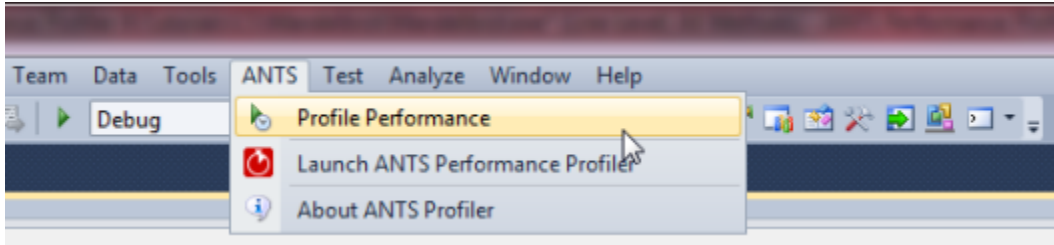
The ANTS Performance Profiler Visual Studio add-in allows you to:

- Launch ANTS Memory Profiler from your IDE.
- Switch straight to your source code from ANTS Performance Profiler.

## Launching ANTS Performance Profiler from Visual Studio

Installing the add-in adds a new **ANTS** menu in Visual Studio. If you also have ANTS Memory Profiler installed, both profilers will be available under this menu.

Build your solution in Visual Studio and then select **Profile Performance** to profile the build.

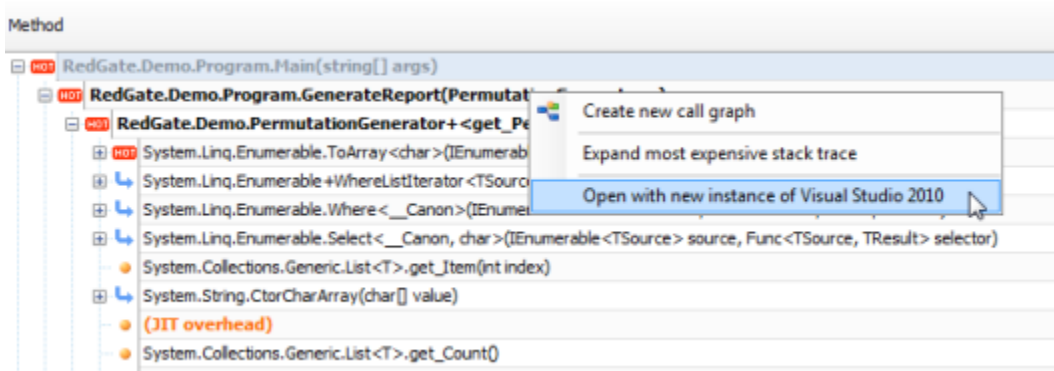


## Switching to your source code from ANTS Performance Profiler

So that ANTS Performance Profiler can identify classes with source code, you must ensure that the *.pdb* file is in the same directory as the application. See [Resolving .PDB problems](#) for more details.

You can switch to source code from the call tree and the methods grid. In both, classes with source code are shown in bold. You can use the **Find** bar to search for your class's namespace.

Right-click a class with source code to show the context menu.



To open only the source code associated with that class, select **Open with new instance of Visual Studio 20xx**.

It is often more useful to open the source code inside its solution. To do this, open the solution in Visual Studio. Return to ANTS Performance Profiler and open the context menu. On the context menu, select **Open with (Solution Name) - Microsoft Visual Studio (Visual Studio 20xx)**.

## Troubleshooting

- If the path to the source code in the *.pdb* file is invalid, the class is still shown in bold, but the context menu does not display the Visual Studio options. Recompile the application on the computer you are using to profile it.
- If the solution was opened with elevated privileges (Visual Studio is running as administrator), the option for opening the source code inside the solution might not be shown. Restart Visual Studio under the same credentials as ANTS Performance Profiler.
- The Visual Studio add-in is a separate program from ANTS Performance Profiler, and is installed by default as part of Redgate's .NET Developer Bundle. If you purchased ANTS Performance Profiler as a standalone product, you might not have the add-in. In that case, download the free trial of the [.NET Developer Bundle](#) from the Redgate website and install **ANTS Profiler Visual Studio Add-in 1.0**. You do not need a new license for the add-in, and the add-in will not expire when the bundle's trial period ends.

## Setting up continuous profiling

Continuous Profiling was an experimental feature in ANTS Performance Profiler 7.0 only. It was removed in subsequent versions.

### ANTS Performance Profiler includes an early access build of a continuous profiling tool, which lets you collect performance data on running ASP.NET applications hosted in IIS 7 or 7.5.

The tool uses an IIS module to monitor the .NET call stacks in each IIS worker process (*w3wp.exe*), recording hit counts, timings, and CPU usage data for each method call. Profiling results are displayed in a user interface accessible via a web browser.

Continuous profiling is under active development and ANTS Performance Profiler v7.0 includes an early access version of the functionality. If you have any questions or comments, please visit the [Continuous profiling feedback forum](#).

#### To enable continuous profiling:

1. From your computer's Start menu, launch the **Continuous Profiling Configuration Tool**.
2. Click **Install**.

The continuous profiling tool installs a profiling module into IIS. The module collects profiling data on all web applications running in IIS.

Results are written to *C:\inetpub\ProfilerLogs* in the current early access build, this directory setting cannot be changed. If this directory does not exist, ANTS Performance Profiler creates it. Results are deleted automatically after 6 hours, or when the total size of results files reaches 500MB whichever is sooner.

The screenshot shows the 'Continuous Profiling Configuration Tool' window. It has two main sections: 'IIS Profiler Module' and 'Web-based profiler interface'. The 'IIS Profiler Module' section shows a status of 'Not installed' and a 'Data Directory' of 'C:\inetpub\ProfilerLogs'. The 'Web-based profiler interface' section also shows a status of 'Not installed'. It includes fields for 'Installation Path' (C:\Program Files\Common Files\Red Gate\), 'Read Profiling Results From' (C:\inetpub\ProfilerLogs), a checked checkbox for 'Create an application in IIS', a 'Target Site' dropdown (Default Web Site), a 'Virtual Directory' field (/ANTSPProfiler), and a 'Run in Application Pool' field (ANTS Performance Profiler). Each section has 'Install' and 'Uninstall' buttons.

#### To install and deploy the continuous profiling user interface:

1. Choose the **Installation Path**.

In the current early access build, the **Read Profiling Results From** path cannot be changed from the default **Data Directory**, *C:\inetpub\ProfilerLogs*.

2. Select **Create site in IIS**.
3. Choose the **Target Site** in IIS in which the profiler will create a virtual directory to host the web interface.

This should be a site you have already created in IIS. By default your IIS instance's *Default Web Site* is used.

4. Name the **Virtual Directory** in which the profiler interface will be created within the target site.

By default the directory is named *ANTS Profiler*.



If you have a custom IIS configuration, and want to host the web interface at a particular URL, you should first use IIS to create the site in which you want to host the web interface application. Then, in the continuous profiling configuration tool, choose your custom site from the **Target Site** dropdown and set the **Virtual Directory** to the path at which you want the web interface to appear.

5. Click **Install**.

By default, ANTS Performance Profiler creates the web interface in its own dedicated application pool, specified in **Run in Application Pool**.

## Viewing continuous profiling results

For information on viewing continuous profiling results in the web interface, see [Working with continuous profiling](#).

## Compatibility with other profilers

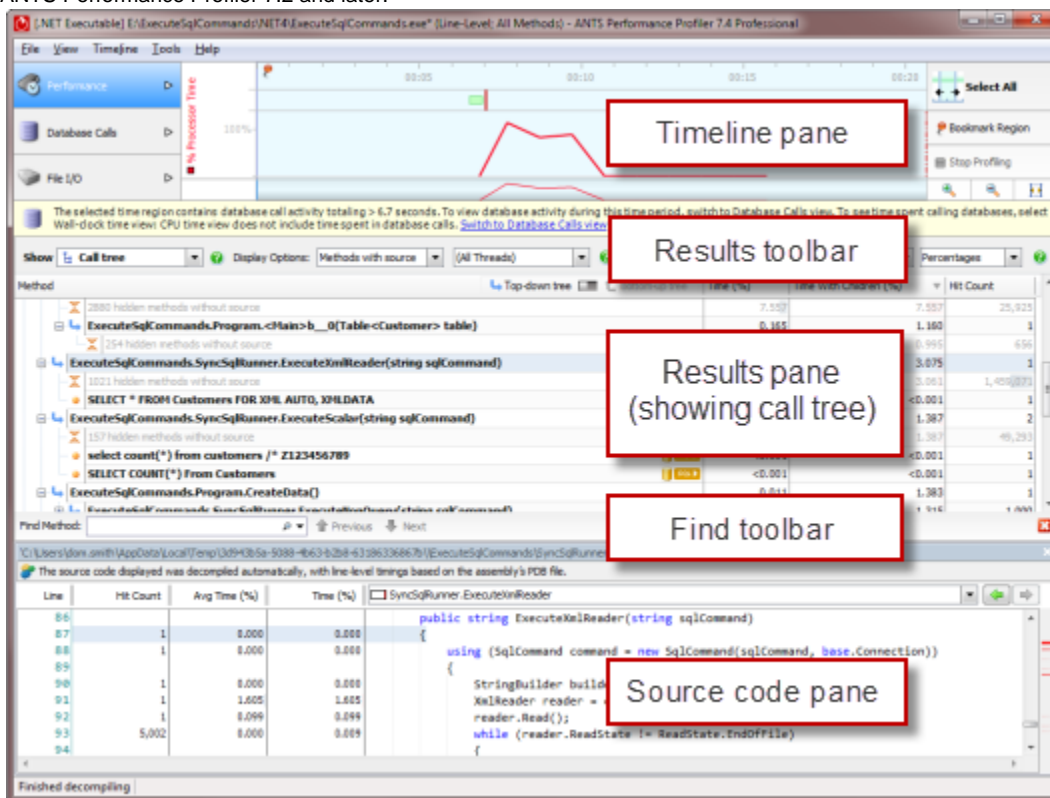
The continuous profiling tool currently runs as a separate tool from the main ANTS Performance Profiler product. If you have installed an early access build of the ANTS Performance Profiler continuous profiler IIS module, other profilers - including the desktop ANTS Performance Profiler product - will be unable to profile applications running in IIS on this computer. To re-enable other profilers with IIS, uninstall the IIS Profiler Module:

1. From your computer's Start menu, launch the **Continuous Profiling Configuration Tool**.
2. Click **Uninstall**.

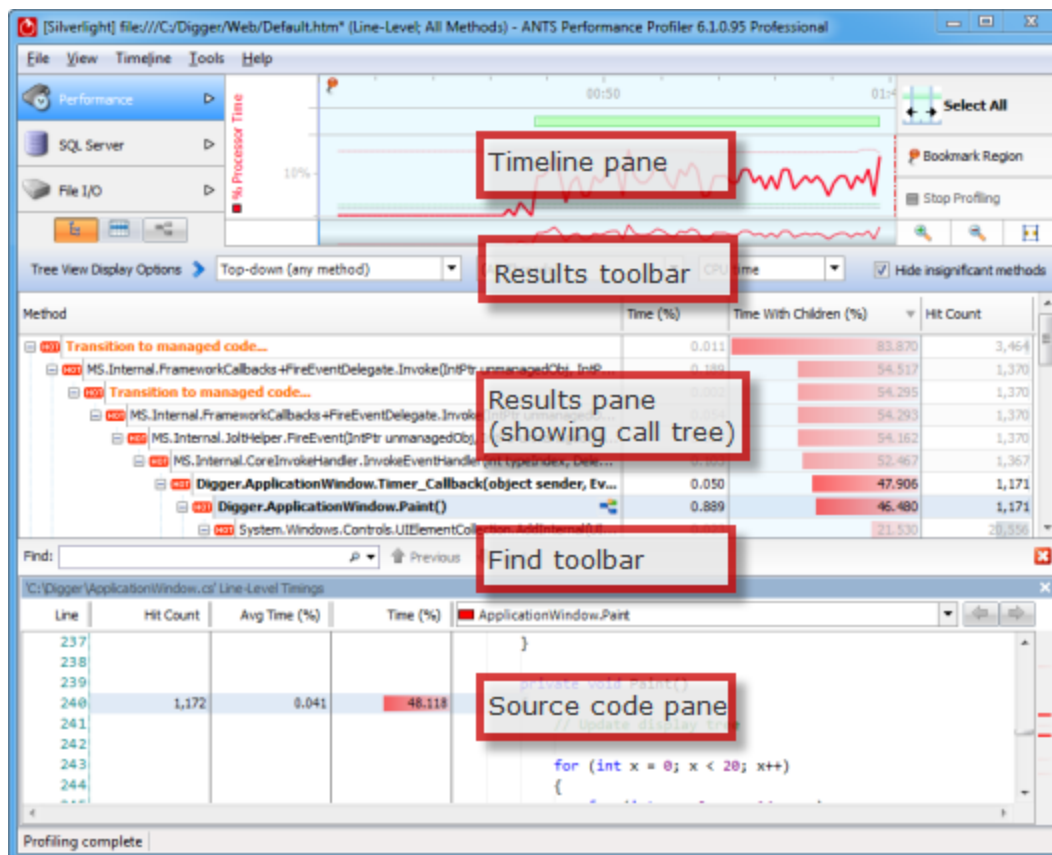
# Working with profiling results

Once you have run a profiling session and displayed some profiling results you can start analyzing the results in the results pane using the three main display types: call tree, methods grid, and call graph.


ANTS Performance Profiler 7.2 and later:



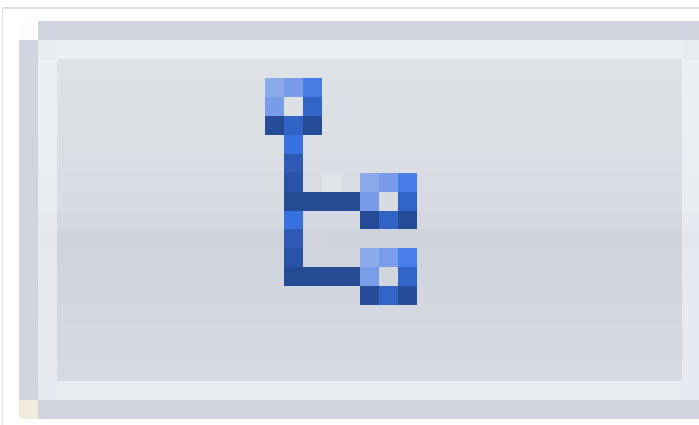
ANTS Performance Profiler 7.0:



Profiling results are shown:

- After you click the  **Stop Profiling** button.
- During profiling, when you select a period on the timeline; see [Working with the timeline](#).

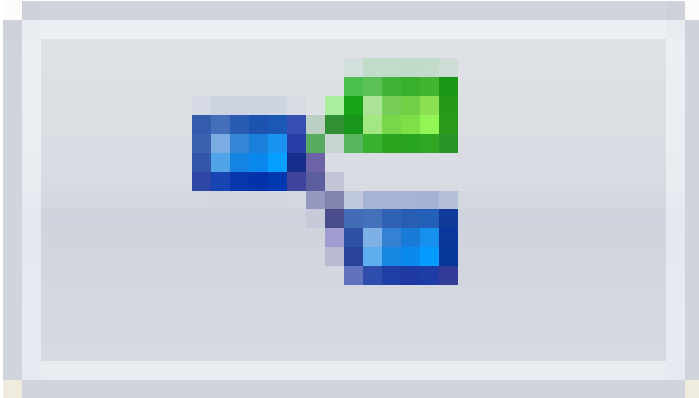
Initially, the profiling results are shown on the call tree. Use the dropdown menu in the results toolbar to switch between display types (or the buttons on the timeline pane in ANTS Performance Profiler 7.0):



**Call tree:** shows stack traces that were executed by your application during the time period you have selected.



**Methods grid:** lists each method that was executed by your application during the time period you have selected.



**Call graph:** shows the calling relationships between methods executed by your application, for the time period you have selected.  
(The call-graph option is disabled until you have [created a new call graph](#).)

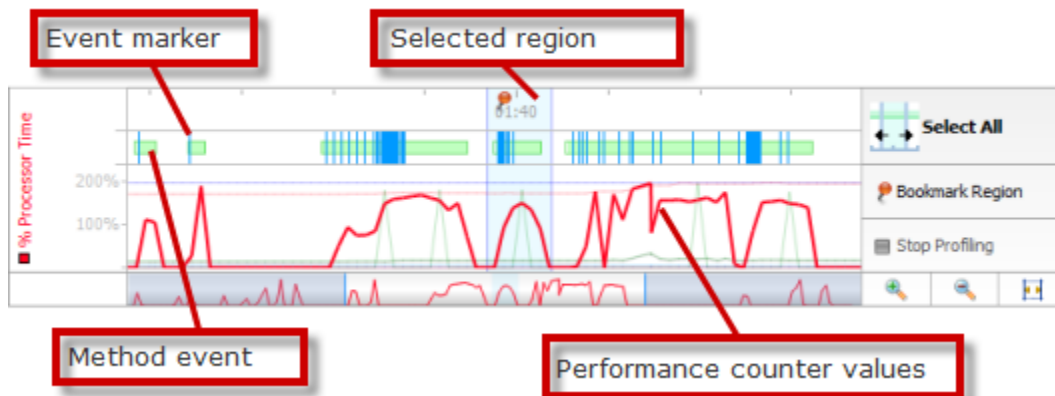
## Working with the timeline

The timeline is visible throughout a profiling session, and provides a frequently updated display of performance-counter values and instances of events related to the application you are profiling. You can use this overview of application activity to isolate performance-profiling results for specific time periods.

The timeline enables you to select a region (corresponding to a time period during execution of your application) for which you wish to display profiling results. You can select and reselect any region as often as you need to, both during profiling and after you have stopped profiling and closed your application. You can also create bookmarks for selected regions, enabling you to define multiple regions and switch between them to look at data for different periods during a profiling session.

The main section of the timeline shows the values for a selection of Windows performance counters. You can choose which performance counters to display before you start profiling your application. See [Setting up Charting Options](#) for more information.

The event bar on the timeline shows event markers. These indicate when certain types of event occur within your application, for example, button clicks, window activations, and exceptions. When you move the mouse pointer over an event marker a tooltip provides more information about the event. See [Working with event markers and method events](#) for more information.



## Working with regions on the timeline

You can select, clear, and bookmark regions on the timeline. Whenever you select a region, profiling results are displayed that relate to the selected period only.

### Selecting a region

To select a region, click and drag the mouse pointer



across the timeline. The results pane (beneath the timeline) updates to show profiling results for the selected region.

### Resetting a region

To reset the currently selected region to cover the whole timeline, click **Select All**. The results pane updates to show profiling results for the entire time period for which your application was running.

### Bookmarking a region

You can create a bookmark on the timeline, for a selected region. This is useful if there are several periods for which you want to view or compare profiling results: you can easily switch between bookmarked regions to redisplay profiling results.

To bookmark a selected region, click



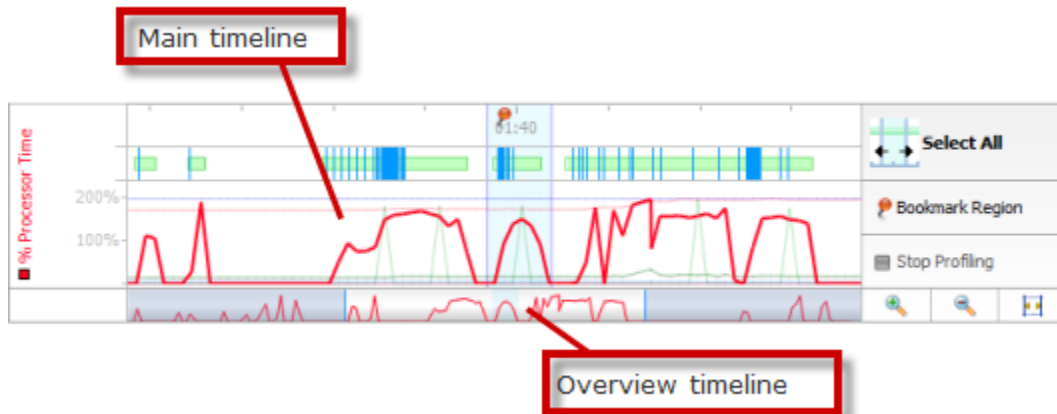
within the selected region. This region is now bookmarked (this is indicated by a highlighted bar at the top of the timeline). To select this region again later, click within the highlighted bar.

You can name a bookmark to make it easier to identify. To name a bookmark, click the highlighted bar for the bookmark and click



. The name you type will be shown on the bookmark's tooltip.

To delete a bookmark, click the highlighted bar for the bookmark and click



## Adjusting the time scale

You can change the time scale to view performance-counter data in more or less detail by rotating the mouse wheel, or by using the zoom-control buttons (zoom in



, zoom out



, and zoom to fit



). You can also use the following keyboard shortcuts: CTRL+PLUS to zoom in; CTRL+MINUS to zoom out.

To pan the main timeline, move the mouse pointer over the highlighted area in the overview timeline (



), and drag to the left or right.

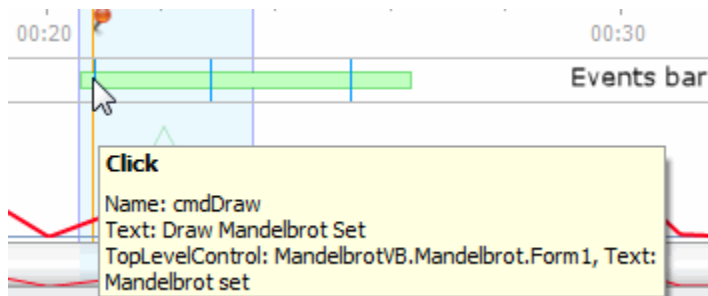
## Working with performance counters

The performance counters available for the current profiling session are listed to the left of the timeline. You can choose which performance counters to display when you set up a new profiling session (see [Setting up Charting Options](#)).

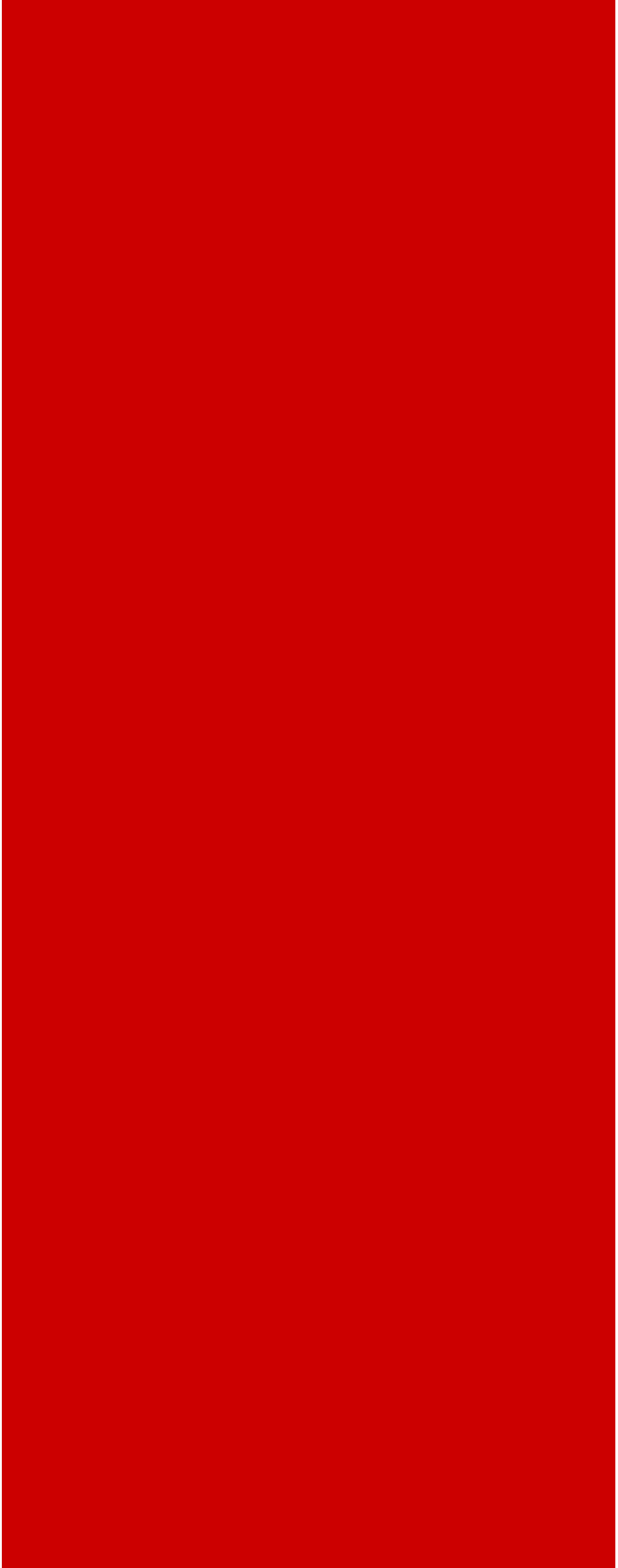
To highlight a particular performance counter on the timeline, click its description in the **Performance Counters** list. Values for the selected performance counter are shown on a tooltip when you move your mouse pointer over the main timeline.

## Working with event markers and method events

The events bar (directly above the main timeline) indicates when certain events occurred within the application you are profiling. To display more information about the event, move your mouse pointer over an event marker.



Events are displayed using colored markers:

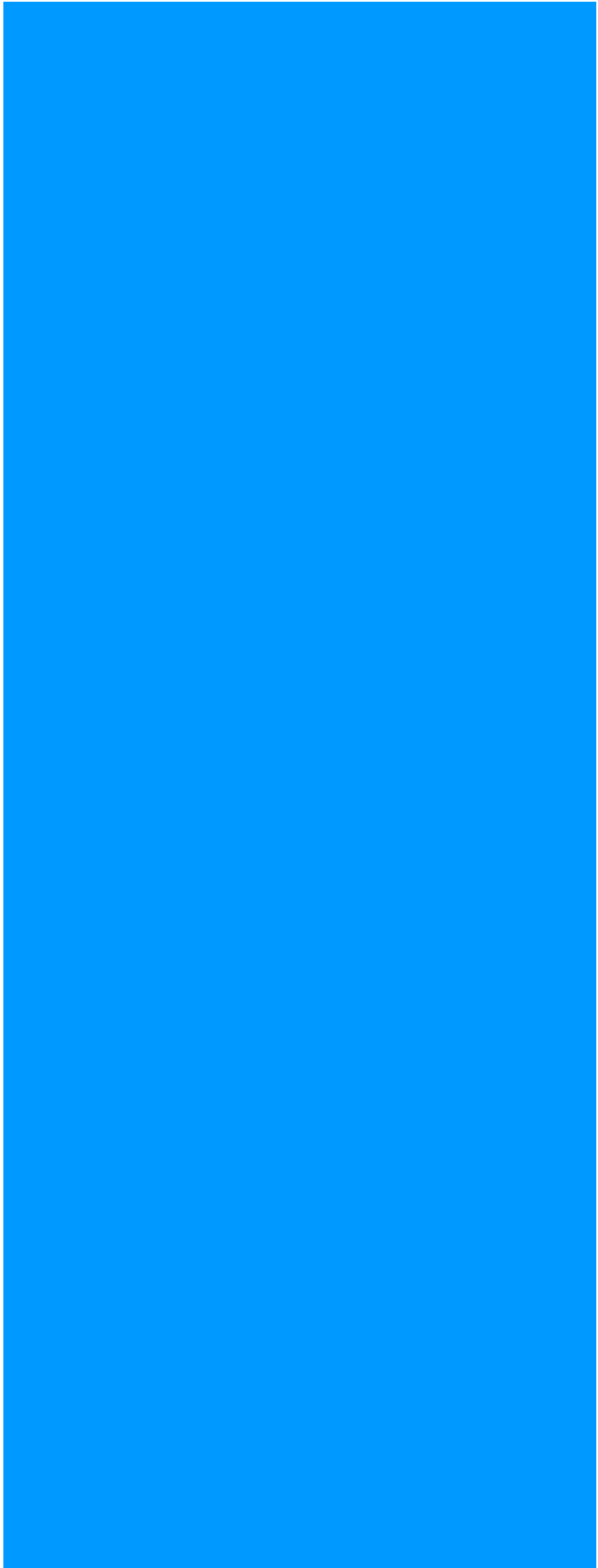
Event type	Color
Exception	

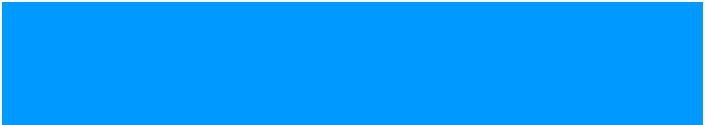


Click

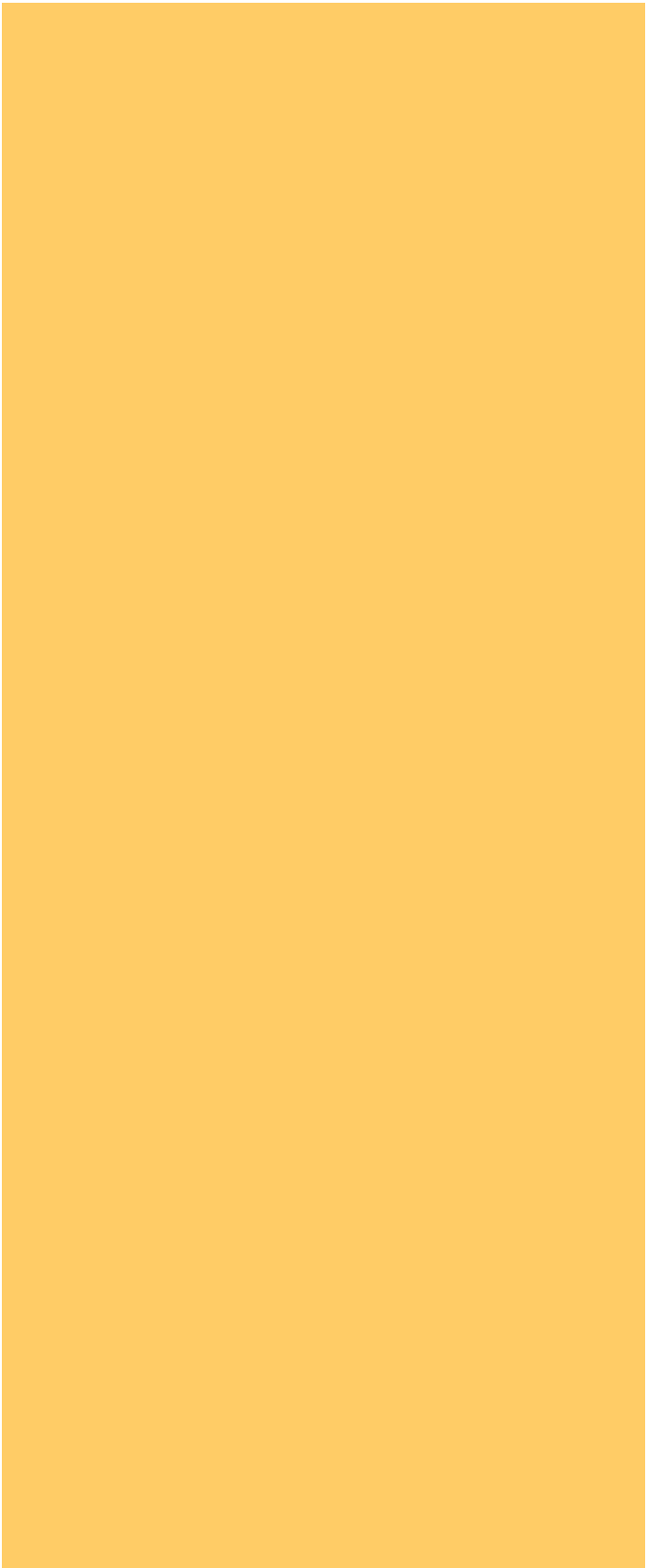






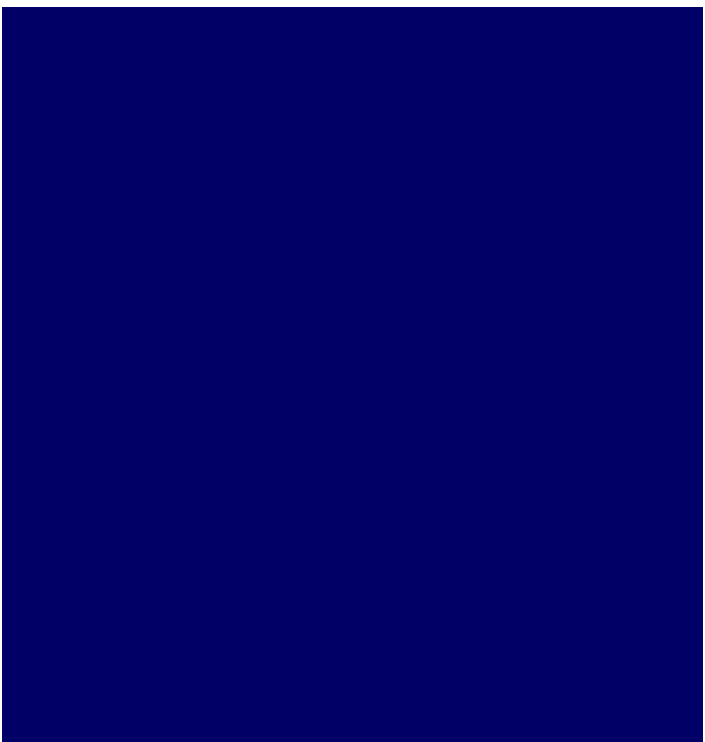


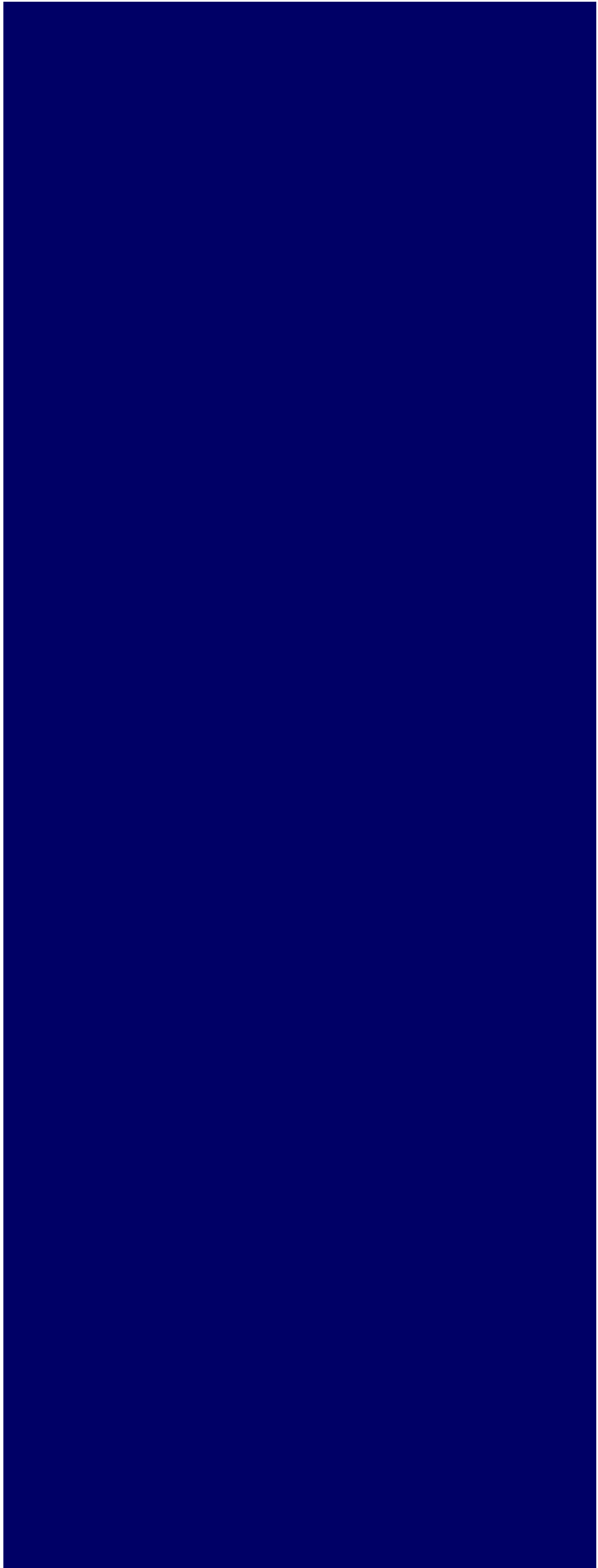
Window activated and window closed





All other events







To see which method threw a given exception, click on the exception marker: the method will be selected in the call tree or method grid below, and its source, if available, displayed in the source code view. (ANTS Performance Profiler 7.3 and later).

If you click on a method in the call tree or on the methods grid (and when you display the call graph for a method), the times when that method executed are shown as light green areas in the events bar. These areas are called 'method events'. If the method called unmanaged code, the period when the unmanaged code ran is shown as dark green.

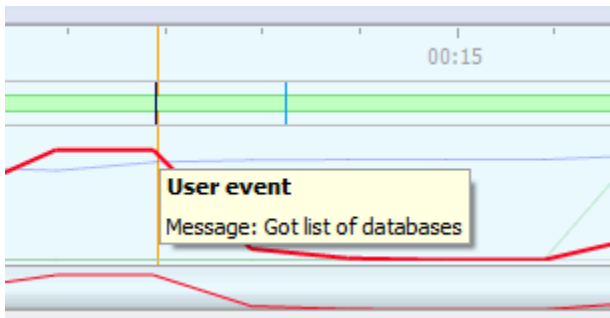
## Adding custom event markers

It is possible to add your own event markers to the timeline. For example, you might want to know when a certain item is loaded into memory or when an ASP.NET page is loaded or unloaded. To add a custom marker:

1. Copy *%ProgramFiles%/Red Gate/ANTS Performance Profiler 7/RedGate.Profiler.UserEvents.dll* to your application's *bin* folder. (Create the *bin* folder using Windows Explorer if it does not already exist.)
2. Reference the DLL from your application.
3. Add the following line to your application wherever you want to see an event marker on the profiler's timeline. (Replace 'The message' with a short description of the event.)

```
RedGate.Profiler.UserEvents.ProfilerEvent.SignalEvent("The message");
```

4. When you profile your application, the event is shown in the ANTS Performance Profiler timeline with a black event marker. Move your mouse over the event marker to display the message.



Note that you cannot use custom event markers when profiling Silverlight applications.

## Working with the call tree

The call tree shows the stack traces that were executed by your application during the time period you have selected. By default, stack traces are displayed top-down (calling method above called method). The "hottest" stack trace (the one that took the most time to run) is displayed at the top of the call tree, and is automatically expanded. If a method was called in several contexts, it is displayed once for each context in the call tree.

See [Tips on using the call tree](#) for more information on how to use the call tree effectively.

Method	Time (%)	Time With Children (%)	Hit Count
Mandelbrot.Form1.Main() *	<0.001	93.836	1
(Collapsed methods without source, such as framework class library methods)	38.849	56.793	721,897
Mandelbrot.Form1.cmdDraw_Click(object sender, EventArgs e)	0.001	20.491	2
Mandelbrot.Form1.DrawMandelbrot()	0.122	20.287	2
Mandelbrot.Image.SetPixelColor(int i, int j, int iterations)	0.174	15.510	230,208
(Collapsed methods without source, such as framework class library methods)	14.839	14.839	4,301,442
Mandelbrot.ColorScheme.ColorFromIterations(int iterations)	0.129	0.497	230,208
(Collapsed methods without source, such as framework class library methods)	4.020	4.020	126,546
Mandelbrot.Algorithm.Evaluate(double x, double y)	0.124	0.589	230,208
Mandelbrot.Image.get_Height()	0.046	0.046	231,082
Mandelbrot.Settings.get_YMin()	<0.001	<0.001	874
Mandelbrot.Image.get_Width()	<0.001	<0.001	876

## Reading the call tree

The following data is shown for each method within the stack trace, for the selected time period:

- **Time:** the total execution time for the method within this stack trace.
- **Time With Children:** the total execution time for the method and all its children within this stack trace.
- **Hit Count:** the number of times the call was made within this stack trace.

When time is shown as a percentage, the **Time (%)** for each method shows the proportion of the total execution time that the method contributed during the selected period. The total percentage for all methods can sum to over 100 on machines using multiple CPU cores.

## SQL calls (ANTS Performance Profiler 7.2 and later)

In ANTS Performance Profiler Professional edition, SQL calls made by the application are also shown within the stack trace, with the following information:

- **Time:** The time until the first result for the query was returned.
- **Time With Children:** The total execution time for the call and all its children within this stack trace.
- **Hit Count:** The number of times the query was called within this stack trace.

## HTTP calls (ANTS Performance Profiler 7.3 and later)

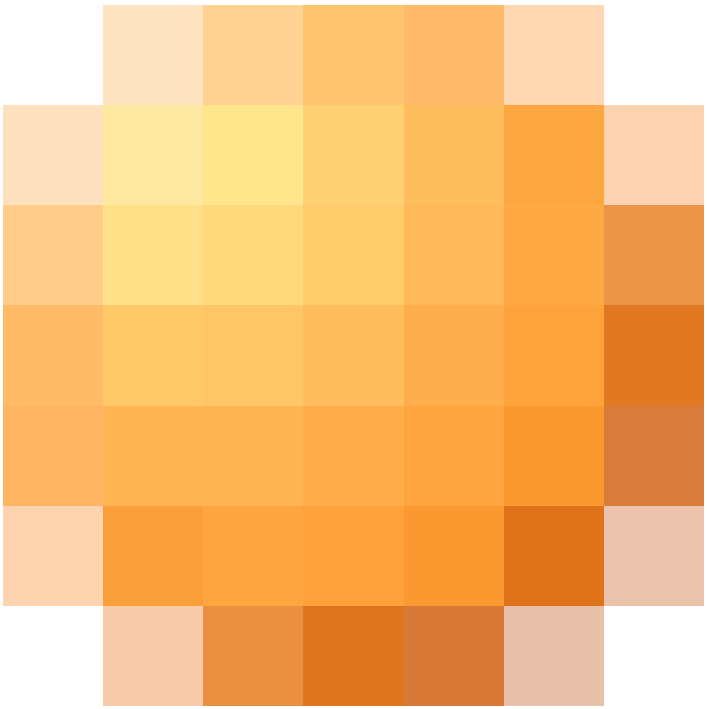
In ANTS Performance Profiler Professional edition, if your application is an ASP.NET site, you will also see incoming HTTP requests in the call tree, with the following information:

- **Time:** The time taken to execute the .NET helper method(s) that ran the request.
- **Time With Children:** The total execution time for the request and all the .NET methods executed because it was made.
- **Hit Count:** The number of times the request was called within this stack trace.

To collapse all nodes except HTTP requests, right-click anywhere in the call tree and select **Show only incoming HTTP calls**. SQL calls will also be shown.

## Call tree icons

Each tree item is shown with one or more of the following icons:



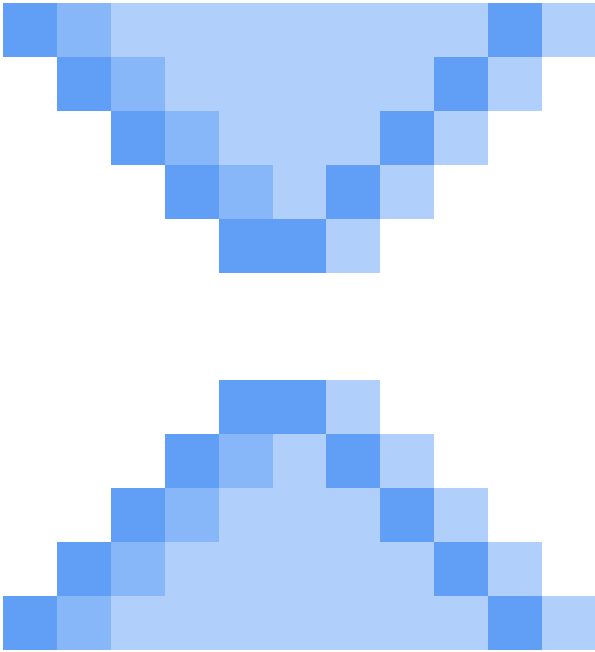
Root method or leaf method. Root methods are not called by any other method; leaf methods do not call any other method.



Indicates call flow when the call tree direction is top-down (calling method above called method).

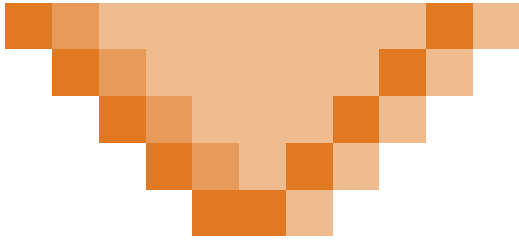


Indicates call flow when the call tree direction is bottom-up (called method above calling method).

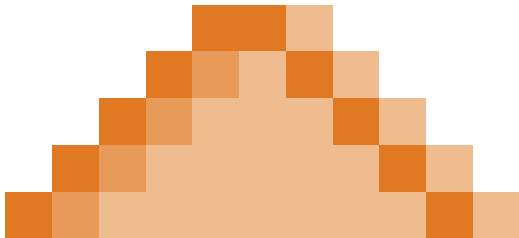


Indicates one or more collapsed methods that can be expanded.





Indicates one or more collapsed methods that cannot be expanded, because they are all leaf nodes that have been filtered out of the current view - see [Filtering the call tree](#). (ANTS Performance Profiler 7.2 and later)



Method is part of the hottest (longest running) stack trace. Used instead of the root/leaf or call-flow icons.

\*

A method name followed by an orange asterisk: Method is optimizable. (ANTS Performance Profiler 7.2 and earlier)



Indicates a SQL call generated by the parent .NET method.



Indicates an HTTP request: the .NET methods listed as children of this item were called when this HTTP request was made. (ANTS Performance Profiler 7.3 and later)

Methods listed in **bold** have source code available. To display the method's [source code](#), click any bold method. Line-level timings are also available in the source-code pane if you use one of the *Line-level ...* profiling modes.

You may also see the following items in the call tree. These are shown in bold orange text, and represent time spent in your application that is in addition to time spent executing specific methods:

- **Waiting for synchronization:** A thread was waiting. For example, if you have called the `Monitor.Wait` method, the thread will wait for synchronization until the lock is reacquired. Another cause is that the finalizer thread spends most of its time waiting. Expand the [call graph](#) to see what caused the wait. Items that are *Waiting for synchronization* only contribute to timings in the call tree when the **Timing** display option is set to *Wall-clock time*. To exclude time due to *Waiting for synchronization* items, select *CPU time* from the **Timing** display option.
- **Thread blocked:** The executing thread was blocked. For example, the thread may have been sleeping, or waiting for access to a shared resource. *Thread blocked* items only contribute to timings in the call tree when the **Timing** display option is set to *Wall-clock time*. To exclude time due to *Thread blocked* items, select *CPU time* from the **Timing** display option. The call-tree display options are described below.
- **Waiting for I/O to complete:** The executing thread was blocked waiting for file I/O. *Waiting for I/O to complete* items only contribute to timings in the call tree when the **Timing** display option is set to *Wall-clock time*. To exclude time due to *Waiting for I/O to complete* items, select *CPU time* from the **Timing** display option. The call-tree display options are described below.
- **Transition to unmanaged code:** A transition from managed code to unmanaged code occurred at this point in the stack trace. In general, line-level and method-level timings are not available for the unmanaged code. However, for unmanaged methods that are declared with `extern` within managed code, method-level timings are available.
- **Transition to managed code:** A transition from unmanaged code to managed code occurred at this point in the stack trace.
- **JIT overhead:** JIT compilation occurred at this point in the stack trace during execution of your application. The method that needed to perform the compilation is shown as the parent of a *JIT overhead* item.
- **Profiler overhead:** Additional overhead introduced by ANTS Performance Profiler. This is unlikely to be seen when the option to [adjust timings to compensate for overhead added by the profiler](#) is enabled.
- **Assembly load or unload:** A .NET assembly was loaded or unloaded.
- **Module load or unload:** A .NET module was loaded or unloaded.

To create a new [call graph](#) based on a particular method, select the method in the call tree, and click the new call graph button



in the **Method** column.

To switch to view a SQL query in [Database Calls view](#), select the method in the call tree, and click the



button in the **Method** column. (ANTS Performance Profiler 7.2 and later)

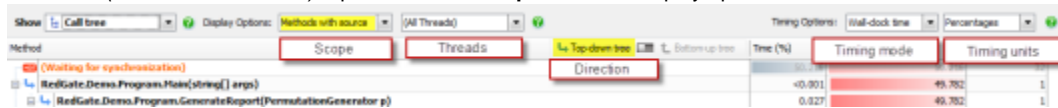
## Tips on using the call tree

To locate methods that may be good candidates for optimization:

1. Order the call tree with the slowest stack traces at the top (top-down). If necessary, click the **Time With Children** column heading to change the stack-trace order.
2. Starting with the slowest stack traces, look for method pairs where subsequent values for **Time With Children** reduce substantially as you move down the stack trace. Methods with higher values in such pairs may be good candidates for optimization. ANTS Performance Profiler can optionally suggest methods that may be good candidates for optimization. To show suggested methods, on the **Tools** menu, click **Suggest methods to optimize**. Suggested method names are marked with an asterisk (\*). In general, the better you understand the structure and meaning of your code, the more easily you will be able to interpret the data collected by the profiler.

To reduce the number of methods shown, you can do either of the following:

1. Choose a "(methods with source)" option from the **Scope** list in the display options.



2. Select a shorter region on the timeline.

To find a particular method:

1. On the **Tools** menu, click **Find**.  
The **Find** bar is displayed beneath the call tree.
2. Type all or part of the method name you are looking for, and press ENTER.  
The first matching row in the call tree is highlighted.

Click



**Previous** or



**Next** to move between matching method names.

## Waiting for synchronization

The Call Tree may display '(Waiting for synchronization)'. This means that a thread is waiting for another thread to finish.

If '(Waiting for synchronization)' is the only item in the Call Tree, it is likely that you are viewing the garbage collector thread only. To see all results, select **Top-down (Any method)**, **(All threads)**, **Wall-clock time** and **Hide insignificant methods**.

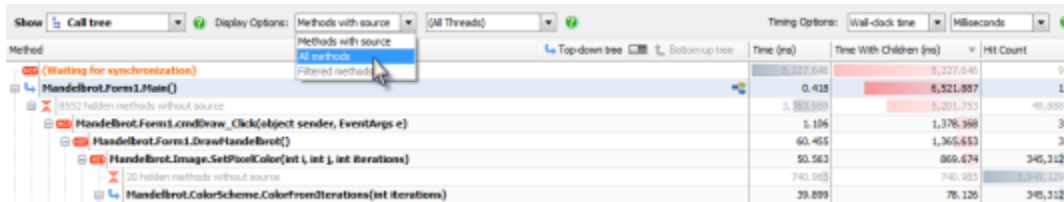
## Working with integrated decompilation

ANTS Performance Profiler lets you decompile methods without source directly from the [call tree](#). Decompiling a method generates line-level code from any .NET assembly. This allows you to identify the causes of performance problems in previously inaccessible code for example, third-party components or legacy assemblies called by your application.

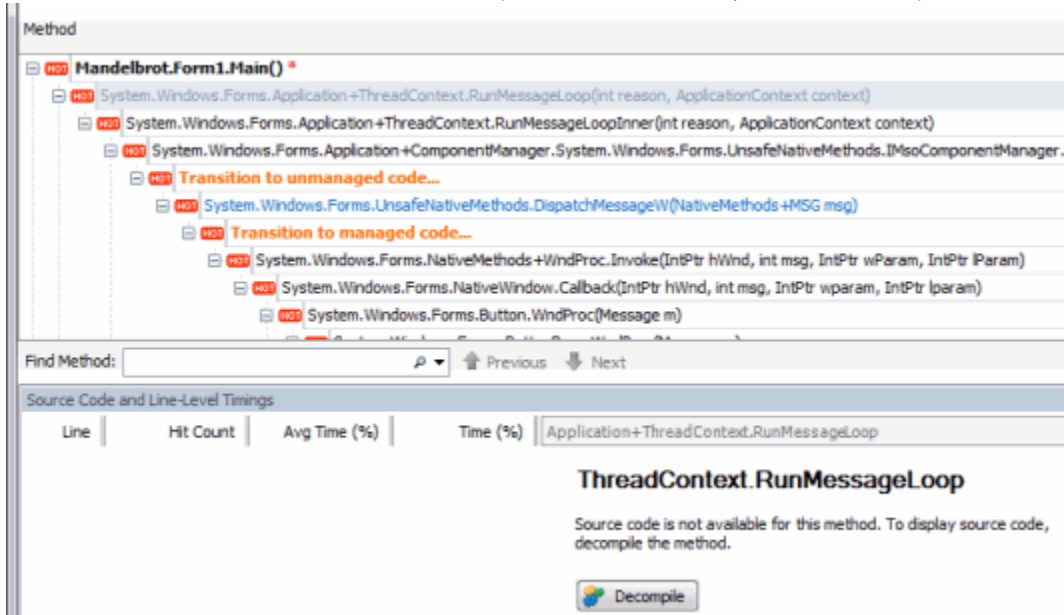
Integrated decompilation is powered by .NET Reflector; however, you do not need to have .NET Reflector installed to use integrated decompilation.

### To decompile a method from an assembly with no PDB in ANTS Performance Profiler 7.2 and later:

1. In the **Results toolbar**, directly above the call tree, select **All methods** to view methods without source.



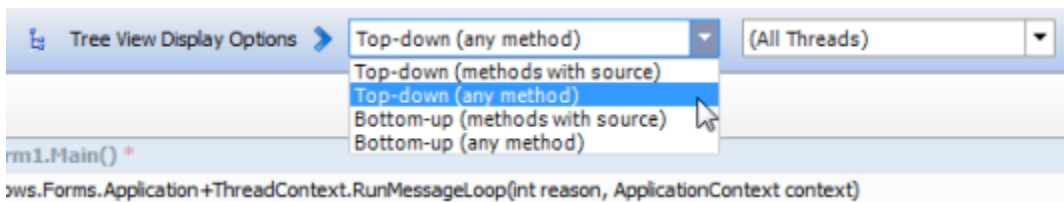
2. In the call tree, select a method without source code (shown in the call tree in plain, unbolded text).



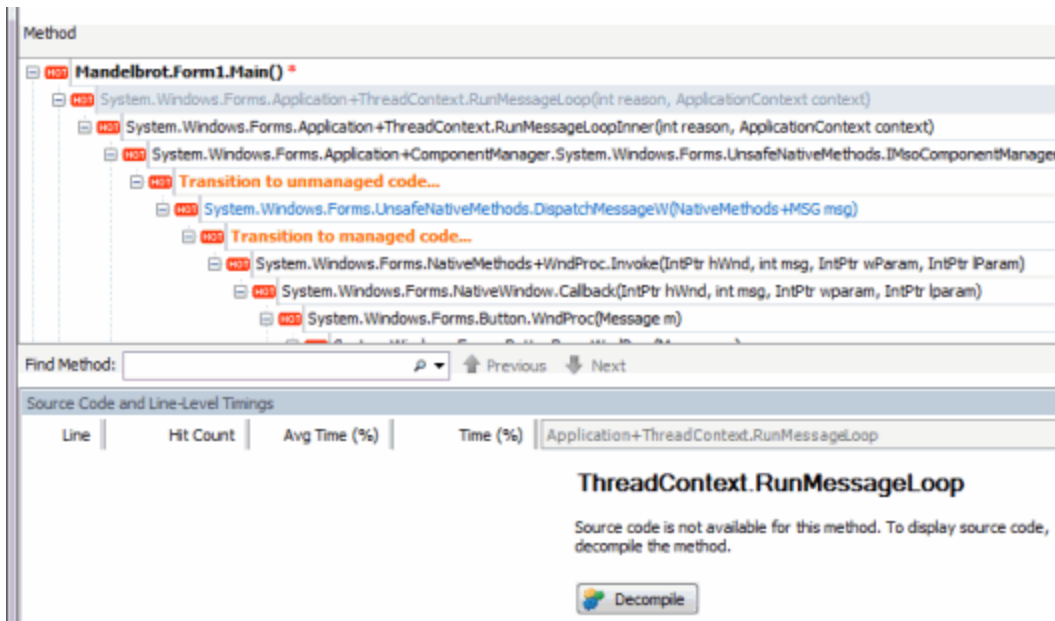
3. Select **Decompile** to generate source code for the selected method.  
ANTS Performance Profiler now decompiles the selected method, and displays automatically-generated source code for the selected method in the source-code pane. Without a PDB file, line-level timings are unavailable, but method-level timings are shown. For more information on navigating your decompiled source, see [Working with source code](#).

### To decompile a method from an assembly with no PDB in ANTS Performance Profiler 7.0:

1. In the **Tree View Display Options** menu, directly above the call tree, select **Top-down (any method)** to view methods without source.



2. In the call tree, select a method without source code (shown in the call tree in plain, unbolded text).



3. Select **Decompile** to generate source code for the selected method. ANTS Performance Profiler now decompiles the selected method, and displays automatically-generated source code for the selected method in the source-code pane. Without a PDB file, line-level timings are unavailable, but method-level timings are shown. For more information on navigating your decompiled source, see [Working with source code](#).

## To decompile a method from an assembly with a PDB:

1. In the call tree, select a method from an assembly for which you have a PDB but no source code (shown in the call tree in plain, unbolded text).
2. Click **Decompile** to generate source code for the entire assembly. ANTS Performance Profiler now decompiles the selected assembly or method, and displays automatically-generated source code for the selected method in the source-code pane. If you chose a profiling mode with line-level timings, ANTS Performance Profiler will match these timings to the decompiled code, though the timings may be inexact. For more information on navigating your decompiled source, see [Working with source code](#).

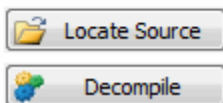
Methods will also display without source if you have the assembly's source code but ANTS Performance Profiler cannot automatically locate it. To manually specify a path to the source code for an assembly, select **Locate source**.

### Mandelbrot.exe

C:\Program Files\Red Gate\ANTS Performance Profiler  
6\Tutorials\CS\Mandelbrot\Algorithm.cs

Source code was not found for this file.

You can locate the source manually, or decompile source code from the assembly.



## Decompiling Silverlight applications

Code from Silverlight browser applications embedded using Javascript cannot be decompiled.

Silverlight browser applications embedded using HTML can be decompiled if ANTS Performance Profiler can locate the associated XAP file. For information on configuring your Silverlight application so that ANTS Performance Profiler can find the XAP file, see [How to: Add Silverlight to a Web Page by using HTML \(MSDN\)](#).

If ANTS Performance Profiler cannot find the XAP file automatically, the message "Could not identify a file path" will be displayed.

To manually specify the path to the XAP file, on the **Application Settings** dialog, perform the following steps:

1. Under **Choose application type to profile**, click **Silverlight 4+ application**.
2. In the Silverlight application URL field, enter the path to the XAP file (rather than to the website).

Code from Silverlight out-of-browser applications can also be decompiled if the profiled application is configured to work as an out-of-browser application.

## Decompiling obfuscated code

Some obfuscated code cannot be decompiled, or can be decompiled only partially. Partially obfuscated code, when decompiled, may display mangled method names and absent lines of code.

If an obfuscated method cannot be decompiled, the method's contents are replaced with the comment "This code is obfuscated and can not be translated" in the source code pane.

## Saving profiling results

When saving profiling results, any decompiled code generated during the profiling session is not saved. On opening a saved session, to view source for methods from third-party assemblies, follow the decompilation steps above.

## Working with the methods grid

The methods grid lists each method that was called by your application during the [time period you have selected](#). Even if a given method is called in several contexts, it is shown only once in the methods grid, with aggregated data that accounts for all contexts. You can order the data by any column by clicking the column heading. Data is ordered by **Time With Children** by default.

Namespace	Method Name	Time (%)	Time With Children (%) ▾	Hit Count	Source File
Mandelbrot	<b>Form1.Main()</b>	0.000	94.495	1	Form1.cs
Mandelbrot	<b>Form1..ctor()</b>	0.003	40.333	1	Form1.cs
Mandelbrot	<b>Form1.cmdDraw_Click(object sender, EventArgs e)</b>	0.001	21.003	2	Form1.cs
Mandelbrot	<b>Form1.DrawMandelbrot()</b>	0.082	20.746	2	Form1.cs
Mandelbrot	<b>Image.SetPixelColor(int i, int j, int iterations)</b>	0.168	16.267	230,208	Image.cs
Mandelbrot	<b>Form1.InitializeComponent()</b>	0.007	7.750	1	Form1.cs
Mandelbrot	<b>Form1.Dispose(bool disposing)</b>	0.000	1.520	1	Form1.cs
Mandelbrot	<b>Algorithm.EvaluateUsingDoubles(double x, double y)</b>	0.393	0.393	115,104	Algorithm.cs
Mandelbrot	<b>ColorScheme.ColorFromIterations(int iterations)</b>	0.107	0.308	230,208	ColorScheme.cs
Mandelbrot	<b>Image..ctor(int width, int height)</b>	0.001	0.116	1	Image.cs
Mandelbrot	<b>Algorithm.Evaluate(double x, double y)</b>	0.117	0.113	230,208	Algorithm.cs
Mandelbrot	<b>Image.get_Height()</b>	0.032	0.032	231,082	Image.cs

The following data is shown for each method, for the time period you have selected:

- **Time**: the total execution time for the method (in all contexts).
- **Time With Children**: the total execution time for the method and all its children.
- **Hit Count**: the number of times the method was called.

Methods listed in **bold** have source code available. To display the method's [source code](#), click any bold method. Line-level timings are also available in the source-code pane if you used the *Line-level and method-level timings; all methods* or *Line-level and method-level timings; only methods with source profiling mode*.

## Changing the methods-grid display options

You can change the way data is displayed in the methods grid, using the display options on the results toolbar.

Namespace	Method Name	Scope	Threads	With Children (%)	Hit Count	Timing mode	Timing units
133 hidden methods without source				99.077	7,857,191		
RedGate.Demo	<b>Program.GenerateReport(PermutationGener...</b>			49.782	1		
RedGate.Demo	<b>Program.Main(string[] args)</b>			49.782	1		

✓ IN ANTS Performance Profiler 7.4...

- **Scope**: controls whether any method, or only methods with source, are shown.  
If you choose to display any method, the methods grid will include details for .NET Framework class-library methods.
- **Threads**: filters the display of stack traces by thread.
- **Timing mode**: controls the way in which method timings are calculated.  
You can choose from **Wall-clock time** which includes blocking such as waiting for I/O, or **CPU time** which excludes blocking.
- **Timing units**: you can choose whether to display time in **Percentages**, **Ticks**, **Milliseconds**, or **Seconds**.  
Choosing **Percentages** displays the time the method took as a percentage of the period currently selected on the timeline.

Namespace	Method Name	Scope	Time (%)	Threads	Children	Timing	Source File
MandelbrotVB.Ha...	<b>Form1.Main()</b>		0.000		100.000		Form1.vb
MandelbrotVB.Ha...	<b>Form1.cmdDraw_Click(object sender, Eve...</b>		0.084		87.513		Form1.vb
MandelbrotVB.Ha...	<b>Form1.DrawMandelbrot()</b>		2.922		85.412		Form1.vb

✓ In ANTS Performance Profiler 7.3 and earlier...

- **Scope**: controls whether any method, or only methods with source, are shown.  
If you choose to display any method, the methods grid will include details for .NET Framework class-library methods.
- **Threads**: filters the display of stack traces by thread.
- **Timing**: controls the way in which method timings are calculated.  
You can choose from **Wall-clock time** which includes blocking such as waiting for I/O, or **CPU time** which excludes blocking.
- **Hide insignificant methods**: select this check box to hide methods that contribute less than 1% of the total execution time for the currently selected time period.

## Tips on using the methods grid

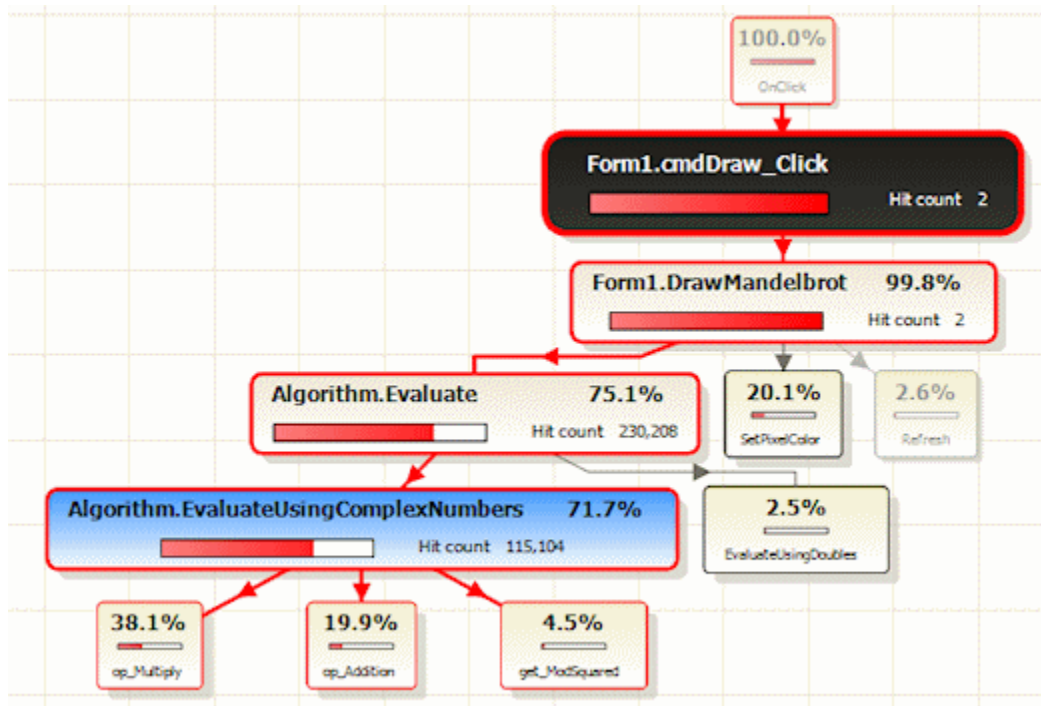
To find particular methods:

1. On the **Tools** menu, click **Find**.  
The **Find** bar is displayed beneath the methods grid.
2. Type all or part of the method name you are looking for.  
As you type, the methods are filtered to display only those that match your text.



## Working with the call graph

The call graph shows the calling relationships between methods during the execution of your application, and is focused on a method of your choice (the *chosen* method; shown in black in the example below). If a given method is called in several contexts, it is shown once for each context in the call graph. The chosen method is shown only once in the call graph, unless it is called recursively.



Selecting a chosen method makes it easy for you to visualize all the callers and callees for that method.

The percentage value shown in each method is calculated with respect to the chosen method as follows:

- For a method called by the chosen method, this is the percentage of the chosen method's execution time that the method accounts for, relative to the chosen method's total execution time.
- For a method that calls the chosen method, this is the percentage of the chosen method's total execution time that is due to the calling method.

Calculations are always made with respect to the selected region on the timeline, or the whole profiling period if you have not selected a region.

### Creating a new call graph

Every instance of a call graph is based on a particular method, so you must first select a method in the call tree, methods grid, or source code, then click the create new call graph button



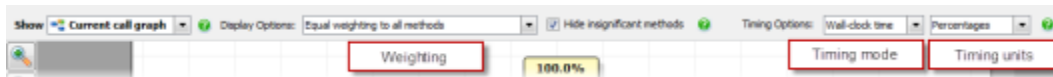
		0.007	9.358	1
library methods)		1.591	45,253	1,718
EventArgs e)		0.001	13,302	1
		0.063	13,101	1
int iterations)	Create a new call graph for this method.		9.711	115,104
work class library methods)		1.280	2,593	810
		<0.001	7,405	1

Alternatively, right-click the method and select **Create new call graph** on the short-cut menu.

The call graph is displayed in the results pane.

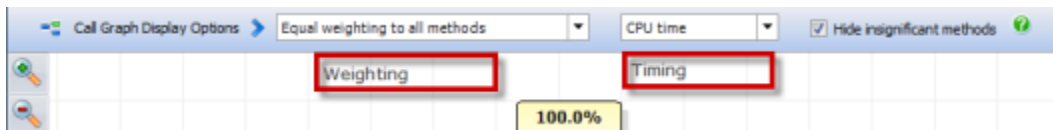
### Changing the call-graph display options

You can change the way data is displayed in the call graph, using the display options on the results toolbar:



✓ In ANTS Performance Profiler 7.4...

- **Weighting:** controls the way that methods are drawn on the call graph. *Equal weighting to all methods* is useful when you need to see how methods without source code (for example, .NET Framework library methods) affect the execution times in your application. *Emphasize methods with source* draws the call graph with much smaller boxes for those methods that do not have source code available. This allows you to concentrate on the timings for those methods for which you have the source code.
- **Timing mode:** controls the way in which method timings are calculated. You can choose from **Wall-clock time** which includes blocking such as waiting for I/O, or **CPU time** which excludes blocking.
- **Timing units:** you can choose whether to display time in **Percentages**, **Ticks**, **Milliseconds**, or **Seconds**. Choosing **Percentages** displays the time the method took as a percentage of the period currently selected on the timeline.



✓ In ANTS Performance Profiler 7.3 and earlier...

- **Weighting:** controls the way that methods are drawn on the call graph. *Equal weighting to all methods* is useful when you need to see how methods without source code (for example, .NET Framework library methods) affect the execution times in your application. *Emphasize methods with source* draws the call graph with much smaller boxes for those methods that do not have source code available. This allows you to concentrate on the timings for those methods for which you have the source code.
- **Timing:** controls the way in which method timings are calculated. You can choose from **Wall-clock time** which includes blocking such as waiting for I/O, or **CPU time** which excludes blocking.
- **Hide insignificant methods:** select this check box to hide methods that contribute less than 1% of the total execution time (for the currently selected time period).

## Navigating the call graph

You can resize the call graph by rotating the mouse wheel, or by using the zoom controls to the left of the call graph. You can pan the call graph by clicking and dragging on a blank part of the graph.

To expand a method on the call graph (that is, to show the method's immediate children or parents), click the method.

To expand the most expensive path from a particular method, hold down the CTRL key and click the method. Alternatively, right-click and select **Expand most expensive stack trace**.

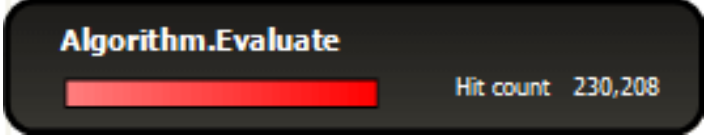
To expand the most expensive path for all children from a particular method, hold down the SHIFT key and click the method. Alternatively, right-click and select **Expand most expensive stack trace for all callees**.

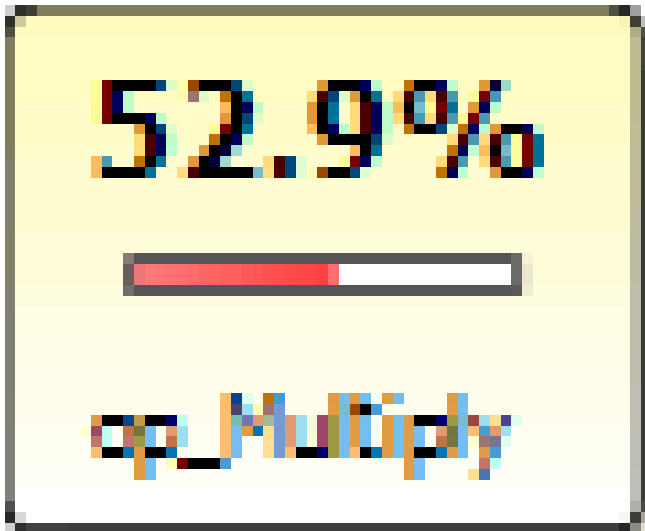
To expand the most expensive path for all parents of a particular method, hold down the SHIFT key and click the method. Alternatively, right-click and select **Expand most expensive stack trace through all callers**.

To collapse a method on the call graph, double-click the method.

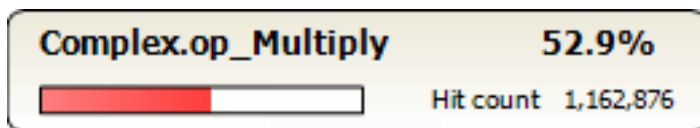
## More about call graphs

Methods are drawn in several different styles in a call graph:

 A call graph node for the method 'Algorithm.Evaluate'. It has a red progress bar and a 'Hit count' of 230,208.	Chosen method (the method you chose when creating the call graph). Execution-time percentages are calculated with respect to this method.
--	---



**(Minimized)**



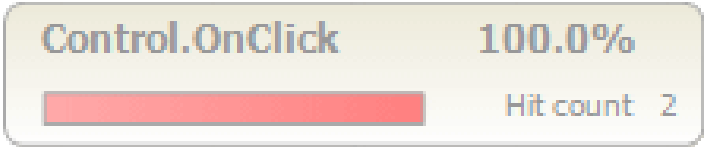
**(Maximized)**

Method with source code.



Method without source code. This style is used when all methods h  
equal weighting.

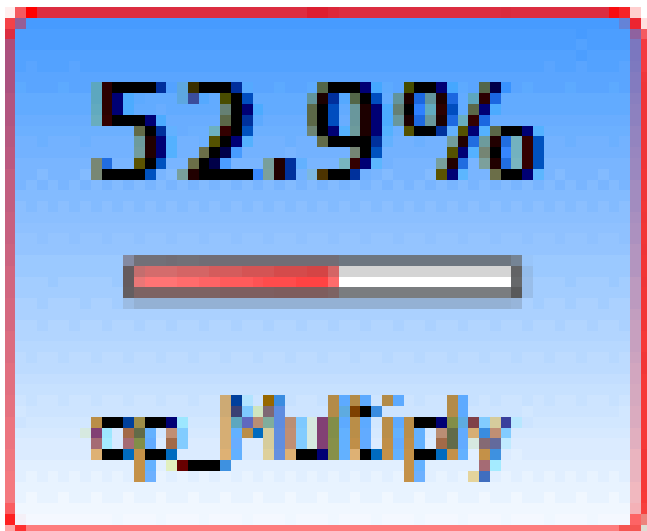
**(Minimized)**



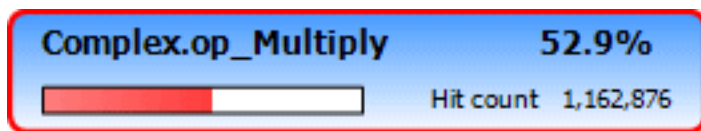
**(Maximized)**



Method without source code. This style is used when methods with  
code are emphasized.



(Minimized)

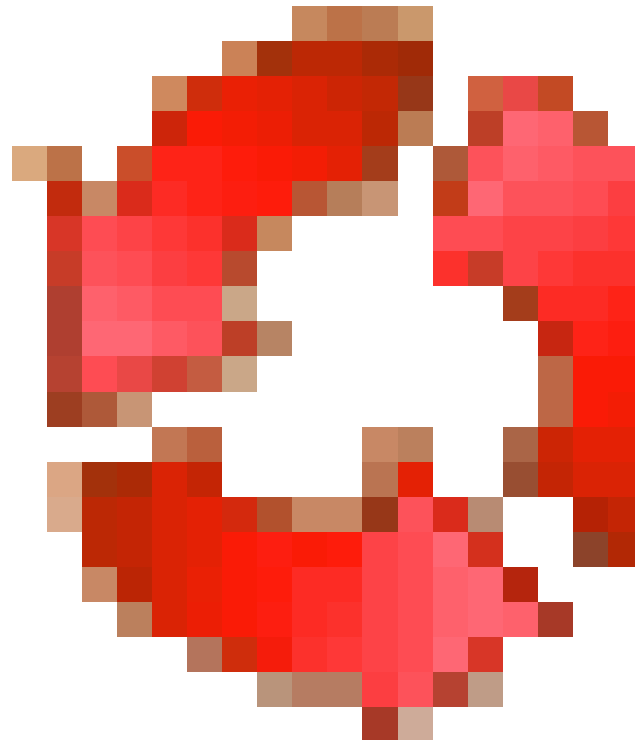


(Maximized)



Selected method. When a method is selected, *all* methods in that stack trace are also outlined in red.

Recursive method. The



symbol is added to any method that is called recursively within your application.

Call graphs always include methods for which no source code is available (for example, methods from the .NET Framework class library) and methods for all threads running in your application during profiling.

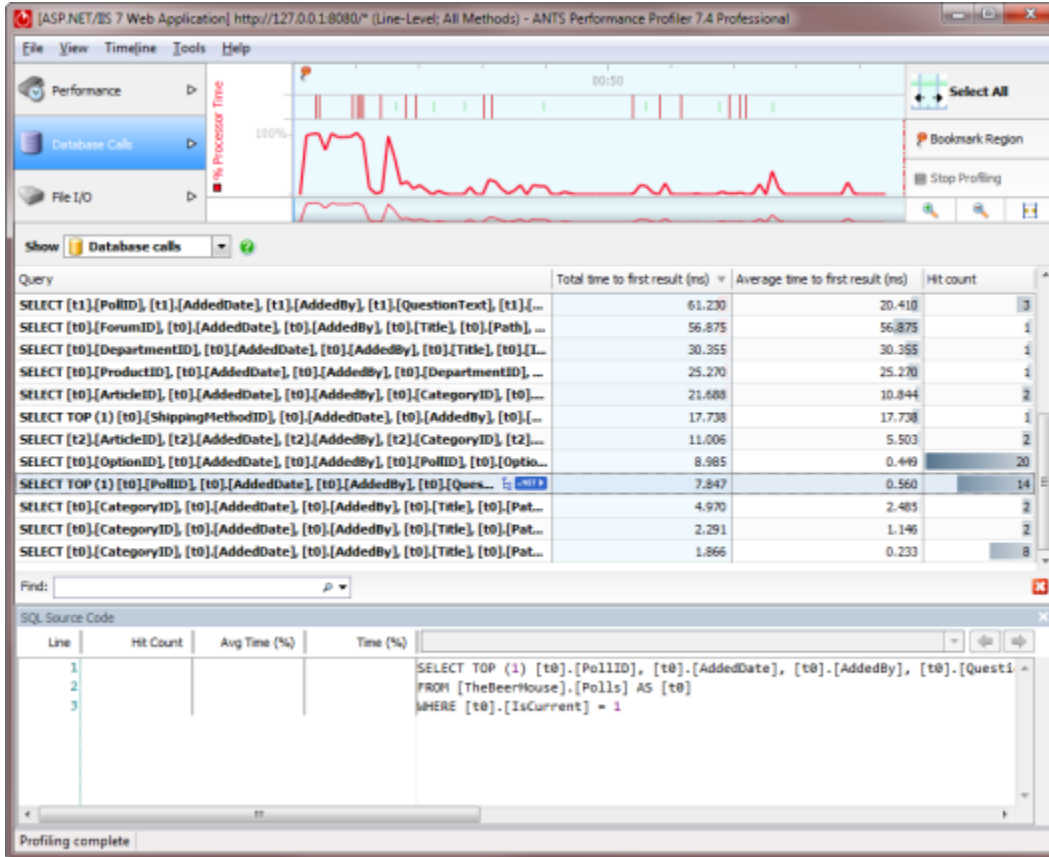
It is not possible to change the time period covered by an existing call graph. To create a call graph for a different time period, return to the call-tree or methods-grid display, reselect the required period on the timeline, and create a new call graph.

## Working with the database calls view

In ANTS Performance Profiler Professional, the **Database Calls** view shows performance data for all the database calls your application made during the time period you have selected.

If a call was made multiple times, it is still shown only once in the database calls view, with timing and hit count data aggregated from each instance of the call.

Note that database call performance data is not recorded when profiling in sampling mode.



The following data is shown for each database call, for the time period you have selected:

- **Query:** The first line of the database query (excluding any initial blank lines). To view the full query text, move the mouse over the row.
- **Total time to first result (ms):** The time taken (in milliseconds) until the query returned its first result.  
Where a call was made multiple times, **Total time to first result** shows the sum of the time to return a first result taken by all instances of the call.
- **Average time to first result (ms):** The time taken (in milliseconds) until the query returned its first result. (ANTS Performance Profiler 7.4 only)  
Where a call was made multiple times, **Average time to first result** shows the average time taken for each instance of the call to return the first result.
- **Hit count:** The number of times the call was made.

When a query is selected, a green highlight on the timeline shows the period when the call was running. Source code for the selected query is shown below the list of queries, in **SQL Source View**.

## Linking back to your code

To find out which of your code's methods ran a particular SQL query:

✓ In ANTS Performance Profiler 7.3 and later...

1. In Database Calls view, select the query.

A



icon appears on the right-hand side of the **Query** column.

2. Click on the



icon.

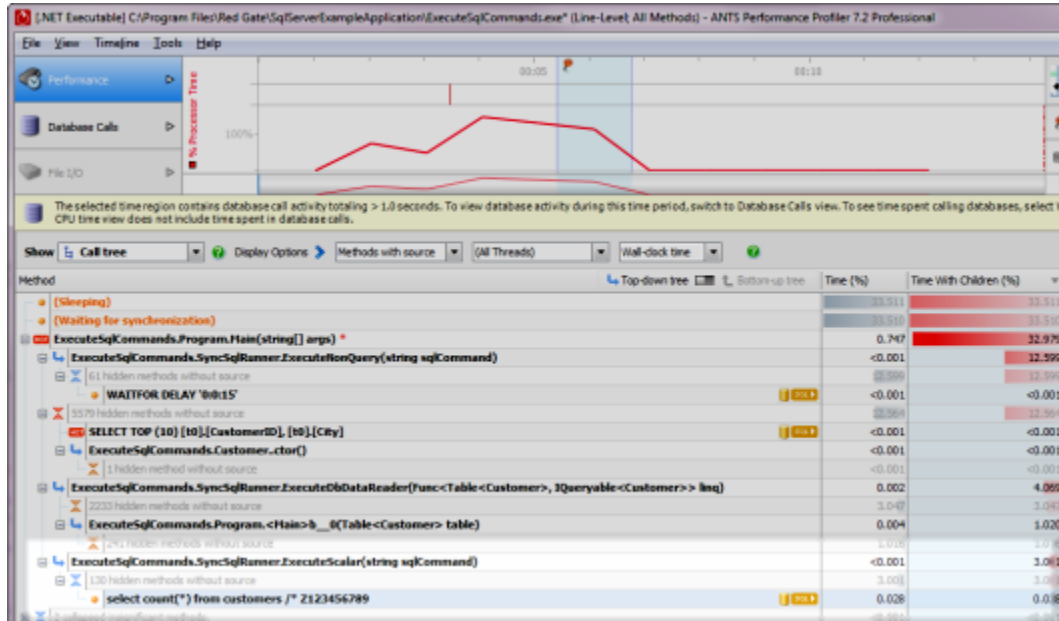
The display switches to **Performance** view, displaying the [call tree](#), with the SQL query selected.

You can scroll up the call tree to locate the query's .NET parent, and browse its timings to determine whether the problem lies in the way the query is called.

For more information on understanding .NET method timings, see [Working with the call tree](#).

▼ In ANTS Performance Profiler 7.2 and earlier...

1. In Database Calls view, select the time when the query ran on the timeline. Include some time just before the SQL query ran, and the server load increased: this will ensure your selected range includes the time when the method that runs the query was called.
2. Switch back to **Performance** view.



3. When the **call tree** is displayed, the selected SQL query is shown as a child of the .NET method that originated it. Browse the timings for the parent .NET code to determine whether the problem lies in the way the query is called.



## Working with source code

When you select a method in the results pane and source code is available for that method, the code is displayed in the source-code pane with the first line of the method body highlighted.

With ANTS Performance Profiler Professional, you can also decompile methods from third-party and framework assemblies directly from the call tree, and profile and navigate the resulting code in the same way as the original source code. For more information see [Working with integrated decompilation](#).

Line	Hit Count	Avg Time (%)	Time (%)
29			
30			
31			
32			
33			
34			
35			
36			
37	1,162,876	0.000	12.432
38			
39			
40			
41			
42	1,162,876	0.000	23.070
43			
44			
45			
46			

```
public override string ToString()
{
    return(String.Format("{0} + {1}i", X, Y));
}

public static Complex operator +(Complex c1, Complex c2)
{
    return new Complex(c1.X + c2.X, c1.Y + c2.Y);
}

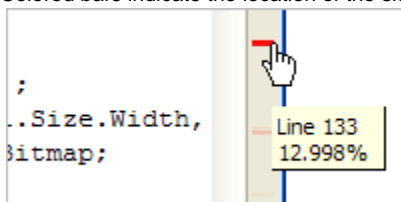
public static Complex operator *(Complex c1, Complex c2)
{
    return new Complex(c1.X * c2.X - c1.Y * c2.Y, c1.X *
    /// <summary>
    /// Real part.
```

If you used *Line-level and method-level timings; all methods* or *Line-level and method-level timings; only methods with sourceprofiling* mode, line-level timings are shown for each line of code (as well as average time and hit count). Note that line-level timings are unavailable for Silverlight applications.

## Navigating through source code

You can navigate through the source code for a particular file in several ways:

- To jump directly to lines of code that accounted for the most execution time, use the heat map alongside the vertical scroll bar. Colored bars indicate the location of the slowest lines of code in the source-code file:



- Click a bar to jump to the relevant line. The heat map is not available if you did not collect line-level timings.
- To jump directly to a particular method, select the method from the list directly above the source code. A colored bar against each method indicates the relative time spent within the method.
- To jump between methods from within the source code, click the method call. This click-through navigation works for most method calls where the called method has source code available.
- Use the forward



and back



history buttons.

You can also create a new call graph from within the source-code pane: right-click the method that you want to base the call graph on, and select **Draw Call Graph** > <method name>.

## Filtering the call tree and methods grid

Filtering the call tree and methods grid is only available in ANTS Performance Profiler 7.4.

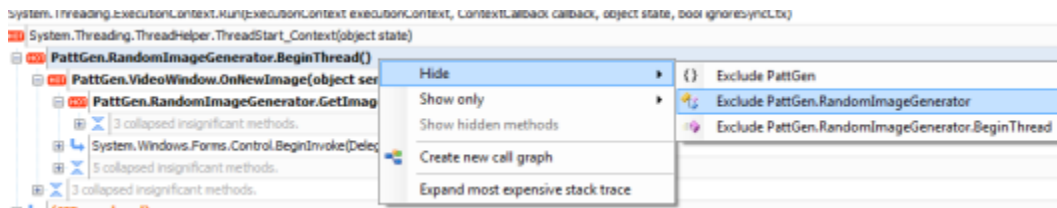
You can apply filters to limit the methods shown in the [call tree](#) or [methods grid](#).

To apply filters, right-click on the name of a method in either the call tree or the methods grid, and choose:

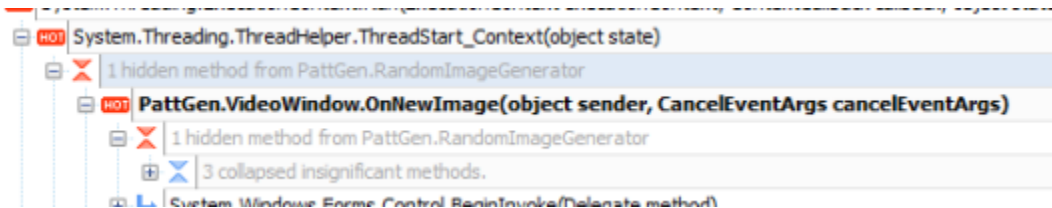
- **Hide**: Stops displaying the selected method, class, or namespace from the results shown in the call tree.
- **Show only**: Removes all but the selected method, class, or namespace from the results shown in the call tree.
- **Show hidden methods**: Remove all applied filters and display all methods in the call tree.

Methods cannot be hidden if they are the first node in a call stack.

When filters are applied, the **Display Options Scope** control for the selected view switches automatically to *Filtered methods*. After applying or removing a filter, ANTS Performance Profiler redraws the call tree. The timings shown for other methods are not affected.



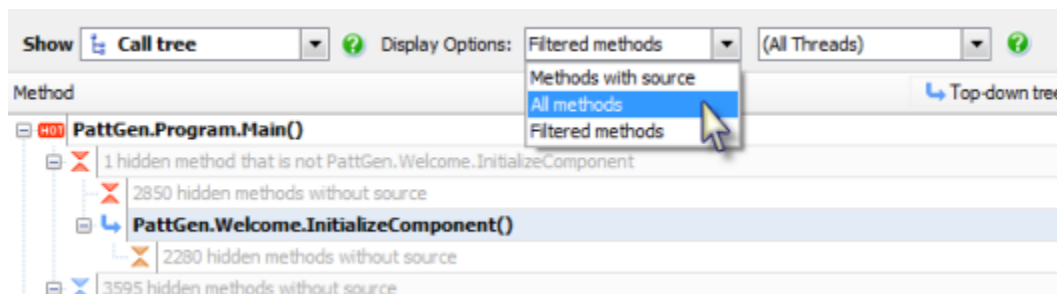
Hidden methods, classes, and namespaces are represented in the call tree and method by a grayed-out line:



To display the hidden methods, right-click on this grayed-out line and select **Show hidden methods**.

To remove the filter from throughout the call tree or methods grid, right-click on the grayed-out line and select **Remove filter**.

To remove all filters, use the **Display Options Scope** control to switch from *Filtered methods* to *All methods* or *Methods with source*:

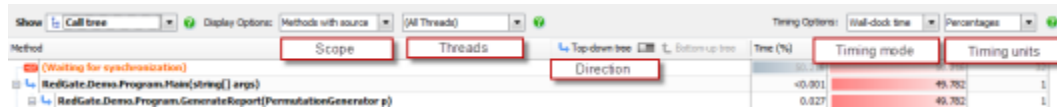


For more information, see [Changing results display options](#).

## Changing results display options

Changing the results display options is only available in ANTS Performance Profiler 7.4.

You can change the way data is displayed in the [call tree](#) and [methods grid](#), using the display options on the results toolbar:



- **Scope:** Controls which methods are displayed in the call tree or methods grid:
  - *All methods:* Shows all methods called during the selected period, including .NET framework class-library methods.
  - *Methods with source:* Shows only those methods for which the profiler can locate source code.
  - If you have selected methods with source and are not seeing certain methods that you expect, *Filtered methods:* Shows methods according to filters you have selected. For details on filtering, see **Filtering** below.
- **Threads:** filters the display of stack traces by thread.
- **Timing mode:** controls the way in which method timings are calculated. You can choose from:
  - *Wall-clock time:* includes blocking, such as waiting for I/O. Can be less reliable on virtual machines.
  - *CPU time:* excludes blocking.

Note that the profiler initially records timings in CPU ticks, and converts them to milliseconds when *Wall-clock time* is selected. CPU tick recording can be unreliable if a process runs on more than one processor.

- **Timing units:** controls how to display time. You can choose from:
  - *Percentages:* displays the time the method took as a percentage of the period currently selected on the timeline.
  - *Ticks*
  - *Milliseconds*
  - *Seconds*
- **Direction** (call tree only): controls whether the call tree is displayed *top-down* (calling methods above called methods) or *bottom-up* (called methods above calling methods)

You can also reorder the call tree. To change the stack-trace order, click the **Time With Children (%)** column heading.

## Working with continuous profiling

Continuous Profiling was an experimental feature in ANTS Performance Profiler 7.0 only. It was removed in subsequent versions.

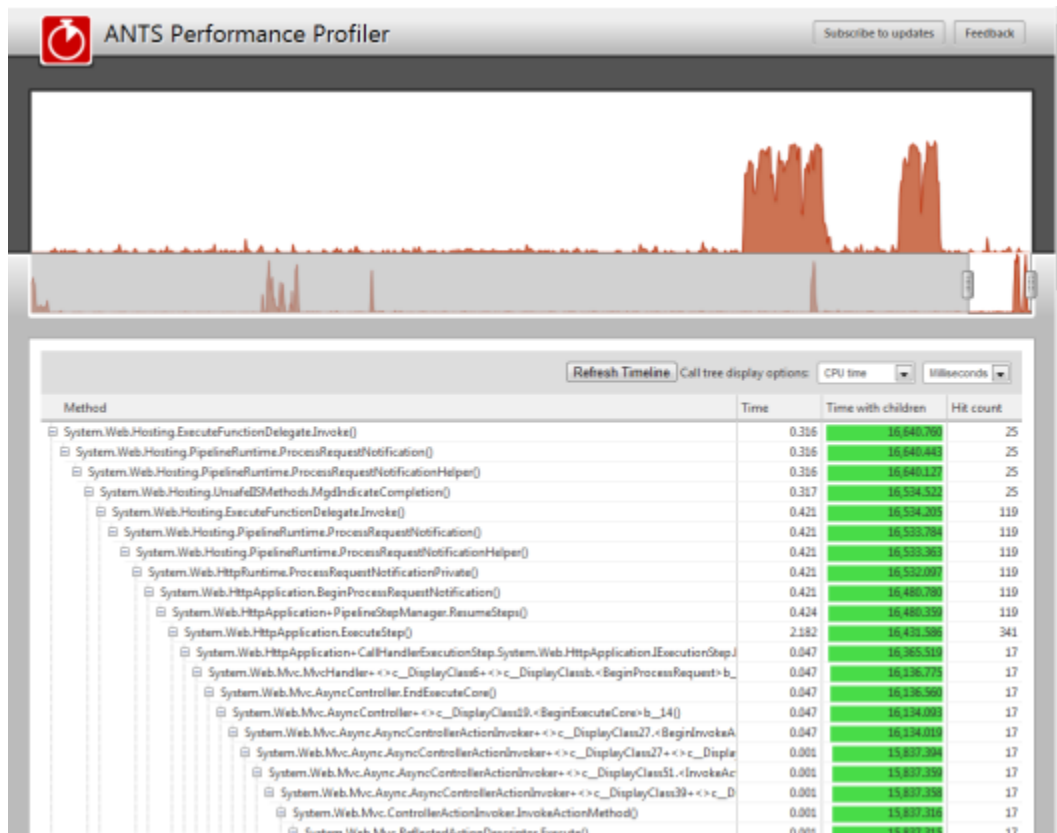
For instructions on installing and setting up continuous profiling, see [Setting up continuous profiling](#).

Continuous profiling is under active development and ANTS Performance Profiler v7.0 includes an early access version of the functionality. If you have any questions or comments, please visit the [Continuous Profiling feedback forum](#).

## To view continuous profiling results:

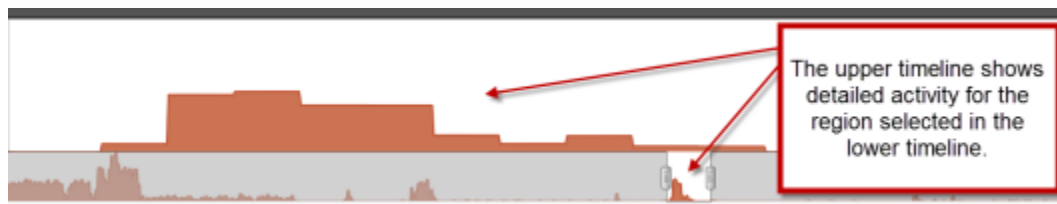
1. Launch the profiling results web interface in Chrome, Firefox, or Internet Explorer 9.

By default the interface is located at <http://localhost/ANTSPProfiler>. If you specified another target site during installation, the interface will be located at that address instead.



The web interface displays method calls made in all sites running in IIS, except in the application pool where the profiler interface is hosted.

The interface displays two timelines: the lower timeline shows CPU usage for up to the last six hours (a shorter interval is shown if 500MB of results were generated in under six hours). To view detailed CPU usage information in the upper timeline, select a region on the lower timeline.

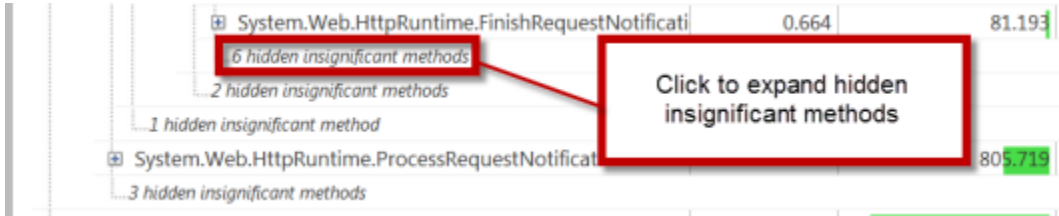


When you select a region, a call tree appears below the timelines, showing all methods called during the selected interval. Use the

☐ +  
and  
☐ -

buttons to expand and collapse the call stacks.

Insignificant methods, with an execution time totaling less than 1% of the parent method's execution time, are hidden in the call tree and indicated with the message "hidden insignificant methods". To display these methods, click "[x] hidden insignificant methods".



For advice on interpreting results, see [Working with the call tree](#).

## Compatibility with other profilers

The continuous profiling tool currently runs as a separate tool from the main ANTS Performance Profiler product. If you have installed an early access build of the ANTS Performance Profiler continuous profiler IIS module, other profilers - including the desktop ANTS Performance Profiler product - will be unable to profile applications running in IIS on this computer. To re-enable other profilers with IIS, uninstall the IIS Profiler Module:

1. From your computer's Start menu, launch the **Continuous Profiling Configuration Tool**.
2. Click **Uninstall**.

## Worked examples

Learn more about performance profiling with ANTS Performance Profiler by following these examples:

- [Worked example: Profiling performance of an algorithm](#)
- [Worked example: Profiling network overheads](#)
- [Worked example: Profiling an ASP.NET application](#)
- [Worked example: Profiling from the command line](#)

## Worked example - Profiling performance of an algorithm

This worked example shows how you can use ANTS Performance Profiler to identify the most time-critical parts of a demonstration application, particularly where two different approaches to a problem are optimal in different circumstances.

Note that the demonstration application used in this worked example is supplied with ANTS Performance Profiler 6.3 and later.

### Introducing TimeLineDemo

This worked example uses *TimeLineDemo*.

*TimeLineDemo* is a simple Windows application that checks which of a set of numbers are prime.

There are two different versions of *TimeLineDemo*:

- In *Brute Force* configuration, a brute-force method is used to check whether the number is prime.

The *Brute Force* build of *TimeLineDemo* is found in %Program Files%\Red Gate\ANTS Performance Profiler 7\Tutorials\CS\Precompiled\TimeLine\BruteForce\

- In *Miller-Rabin* configuration, the Miller-Rabin algorithm is used to check whether the number is prime.

The *Miller-Rabin* build of *TimeLineDemo* is found in %Program Files%\Red Gate\ANTS Performance Profiler 7\Tutorials\CS\Precompiled\TimeLine\MillerRabin\

The application has two options:

- Max Random
- Sample Size

When *TimeLineDemo* first starts, it creates a list of as many prime numbers as possible in 15 seconds.

When you click **Go**, *TimeLineDemo* creates a list of positive integers, which is *Sample Size* items long. All of the integers are random numbers between 1 and *Max Random*. For each integer in the list, *TimeLineDemo* checks whether the number is prime, using the following algorithm:

1. If the number is in the list created in the first 15 seconds, it is prime.
2. If a number in the list created in the first 15 seconds is a factor of the current number, the current number is not prime.
3. In all other cases, use either a brute-force mechanism or the Miller-Rabin algorithm to check whether the number is prime (depending on the build in use).

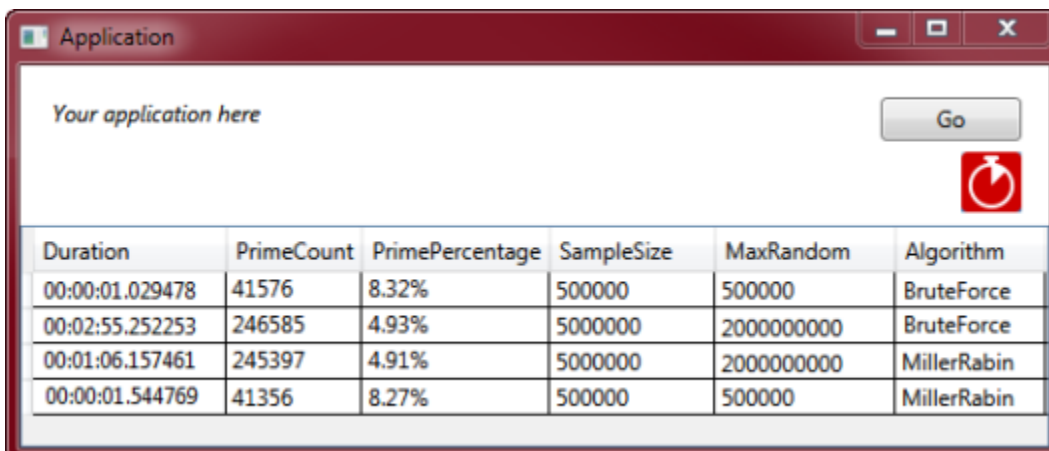
Note: Strictly speaking, the Miller-Rabin algorithm is probabilistic. For the purposes of this demonstration, however, the algorithm can be assumed to be deterministic. This is because the algorithm's results are exact for integers which are smaller than the largest integer that the field accepts.

### Walkthrough

Error rendering macro 'multimedia' : null

### Conclusion

This walkthrough has demonstrated a realistic programming scenario. It has compared two methods for testing if a number is prime and shown that Miller-Rabin is more efficient than the brute force approach when the number being tested is large.



Duration	PrimeCount	PrimePercentage	SampleSize	MaxRandom	Algorithm
00:00:01.029478	41576	8.32%	500000	500000	BruteForce
00:02:55.252253	246585	4.93%	5000000	2000000000	BruteForce
00:01:06.157461	245397	4.91%	5000000	2000000000	MillerRabin
00:00:01.544769	41356	8.27%	500000	500000	MillerRabin

This walkthrough has also demonstrated that you can select just a portion of the timeline when analyzing results, allowing you to ignore a slow

initialization phase, for example.

### Learning more

To perform the task demonstrated in this walkthrough more efficiently, you can use the following additional functionality:

- To help you identify the section of the timeline to select, you can refer to the Event markers in the Events bar.
- You can name and bookmark sections of the timeline.

For more information, see [Working with the timeline](#).



## Worked example - Profiling network overheads

This worked example demonstrates how to use ANTS Performance Profiler to profile a .NET executable that exhibits latency problems caused by fetching HTTP data from a network.

Note that the demonstration application used in this worked example is supplied with ANTS Performance Profiler 6.3 and later.

### Introducing LatencyDemo

This worked example uses *LatencyDemo*.

*LatencyDemo* is a simple Windows application that fetches the RSS feeds from the ANTS Memory Profiler and ANTS Performance Profiler forums on the Redgate website.

A copy of *LatencyDemo* is supplied with ANTS Performance Profiler in %Program Files%\Red Gate\ANTS Performance Profiler 7\Tutorials\CS\Precompiled\LatencyDemo\

### Walkthrough

Error rendering macro 'multimedia' : null

## Worked example - Profiling an ASP.NET application - NerdDinner

This worked example applies to ANTS Performance Profiler 7.3 and later.

This worked example describes how to profile a sample ASP.NET website called NerdDinner. You can [download the original ASP.NET MVC source code for NerdDinner from CodePlex](#): in this example, the NerdDinner code has been modified to illustrate a performance problem.

In this example, NerdDinner is installed on the same computer as the one being used to profile it, and can be accessed in a web-browser at <http://127.0.0.1:8090>.

Imagine that the problem with NerdDinner is that it is slow to return results for a search involving a SQL query. Imagine you want to know whether you can do anything in the application's .NET code to improve the site's performance, before you investigate performance on the database, or spend money to improve the hardware the site runs on.

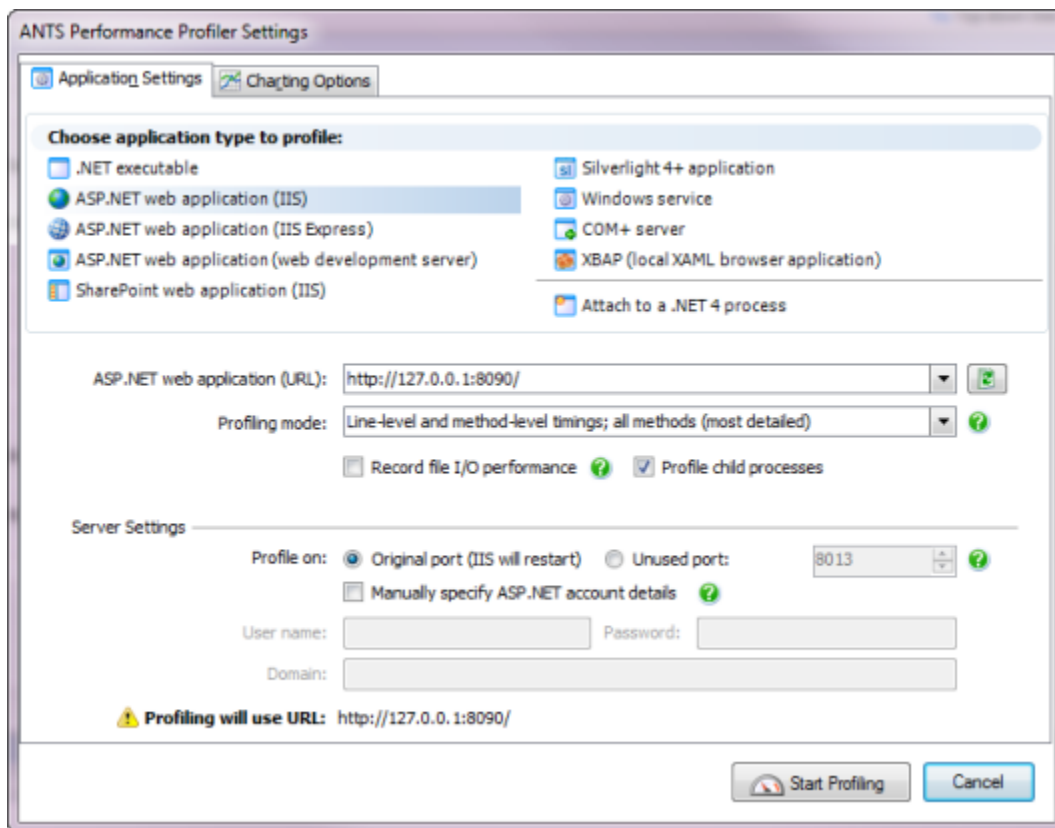
There are three main steps:

1. Set up ANTS Performance Profiler
2. Use NerdDinner
3. Analyze the profiler's results

### Setting up ANTS Performance Profiler

To set up ANTS Performance Profiler:

1. In the ANTS Performance Profiler settings, on the Application Settings tab, select **ASP.NET web application (IIS)**.
2. In the **ASP.NET web application (URL)** dropdown menu, select the site's URL: <http://127.0.0.1:8090>.
3. Choose the required profiling mode. Here, we'll use line-level and method-level timings.  
For more on profiling modes, see [Choosing a profiling mode](#).



4. Select the port on which to profile your application:
  - If you are using IIS 6 or IIS 7, select **Unused port** and choose a port that is not used by IIS. IIS will start a new instance of your application on the specified port. Note that this will not work if your application's code binds to a specific port.
  - If you are using IIS 5, or if you are using IIS 6 or 7 and your application binds to a specific port, select **Original port**. IIS will restart so that the profiler can attach to the port. Note that restarting IIS stops IIS and restarts only the application that you are profiling. Other websites on the same

IIS instance will not be present when IIS restarts.

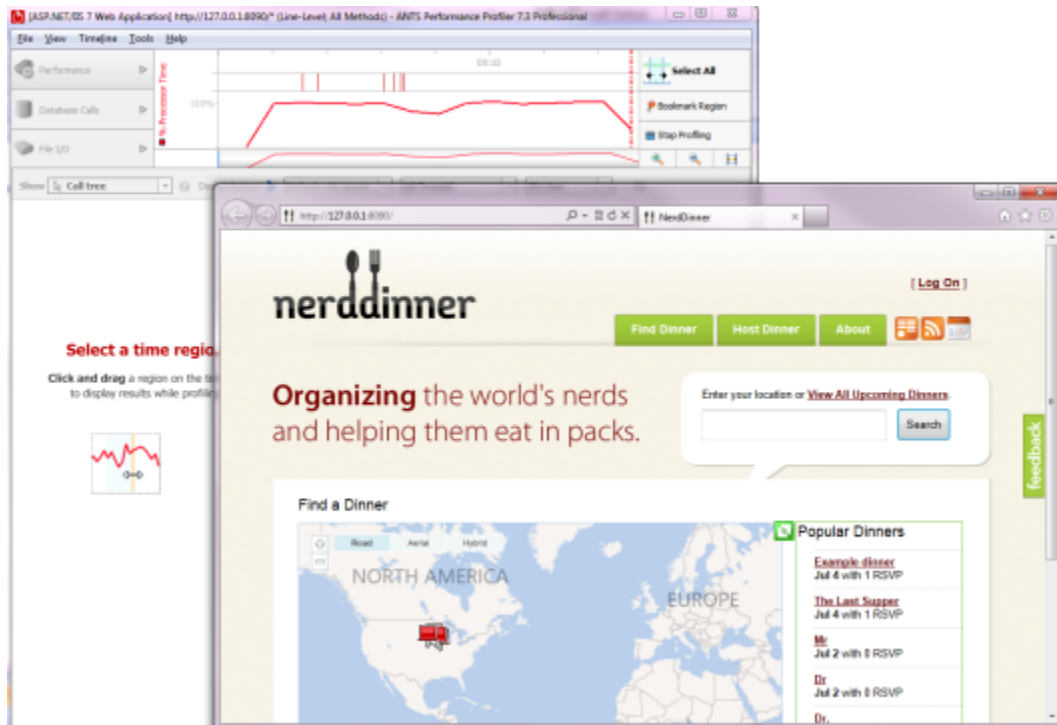
The port where the application will be profiled is displayed at the bottom of the ANTS Performance Profiler Settings dialog box. For more on profiling web applications in IIS, see [Profiling ASP.NET applications running on IIS](#) and [Troubleshooting IIS profiling](#).

5. Click



6. Internet Explorer launches and loads NerdDinner.

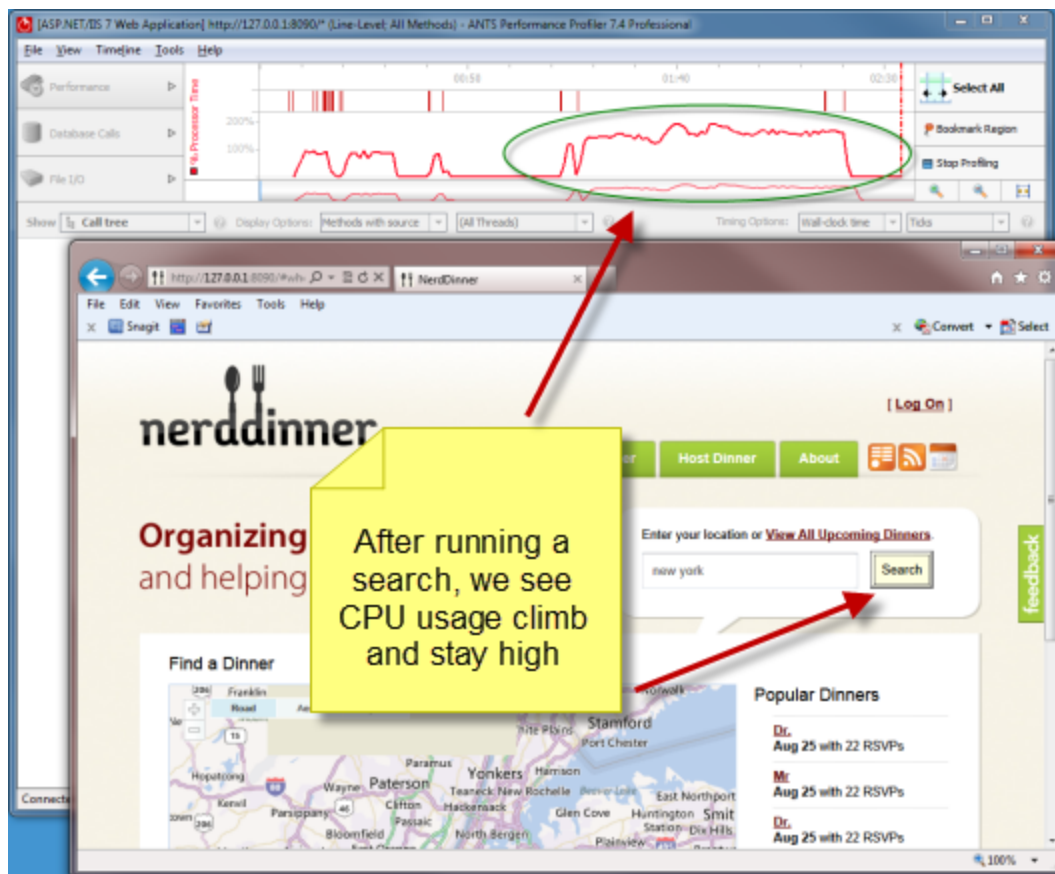
If you prefer not to use Internet Explorer, you can open a different browser at the same address. You must, however, leave open the instance of Internet Explorer created by the profiler.



## Using NerdDinner

In this example, the NerdDinner site includes several pages that rely heavily on database queries, as well as some static HTML. Imagine users have reported that the site's location search feature is slow: we start the investigation by exercising that feature, entering a place name in the search box and hitting **Search**. The search is designed to return a list of events near a certain location.

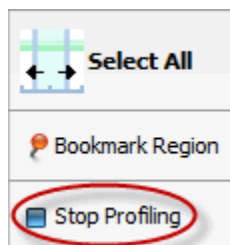
While we use the site, ANTS Performance Profiler's timeline shows the CPU usage the application has caused:



There's a brief, expected CPU peak when the site launches. CPU then returns to near zero until we start our search, when it begins to grow to nearly 200%, staying high for long after the first results were returned. This clearly indicates a performance bottleneck in the search feature.

For more information on analyzing CPU usage, see [Working with the timeline](#).

Now we see where the bottleneck may lie, we can explore the results in detail: In the ANTS Performance Profiler window, click **Stop Profiling**.

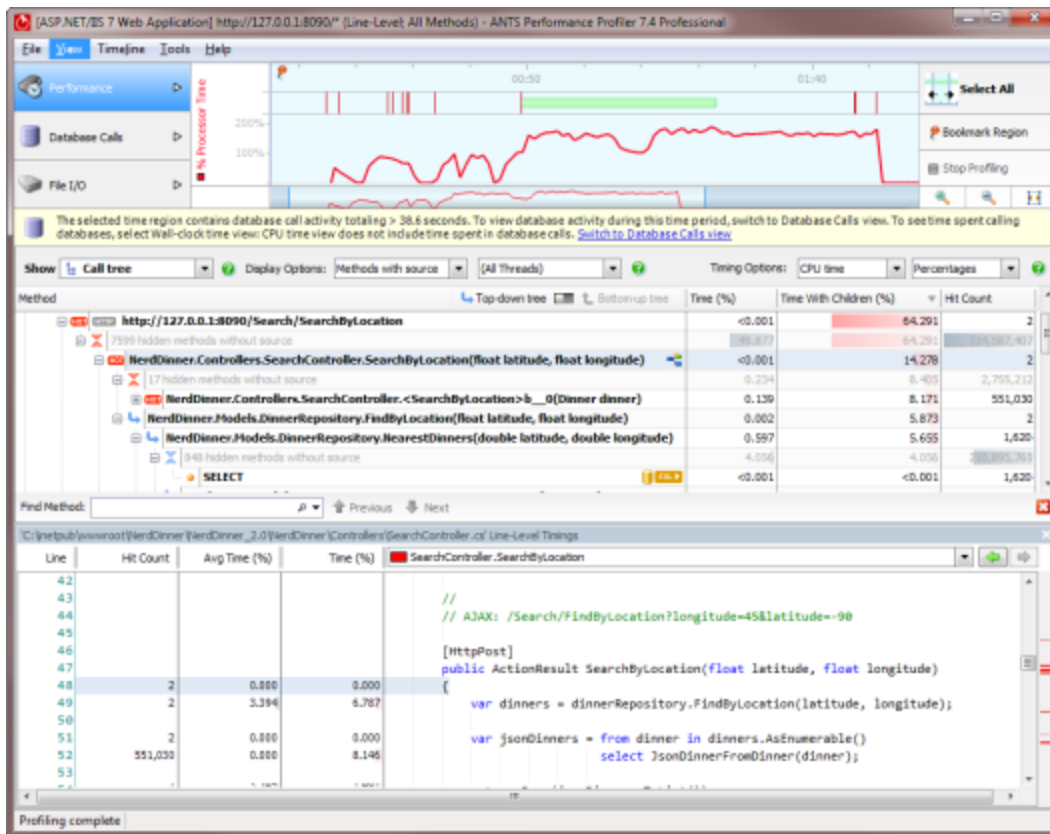


The website closes.

## Analyzing the profiler's results

After a few moments, full results are shown.

At the top of the call tree, ANTS Performance Profiler shows the 'hottest' stack trace; that is, the code that contributes the greatest proportion of CPU time. This is usually a good place to start looking for opportunities to optimize the code.



At the top of each call tree stack, we see the HTTP request that triggered the calling of the .NET methods. As we expected, the hottest stack trace during the whole profiling session descends from the request `http://127.0.0.1:8090/Search/SearchByLocation`, child methods of which account for about 64% of the total time spent in the profiling session.

Looking down the hot stack, we can see that this request called a .NET method, `NerdDinner.Models.DinnerRepository.NearestDinner(double latitude, double longitude)`, that was hit 1620 times - as was the SQL `SELECT` query it ultimately runs.

For more information on hot stacks and HTTP nodes, see [Working with the call tree](#).

If we select the method's parent, `NerdDinner.Controllers.SearchController.SearchByLocation(float latitude, float longitude)`, we can view its source code. Because we used a profiling mode with line-level timings, we can also see where inside the method the greatest time was spent. This shows us that the method retrieves the full list of all recorded events from the database:

Hit Count	Avg Time (%)	Time (%)	SearchController.SearchByLocation
2	0.000	0.000	[HttpPost]
2	3.240	6.480	public ActionResult SearchByLocation(float latitude, float longitude)
2	0.000	0.000	{
581,214	0.000	7.816	var dinners = dinnerRepository.FindByLocation(latitude, longitude);
2	3.558	7.116	var jsonDinners = from dinner in dinners.AsEnumerable()
2	0.000	0.000	select JsonDinnerFromDinner(dinner);
			return Json(jsonDinners.ToList());
			}

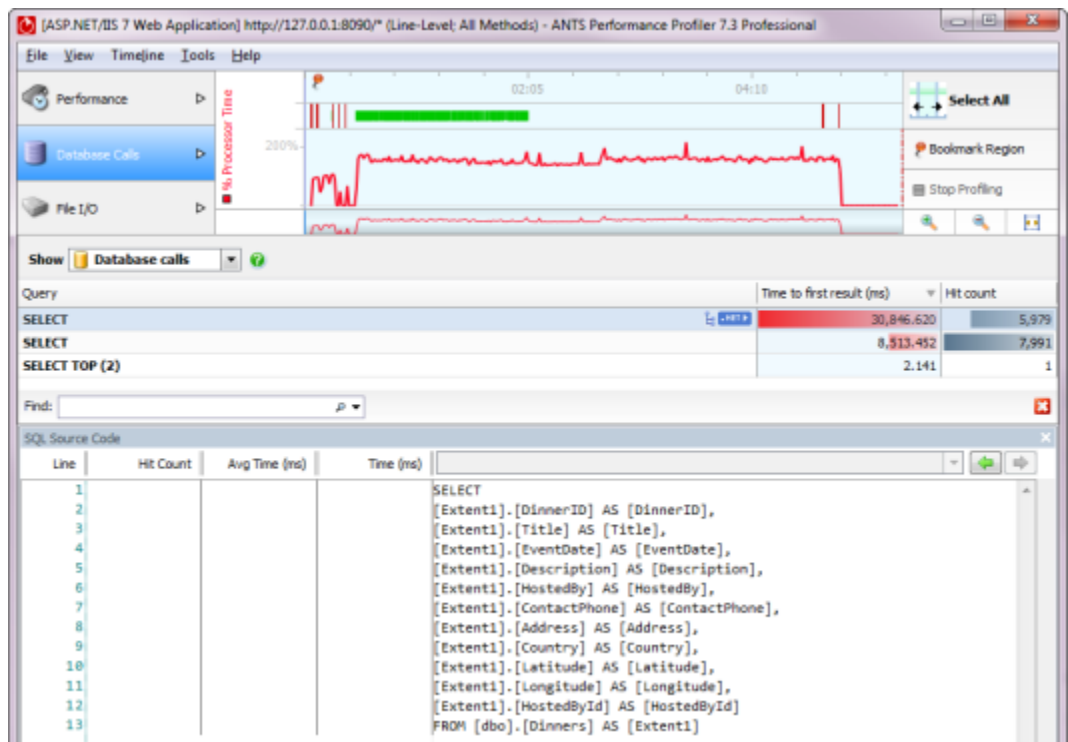
The method `NerdDinner.Models.DinnerRepository.FindByLocation(float latitude, float longitude)` then tries to process the result set in the web page, to filter by location:

Hit Count	Avg Time (%)	Time (%)	DinnerRepository.NearestDinner
1,620	0.000	0.000	public IQueryable<Dinner> NearestDinner(double latitude, double longitude)
1,620	0.000	0.000	{
1,620	0.000	0.000	var dinners = db.Dinners;
1,620	0.000	0.003	var nearestDinners = new List<Dinner>();
1,620,000	0.000	4.406	var thePoint = new GeoCoordinate(latitude, longitude);
1,618,380	0.000	0.009	foreach (var dinner in dinners)
1,618,380	0.000	0.839	{
581,214	0.000	0.025	if (DistanceBetweenPoints(thePoint, new GeoCoordinate(dinner.Latitude, dinner.Longitude)) < 0.001)
1,618,380	0.000	0.009	nearestDinners.Add(dinner);
			}

These methods are good candidates for optimization. The same results could be achieved via AJAX calls, and by returning from the database only events that meet specified latitude and longitude criteria.

For more on the source code view, see [Working with source code](#).

Switching to Database Calls view, we can see that the query to return the full results set was run thousands of times, summing to over 14 seconds just to return the first result for all the instances of the query:



Again, it's clear that this very broad request is being run repeatedly, contributing a large total running time. It would be more efficient to run a more precise request fewer times.

For more on SQL query timings, see [Working with the database calls view](#).

After profiling, we now have a clear idea of which HTTP requests are associated with slow performance, and which of our .NET methods contain the source of those slowdowns. We know which methods to rewrite to remove bottlenecks, and the steps we'll need to reproduce in the application to check that the problem has gone.

## Worked example - Profiling an ASP.NET application - TheBeerHouse

This worked example applies to ANTS Performance Profiler version 7.2 only.

This worked example describes how to profile a sample ASP.NET website called TheBeerHouse. The original ASP.NET MVC source code for TheBeerHouse can be obtained from [CodePlex](#). To create this worked example, TheBeerHouse has been recompiled for .NET 4.

TheBeerHouse has been installed on the same computer as the one being used to profile it, and it can be accessed in a web-browser from the address [http://localhost/TBH\\_Web/](http://localhost/TBH_Web/)

Imagine that the problem with TheBeerHouse is just that it is slow when loading pages. I want to know whether I can do anything to improve the site's performance, before I spend lots of money on improving the hardware it runs on.

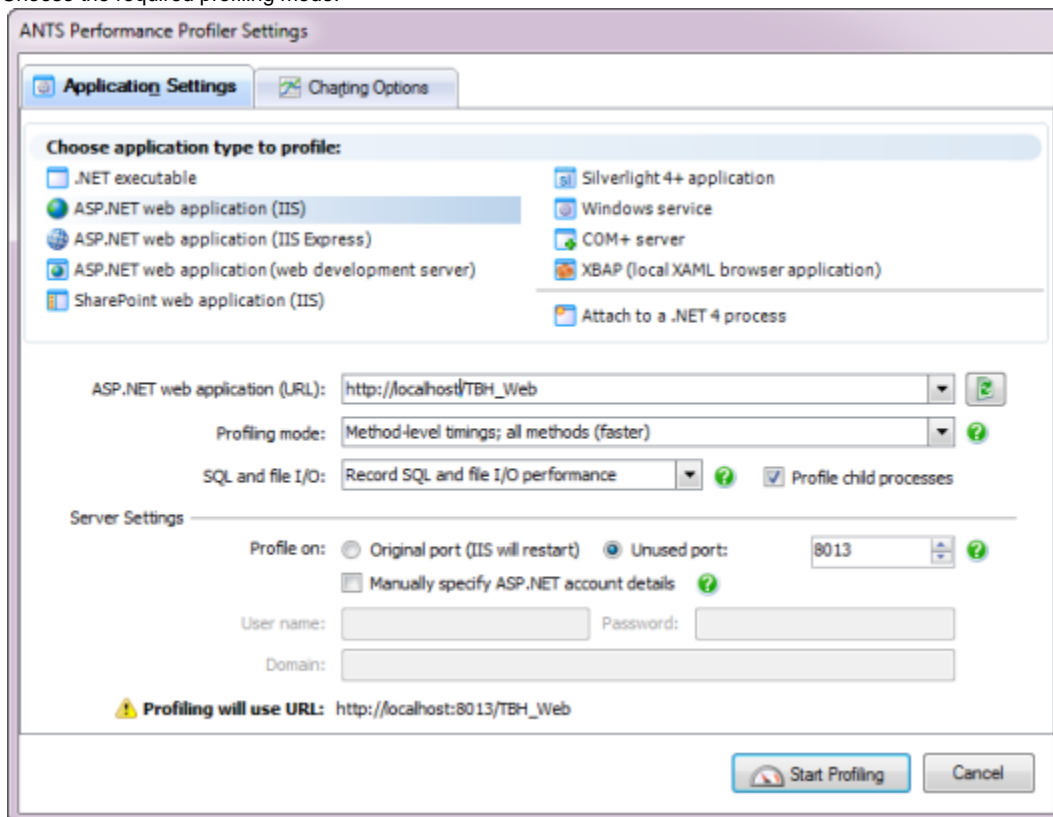
There are three main steps:

1. Set up ANTS Performance Profiler
2. Use TheBeerHouse
3. Analyze the profiler's results

### Setting up ANTS Performance Profiler

To set up ANTS Performance Profiler:

1. In the ANTS Performance Profiler settings, on the Application Settings tab, select **ASP.NET web application (IIS)**.
2. In the **ASP.NET web application (URL)** dropdown menu, select the site's URL: [http://localhost/TBH\\_Web/](http://localhost/TBH_Web/).
3. Choose the required profiling mode.



4. Select the port on which to profile your application:

- If you are using IIS 6 or IIS 7, select **Unused port** and choose a port that is not used by IIS.

IIS will start a new instance of your application on the specified port.

Note that this will not work if your application's code binds to a specific port.

- If you are using IIS 5, or if you are using IIS 6 or 7 and your application binds to a specific port, select **Original port**.

IIS will restart so that the profiler can attach to the port.

Note that restarting IIS stops IIS and only restarts the application that you are profiling. If your website depends on another site running

on the same IIS instance, that other site will not be present when IIS restarts.

If your application takes too long to start, IIS might not restart correctly. Use IIS Manager to stop the website manually until you have finished profiling.

The port where the application will be profiled is displayed at the bottom of the ANTS Performance Profiler Settings dialog box.

1. Click



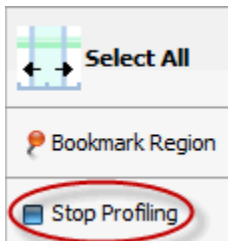
2. Internet Explorer launches and shows TheBeerHouse. If you prefer not to use Internet Explorer, you can open a different browser at the same address. You must leave the instance of Internet Explorer created by the profiler open, however.



## Using TheBeerHouse

In this scenario, it is not known exactly where the performance problem is, and so initially a number of different pages are accessed, including some which are known to rely heavily on database queries, and some which mainly contain static HTML. After any particular performance problems have been identified, those pages can be profiled again in a more systematic manner.

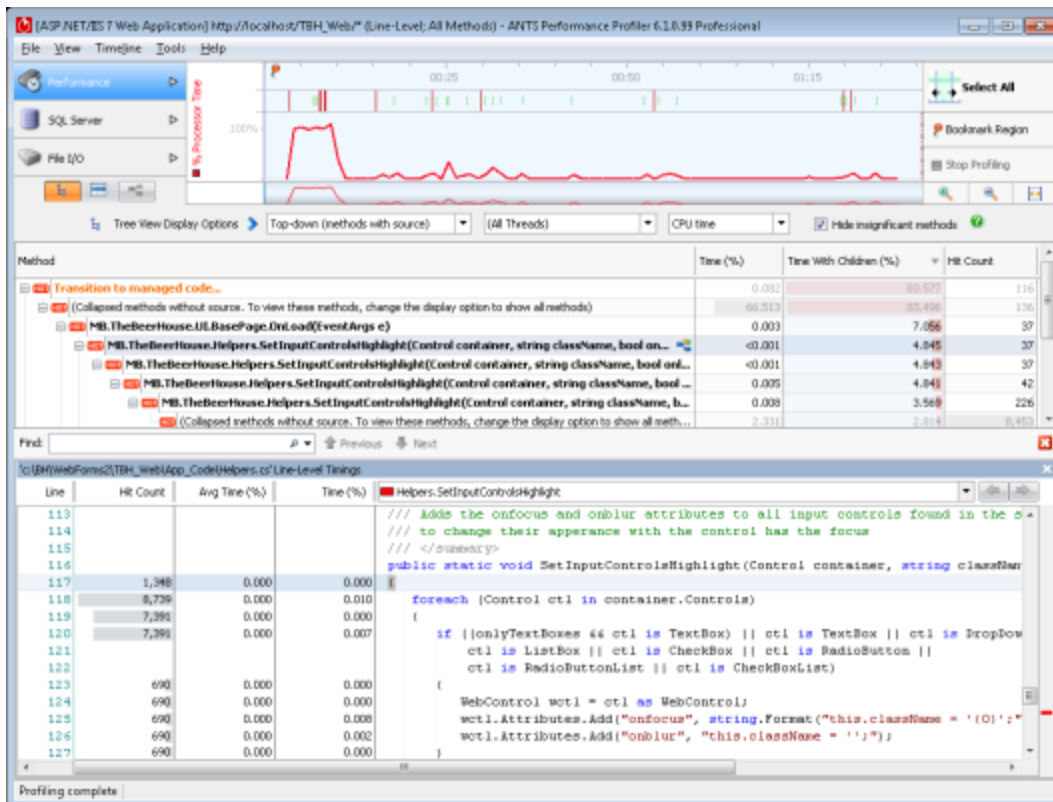
After a number of different pages have been opened, in the ANTS Performance Profiler window, click **Stop Profiling**.



## Analyzing the profiler's results

After a few moments, the results are shown. ANTS Performance Profiler shows the 'hottest' stack trace; that is, the code which is using the greatest amount of CPU time. This is usually a good place to start looking for opportunities to optimize the code.



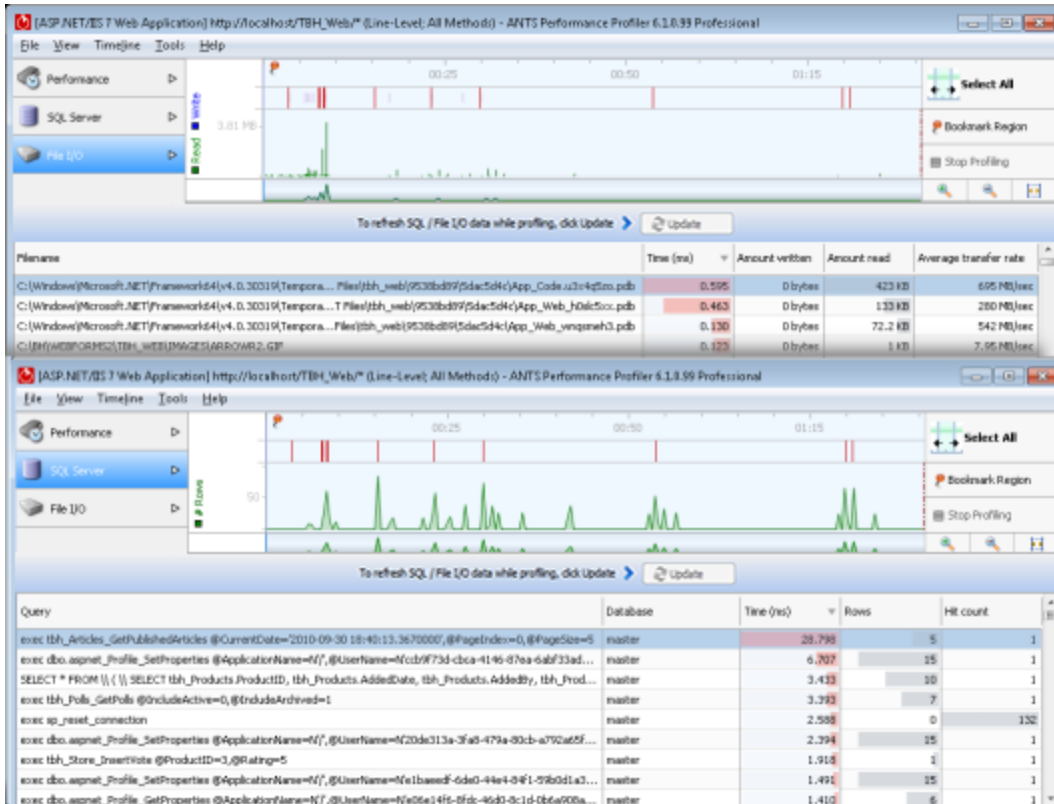


TheBeerHouse is already quite well optimized, with 66% of processor time being spent on very inexpensive methods. The site could be improved, however, because nearly 5% of the processor's time is spent on a method called `SetInputControlsHighlight()`, which runs when each page loads.

Select that row.

Because the source code for TheBeerHouse is available, ANTS Performance Profiler shows the source code for this method in the lower pane. Every time a page loads, `SetInputControlsHighlight()` iterates over the input fields it contains, and adds `onfocus` and `onblur` attributes to the HTML output, in turn causing the DOM to change their class when the input has focus. This is clearly a good candidate for optimization, because the same result can be achieved by just changing the CSS file to add the `:focus` pseudo-class.

In this instance, the File I/O and database call results do not show anything abnormal.



## Worked example - Profiling from the command line

This worked example demonstrates how you can use ANTS Performance Profiler from the command line.

Profiling from the command line is useful if you want to integrate performance profiling into your usual testing or build processes. The profiler results can be output to CSV, XML or HTML, which means that you can easily check the results for abnormal values as part of your automated routines.

This example uses the following simple C# Console Application (called *SimpleApp.exe*) which prints '.' to the console 100 times:

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

namespace SimpleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("The application has started");
            // Count from 0-99
            int i = 0;
            while (i < 100)
            {
                Console.Write('.');
                i++;
            }
            Console.WriteLine("The application is exiting");
        }
    }
}
```

Another simple C# Console Application can read the results CSV file created by ANTS Performance Profiler, to check that Write() is called exactly 100 times, thereby verifying that *SimpleApp* is performing correctly:

```

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

namespace ReadOutput
{
    class TextFileReader
    {
        static void Main(string[] args)
        {
            StreamReader sr = new StreamReader(new
FileStream("C:\\testing\\results.csv", FileMode.Open, FileAccess.Read));
            string line;
            int ok = 1;
            // Read file line-by-line
            while ((line = sr.ReadLine()) != null)
            {
                char separator = ',';
                string[] linedata = new string[10];
                linedata = line.Split(separator);
                // Try to cast linedata[3] as int, not string
                int i;
                try
                {
                    i = (int)int.Parse(linedata[3]);
                    // Check the number of times that Write(char value) is run
                    if ((linedata[2] == "Write(char value)") && (i != 100))
                    {
                        Console.WriteLine("Test failed");
                        ok = 0;
                    }
                }
                catch
                {
                    // Do nothing
                }
            }
            sr.Close();
            if (ok == 1)
            {
                Console.WriteLine("Test passed OK");
            }
        }
    }
}

```

Finally, an MS-DOS batch file can be written to profile *SimpleApp* in ANTS Performance Profiler and, when this is complete, to check that the test passed:

```
C:
CD /
CD "Program Files\Red Gate\ANTS Performance Profiler 7\"
Profile.exe /e:"C:\testing\SimpleApp.exe" /ll /csv:"C:\testing\results.csv"
C:
CD testing
ReadOutput.exe
```

The console shows:

```
C:\testing>ReadOutput.exe
Test passed OK
```

This confirms that SimpleApp is performing correctly.

For a list of all available command line arguments, see [Profiling from the command line \(API\)](#).

For a more complex example describing how to integrate ANTS Performance Profiler results with an NUnit test, see [Integrating Performance Profiling into the Build Process](#).

# Troubleshooting

## Common issues

- Troubleshooting licensing and activation errors
- Troubleshooting application crashes
- Troubleshooting missing results
- Troubleshooting PDB problems
- Troubleshooting SharePoint Profiling
- Troubleshooting IIS profiling
- Troubleshooting SQL and HTTP call profiling

## Error messages

- "No Disk" error occurring while profiling application
- Couldn't open metabase
- Error stopping IISAdmin profiling IIS web application on Windows XP
- Failed to CoCreate Profiler
- IIS ceases to work after profiling web applications
- Method not found: 'UInt32 <Module>.\_ANTS\_Begin\_Sql(System.String)'
- No .NET methods were profiled on web application
- Operation could destabilize the runtime error profiling ASP.NET
- Please specify a valid URL message profiling ASP.NET
- The system cannot find the file specified

## 3rd party components

- Profiling assemblies protected with DeployLX

## Unexpected behavior / technical questions

- ANTS Performance Profiler menu items not showing in Visual Studio 2010
- Attach to process unavailable with some anti-virus software
- Call graph percentages do not add up exactly
- Can I profile Compact Framework applications?
- Double hit counts occurring on one line
- Enabling line-level timings for SecurityTransparent code
- Failed to coCreate Profiler on ASP .NET web application
- Forcing your application to use .NET 4
- HTTP request timings in IIS
- Isolating single ASP .NET pages in ANTS Profiler results
- Log files
- Memory leaks observed when profiling Windows Presentation Foundation (WPF) applications
- Missing hits for lines in the source code view
- Problems synchronizing results
- Profiler prompts for location of source code which is not your own source code
- Profiler stopping while profiling an in-browser Silverlight application
- Profiling an assembly in the Global Assembly Cache (GAC)
- Profiling ClickOnce applications deployed to IIS
- Profiling Microsoft Office managed-code add-ins
- Profiling unit tests using Nunit
- Profiling web services in IIS Express
- Setting file IO and child process profiling in high DPI modes
- Showing the amount of time taken for a method in one particular thread
- Times in source code window are greater than the times showing in the method grid or tree view
- Times on individual lines do not add up to method time
- Windows service profiling fails if the service uses a system account

## Common issues

- [Troubleshooting licensing and activation errors](#)
- [Troubleshooting application crashes](#)
- [Troubleshooting missing results](#)
- [Troubleshooting PDB problems](#)
- [Troubleshooting SharePoint Profiling](#)
- [Troubleshooting IIS profiling](#)
- [Troubleshooting SQL and HTTP call profiling](#)

## Troubleshooting licensing and activation errors

This page provides information about errors you may encounter when you activate Redgate products:

- [The number of activations for this serial number has been exceeded](#)
- [This serial number has been disabled](#)
- [This serial number was for a trial extension](#)
- [This serial number is not registered with the activation server](#)
- [This serial number is not for <product name>](#)
- [This serial number is not for this version](#)
- [The activation request is in the wrong format](#)
- [The activation request contains an invalid machine hash](#)
- [The activation request contains an invalid session](#)
- [The activation request contains an invalid serial number](#)
- [The activation request contains an invalid product code or version number](#)
- [There's a problem deactivating your serial number](#)
- [This serial number is not activated on this computer](#)
- [Products not activated on this computer](#)

### The number of activations for this serial number has been exceeded

This error message is displayed when a serial number is activated on more computers than the number of licenses that were purchased for that serial number.

When you purchase products from Redgate, we send you an invoice that includes your serial numbers. The serial numbers enable you to activate the software a number of times, depending on how many licenses you purchased and the terms in the [license agreement](#). When this limit is reached, you will see this error message.

To fix the problem, you can:

- [deactivate](#) the product on another computer to free up a license
- [purchase](#) more licenses
- [request additional activations](#) for your serial number

### This serial number has been disabled

This error message is displayed when you try to activate a product using a serial number that Redgate has disabled.

When you upgrade a product, your existing serial numbers will be disabled and we will issue new ones with your invoice. If you cannot find your new serial numbers, you can review them at <http://www.red-gate.com/myserialnumbers>

Redgate will also disable serial numbers for non-payment of invoices or breach of the terms in the [license agreement](#). If you think we have disabled your serial numbers in error, email [licensing@red-gate.com](mailto:licensing@red-gate.com)

### This serial number was for a trial extension

This error message is displayed when you have requested a trial extension and you try to reuse the serial number that was provided for the trial extension; trial extensions can be used one time only.

To continue using the product, you need to [purchase it](#).

### This serial number is not registered with the activation server

This error message is displayed when the serial number you entered does not exist on the Redgate activation server.

To find out your serial numbers, check your invoice or go to <http://www.red-gate.com/myserialnumbers>

### This serial number is not for <product name>

This error message is displayed when the serial number you entered is not for the product you are trying to activate.

To find out your serial numbers, check your invoice or go to <http://www.red-gate.com/myserialnumbers>



### This serial number is not for this version

This error message is displayed when the serial number you entered is for a different version of the product you are trying to activate.

If the serial number is for an older version of the product, and you don't have that version installed on your computer, you can download it from the Release notes and other versions page.

If you want to upgrade to the latest version of the product, go to the [Upgrade center](#) to get a quote or purchase an upgrade, or email [sales@redgate.com](mailto:sales@redgate.com).

### The activation request is in the wrong format

This error message is displayed:

- if your internet connection does not allow SOAP requests.  
Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed.
- if you are activating by email and there is a problem with the format of the activation request.  
Check that you copied and pasted all of the activation request.  
Alternatively, try using manual activation. Go to <http://www.red-gate.com/activate> and paste your activation request under **Step 1**.
- when you are using manual activation and there is a problem with the format of the activation request. If the format is incorrect, for example part of the request is missing, the Redgate activation server cannot process the request.  
Check that you copied and pasted all of the activation request.

For more information about activating manually, see [Manual activation](#).

### The activation request contains an invalid machine hash

This error message is displayed:

- if your internet connection does not allow SOAP requests.  
Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see [Manual activation](#).
- when you are using manual activation and there is a problem with the format of the *machinehash* element in the activation request. The *machinehash* is a checksum of attributes from your computer. We use the *machinehash* to identify computers on which our products have been activated. If the format of the *machinehash* element is incorrect, the Redgate activation server cannot process the request.  
Check that you copied and pasted the activation request correctly.

### The activation request contains an invalid session

This error message is displayed:

- if your internet connection does not allow SOAP requests.  
Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see [Manual activation](#).
- when you are using manual activation and there is a problem with the format of the activation request. If the format of the *session* element is incorrect, the Redgate activation server cannot process the request.  
Check that you copied and pasted the activation request correctly.

### The activation request contains an invalid serial number

This error message is displayed:

- if your internet connection does not allow SOAP requests.  
Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see [Manual activation](#).
- when you are using manual activation and there is a problem with the format of the activation request. If the format of the serial number is incorrect, the Redgate activation server cannot process the request.  
Check that you copied and pasted the activation request correctly.

### The activation request contains an invalid product code or version number

This error message is displayed:

- if your internet connection does not allow SOAP requests.  
Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see [Manual activation](#).
- when you are using manual activation and there is a problem with the format of the activation request. If the product code or version numbers are incorrect, the Redgate activation server cannot process the request.  
Check that you copied and pasted the activation request correctly.

### There's a problem deactivating your serial number

This error message is displayed if your computer is not connected to the internet or your internet connection does not allow SOAP requests. You cannot deactivate a serial number if your computer does not have an internet connection.

Try deactivating again later. If the problem persists, contact your system administrator.

If you require more activations because you cannot deactivate your serial number, you can request them on the [Request Extra Activations](#) page.

### This serial number is not activated on this computer

This error message is displayed when you try to deactivate a serial number that has not been activated on your computer.

If you think the product installation on your computer is corrupt, you can try re-activating the product, and then deactivating the product again.

If you require more activations because you cannot deactivate your serial number, you can request them on the [Request Extra Activations](#) page.

### Products not activated on this computer

This error message is displayed when you try to deactivate a serial number for a bundle of Redgate products and those products were not activated on your computer.

If you think the product installation on your computer is corrupt, you can try re-activating the product, and then deactivating the product again.

If you require more activations because you cannot deactivate your serial number, you can request them on the [Request Extra Activations](#) page.

## Troubleshooting application crashes

If the application you are profiling stops responding, fails to load, or closes unexpectedly, the profiling process may have caused the profiled application to crash. If this happens, you may not receive profiling results for your session, or you may receive incomplete results captured before the crash.

If your profiling session returns no results, or incomplete results, but does not hang or crash, see [Troubleshooting missing results](#).

### Troubleshooting all crashes

If the application crashed but did not return the above error message, try the following approaches. They are listed in order of how likely they are to resolve common causes of crashes.

#### ***Force your application to use .NET 4***

Many crashes can be avoided by rebuilding the application as a .NET 4 executable, and profiling it using the **Attach to a .NET 4 process** option in Application Settings.

For instructions on reconfiguring your application for use with this profiling method, see [Forcing your application to use .NET 4](#).

The **Attach to a .NET 4 process** approach profiles your application when it is already running, rather than launching a new instance of the application. Because attaching to a running process does not change the way the application's code executes, it is less likely to cause a crash.

#### ***Run ANTS Performance Profiler from the Visual Studio menu***

If the crash occurs in a profiling session launched directly from ANTS Performance Profiler, try profiling using the Visual Studio ANTS Performance Profiler add-in instead.

For instructions see [Using the Visual Studio add-in](#).

#### ***Delete corrupt third-party PDBs***

The following third-party assemblies sometimes contain corrupt .pdb files, which can cause crashes during profiling:

- Microsoft.Practices library
- AjaxControlToolkit

For instructions on finding and deleting corrupt .pdb files, see [Troubleshooting PDB problems](#).

### Troubleshooting web application crashes

If the application you are profiling is a web application, try the following two steps.

#### ***Profile using another web server application***

Some applications that crash under profiling can be profiled successfully if run on another type of server application:

- If you are profiling in IIS, switch to the ASP.NET built-in web development server.
- If you are profiling in the ASP.NET web development server, switch to IIS.

Note that changing environments will work only if the server application you select supports all your application's features:

- Applications using integrated pipeline mode, impersonation, or HTTPS will not run on the built-in web-development server.
- Applications that cannot operate in a restricted security context may not run in IIS.

Profiling results obtained in from applications running on the web development server may also differ from performance in a production environment.

#### ***IIS: ensure you are profiling on the correct port***

- If you are using **IIS 6** or **IIS 7**, select **Unused port** and choose a port that is not used by IIS.  
Note that this will not work if your application's code binds to a specific port.
- If you are using **IIS 5**, or if you are using **IIS 6 or 7** and your application binds to a specific port, select **Original port**.  
IIS will restart so that the profiler can attach to the port.

### Troubleshooting error messages

#### ***StackOverflow Exception***

During line-level profiling, if the application you are profiling is very large or the profiling session lasts a long time, a stack overflow can occur. If this happens, the application will crash and may record a "StackOverflow Exception" error in the log. (For instructions on finding the ANTS Performance Profiler logs, see [Log files](#).)

To reduce the risk of an overflow, open the **Tools** menu, click **Options**, and select **Simplify very complex stack traces to save memory**.

If this does not prevent the crash, try profiling in sampling mode: on the **Application Settings dialog box**, set the **Profiling mode** to **Sample method-level timings (fastest, least detail, no hit counts)**.

### Getting more help

If the steps above do not prevent the profiled application from crashing, please [contact Redgate support](#), including, if possible, a log of the application crash.

## Troubleshooting missing results

ANTS Performance Profiler may show no profiling results, may show no results for some methods, or may show the error message "The profiler did not find any methods with source code", when:

- ANTS Performance Profiler cannot find usable *.pdb* files for your application
- ANTS Performance Profiler cannot find any code to profile
- ANTS Performance Profiler cannot read a performance counter

This page describes how to resolve these problems.

If your results are only missing SQL or HTTP call profiling data, see [Troubleshooting SQL and HTTP call profiling](#).

You may also receive no results, or incomplete results, if the application you are profiling crashes. For more information, see [Troubleshooting application crashes](#).

### ANTS Performance Profiler cannot find usable *.pdb* files for your application

See [Troubleshooting .pdb problems](#).

### ANTS Performance Profiler cannot find any code to profile

ANTS Performance Profiler may be unable to find your code if:

- The application you are profiling is not installed on the same computer as ANTS Performance Profiler.  
It is not possible to profile remotely using ANTS Performance Profiler. Ensure that the profiler is installed on the computer running the application you want to profile.
- You are profiling with 'Hide insignificant methods' selected.  
This setting hides any method that contributes less than 1% of your application's total CPU time. To display all profiled methods, clear the 'Hide insignificant methods' checkbox.
- Your application contains no managed code.  
Line-level and method-level timings are not available for unmanaged code. If all your application's code is unmanaged, it cannot be profiled.  
Method-level timings for unmanaged methods can be shown if the methods are declared with `extern` within your managed code.

#### For web applications and WCF services running in IIS:

- The application is using an unprofiled port.  
If you are profiling on an unused port, configure your application to communicate on the same port that you selected on the Application Settings dialog in ANTS Performance Profiler. By default this is port 8013. For more information, see [Profiling ASP.NET applications running on IIS](#).

### ANTS Performance Profiler cannot read a performance counter

If any performance counters are missing from profiling results, rebuild the counters and try profiling again.

To rebuild performance counters:

- For Windows 2000, Windows XP and Windows Server 2003: see [How to manually rebuild Performance Counter Library values \(MSDN\)](#)
- For Windows Vista, Windows Server 2008 and Windows 7: see [How to rebuild performance counters \(MSDN\)](#)

### Contacting Redgate support

If you are still unable to resolve this problem, [contact Redgate support](#). Ensure that you include:

- the version of ANTS Performance Profiler you are using
- your computer's operating system
- the steps you have already tried
- any error messages ANTS Performance Profiler has generated, including any in the log files.  
More information about locating log files for ANTS Performance Profiler can be found [here](#).

## Troubleshooting PDB problems

If ANTS Performance Profiler can't locate usable *.pdb* (debugging symbols) files for the application you are profiling, it can't display method source code or show line-level timings. You may receive the message "The profiler did not find any methods with source code."

This can indicate one or more of the following:

- No *.pdb* file exists for the application.
- A *.pdb* file exists, but ANTS Performance Profiler cannot locate it.
- A *.pdb* file exists, but is out of date.
  - A corrupt *.pdb* file causes ANTS Performance Profiler to crash.

This page explains how to resolve these issues.

To resolve other issues causing missing results, see [Troubleshooting missing results](#).

### No *.pdb* file exists for the application

The method for creating a PDB depends on the type of application you are profiling.

Note that, if the application's namespace includes multiple DLLs, ANTS Performance Profiler will look for a PDB for each of them. Line-level profiling results and source-code display will be available only for methods in those DLLs for which ANTS Performance Profiler can locate the PDB.

#### ASP.NET web applications:

1. Close ANTS Performance Profiler.
2. Open the application's *web.config* file.  
For instructions, see [How to enable debugging for ASP.NET applications](#) (MSDN)
3. Find the `compilation` tag and set its `debug` attribute to *true*.
4. Restart ANTS Performance Profiler and try to profile your application again.

Note that simply rebuilding the assembly will not work unless debugging is enabled. Once the *.pdb* file has been created, you can set `debug` to *false* without affecting it.

#### Other types of application:

See [Debug settings and preparation](#) (MSDN).

### Decompiling methods without source

If you have no usable PDB file for an assembly, you can still display source code in ANTS Performance Profiler using the integrated decompilation feature. Line-level timings will still be unavailable. For details, see [Working with integrated decompilation](#).

### A *.pdb* file exists, but ANTS Performance Profiler cannot locate it

For each of an application's assemblies, ANTS Performance Profiler looks for PDBs in the directory where the assembly's DLL is stored. (For ASP.NET web applications, this is by default the *bin* or *app\_bin* folder.) If the PDB for a profiled assembly is located somewhere else, move it into this folder to enable line-level profiling and source code display.

### Using a global PDB directory

If you are unable to move the PDBs for all your DLLs into a folder where ANTS Performance Profiler can find them (for example, if your application uses assemblies from the Global Assemblies Cache), you can bypass the problem by [creating a global debugging symbols \(PDB\) directory](#).

### A *.pdb* file exists, but is out of date

If the assembly has been changed since its *.pdb* file was generated, the PDB may be out of date. To update it:

1. Enable debugging by following the steps under "No *.pdb* file exists for the application" above.
2. Recompile your assembly.
  - a. Restart ANTS Performance Profiler and try to profile your application again.

Note that simply rebuilding the assembly will not work unless debugging is enabled.

### A corrupt *.pdb* file causes ANTS Performance Profiler to crash

If ANTS Performance Profiler encounters a corrupt PDB during profiling, the profiler may crash and fail to return results. The corrupt PDB might belong to a third-party DLL or to the application you want to profile.

There are two ways to profile an application with a corrupt PDB:

- Delete the corrupt PDB.  
This will prevent you from viewing source code referenced in the deleted PDB, but will display line-level code for all other methods.
- On the Application Settings dialog, in the **Profiling mode** dropdown menu, choose **Method-level timings; all methods (faster)**.  
This will prevent the corrupt PDB from being read, avoiding the cause of the crash, but will also prevent you from viewing source code referenced in the deleted PDB. Line-level timings are also unavailable in this profiling mode.

## Getting help

If you are still unable to resolve this problem, please [contact Redgate support](#). Supply as much information as you can in the **Description** box, including:

- the type of application you are trying to profile
- the version of ANTS Performance Profiler you are using
- your computer's operating system
- the steps you have already tried
- any error messages ANTS Performance Profiler has generated, including any in the log files. To locate log files for ANTS Performance Profiler, see [Log files](#).

## Creating a global debugging symbols (PDB) directory

ANTS Performance Profiler requires a PDB file corresponding to the assembly being profiled in order to locate source code for line-level timings and filtering functions. Normally, the PDB can be resolved if it is in the same folder as the assembly. Other methods for debugging symbols storage, such as a symbol server, aren't used by ANTS Profiler, but it is still possible to create a "global" symbol store by leveraging the features of the Visual C++ runtime.

To create a global symbol store for ANTS Profiler, create a subfolder called "Symbols" inside the %SYSTEMROOT% folder. Underneath this, create two more subfolders called "dll" and "exe". The dll folder can be used to store your class library symbols, while the exe folder is used to store the PDB files corresponding to executable assemblies that you may want to profile.

For instance, you can copy your class library debugging symbols to

c:\windows\symbols\dll

...and you can copy your executable symbols to...

c:\windows\symbols\exe

Please note that using this approach will more than likely cause problems if you have many versions of the same assembly that you want to profile. There is no way to store many versions of the same assembly's pdb in the exe or dll folders, unlike a proper symbol server that can segregate symbol databases into different folders by their version using the symstore.exe utility, for instance.



## Troubleshooting SharePoint Profiling

There are two main problems that you may encounter while profiling SharePoint:

- ANTS Performance Profiler cannot read from the directory which SharePoint writes data to.
- ANTS Performance Profiler cannot profile SharePoint because of security restrictions in ASP.NET.

### ANTS Performance Profiler cannot read from the directory to which SharePoint writes

The most likely cause of problems while profiling Sharepoint is that ANTS Performance Profiler cannot read from the directory which SharePoint writes data to. To fix this:

1. Create a temporary directory
2. If you are not on a sensitive system, allow full read/write access to this temporary directory to all users. If you are on a sensitive system, ensuring that the local system account has read/write access should suffice.
3. Use **Control Panel** to add a new environment variable. The variable must be called *RGIISTEMP* and the value is the path to the temporary directory you just created.

For more information, see Chris Allen's blog post, '[Profiling SharePoint with ANTS Performance Profiler 5.2](#)' (This blog post is also valid for ANTS Performance Profiler 7.4.)

### ANTS Performance Profiler cannot profile SharePoint because of security restrictions in ASP.NET

Security features in ASP.NET may cause problems on some systems. To fix this:

1. Obtain the required information.
2. Grant permissions.
3. Ensure that compilation with be in DEBUG configuration.
4. Copy PDBs and web part DLLs to the app\_bin directory.

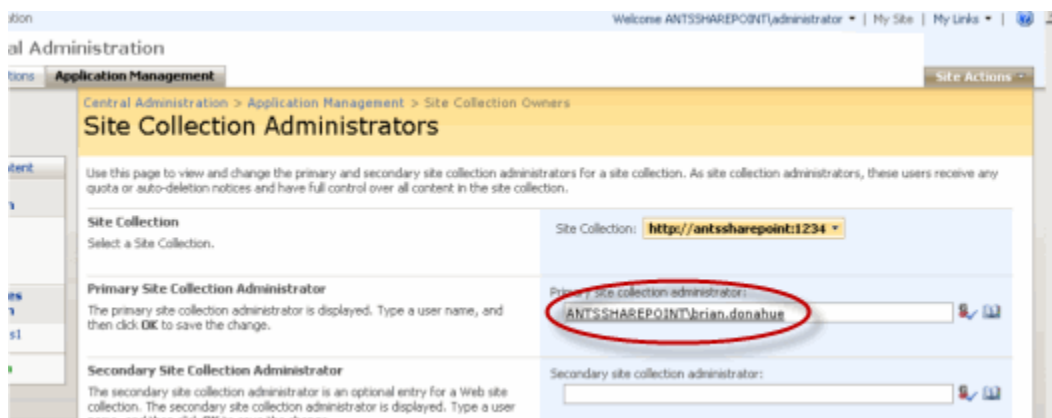
#### 1. Obtain the required information

Before you start, you need to know the following details for the site collection which you are profiling:

- the URL
- the TCP port
- the name of the primary site collection administrator
- the primary site collection administrator's password

To find the name of the primary site collection administrator:

1. Open the SharePoint Central Administration website using the Start menu item.
2. Click the **Application Management** tab.
3. Under the **SharePoint Site Management** heading, click **Site Collection Administrators**.
4. Select the name of the site collection hosting your web part from the dropdown list.
5. Make a note of the account set in the **Primary Site Collection Administrator** box.

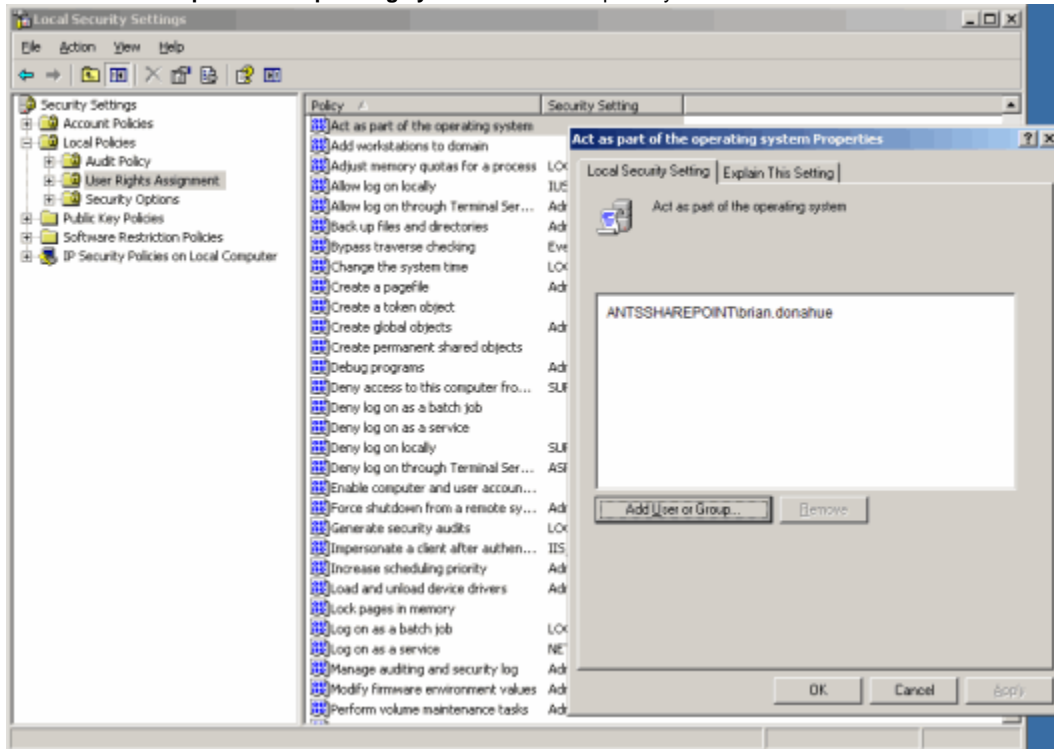


#### 2. Grant permissions

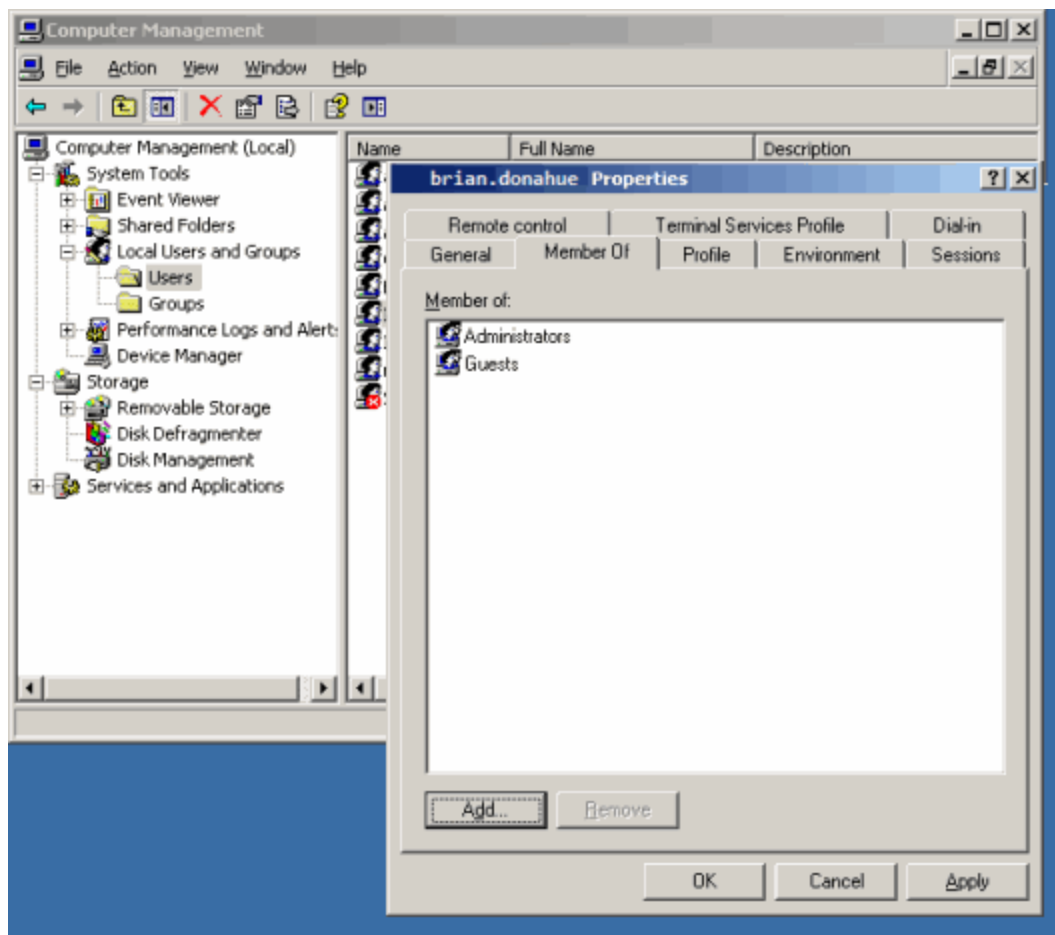
The primary site collection administrator must have permission to launch an IIS worker process. To grant this permission:

1. Open **Administrative Tools** then open **Local Security Policy**.
2. Under **Local Policies**, click **User Rights Assignment**.

3. Double-click **Act as part of the operating system** and add the primary site collection administrator's account.

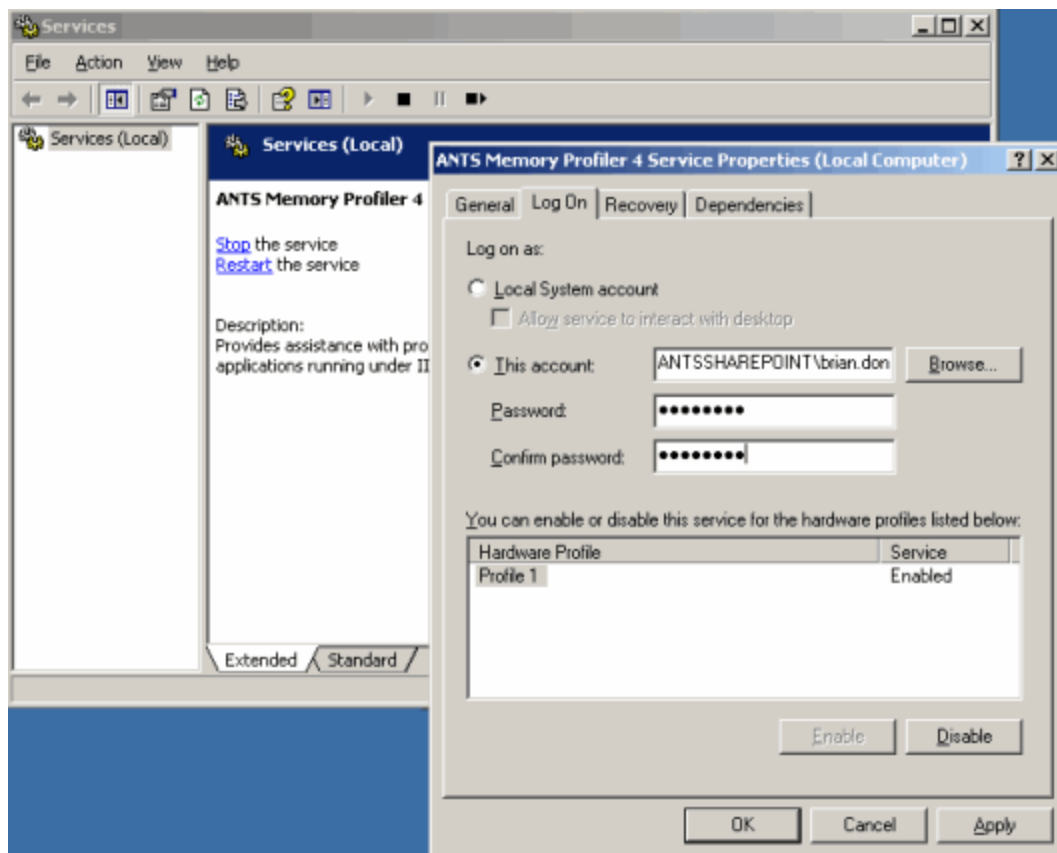


4. In the same way, double-click **Impersonate a client after authentication** and add the primary site collection administrator's account.
5. Open a command prompt and run `gpupdate /force` to enforce the new settings.
6. Open **Administrative Tools** and go to **Computer Management**.
7. Under **Local Users and Groups**, open **Users**.
8. Double-click the primary site collection administrator's account and open the **Member Of** tab.
9. Add the **Administrators** group.



The ANTS Performance Profiler Service must use the primary site collection administrator's account when it starts. To configure this:

1. Open **Administrative Tools** then open **Services**.
2. Double-click **ANTS Performance Profiler 7.4 Service**.
3. Click the **Log On** tab.
4. Select **This Account** and enter the primary site collection administrator's username and password.
5. Click **OK**.
6. If the status of ANTS Performance Profiler Service is *Started*, right-click the service and click **Restart**.



### 3a. Ensure that compilation will be in DEBUG configuration (IIS 7)

(Instructions for IIS 6 are below)

To set the application in DEBUG configuration in IIS 7:

1. Load **IIS Manager**.
2. Click the web application you want to profile.
3. Click the **.NET Compilation** option.
4. Under the **Behavior** group, set **Debug** to **True**.

Continue reading the instructions in section "4. Copy PDBs and web part DLLs to the app\_bin directory", below.

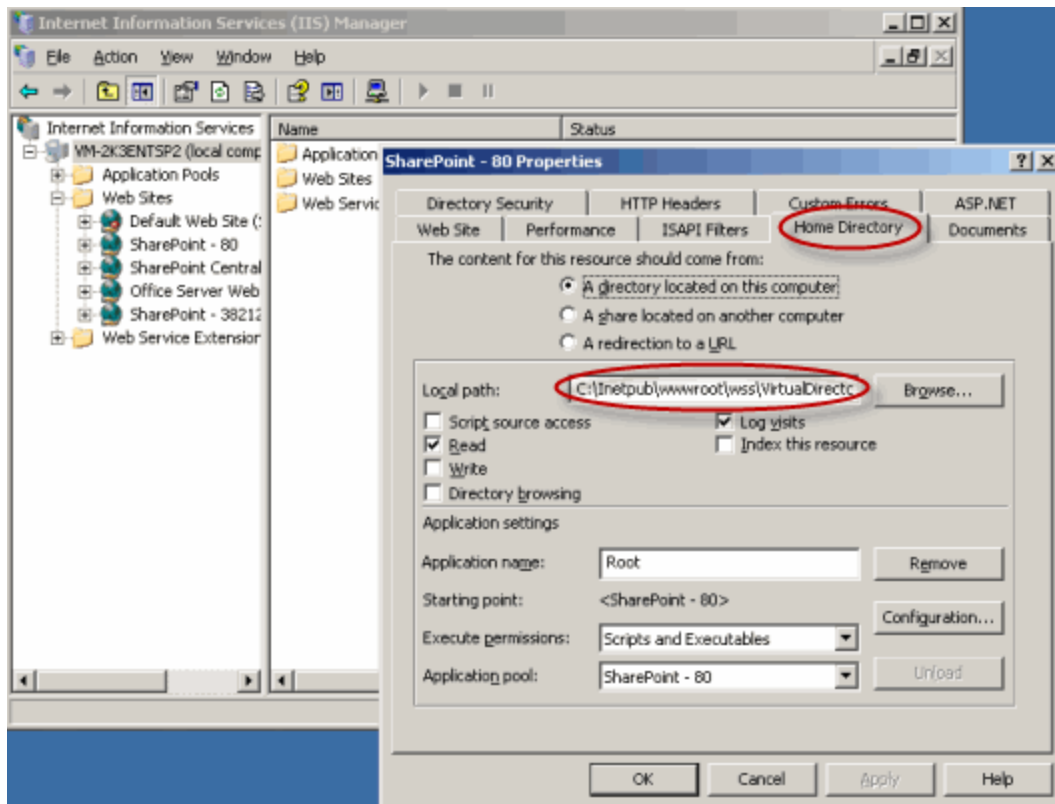
### 3b. Ensure that compilation will be in DEBUG configuration (IIS 6)

To profile a SharePoint collection, the ASP .NET compilation must be done in DEBUG configuration. This will allow ANTS Performance Profiler to locate the source code for any web parts or other extensions you have written for the site collection. DEBUG configuration will also relax some unmanaged code restrictions that prevent profiling and stop the site from timing out.

To set DEBUG configuration, you must know the physical path to the root of the site collection website.

To find this path:

1. Open **Administrative Tools**.
2. Open **Internet Information Server (IIS) Manager**.
3. Right-click the website containing the site collection then click **Properties**.
4. Open the **Home Directory** tab.
5. Note the path in the **Local path** box.



You must now locate and edit the *web.config* file for the site collection using an XML editor.

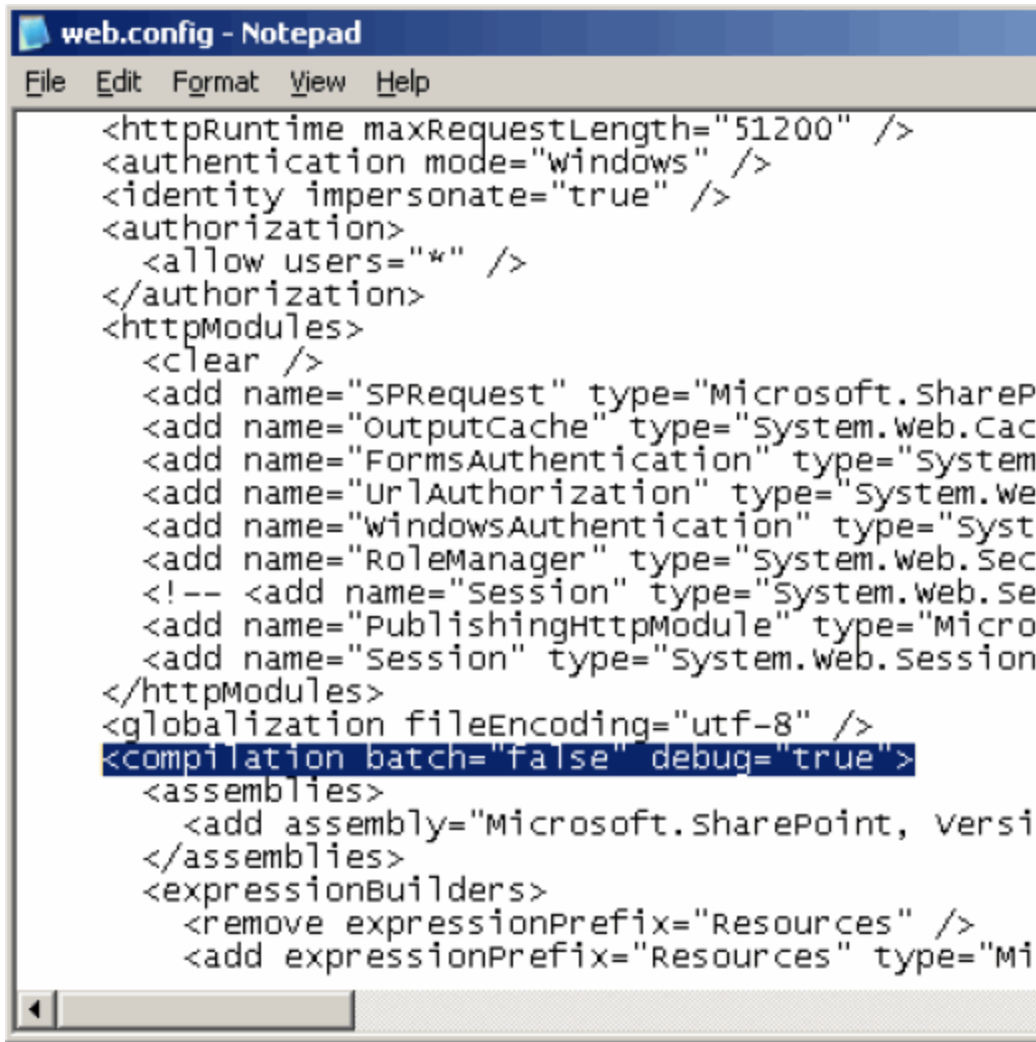
1. Use Windows Explorer to navigate to the site collection root's physical path.
2. Right-click the *web.config* file.
3. Open the *web.config* file using Notepad. Search for the text **debug**.
4. Change
 

```
<compilation batch="false" debug="false">
```

 to
 

```
<compilation batch="false" debug="true">
```

 and save the file.



```
<httpRuntime maxRequestLength="51200" />
<authentication mode="windows" />
<identity impersonate="true" />
<authorization>
  <allow users="*" />
</authorization>
<httpModules>
  <clear />
  <add name="SPRequest" type="Microsoft.ShareP
  <add name="OutputCache" type="System.Web.Cac
  <add name="FormsAuthentication" type="System
  <add name="UrlAuthorization" type="System.We
  <add name="windowsAuthentication" type="syst
  <add name="RoleManager" type="system.web.sec
  <!-- <add name="Session" type="System.Web.Se
  <add name="PublishingHttpModule" type="Micro
  <add name="Session" type="System.Web.Session
</httpModules>
<globalization fileEncoding="utf-8" />
<compilation batch="false" debug="true">
  <assemblies>
    <add assembly="Microsoft.SharePoint, versi
  </assemblies>
  <expressionBuilders>
    <remove expressionPrefix="Resources" />
    <add expressionPrefix="Resources" type="Mi
```

#### 4. Copy PDBs and web part DLLs to the app\_bin directory

If you want to be able to filter out all methods except the ones run by your code when viewing the results, you must copy the relevant files into the site's app\_bin directory. To do this:

1. Copy all PDB files and any web part DLLs used by your site to the Clipboard.
2. Use Windows Explorer to navigate to the site collection root's physical path.
3. Open the *app\_bin* directory.
4. Paste all PDB files and any web part DLLs used by your site into this directory.

Continue following the instructions in Setting up the Performance Profiler, above.

## Troubleshooting IIS profiling

When you click **Start Profiling** for an ASP.NET web application (IIS), profiling may not start, and a "Cannot start IIS" error may be displayed. This indicates one or more of the following:

- The logged-in user has insufficient account permissions to run the web application.
- Internet Explorer is running in protected mode.
- IIS is unable to resolve the web application's URL.
- ANTS Performance Profiler encounters a conflict with another performance profiler installed on your computer.

This page describes how to resolve these issues.

### The logged-in user has insufficient account permissions to run the web application

ANTS Performance Profiler starts the IIS application pool with permissions inherited from the **currently logged-in user account**, rather than using the IIS application settings.

- If possible, run ANTS Performance Profiler as an administrator.
- If you are unable to run ANTS Performance Profiler as an administrator, grant the logged-in user account permissions to access the IIS configuration system and write to the ASP.NET temporary files. For more information on how to do this, see [Assign ASP.NET Permissions to the New Account](#) (MSDN).

### Manually specifying the ASP.NET account

The error may also occur when using "Manually specify ASP.NET account details". Check that the specified account is a valid user, has administrator privileges, and has read access to `%ProgramFiles%\Red Gate\ANTS Performance Profiler 7\ProfilerCore.dll`.

### Internet Explorer is running in protected mode

If Protected Mode is turned on in Internet Explorer, when you click **Start Profiling**, a browser session may invisibly start and quickly terminate. If this occurs, Internet Explorer may launch with the message "Internet Explorer cannot display the webpage" or "Could not connect to the remote server", and no profiling results are displayed.

To prevent this error:

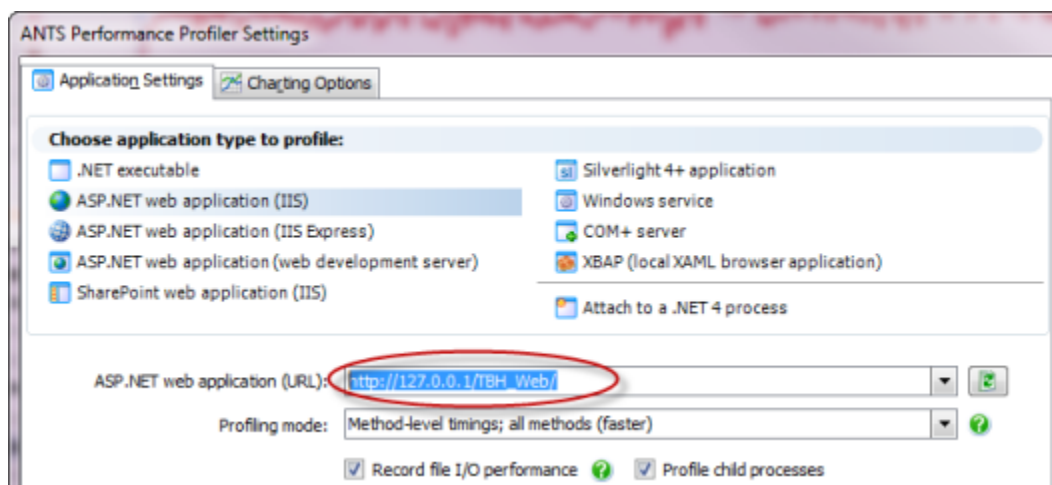
- If possible, turn off Protected Mode in Internet Explorer before starting a profiling session.
- If you need to use Protected Mode, add `localhost` to the list of trusted sites, and try to profile your application again. For instructions on adding a site to Internet Explorer's Trusted Sites list, see [Security zones: adding or removing websites](#) (Microsoft documentation).

### IIS cannot resolve the web application's URL

If the bindings in IIS have been changed from the default, ANTS Performance Profiler may be unable to resolve your site's hostname. If this problem occurs, the following error message is usually shown:

"Couldn't determine the IIS Site associated with URL 'http://< URL>:port'. Please check that the URL is serviced by the instance of IIS running on this machine."

In the **ASP.NET web application (URL)** field, enter `localhost` or the loopback IP address (`127.0.0.1`) and try to profile your application again:



## ANTS Performance Profiler encounters a conflict with another performance profiler installed on your computer

IIS can fail to start if ANTS Performance Profiler encounters a conflict with another performance profiler. We recommend uninstalling other profilers while profiling with ANTS Performance Profiler.

If you had ANTS Performance Profiler version 7.0 and installed the early access build of the continuous profiling tool, an IIS module installed by the continuous profiler may be preventing profiling. To re-enable other profilers with IIS, uninstall the IIS Profiler Module:

1. From your computer's Start menu, launch the **Continuous Profiling Configuration Tool**.
2. Click **Uninstall**.

For more information on configuring continuous profiling, see [Setting up continuous profiling](#).

### If you are running ANTS Performance Profiler Version 5.2 and earlier:

If the error persists after uninstalling other profilers, you may need to remove environment variables left behind by an earlier profiling session:

1. Close ANTS Performance Profiler and IIS.
2. In regedit.exe, locate the following registry key:  
If you are running IIS version 6.0 or earlier: *HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\IISADMIN*  
If you are running IIS version 7.0 or later: *HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC*
3. Expand the key and modify the **Environment** subkey to delete the following values:  
*COR\_ENABLE\_PROFILING=1* and *COR\_PROFILER={a GUID}*.  
**Note:** if the **Environment** subkey does not exist, please contact Red Gate support (see below for more details).
4. Close the Registry Editor, restart ANTS Performance Profiler, and try to profile your application again.

## Contacting Red Gate support

If you are unable to resolve this problem using the information in this topic, please [contact support](#), and supply as much information, including:

- the versions of IIS and ANTS Performance Profiler you are using
- your computer's operating system
- the steps you have already tried
- any error messages ANTS Performance Profiler has generated, including any in the log files. More information about locating log files for ANTS Performance Profiler can be found [here](#).



## Troubleshooting SQL and HTTP call profiling

ANTS Performance Profiler automatically captures performance data for inbound HTTP calls and for any SQL calls your application makes during profiling.

Missing SQL and HTTP calls in your results may be due to one or more of the following:

- You are using ANTS Performance Profiler Standard edition.  
SQL and HTTP call profiling are available in ANTS Performance Profiler Professional edition only.  
For information on upgrading ANTS Performance Profiler, see [Upgrading](#).
- You are profiling in sampling mode.  
SQL and HTTP call data are captured only in line-level and method-level profiling modes.  
To compare sampling, line-level, and method-level modes, see [Choosing a profiling mode](#).
- You are profiling an application compiled into an NGen profile image.  
ANTS Performance Profiler cannot intercept SQL and HTTP calls for NGen profile images loaded at runtime.  
To disable the NGen image for the profiled application, at the command prompt, enter `ngen uninstall * /profile` and start a new profiling session.

### Missing SQL calls only:

- Your application calls a database using unsupported ADO.NET drivers.  
Currently, ANTS Performance Profiler captures timing data for calls to all SQL, Oracle, and SQL Services (formerly SQL Azure) servers, wherever they are hosted.  
Calls to databases that use other ADO.NET drivers (including MySQL, MongoDB, and PostgreSQL) are not currently instrumented, so no timing data for these SQL calls are shown.  
You can request support for other server types at the [feature suggestions forum](#).
- Your SQL calls are made asynchronously.  
ANTS Performance Profiler 7.3 does not capture timing information for asynchronous SQL operations. `Begin` calls (e.g. `BeginExecuteReader`) to `SqlCommand` are not captured. Standard `Execute` commands (e.g. `ExecuteReader`) will be profiled.
- Your calls use a non-standard SQL API.  
ANTS Performance Profiler 7.3 instruments several standard interfaces for calling databases, but calls made via less common APIs may not be captured. If you think this is the case for your application, please contact Support, giving as many details as possible of your implementation.

### Missing HTTP calls only:

- The page consists entirely of static content, not content generated dynamically by the ASP.NET application.  
By default, ANTS Performance Profiler does not capture page load events that render only static content.  
If you want to see static content load events in the call tree, add the following setting to your application's *web.config* file:

```
<system.webServer>
  <modules runAllManagedModulesForAllRequests="true" />
</system.webServer>
```

## Error messages

- "No Disk" error occurring while profiling application
- Couldn't open metabase
- Error stopping IISAdmin profiling IIS web application on Windows XP
- Failed to CoCreate Profiler
- IIS ceases to work after profiling web applications
- Method not found: 'UInt32 <Module>.\_ANTS\_Begin\_Sql(System.String)'
- No .NET methods were profiled on web application
- Operation could destabilize the runtime error profiling ASP.NET
- Please specify a valid URL message profiling ASP.NET
- The system cannot find the file specified

## "No Disk" error occurring while profiling application

When performance or memory profiling an application, it may return an error message similar to:

MyApplication - No Disk

-----  
There is no disk in the drive. Please insert a disk into drive  
\\Device\\Harddisk1\\DR2

This can happen when the source code location matches a removable drive that has been disconnected. It does not matter if the software was built on a completely different computer - if the drive letter is the same as a disconnected removable drive on the computer used for profiling, this error will occur.

There are a few possible ways to fix this:

- Reconnect the removable drive
- Disconnect the removable device that normally mounts the drive
- Change the drive letter using Windows Computer Management Console (Disk Management)
- Delete the PDB files applicable to the code you are profiling

## Couldn't open metabase

This article relates to both ANTS Performance Profiler and ANTS Memory Profiler.

When profiling an ASP.NET web application hosted in IIS, an exception may occur at the start of profiling. Clicking **Show Details** reveals the following information, as well as some stack traces:

*Could not start IIS.*

*Couldn't open metabase - Please ensure that IIS is installed*

This error can happen when you are logged into the computer with an account that has reduced privileges.

To correct this:

1. Log out of the computer and log back in using an Administrator account, or one that has administrative privileges on the machine.
2. Open a command prompt, and change the working directory to the installation directory of the version of Microsoft .NET Framework that the ASP.NET application uses as its runtime.

For instance, if the web application that you want to profile uses ASP .NET 2.0:

```
cd \  
cd %systemroot%\microsoft.net\framework\v2.0.50727
```

3. Run the ASPNET\_REGIIS utility, which will grant permissions to the IIS metabase and ASP .NET temporary files locations. To allow a domain user called MyDomain\MyUser, this would be the correct command:

```
aspnet_regiis -ga MyDomain\MyUser
```

After logging the administrator account off and logging your low-privilege user back on, the account should now have access to IIS and ASP .NET appropriate for profiling the web application hosted in IIS.

## Error stopping IISAdmin profiling IIS web application on Windows XP

This article relates to both ANTS Performance Profiler and ANTS Memory Profiler.

When profiling a web application hosted in IIS 5.x (Windows XP), the following error may occur before code profiling actually takes place:  
*Error stopping IISAdmin*

Because the error occurs while stopping and starting IIS, the error message also covers a failure to START IISAdmin as well, so it is a good idea to check your application event log at this point. ANTS Performance/Memory Profiler will log the error actually returned by Windows at that location.

If the event log contains this entry:

*System.Exception: Timed out starting 'IISADMIN'*

the problem is actually a failure to start the service due to a timeout. In some cases, the default service timeout setting for Windows does not allow enough time for the service to start.

To increase the timeout, edit the following registry key:

*HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\WaitToKillServiceTimeout*

The value of this registry key contains the number of milliseconds that Windows will wait for a service to start, the default being 30000 (30 seconds). Increasing this to 60000 (60 seconds) or slightly higher may allow the IISAdmin service to start successfully.

If you change the registry key, restart the computer so that the changes will take effect.

## Failed to CoCreate Profiler

This article applies to both ANTS Performance Profiler and ANTS Memory Profiler.

If ANTS Performance/Memory Profiler is used to profile a Windows Service, it modifies the running environment of the service temporarily, then undoes the changes when profiling is stopped. If the service crashes with an unhandled exception during profiling, however, the modified environment may be left behind and the service will be permanently 'hooked' into ANTS Performance/Memory Profiler so the service will attempt to load the Profiler's "core" component at service startup time. Because ANTS Performance/Memory Profiler is not running, the following error entry is written to the application event log:

Source: .NET Runtime

Category: None

Event ID: 1022

Description: .NET Runtime version 2.0.50727.832 - Failed to CoCreate profiler.

In order to restore the normal environment of the service, the registry editor must be used. Click the start bar, then 'run', then type regedit and press enter.

Look for the short name of your service in the key:

*HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services*

Open the key that has the same name as your service, and in the right-hand pane, right-click Environment and select Modify.

Locate the entries *COR\_ENABLE\_PROFILING=1* and *COR\_PROFILER={a GUID}* and remove these name/value pairs from the registry value. When the service is restarted, the error should no longer occur.

## IIS ceases to work after profiling web applications

After profiling a web application hosted in IIS, IIS may stop working altogether, even when not profiling, usually resulting in a page claiming "Server Application Unavailable" when accessed by a web browser.

A change was introduced in version 4.3 of the performance profile that can permanently affect the state of Internet Information Services when the web application needs to write files to a temporary directory. If the profiler exits badly (like it crashes or you close it in task manager), the broken configuration remains. This will be addressed in a future version, but for now, there is a fix that should get IIS working properly again:

- Run the Windows Registry Editor (regedit.exe)
- Navigate to HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC
- Delete the value called "Environment"

## Method not found: 'UInt32 <Module>.\_ANTS\_Begin\_Sql(System.String)'

When profiling an application containing an assembly compiled into an [NGen](#) Profile image, profiling may fail with the error :

```
Method not found: 'UInt32 <Module>._ANTS_Begin_Sql(System.String)'
```

This error is caused by a bug in the [.NET profiling API](#), whereby the profiler loads the NGen profile image at runtime but cannot instrument it. NGen profile images are created by tools such as [BugAid](#), including for .NET framework assemblies, in order to improve application performance.

To enable profiling if this message has appeared, do one of the following:

- At the command prompt, enter `ngen uninstall * /profile` and start a new profiling session.  
This removes the NGen profile images and allows the profiler to instrument the application.
- Start a new profiling session in sampling mode.  
This prevents ANTS Performance Profiler from attempting to instrument the application, and avoids loading the NGen profile image.



## No .NET methods were profiled on web application

When profiling an ASP .NET web application, the following warning may be displayed when gathering the results:

No .NET methods have been profiled

There are four possible causes of this error:

- IIS is caching the previous request for an ASP .NET page, so that no code needs to be executed to satisfy the request. Caching can be configured in a page directive, causing this behavior.
- The profiler's core dll was not loaded into the application.  
When the .NET Execution Engine starts, it checks to see if a profiler is configured by examining the COR\_PROFILER and COR\_ENABLE\_PROFILING environment variables. If they are not set for the IIS worker process, profiling will not take place.
- An attempt was made to load the Core dll, but the wrong dll is registered, the registry is incorrect, or the dll file itself is corrupt or missing.
- The profiler's Core dll loaded correctly, but encountered an error.

### Verify that the .NET Execution Engine is loading

In this case, it's necessary to determine whether or not your application is executing managed .NET code in the runtime. This can be done by attaching a debugger or using a process monitoring tool such as Microsoft's Process Explorer.

If you notice that the ASP .NET worker process (aspnet\_wp.exe or w3wp.exe) has loaded the *mscorlib.dll* or "mscorlib.ni.dll" files, the runtime should have executed managed code. To determine the process that needs to be checked on IIS 6 and 7, the [iisapp.vbs](#) (IIS 6) or [appcmd.exe](#) (IIS7) tools can be used to resolve an IIS application pool to a Windows Process ID.

If the runtime is not present in your worker process, the site may contain nothing but static HTML, traditional ASP, other scripting code such as PHP, or the page is cached by IIS. You can determine the latter by looking for the following at the top of your .aspx files:

```
<%@ OutputCache Duration="Seconds" %>
```

### Verify that the process has a profiler configured

On IIS 6 and 7, the only way to check that a profiler will be attached to a worker process would be to examine the process environment variables for the COR\_PROFILER and COR\_ENABLE\_PROFILING variables, again, using [iisapp.vbs/appcmd.exe](#) and Process Explorer as in the previous case.

IIS 5 is slightly more complicated. Because IIS 5 does not implement the IIS 6 Worker Process Isolation Model, it is necessary to work out the user who will launch the worker process (aspnet\_wp.exe) and modify their user profile so that the above environment variables are set in the user environment. This way, any process launched by this user (default: ASPNET) will be profiled.

If the environment can't be modified, a different error message will be shown, so that possibility will not be discussed in this section. If the environment variables are incorrect, this would result in a "failed to cocreate profiler" error (see below). The remaining possibility is that the variables had been set in the wrong profile.

Because .NET supports the parallel installation of many runtime versions, it's important to ensure that you have chosen the correct .NET version in the web application profiling project wizard. The user who will run aspnet\_wp.exe is specified in the .NET installation's machine.config file, under the processModel section. If, for instance, your web application is configured to run .NET 2.0, which is configured to run as "USERA" and you also have configured .NET 1.1 to run as "USERB", choosing .NET 1.1 will set the environment for "USERB" and the profiler will not be attached to your web application. Normally, ANTS will automatically detect the correct .NET version, but if the metabase is corrupt or no application is configured for your web app's virtual directory, the wrong user could be configured.

### Verify that the Profiler Core is correctly registered

Rather than checking the registry, you can ensure the registration is correct by re-registering the Core dll. First, determine the location of the dll. In ANTS v3, it is in %programfiles%\Red Gate\ANTS Profiler 3\RedGate.Profiler.Core2.dll. In Performance Profiler 4, it is in %programfiles%\Red Gate\ANTS Profiler 4\ProfilerCore.dll. In Memory Profiler 4.0, it is at %programfiles%\Red Gate\ANTS Profiler 4\Memory\RedGate.Profiler.Core2.dll.

Open a command prompt, change the working directory to the folder containing the dll, and run  
regsvr32 <name of dll>

If this is successful, then retry profiling. If not, you may need to reinstall ANTS Profiler because the dll is corrupt.

### Verify that the dll is functioning correctly

If the Core dll does load successfully, it may have failed during initialization. When this happens, an entry is written into the Application event log,

with the source shown as ".NET Runtime":

Failed to coCreate the profiler.

Note that incorrect environment variables or incorrect registry information can also cause the same event log entry to appear. In that case, re-registering the dll as above or checking that the COR\_PROFILER environment variable is correctly set to the GUID corresponding to the dll would fix this.

[ANTS Profiler version 3.x]

Name: COR\_PROFILER Value: {9AE7D44D-DE91-47cb-9ABA-84BCAA0E5F54}

[ANTS Performance Profiler version 4.x]

Name: COR\_PROFILER Value: {A07CBDF0-B23B-4A96-A889-18E3C3004EB2}

[ANTS Memory Profiler version 4.x]

Name: COR\_PROFILER Value: {4befac55-31fa-4cf0-84b7-327248147851}

[ANTS Memory Profiler version 5.x]

Name: COR\_PROFILER Value: {D60F7519-2600-4865-8AC1-C621C9CE41A2}

[ANTS Performance Profiler 5.x]

Name: COR\_PROFILER Value: {60E3DCF2-1A65-4a44-9F41-038582C8181C}

If the environment is set correctly and the "Core" is registered, the only way to find the cause of the problem would be to attach a debugger to the worker process. Since there is probably a bug in the dll at this point, the only reliable way of troubleshooting this would be to send a memory dump of the worker process to [support@red-gate.com](mailto:support@red-gate.com). Microsoft's DebugDiag can be used to create this dump file.

< <http://www.microsoft.com/downloads/details.aspx?familyid=28bd5941-c458-46f1-b24d-f60151d875a3&displaylang=en> >

If it is not necessary to use IIS as a platform to run your web application, it is much simpler and less error-prone to use the "development web server" option to profile web applications. This application had shipped with .NET Framework 2.0 as a cut-down webserver to use for debugging purposes.

## Operation could destabilize the runtime error profiling ASP.NET

When profiling an ASP.NET web application, the following error may appear in the web browser window:

Operation could destabilize the runtime

This error occurs sporadically depending on the nature of the code being run in the website process and the causes of this message can be varied. This is because the error concerns a failure by the runtime to validate Intermediate Language code at a very low-level.

One possible fix for this problem is to enable full trust for the web application. To do this, open the web.config file in the web application's root directory and change the trust level, for instance:

```
<configuration>
<system.web>
    <trust level="Full" />
...

```

## Please specify a valid URL message profiling ASP.NET

When attempting to profile a web application hosted in IIS, the following message may appear in the "Application Settings" window:

Please specify a valid URL for your web application

The **Start Profiling** button is disabled.

Before profiling, ANTS Profiler attempts to check that the web address in the "URL" textbox has a valid structure by checking the scheme, host, and path components. If the "path" is empty, ANTS Profiler issues this warning, preventing profiling.

If the URL to be profiled is the root web application, a "forward-slash" character must be specified as the application path.

For instance, the URL *http://localhost* would be considered invalid. Changing this to *http://localhost/* will allow profiling to continue.

## The system cannot find the file specified

This page relates to both ANTS Performance Profiler and ANTS Memory Profiler.

When profiling an ASP .NET web application, ANTS Performance/Memory Profiler may throw the following exception:

Unable to start profiler - exception details

System.ComponentModel.Win32Exception: The system cannot find the file specified

ANTS Performance/Memory Profiler relies on an number of programs to be installed on the system to work properly. Some are web browser components and some are to verify that application pools are running. In short, this error can occur if one of the following files are missing:

- *%systemroot%\shdocvw.dll*
- *w3wp.exe*, the path to which is found in the registry at *HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\IISAdmin\ImagePath*, substituting *w3wp.exe* for *inetinfo.exe*
- **In IIS 6 only:** *%systemroot%\system32\cscript.exe*
- **In IIS 6 only:** *%systemroot%\system32\iisapp.exe*
- **In IIS 7 only:** *appcmd.exe*, located in the same folder as *w3wp.exe* above

## 3rd party components

- Profiling assemblies protected with DeployLX

## **Profiling assemblies protected with DeployLX**

If your application is protected and/or licensed using XHEO INC's DeployLX, ANTS Performance Profiler will crash on attempting to profile it. This is because DeployLX's copy protection features block ANTS Performance Profiler's access to your code.

To work around the problem, compile a version of your code without DeployLX protection and licensing enabled, then profile this version instead.

## Unexpected behavior / technical questions

- ANTS Performance Profiler menu items not showing in Visual Studio 2010
- Attach to process unavailable with some anti-virus software
- Call graph percentages do not add up exactly
- Can I profile Compact Framework applications?
- Double hit counts occurring on one line
- Enabling line-level timings for SecurityTransparent code
- Failed to coCreate Profiler on ASP .NET web application
- Forcing your application to use .NET 4
- HTTP request timings in IIS
- Isolating single ASP .NET pages in ANTS Profiler results
- Log files
- Memory leaks observed when profiling Windows Presentation Foundation (WPF) applications
- Missing hits for lines in the source code view
- Problems synchronizing results
- Profiler prompts for location of source code which is not your own source code
- Profiler stopping while profiling an in-browser Silverlight application
- Profiling an assembly in the Global Assembly Cache (GAC)
- Profiling ClickOnce applications deployed to IIS
- Profiling Microsoft Office managed-code add-ins
- Profiling unit tests using NUnit
- Profiling web services in IIS Express
- Setting file IO and child process profiling in high DPI modes
- Showing the amount of time taken for a method in one particular thread
- Times in source code window are greater than the times showing in the method grid or tree view
- Times on individual lines do not add up to method time
- Windows service profiling fails if the service uses a system account



## **ANTS Performance Profiler menu items not showing in Visual Studio 2010**

The ANTS Performance Profiler menu items may not show in the Visual Studio 2010 menu bar

Try deleting the following registry key:

HKEY\_CURRENT\_USER\Software\Red Gate\ANTS Profiler Visual Studio Add-in 1

That is where the location of the menu is stored. It should go back to the default location if you delete it. Then restart Visual Studio and see if the menu items are available.

If you still don't see the menu, try re-registering the add-in:

```
regasm /codebase "c:\Program Files\Red Gate\ANTS Profiler Visual Studio Add-in 1\RedGate.ANTSVsAddin.dll"
```

## **Attach to process unavailable with some anti-virus software**

When using ANTS Memory Profiler or ANTS Performance Profiler, you may find that you cannot attach to any .NET 4 process. This can be caused by anti-virus software.

To create the list of .NET 4 processes you can attach to, the profiler checks all running processes to check whether they are managed processes. For managed processes, the profiler obtains the CLR version number.

Processes that are part of some anti-virus programs protect themselves by changing their entries in the Access Control List. If a running process is protected in this way, the profiler cannot check the CLR version of that process, or of any other running processes.

To work around this problem, uninstall your anti-virus software while profiling.

## Call graph percentages do not add up exactly

After creating a call graph for a method in ANTS Performance Profiler, the percentage of time may not add up exactly across one row to match the percentage of time taken by the parent method.

In the example graph, the percentages in the second row should add up to "100%", but they add up to 98.9%.

```
| Program.Main (25.365) |  
|-----|  
| Application.Run 71.6% | | ctor 22.9% | | JIT Overhead 2.4% | | MethodA 1.2% | | MethodB 0.5% | | MethodC 0.3% |
```

There are a number of reasons why the percentages do not add up.

- Rounding on very small percentages
- Skew caused by ANTS Performance Profiler's overhead calculations
- Code that doesn't correspond to lines of code in the program, for example Visual Basic error handling code added automatically by the language compiler or the implicit Dispose() call added by the C# language after exiting a using block.

## **Can I profile Compact Framework applications?**

Can ANTS Performance Profiler profile applications for mobile devices running Windows CE and the .NET Compact Framework?

Managed-code Windows CE applications cannot be profiled at this time, however, once Microsoft create a code instrumentation API for the Compact Framework, support for CE will be re-evaluated.

## Double hit counts occurring on one line

ANTS Performance Profiler may be showing unrealistic hit counts in the source code window. For instance:

```
100 ->if (i=o)
200 -> { dosomething();
100 -> }
```

It may seem odd that the line with the if statement executes half as often as the code inside the condition. However, this is expected behavior for ANTS Performance Profiler. The curly-brace is a potential breakpoint in the code and therefore, technically, this line executes twice.

## Enabling line-level timings for SecurityTransparent code

ANTS Performance Profiler will not show line-level profiling results for methods from any module marked as SecurityTransparent (or, in .NET 3.5 and earlier, marked as partially trusted). ANTS Performance Profiler shows method-level timings for these methods, even if you have selected a line-level profiling mode.

To enable line-level timings for SecurityTransparent or partially trusted modules, modify the assembly and then profile the application again:

1. Remove the **AllowPartiallyTrustedCallers** attribute from the profiled assembly and from any of its dependencies.
2. Increase the code's trust level:
  - In **.NET 4**: Set the application's code to **SecurityCritical** or **SecuritySafeCritical** mode. (This requires the code to have full trust.)  
  
For instructions on setting these attributes, see [SecurityCriticalAttribute Class](#) or [SecuritySafeCriticalAttribute Class](#) (MSDN).
  - In **.NET 3.5** and earlier: Set the application's code trust level to **FullTrust**.  
  
For details, see [Named Permission Sets](#) (MSDN).
3. Rebuild and profile the application again.

## Failed to coCreate Profiler on ASP .NET web application

When profiling an ASP .NET web application, the profiling may yield no results. When examining the application event log, the following entry is shown:

Event Type: Error  
Event Source: .NET Runtime  
Event Category: None  
Event ID: 1022  
Date: 14/10/2008  
Time: 14:38:17  
User: N/A  
Computer: MINE  
Description:  
.NET Runtime version 2.0.50727.1433 - Failed to CoCreate profiler. For more information, see Help and Support Center at <http://go.microsoft.com/fwlink/events.asp>.

One possible cause for this error is that the ASPNET user does not have read permissions to the ANTS Profiler program directory. The ASP .NET worker process needs to load ProfilerCore.dll (RedGate.Memory.Core.dll for memory profiler) in order to send profiling metrics from the worker process to ANTS Profiler.

By default, ASP .NET web applications run under the local machine's ASPNET user account in IIS version 5. This is configurable, however, in the processModel section of the machine.config file. For example, if the ASP .NET web application uses the .NET 2.0 runtime, the location for the configuration file will be "%systemroot%\microsoft.net\Framework\v2.0.50727\config\machine.config". If the processModel is set to "Autoconfig=Yes", then ASP .NET is also using the ASPNET account.

In IIS version 6 and 7, the web application pool's Windows user is configurable. It's normally the user that runs the application pool (for instance NT AUTHORITY\NETWORK SERVICE).

Once you have determined the correct account that runs ASP .NET code, you may use Windows Explorer to grant READ permissions to the Profiler installation folder (%programfiles%\red gate\ants profiler x).

## Forcing your application to use .NET 4

To use the 'attach to process' feature in ANTS Memory Profiler 6 and above, and ANTS Performance Profiler 6 and above, your assembly must use the .NET 4 CLR.

You can change the CLR version that your application uses without rebuilding it. This allows you to use 'attach to process' on applications compiled against previous versions. Note that this will only work if you have not hard-coded a reference to a specific version of .NET in your code.

To force your application to use .NET 4:

1. Copy the following XML to a new file:  

```
<configuration>

  <startup>

    <supportedRuntime version="v4.0"/>

  </startup>

</configuration>
```
2. Save the file in the same location as the application's *.exe* file, adding *.config* to the *.exe*'s file name. For example, if your application is saved at *C:/Program Files/Company/Program/myApplication.exe*, you must save the XML file as *C:/Program Files/Company/Program/myApplication.exe.config*.
3. Run your application as normal. .NET will automatically load the *.exe.config* file and the application will run against .NET 4.
4. While the application is running, you can attach the profilers to it, as for a normal .NET 4 application.

Note: If the *\*.exe.config* file already exists, edit it with an XML editor to add the `<supportedRuntime>` node.



## HTTP request timings in IIS

In the call tree, when profiling applications hosted in IIS or IIS Express, timings and hit counts shown for HTTP nodes may be greater in **Methods with source** view than in **All methods** view.

This happens because the first time a user of the profiled application generates a content request, IIS automatically initiates an extra HTTP request to the same URL. These IIS-initiated requests call only framework methods, do not return content, and do not involve calls to methods with source.

ANTS Performance Profiler captures timing data for these IIS-initiated calls, and displays it according to the following logic:

- In **All methods** view, two nodes are shown for each requested URL: The first node displays timings for user-generated calls. The second (usually displayed lower on the call tree) displays timings for the IIS-initiated calls.
- In **Methods with source** view, only one node is shown for each requested URL. It aggregates the timings for user-generated calls (which may call methods with source) and for IIS-initiated calls (which only ever call framework methods).

For more information on HTTP node timings, see [Working with the call tree](#).

## Isolating single ASP .NET pages in ANTS Profiler results

ANTS Profiler does not offer any features specific to ASP .NET web applications, however, it is possible to mark the timeline of ANTS Profiler at the point that an ASP .NET page loads and unloads. This way the results can be filtered to more or less only the code that had been run by a particular page.

Creating an event on the ANTS Profiler timeline requires referencing one of ANTS Profiler's DLLs and adding a line of code to the page's "Page\_Load" and "Page\_Unload" event handlers. ASP .NET automatically looks for methods called "Page\_Load" and "Page\_Unload" on an ASP .NET code file by default, or when the "AutoEventWireup" page directive is used. One or both of these methods may exist in your code. If they do not, they can be created manually using the standard parameters for an event handler as shown in the examples below.

First, copy the file "RedGate.Profiler.UserEvents.dll" from the "%programfiles%\red gate\ants profiler 4" folder to the "Bin" folder of your web application. If the Bin folder does not exist, it can be created manually using Windows Explorer. Next, add the code necessary to trigger a user event in the ANTS Profiler timeline into your ASP .NET page code file:

```
public partial class _Default : System.Web.UI.Page
{
    ...
    protected void Page_Load(object sender, System.EventArgs e)
    {
        // Will run when the page loads
        RedGate.Profiler.UserEvents.ProfilerEvent.SignalEvent( "Default.aspx
Loaded." );
    }

    protected void Page_Unload(object sender, System.EventArgs e)
    {
        // Will un when the page unloads
        RedGate.Profiler.UserEvents.ProfilerEvent.SignalEvent( "Default.aspx
Unloaded." );
    }
    ...
}
```

When this page is saved and the web application profiled, the following events will appear at two different points in the timeline:

User event  
Message: Default.aspx Loaded.  
...  
User event  
Message: Default.aspx Unloaded.

Clicking in the timeline directly under the first event and dragging until the second event is reached will narrow down the code to only code run in the page "Default.aspx", provided there were no concurrent accesses to this page or background threads started by this page.

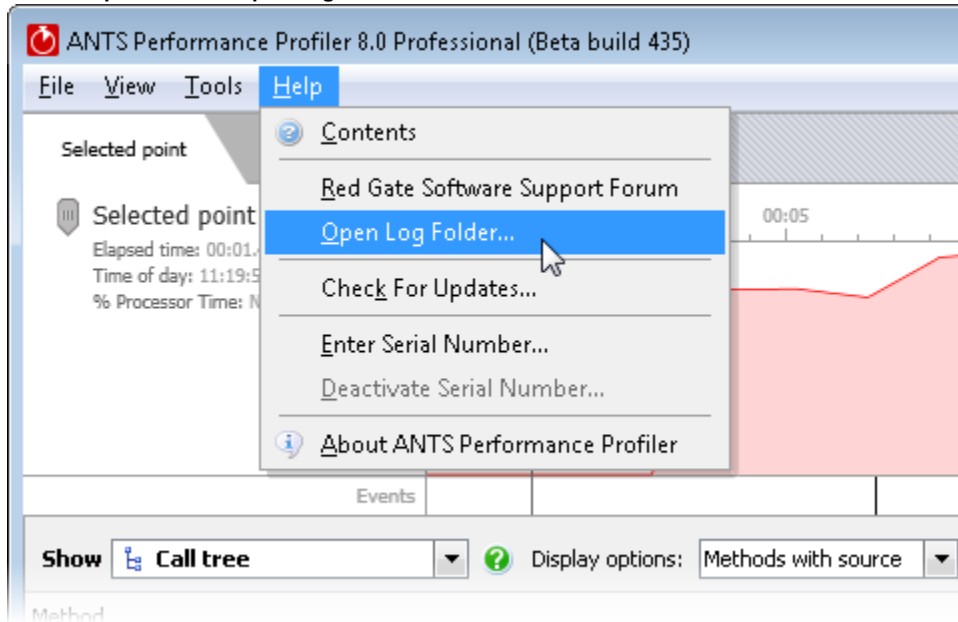
## Log files

The information on this page relates to ANTS Performance Profiler, ANTS Memory Profiler and Exception Hunter only.  
For other products, see [Log file locations](#)

Log files collect information about the application while you are using it. These files are useful to us if you have encountered a problem.

To open the folder where the log files are stored:

- On the **Help** menu, click **Open Log Folder**:



By default the log files are stored in:

`%ALLUSERSPROFILE%\Local Settings\Application Data\Red Gate\ProductName`

If you are profiling an ASP.NET application in IIS, additional log files for the service and the trigger process are created. These are stored by default in:

`%LOCALAPPDATA%\Red Gate\ProductName`

The local app data folder used is that of the user who launched the product.

### Changing the log file location

To change where log files are saved, create a new environment variable with the name *RGTEMP*. The value is the path for your log files.

The *RGTEMP* environment variable is also used by the profiler to save temporary results files during profiling. These temporary files could total several gigabytes, so ensure there is sufficient space at this path.

## Memory leaks observed when profiling Windows Presentation Foundation (WPF) applications

When profiling some applications based on the Windows Presentation Foundation (involving controls with collapsed visibility), the private bytes count may increase continually and never decrease.

This is caused a known bug in WPF for which Microsoft has released a hotfix. For more information, see [Microsoft Knowledge Base article 967328](#). You can find a full description of the issue on [Ramon de Klein's blog](#).

## Missing hits for lines in the source code view

When viewing the source code view of the ANTS Performance Profiler results, some lines of code in an individual method only show times and hit count for the first line.

Although ANTS Performance Profiler still shows the source code in the source code view, the settings used for the profiling session can disable line-level profiling. When you use the "Method-level timings; all methods" or "Method-level timings; only methods with source" profiling modes in the performance profiler settings, line-level information is not collected and therefore only the method times are shown in the source code view, by highlighting the start of the method and inserting the method-level time and hit count at that point.

In order to see the hit count and time data for each individual line of source code, you must select the "Line-level and Method-level timings" settings at the start of profiling.

## Problems synchronizing results

When using ANTS Performance Profiler, you may find that the results shown in the profiler's timeline do not match the results in the call tree. This can be caused by one of two problems:

### SpeedStep

Some cheaper motherboards (particularly laptop motherboards) use a technology called SpeedStep to change the CPU's clock speed dynamically, saving power and reducing heat. Leaving this technology enabled may cause the profiler's timeline to lose synchronization if the process being profiled is idle for long periods of time. Using SpeedStep also means that ANTS Performance Profiler cannot accurately estimate the length of time it takes to run a process until profiling is complete. For this reason, selecting a region on the timeline may not give the desired results, especially if you are still profiling the application.

### QueryPerformanceCounter

The alternative reason may be caused by the fact that ANTS Performance Profiler uses the QueryPerformanceCounter function in Windows to obtain timing information. Processors with AMD Cool'n'Quiet technology, and some Intel processors, can cause the QueryPerformanceCounter function to give inaccurate results to ANTS Performance Profiler.

In this case, the solution is to use the /USEPMTIMER flag in boot.ini to force QueryPerformanceCounter to use the Power Management timer instead of the Time Stamp Counter (TSC) timer. This solution will not work in Windows Vista or later, however, because the flag was removed in these editions.

### Sorting methods by time taken can change their order

When you sort methods by the time taken to execute them, and you change the units for displaying time from milliseconds to CPU Ticks, for example, you may find that the order changes. This is because, if the CPU is running slowly, a small number of ticks can equal a large amount of real time, which can alter the order in which the methods are displayed.

In most cases the CPU time is a better reflection of the effort required to run the method, and so this tends to be a more useful unit, if you are searching for performance issues.

## Profiler prompts for location of source code which is not your own source code

This article relates to both ANTS Performance Profiler and ANTS Memory Profiler.

ANTS Performance/Memory Profiler may prompt for the location of a source code file that is not your own source code when viewing the results.

When ANTS Performance/Memory Profiler produces performance profiling results in detailed mode or during memory profiling, information is collected about the assembly being profiled, including the method and object names, as well as the path to the source code files for these items. In some methods internal to the Microsoft .NET Framework, assemblies are compiled dynamically from source code internally without your knowledge. ANTS Performance/Memory Profiler also adds these methods and the source code file to its result set. Because the source code is created, compiled, and disposed of by the .NET Framework, it is no longer available and clicking on some internal Framework objects will trigger ANTS Performance/Memory Profiler to ask for the new source code location because the source code used to create these temporary assemblies no longer exists.

When this happens, the source code file name being requested is normally in the form of a globally unique identifier, for instance 'aaaaaaaa-bbbb-bbbb-bbbb-cccccccccc.vb'

This message can be safely ignored. Alternatively, [.NET Reflector](#) can be used to decompile source code from the executable.

## Profiler stopping while profiling an in-browser Silverlight application

During the profiling of a Silverlight application running in Internet Explorer, ANTS Performance or Memory Profiler may disconnect from the session and start summarizing results.

This may happen under the following circumstances:

- There is already an instance of iexplore running in the background, and the profiler may attach to that instance. To prevent this, use the Task Manager to check that no instances of iexplore.exe are running.
- Internet Explorer may be in "protected mode" as a result of restarting. Ensure that your IE settings allow it to **Run as administrator**.
- Javascript in the application may close the web window browser, signalling to the profiler that profiling is complete.

If all else fails, you may specify the .xap file and profile out-of-browser, if the Silverlight application is configured to run out of browser.



## Profiling an assembly in the Global Assembly Cache (GAC)

When profiling assemblies that are loaded from the Global Assembly Cache, no results may be shown when using the 'profile only methods with source code' setting of the performance profiler. The memory profiler may not show any source code for the methods in a particular assembly loaded from the GAC.

This occurs because the PDB (debugging symbols) file is not saved into the same folder as the assembly being profiled. When the assembly resides in the GAC, this is an additional complication, because the %systemroot%\assembly folder that contains all of the files in the GAC is abstracted by shfusion.dll, which automatically registers assemblies that are dropped onto the folder, making it difficult to add the PDB files into the same folder as the dll. There are two methods to solving this problem.

### 1. Create a global debugging symbols folder

=====

First, you can create a "global" PDB folder and copy your assemblies there. The disadvantage to this method is that you cannot have multiple version of the same assembly's PDBs in this folder, as you can different versions of the same DLL in the GAC folder. To create a PDB folder, create a folder called "symbols" in the system root folder, and a dll subfolder, for instance:

```
%systemroot%\symbols\dll
```

Copy your assemblies' PDB files into this folder, and ANTS Profiler can then resolve the source code location for that assembly installed in your GAC.

### 2. Manually copy the PDB to the GAC

=====

Secondly, you can copy the PDBs into the same folder as the assembly installed in the GAC. In order to view the GAC folder as a normal folder, you will need to unregister shfusion.dll from a command prompt, by typing and running:

```
regsvr32 /u %systemroot%\microsoft.net\framework\v2.0.50727\shfusion.dll
```

Now you may open the GAC folder by using start->run and typing and running %systemroot%\assembly\GAC\_MSIL.

Next, double-click on the folder that has the same name as your assembly. Locating the subfolder containing the assembly being profiles is a matter of knowing the version number. There will be a different subfolder for each individual version of the assembly that you have registered into the GAC. Copy the PDB file into the relevant folder.

Don't forget to put the GAC folder into its original state by re-registering shfusion.dll from the command prompt:

```
regsvr32 %systemroot%\microsoft.net\framework\v2.0.50727\shfusion.dll
```

Now ANTS Profiler will be able to filter the session to show only methods with source code, even though the assembly resides in the GAC.

## Profiling ClickOnce applications deployed to IIS

It is possible to profile ClickOnce applications that are installed locally or only available through Internet Information Server.

ClickOnce deployments send the application to the local computer to be run in a secure environment, so it is possible to profile applications whether they are deployed to the local installation of IIS or on a remote webserver running IIS. There are two ways to launch a ClickOnce application using ANTS Profiler. You may use Internet Explorer, or if you prefer, bypass IE and use the "shim" provided by Windows.

To profile a ClickOnce application using Internet Explorer, use the following settings:

- .NET Executable: c:\program files\Internet Explorer\iexplore.exe
- Arguments: <url to the clickonce application, ie <http://myserver/myapp/my.application>>

To profile a ClickOnce application without the need to launch IE:

- .NET Executable: c:\windows\system32\rundll32.exe
- Arguments: dfshim,ShOpenVerbApplication <http://myserver/myapp/my.application>

Provided that the profiler is configured to profile child processes, it will show results for the ClickOnce application once it has been deployed. In order to profile methods that have source or see results in the source code window, it may be necessary to profile the ClickOnce application on the same computer where it had been built and deployed to the local installation of IIS.

## Profiling Microsoft Office managed-code add-ins

This article relates to both ANTS Performance Profiler and ANTS Memory Profiler.

ANTS Performance/Memory Profiler can profile an add-in produced for any Microsoft Office application using Visual Studio .NET. Add-ins are typically implemented as a dll assembly loaded into the relevant Office application's process. Therefore, you would profile the application that loads your dll in order to get the performance and memory usage data for it from ANTS Performance/Memory Profiler.

For instance, to profile the performance of an add-in for Microsoft Excel, you would:

1. Start the profiler
2. Choose .NET Desktop Application.
3. Browse for and select Excel.exe (*c:\program files\Microsoft Office\OFFICE12\Excel.exe*).
4. Choose to profile **Only .NET methods that have source code** if your assembly has a corresponding PDB file, or choose another option.
5. Excel will start; perform whatever actions are necessary to invoke methods in your add-in.
6. Take snapshots (ANTS Memory Profiler) or view the performance profiling results in ANTS Performance Profiler.

The results reflect the managed code performance of your add-in's .NET code.

## Profiling unit tests using Nunit

Can ANTS Profiler provide performance and memory profiling results for assemblies under unit testing by NUnit?

ANTS Profiler can profile a unit test. The method for doing this is to set ANTS Profiler to profile a .NET desktop application, then choose the path to NUnit-gui.exe. Once ANTS Profiler starts NUnit for you, you can load your test assembly and run your unit test. During the unit test, you can take snapshots from ANTS Profiler, and profiling will stop when NUnit's window is closed.

Please note that if your NUnit installation is configured to use "Shadow Copy", ANTS Profiler may return incomplete results. If this happens, use NUnit's Tools->options menu to disable shadow copy.

It would also be possible to profile the console version on NUnit in the same fashion. In addition to choosing nunit-console.exe as the application to profile, specify the name of the assembly to load and the appropriate options in the Arguments box.

## Profiling web services in IIS Express

When profiling web services deployed in IIS Express, clicking **Start profiling** launches Internet Explorer with a 404 page displayed. The 404 page is shown because ANTS Performance Profiler launches the containing path for your web service, not the .svc file itself, and services in IIS are usually hosted without a default display page.

Although a 404 page is displayed, the service will still be launched and profiled if you leave the Internet Explorer window open. You can collect profiling results for the service by interacting with it via your usual client. Closing the browser window will end your profiling session.

To avoid displaying the 404 page, and to direct the browser to your service's page, do one of the following:

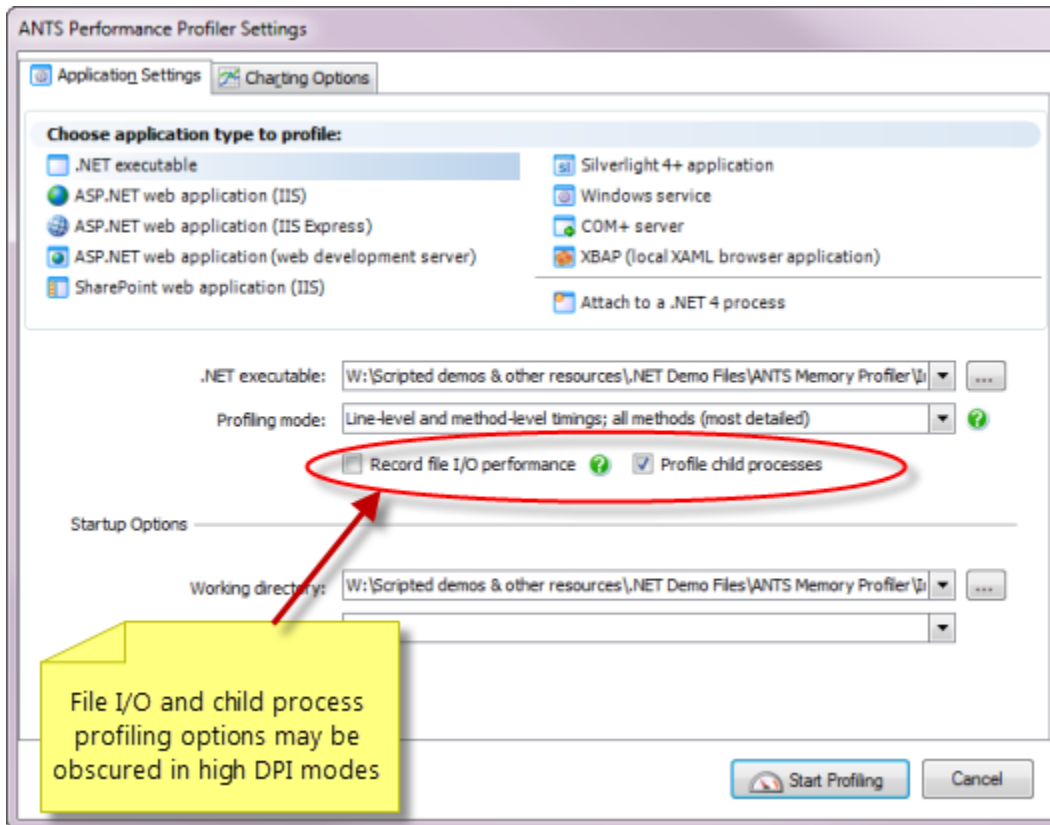
1. Edit your service's web.config file to add a default page for the service, by including the following lines:

```
<system.webServer>
  <modules runAllManagedModulesForAllRequests="true" />
  <defaultDocument>
    <files>
      <add value="RedGateService.svc" />
    </files>
  </defaultDocument>
</system.webServer>
```

2. Use Microsoft WebMatrix to create a default page for your service. For instructions on using WebMatrix, see [Creating a WebService with WebMatrix \(MSDN\)](#).

## Setting file IO and child process profiling in high DPI modes

In high DPI environments, parts of ANTS Performance Profiler's **Application Settings** dialog may be hidden. Above 120% DPI, the selection buttons for File I/O and child process profiling may not be shown.



This bug will be resolved in a future version of ANTS Performance Profiler. Meanwhile, to turn on these settings manually, edit the ANTS Performance Profiler Project Settings configuration file as follows:

1. In `C:\Users\[Username]\AppData\Local\Red Gate\ANTS Performance Profiler 7`, open the `APP7_LastProjectSettings.xml` file in an editing program (e.g. Notepad).
2. Locate the setting you want to activate, and set its `System.Boolean` value to `True`:

- **To profile file I/O**, replace

```
<Property Name="RecordEtwEvents">  
  
<Object Type="System.Boolean">True</Object>  
  
</Property>
```

with

```
<Property Name="RecordEtwEvents">  
  
<Object Type="System.Boolean">False</Object>  
  
</Property>
```

- **To profile child processes**, replace

```
<Property Name="ProfileChildProcesses">  
  
<Object Type="System.Boolean">False</Object>  
  
</Property>
```

with

```
<Property Name="ProfileChildProcesses">  
  
<Object Type="System.Boolean">True</Object>  
  
</Property>
```

The settings are saved until you edit them manually again.

3. To deactivate either setting, replace the relevant `System.Boolean` value with `False`.

## Showing the amount of time taken for a method in one particular thread

ANTS Performance Profiler's results screen has a dropdown list labeled "Thread". This allows you to choose results relative to a particular thread.

For instance, in the example code below, the same method is called from two different threads: one is called "FirstThread" and the other is called "SecondThread". Viewing the profiling results with "all threads" selected will show a hit count of "2" and a total time of "6 seconds", assuming ANTS Performance Profiler is set up to display wall-clock time.

Selecting "FirstThread" or "SecondThread" from the dropdown will show a hit count of "1" for the same method, and a total runtime of "3 seconds". The source code view will reflect the same behavior. The following is the code of the example C# application created to demonstrate this result:

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

namespace TestThreadTimes
{
    class Program
    {
        static void Main(string[] args)
        {
            Thread t1=new Thread(new ThreadStart(WaitThreeSeconds));
            t1.Name = "FirstThread";
            t1.Start();
            t1.Join();
            Thread t2 = new Thread(new ThreadStart(WaitThreeSeconds));
            t2.Name = "SecondThread";
            t2.Start();
            t2.Join();
        }
        static void WaitThreeSeconds()
        {
            Thread.Sleep(3000);
        }
    }
}
```



## **Times in source code window are greater than the times showing in the method grid or tree view**

When adding together the line-level timings for a method in the source code view, the sum may be greater than what is shown in the method views (the grid and tree views).

The line-level timings do not include the profiler's overhead correction, which subtracts the time that ANTS Performance Profiler believes it had spent doing the actual calculations that it needed to do in order to work out the method time. If you go to ANTS Performance Profiler's options and disable overhead correction, all of the times should match.

## **Times on individual lines do not add up to method time**

When viewing the source code view of the ANTS Performance Profiler results, the times on the individual lines of code for a method may not add up to the total method time.

ANTS Performance Profiler subtracts its own overhead from the method times in order to reflect the real-world performance of an application when it is running outside of ANTS Performance Profiler. The overhead calculation is not done for individual lines of code, however, causing the sum of the times on lines of code contained in the method to be greater than the method time.

The reason for this design is because the overhead calculation for individual lines of code will affect the performance of the application being profiled to an intolerable degree, so the line-level times remain more as a rough guide to the slowest individual lines of code.

## Windows service profiling fails if the service uses a system account

Profiling a Windows service may fail if the service runs under the *System*, *LocalService* or *NetworkService* account. If the failure occurs, profiling will stop shortly after starting and the service will not run.

The message "System.ServiceProcess.TimeoutException: Time out has expired and the operation has not been completed" may be written to the profiler log.

To work around this problem, run the service under a specific user account instead of a predefined system account.

This failure happens because ANTS Performance Profiler sets a registry value when it enables profiling: if the service runs as *System*, *LocalService* or *NetworkService*, the profiler cannot acquire the correct value, and sets an erroneous value instead. The service then uses the erroneous registry value to generate a write path: because the value is incorrect, the path is unusable, and the service fails. Running under a specific user account ensures ANTS Performance Profiler can read the system user environment variable, and will set the correct registry value.

## Release notes and other versions

<b>Version 9.3 (current)</b>	October 13th, 2015	<a href="#">Release notes</a>	<a href="#">Documentation</a>
<b>Version 9.2</b>	July 17th, 2015	<a href="#">Release notes</a>	
<b>Version 9.1</b>	April 10th, 2015	<a href="#">Release notes</a>	
<b>Version 9.0</b>	November 20th, 2014	<a href="#">Release notes</a>	
<b>Version 8.6</b>	July 1st, 2014	<a href="#">Release notes</a>	<a href="#">Documentation</a>
<b>Version 8.5</b>	January 23rd, 2014	<a href="#">Release notes</a>	
<b>Version 8.4</b>	December 6th, 2013	<a href="#">Release notes</a>	
<b>Version 8.3</b>	October 7th, 2013	<a href="#">Release notes</a>	
<b>Version 8.2</b>	August 1st, 2013	<a href="#">Release notes</a>	
<b>Version 8.1</b>	June 19th, 2013	<a href="#">Release notes</a>	
<b>Version 8.0</b>	April 17th, 2013	<a href="#">Release notes</a>	
<b>Version 7.4</b>	August 12th, 2012	<a href="#">Release notes</a>	<a href="#">Documentation</a>
<b>Version 7.3</b>	July 10th, 2012	<a href="#">Release notes</a>	
<b>Version 7.2</b>	May 29th, 2012	<a href="#">Release notes</a>	
<b>Version 7.0</b>	February 16th, 2012	<a href="#">Release notes</a>	
<b>Version 6.3</b>	July 12th, 2011	<a href="#">Release notes</a>	<a href="#">Documentation (PDF)</a>
<b>Version 6.2</b>	February 9th, 2011	<a href="#">Release notes</a>	<i>See version 6.3 documentation</i>
<b>Version 6.1</b>	November 4th, 2010	<a href="#">Release notes</a>	
<b>Version 6.0</b>	June 23rd, 2010	<a href="#">Release notes</a>	
<b>Version 5.2</b>	December 17th, 2009	<a href="#">Release notes</a>	<a href="#">Documentation (PDF)</a>
<b>Version 5.1</b>	July 13th, 2009	<a href="#">Release notes</a>	<i>See version 5.2 documentation</i>
<b>Version 5.0</b>	June 18th, 2009	<a href="#">Release notes</a>	
Before version 5.0, ANTS Performance Profiler was part of the ANTS Profiler product. No documentation is available for ANTS Profiler.			

If you need to install an old version of ANTS Performance Profiler, go to [Download old versions of products](#).

## **ANTS Performance Profiler 7.4 release notes**

**August 12th, 2012**

### **Bug fixes and enhancements**

- Adds average time to the first SQL result, useful where a SQL query runs more than once.
- Easier to find timing controls to change timing between milliseconds/seconds/percentage/CPU ticks.
- Supports .NET 4.5.
- Supports Visual Studio 2012.
- Improved the reliability of Find options
- Other bug fixes.

## What's new in version 7.4

Version 7.4 includes the following bug fixes and enhancements:

- Adds average time to the first SQL result, useful where a SQL query runs more than once.
- Easier to find timing controls to change timing between milliseconds/seconds/percentage/CPU ticks.
- Supports .NET 4.5.
- Supports Visual Studio 2012.
- Improved the reliability of Find options.
- Other bug fixes

For more information, see [ANTS Performance Profiler 7.4 release notes](#).

# ANTS Performance Profiler 7.3 release notes

## August 6th, 2012 - Version 7.3.1

### Bug fixes and enhancements

- Re-fixed the bug 'Application pools in IIS should now always restart with the correct version of the .NET framework, and in the correct mode (classic or integrated pipeline mode)', which was accidentally undone in ANTS Performance Profiler 7.3.
- Improved the user experience of the consent dialog box displayed by the Customer Improvement Program. This change will not be visible to users upgrading from ANTS Performance Profiler 7.3 or earlier.

## July 10th, 2012 - Version 7.3

### New features

This release introduces the following new features:

- **HTTP call profiling.** When profiling ASP.NET applications, you can now see inbound HTTP requests in the call tree. This gives you an instant overview of the HTTP calls made while your application ran, and helps you relate your results to user actions in your application: .NET methods are shown as children of the HTTP requests that initiated them. For details, see [Working with the call tree](#).
- **Method grid filtering.** Right-click on any method in the methods grid to hide or show only the selected method, class, or namespace. For details, see [Filtering the call tree and methods grid](#).
- **SQL Server Compact Edition support.** Calls to SQL CE servers are now displayed automatically in the call tree, with full timing and hit count data available in Database Calls view. For details of all servers supported, see [Working with Database Calls view](#).

### Bug fixes and enhancements

- You can now select a query in Database Calls view, and click to see its .NET parents in the call tree view. For details, see [Working with Database Calls view](#).
- The latest version of the ANTS Performance Profiler licensing client lets you optionally associate an email address and username with your activation, to help manage multiple licenses.
- ANTS Performance Profiler Standard edition now lets you save and export results - features previously limited to Professional edition.
- "Find" now works in more cases on the methods grid and call tree.
- This release fixes several race conditions that caused crashes on shut down or on loading results. It also includes a fix to prevent the profiler from misidentifying .NET application pools in IIS.
- ANTS Performance Profiler Standard edition now lets you save and export results (this feature was previously limited to Professional edition).

### Known issues

- In the call tree, when profiling applications hosted in IIS or IIS Express, timings and hit counts shown for HTTP nodes may be greater in "Methods with source" view than in "All methods" view. For full details, see [HTTP request timings in IIS](#).
- In higher DPI modes, the Application Settings options to record file I/O and to profile child processes may disappear. As a workaround, these options can be set manually: see [Setting file IO and child process profiling in high DPI modes](#).
- Unmanaged code that loads .NET COM objects may crash with a heap corruption exception when line-level profiling is used. If this error occurs, use [a profiling mode](#) that excludes line-level timings.
- If you have chosen a specific thread on the call tree view, the database call view link to switch back to the call tree will work only for calls that ran on the selected thread. To enable links, switch to the call tree, and select "All threads" on the results toolbar. For further details, see [Changing results display options](#).
- For some applications, the call graph may show erroneous hit counts for HTTP nodes. Refer to the call tree for correct counts.

# ANTS Performance Profiler 7.2 release notes

May 29th, 2012

## New features

This release introduces the following new features:

- **Database call profiling.** The Professional edition of ANTS Performance Profiler now automatically captures timings for any SQL call made by your .NET code, and displays them in the call tree. See how your .NET code calls any SQL, Oracle, or Amazon RDS server, with timing and hitcount data to help you understand bottlenecks in database performance. For details, see [Profiling SQL queries](#).
- **Remote SQL profiling support.** See performance data for calls to remote databases, as well as those that run on your local machine.
- **Redesigned database calls view.** See an overview of all the database calls your application makes, with details of timings and hitcounts. For details, see [Working with Database Calls view](#).
- **Call tree filtering.** Filter the call tree to hide (or show only) specific methods, classes, or namespaces. For details, see [Working with the call tree](#).

## Bug fixes and enhancements

- Windows Vista or later is no longer required for SQL call profiling.
- Assemblies and methods marked SecurityTransparent (or, in .NET 3.5 and earlier, marked as partially trusted) can now be profiled without crashing. Line level timings are by default unavailable for methods marked in this way, but line-level profiling can be enabled by modifying the assembly. For details, see [Enabling line-level profiling for SecurityTransparent methods](#).
- Application pools in IIS should now always restart with the correct version of the .NET framework, and in the correct mode (classic or integrated pipeline mode). This fixes an issue introduced in the last release.
- Streamlined display controls. Switch more easily between views of your application's call stacks, database calls, and file I/O performance. Include all methods, or only those with source.

## Known issues

- Support for the early access version of continuous profiling, present in version 7.0, has been removed from version 7.2.
- Results files created in version 7.2 cannot be opened in earlier versions of ANTS Performance Profiler. Results files created in earlier versions can be opened in the latest release.
- Assemblies compiled into NGen profile images, e.g. by tools like BugAid, may crash under instrumented profiling modes. For details see [Method not found: 'UInt32 <Module>.\\_ANTS\\_Begin\\_Sql\(System.String\)'](#).

ANTS Performance Profiler version 7.1 was an internal build and not given a public release.



# ANTS Performance Profiler 7.0 release notes

February 16th, 2012

## Features, enhancements, and bug fixes

This release of ANTS Performance Profiler adds full support for profiling SharePoint 2010 sites, web applications running in IIS Express, and web applications deployed to Windows Azure or Amazon EC2.

The new integrated decompilation tool, using .NET Reflector technology, lets you decompile methods without source directly from the ANTS interface. You can navigate decompiled source from third-party and framework assemblies as if it was your own code. If you have the PDB file, you can even view line-level timings for the decompiled code, letting you identify the exact code that caused the bottleneck.

The release also includes an early access version of the ANTS Performance Profiler continuous profiling tool, a new profiler mode under development at Redgate. It's designed to log performance data on live, running ASP.NET applications. The tool logs method calls as they happen on your site, and stores results to display in a web interface: browse back through the timeline to see historical performance data, and analyze problems as they really occurred.

- Watch your code execute while your site is used, with hit counts, timings, and CPU figures.
- View real data captured while performance issues occurred: no need to stop your site, relaunch with a profiler, and guess at reproduction steps.
- Access results directly in your web browser.

This early access build of continuous profiling features an automatic installer, a timeline showing CPU usage, and a call tree that lets you see the method calls executing during a selected period of your application's execution. For full details, see [Setting up continuous profiling](#).

## Supported .NET Framework Versions

- 1.1
- 2.0
- 3.0
- 3.5
- 4.0
- Silverlight 4
- Silverlight 5

The .NET 3.5 runtime must be installed in order for ANTS Performance Profiler to run.

## Supported OS Versions

- Windows XP SP2 or later
- Windows Server 2003
- Windows Vista
- Windows Server 2008
- Windows 7
- Windows Server 2008 R2

Supports 32-bit and 64-bit versions of all listed OSs. Windows 2000 is no longer supported.

Applications running on Windows Azure and Amazon EC2 platforms can also be profiled.

The following feature requires the .NET 4.0 runtime and Windows Vista or later:

- Attach to process

The following features require Windows Vista or later:

- File I/O counters
- SQL counters

## Supported Visual Studio Versions

The following versions of Microsoft Visual Studio are supported by the ANTS Performance Profiler 7 add-in:

- Visual Studio 2005
- Visual Studio 2008
- Visual Studio 2010

## System Requirements

- 512MB RAM (minimum)
- 1GB free hard disk space for profiler results

## Known Issues

- While the continuous profiler IIS module is installed, other profilers – including the desktop – will be unable to profile applications running in IIS on this computer. To re-enable other profilers with IIS, uninstall the IIS profiler module. See [Setting up continuous profiling](#).
- Line-level timings are not available for Silverlight applications.
- Sandboxed web parts created in SharePoint 2010 cannot be profiled as child processes of the parent SharePoint collection. Instead the dedicated SPUserCodeV4 process can be profiled as a Windows service. For details, see [Profiling SharePoint](#)
- SharePoint subsites do not appear in the dropdown SharePoint site selection menu in Application Settings. Subsites can still be profiled by entering the URL directly into the ASP.NET web application (URL) field.
- "My" sites in SharePoint can be profiled only on the original port, not on an unused port. For details on "My" sites, see [Introduction to "My" site \(MSDN\)](#).
- Some cheaper motherboards have an issue that causes their high performance timer to jump backwards or forwards in time by significant periods. This can cause anomalous results to appear in ANTS Performance Profiler 7, since the product relies on that timer to accurately measure application performance. If you experience this issue you should upgrade your motherboard drivers to the latest version.
- A limitation in Windows means that it is not possible to successfully attach to a .NET 4 process more than once.
- If you have Internet Explorer 7 on Windows Vista or later, it is not possible to profile websites on IIS.
- On Windows Vista and Windows 7, after profiling IIS using attach to process, ANTS Performance Profiler 6 sometimes cannot stop the session. If this happens, restart ANTS Performance Profiler before starting a new profiling session.
- In some cases, Windows Services do not restart properly after profiling.
- To profile an XBAP application, Internet Explorer must be set as your default browser and it must be closed before profiling.
- Custom event markers are not available when profiling Silverlight applications.

## ANTS Performance Profiler 6.3 release notes

July 12th, 2011

This version fixes a number of minor bugs present in version 6.2.

As of this version, profiling with Silverlight 5 beta is supported. Although full support for Sharepoint 2010 is not built into this version, a workaround may allow you to profile managed code running on a Sharepoint 2010 server with IIS 7. For details, see [Profiling Sharepoint 2010](#).

Please note that ANTS Performance Profiler 6.3 does not support Windows Phone 7.

## **ANTS Performance Profiler 6.2 release notes**

**February 9th, 2011**

This version removes a dependency on the Microsoft J# runtime, thereby making installation both quicker and more reliable.

A number of other minor bug fixes are also included.

Please note that ANTS Performance Profiler 6.2 does not support SharePoint 2010 or Windows Phone 7.

## ANTS Performance Profiler 6.1 release notes

**November 4th, 2010**

This version fixes a number of bugs present in version 6.0. In particular, the following issues present in version 6.0 have been fixed:

- On Windows Vista (32 bit), if you are profiling Windows Services or websites on IIS, you may have to run ANTS Performance Profiler as an administrator.
- If you have Internet Explorer 7 on Windows Vista or later, it is not possible to profile websites on IIS.

SQL and File I/O profiling is easier to use.

ANTS Performance Profiler 6.0 and 6.1 do not support SharePoint 2010 or Windows Phone 7.

# ANTS Performance Profiler 6.0 release notes

June 23rd, 2010

## Features, Enhancements and Bug Fixes

This release of ANTS adds full support for profiling applications running under .NET 4 and Silverlight. It also adds a new very low overhead profiling mode, the ability to visualise when a method was active on the timeline and a command line tool.

Users running Windows Vista, Windows Server 2008 or later operating systems will also be able to record and analyse SQL and File I/O activity associated with their target application, as well as attach the profiler to running .NET 4 processes without restarting them.

It also includes the vastly improved IIS support, along with XBAP support, that we first shipped with ANTS Profiler 4.3.

ANTS Memory Profiler 5, also available from [www.red-gate.com](http://www.red-gate.com), is our new, completely rewritten memory profiler offering incredible performance and an unparalleled feature set.

## Supported .NET Framework Versions

- 1.1
- 2.0
- 3.0
- 3.5
- 4.0
- Silverlight 4

The .NET 2.0 runtime must be installed in order for ANTS Performance Profiler to run.

## Supported OS Versions

- Windows XP SP2 or later
- Windows Server 2003
- Windows Vista
- Windows Server 2008
- Windows 7
- Windows Server 2008 R2

Supports 32-bit and 64-bit versions of all listed OSs. Windows 2000 is no longer supported.

The following feature requires the .NET 4.0 runtime and Windows Vista or later:

- Attach to process

The following features require Windows Vista or later:

- File I/O counters
- SQL counters

The following feature requires the .NET 2.0 runtime or later:

- Sampled timings

## Supported Visual Studio Versions

The following versions of Microsoft Visual Studio are supported by the ANTS Performance Profiler 6 add-in:

- Visual Studio 2005
- Visual Studio 2008
- Visual Studio 2010

## System Requirements

- 512MB RAM (minimum)
- 1GB free hard disk space for profiler results

## Known Issues

Line-level timings are not available for Silverlight applications.

Some cheaper motherboards have an issue with their high performance timer where it can jump backwards or forwards in time by significant periods. This can cause anomalous results to appear in ANTS Performance Profiler 6 because it relies on that timer to accurately measure application performance. If you experience this issue you should upgrade your motherboard drivers to the latest version.

A limitation in Windows means that it is not possible to successfully attach to a .NET 4 process more than once.

On Windows Vista (32 bit), if you are profiling Windows Services or websites on IIS, you may have to run ANTS Performance Profiler as an administrator.

If you have Internet Explorer 7 on Windows Vista or later, it is not possible to profile websites on IIS.

On Windows Vista and Windows 7, after profiling IIS using attach to process, ANTS Performance Profiler 6 sometimes cannot stop the session. If this happens, restart ANTS Performance Profiler before starting a new profiling session.

In some cases, Windows Services do not restart properly after profiling.

To profile an XBAP application, Internet Explorer must be set as your default browser and it must be closed before profiling.

Custom event markers are not available when profiling Silverlight applications.

## ANTS Performance Profiler 5.2 release notes

December 17th, 2009

ANTS Performance Profiler 5.2 is a bug fix release for ANTS Performance Profiler 5.x. Enhancements and fixes include:

- Profiling of IIS child processes now works, which enables easier Sharepoint profiling
- Fixed StackOverflowException when profiling Sharepoint
- Thread affinity no longer depends upon whether or not inlining is enabled
- Fixed performance issue with locating PDB files for source code view
- Fixed problem with logging configuration which caused some levels to be ignored
- No longer checks for updates on every start-up

This is a recommended upgrade for all users of ANTS Performance Profiler 5.x, particularly those working with Sharepoint.



## **ANTS Performance Profiler 5.1 release notes**

**July 13th, 2009**

- Installer fixes

## ANTS Performance Profiler 5.0 release notes

**June 18th, 2009**

For this release we have added a new profiling mode which offers line level timings, but only instruments methods with source code. This can offer a significant performance improvement when profiling applications that make extensive use of library code with a high level of call graph complexity.

It also includes the vastly improved IIS support, along with XBAP support, that we first shipped with ANTS Profiler 4.3.

Before version 5.0, ANTS Performance Profiler was part of the ANTS Profiler product.