.NET Business Transaction Monitoring

eG Innovations Product Documentation

www.eginnovations.com

Total Performance Visibility

Table of Contents

Table of Figures

Chapter 1: Introduction

Microsoft .NET is one of the most popular technologies in the web application development space, and provides the building blocks for many modern-day, business-critical web applications - eg., Windows client applications, client-server applications, distributed applications, database applications, etc. With millions riding on these .NET applications, it is only natural that administrators constantly fuss over "Application downtime" and "Application slowness". If users frequently complain of application inaccessibility or its poor responsiveness to transaction requests, the enterprise can lose dearly, in terms of revenue, support cycles, productivity, penalties, and reputation!

To avoid this, administrators should continuously observe user interactions with their .NET applications and measure overall user experience with these applications. At the first sign of user dissatisfaction, administrators should identify the precise transactions where users are experiencing slowness, accurately isolate its root-cause, and resolve the bottleneck, well before users complain!

This is where eG Enterprise helps! Using the eG Real User Monitor (RUM), administrators can track transaction requests to a .NET application, measure the responsiveness of each transaction, rapidly identify the slow transactions , and precisely pinpoint what is causing the slowness - is it a problem with the application front-end (i.e., browser)? a flaky network connection? or a server-side processing delay?

If eG RUM reveals issues in server-side processing, then the eG .NET BTM steps in to provide in-depth visibility into transaction performance across the server-side tiers. In the process, the eG .NET BTM leads you the exact source of your transaction troubles.

This document focuses on the **eG .NET Business Transaction Monitor (BTM)**, discusses how it works, and reveals how it helps ensure rapid diagnosis and resolution of delays in your mission-critical .NET transactions.

Chapter 2: The eG .NET Business Transaction Monitor (BTM)

The **eG .NET BTM** employs an advanced 'tag-and-follow' technique to trace the path of each business transaction to a .NET application hosted on an IIS web server. When doing so, it auto-discovers the CLRs the transaction travels through, and also automatically ascertains what remote service calls were made by the transaction when communicating with the CLRs. In the process, the eG .NET BTM measures the following:

- The total response time of each transaction;
- The time spent by the transaction on each CLR node in the path;
- The time spent by the transaction for processing every external service call (including SQL queries);

The eG .NET BTM is also capable of tracing a transaction across hybrid Java and .NET architectures. For instance, if a transaction request to a .NET application makes an HTTP or a Web services call to a Java application, then the eG .NET BTM is capable of discovering this relationship.

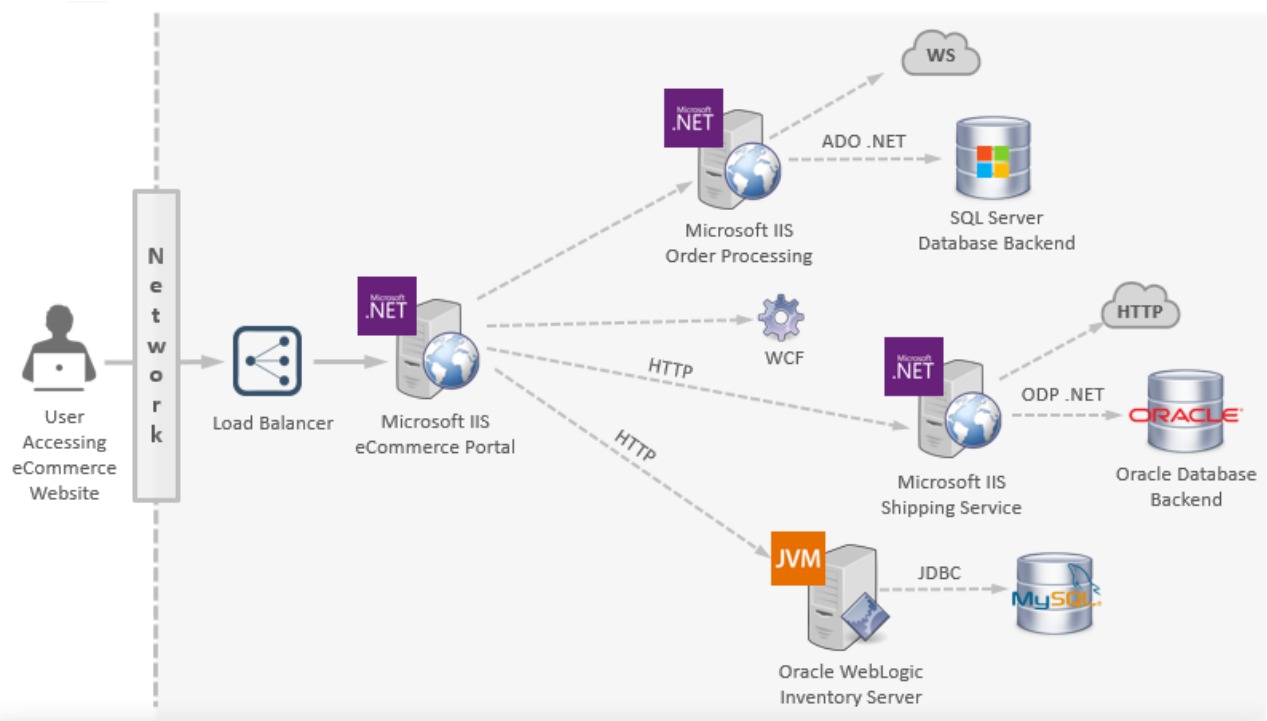


Figure 2.1: Tracing a .NET transaction

Using these analytics, the eG .NET BTM precisely pinpoints the slow, stalled, and failed transactions to the .NET application, enables administrators to accurately isolate where – i.e., on which IIS web server – the transaction was bottlenecked, and helps them figure out exactly what caused the bottleneck – an inefficient or errored query to the database? a slow HTTP/S call to another web / web application server? an issue in asynchronous communication with another server over WCF? a sluggish web service call? or an error in the application code? By quickly leading administrators to the source of transaction failures and delays, the eG .NET BTM facilitates rapid problem resolution, which in turn results in the low downtime of and high user satisfaction with the .NET application.

2.1 How does the eG .NET BTM Work?

The eG .NET BTM is currently capable of monitoring transactions to web sites / web applications hosted on an IIS web server only.

To be able to track the live transactions to web sites on an IIS web server, eG Enterprise requires that a special **eG .NET Profiler** be deployed on that IIS web server.

If more IIS web servers are in the transaction path, then, the profiler will have to be installed on each of the IIS web servers, for end-to-end visibility.

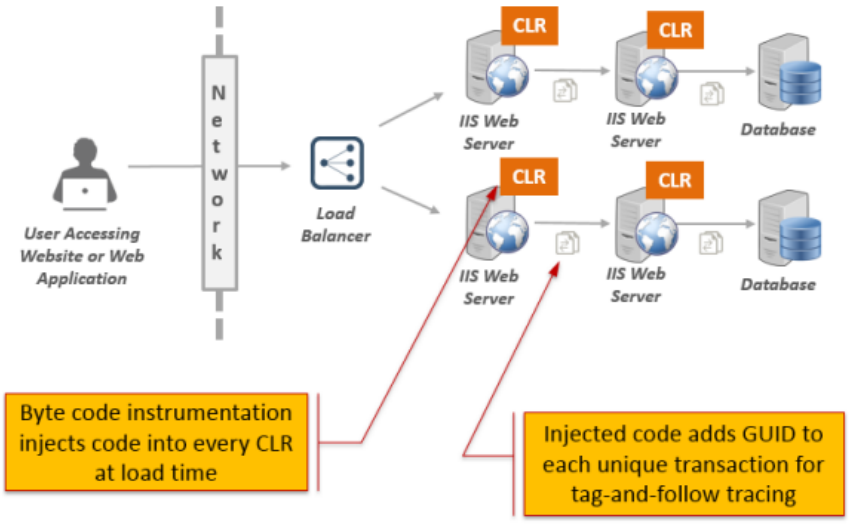


Figure 2.2: How eG .NET BTM Works?

Typically, requests to web site transactions are handled by application pools on an IIS web server. Whenever an end-user requests for a transaction, the application pool spawns a worker process (w3wp.exe) to service that transaction request. Upon receipt of a request, the worker process

automatically invokes an instance of the .NET CLR to process the request. At the same time, the worker process also loads an instance of the .NET profiler.

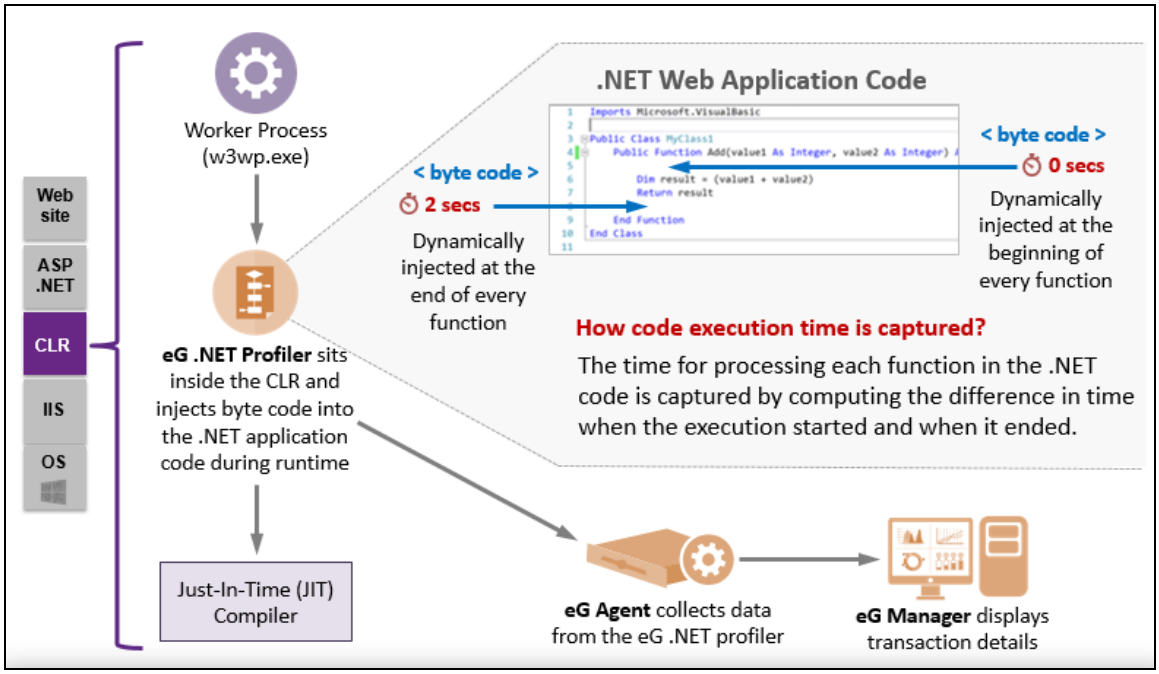


Figure 2.3: How eG .NET BTM traces transaction path and computes transaction responsiveness

Once the profiler latches on to a worker process, it injects a .NET code into the .NET application code. The injected code adds a GUID to each unique transaction to the application, and performs the following tasks:

- Traces the path of a transaction;
- Measures the responsiveness of a transaction by computing the time difference between when the transaction started and when it ended;
- Identifies the slow, stalled, and error transactions, and computes the count of such transactions;
- Discovers the exit calls made by a transaction from the IIS web server, determines the destination of the calls, and measures the average time taken by each call to process the requests for a transaction;

The profiler then sends all these statistics to the eG agent. To know how and when the profiler transmits metrics to the eG agent, refer to the How does the .NET Profiler Communicate with the eG Agent topic.

The eG agent reports these metrics to the eG manager. The eG manager associates these metrics with the **.NET Transactions** layer of the target IIS web server, and displays them in the eG monitoring console.

Figure 2.4: The .NET Transactions layer displaying the .NET business transactions that are being monitored on the target IIS web server

2.2 How Does the .NET Profiler Communicate with the eG Agent?

The eG agent should be deployed on the same IIS web server that hosts the .NET profiler. The profiler communicates with the eG agent via that port number 14001, by default. You can change the default port by following the steps below:

- Edit the eg_DotNetServer.ini file (in the <EG_AGENT_INSTALL_DIR>\agent\config directory)
- Configure the **PORT** parameter in the [EG_DOTNET_SERVER_DATA] section of the file with the new port number.
- Finally, save the file.

Typically, once the eG agent is configured with the details of the web site to be monitored, the profiler contacts the eG agent and downloads these details from it.

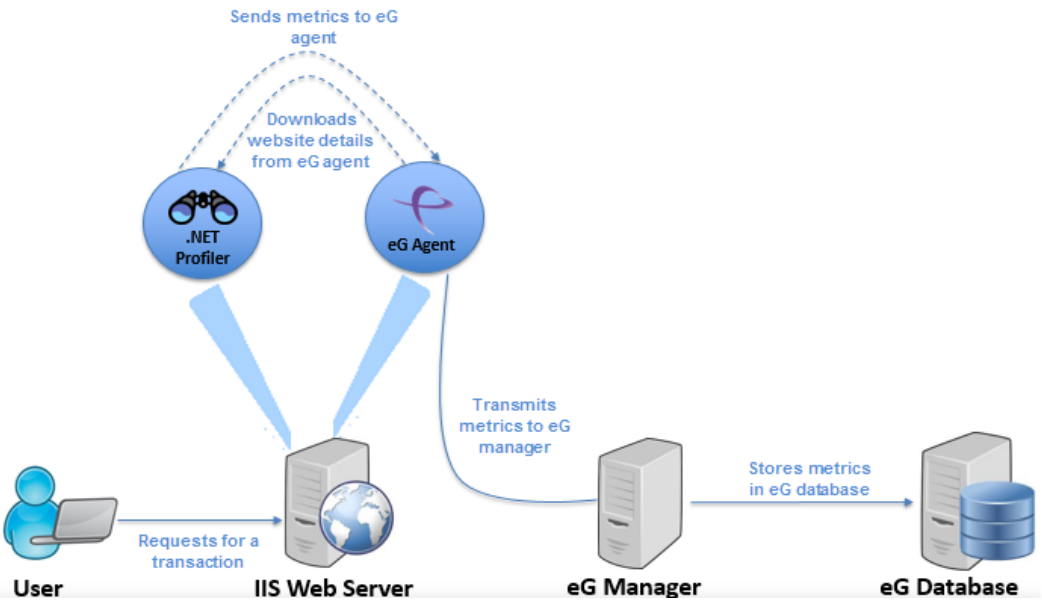


Figure 2.5: Communication between the .NET Profiler and the eG Agent

Then, when a transaction request for the web site comes in, the profiler injects a code in the CLR to trace the path of that request. In the process, the profiler also collects response time metrics related to that transaction. Every 10 seconds, the profiler sends these metrics to the eG agent. The eG agent stores these metrics in memory, until the next time it runs the .NET Business Transactions test. When the test is run, the agent pulls the metrics stored in memory and sends it to the eG manager.

2.3 Pre-requisites for .NET Business Transaction Monitoring

The following are the pre-requisites for performing .NET business transaction monitoring using eG:

- For the eG .NET Business Transaction Monitor to function, your eG Enterprise infrastructure should include:
  - An eG Manager of version 6.3 (or above)
  - eG Agents of version 6.3 (or above)
- The eG .NET BTM can be installed on IIS web servers 7.0, 7.5, 8.0, 8.5, or 10.
- Make sure that the VC++ 2012 Runtime is available on the target IIS web server, prior to BTM-enabling it. On a 64-bit server, both the 32-bit and the 64-bit versions of the VC++ 2012 Runtime should be available. On a 32-bit server on the other hand, make sure that the 32-bit VC++ 2012 Runtime is available.

- Ensure that the **IIS Management Scripts and Tools** feature is installed and enabled on the target IIS web server.
- If any other profiler - eg., NewRelic, AppDynamics, etc. - pre-exists on the IIS web server, then, before BTM-enabling the server, make sure that the profiler is fully and properly uninstalled.
- The eG .NET BTM is supported only in the following environments:
  - **Supported operating systems**
    - Microsoft Windows Server 2008 (32-bit and 64-bit)
    - Microsoft Windows Server 2008 R2
    - Microsoft Windows Server 2012
    - Microsoft Windows Server 2016
    - Microsoft Windows 7, 8, 8.1, 10
  - **Supported Frameworks**
    - Microsoft .NET Framework versions 3.5, 4.0, 4.5, 4.5.2, 4.6. 4.7.2
    - ASP .NET MVC 2, 3. 4, 5
    - Open Web Interface for .NET (OWIN) web API
  - **Supported Runtime Environments**
    - Microsoft IIS versions 7.0, 7.5, 8.0, 8.5,10
    - Microsoft SharePoint 2010, 2013 as services running inside IIS
  - **Supported Remote Procedure Calls**
    - HTTP
    - WCF
    - Web Services including SOAP
  - **Supported Data Storage Types and Clients**

    The .NET profiler supports the ADO.NET data storage type and the following ADO.NET clients:

Database Name	Client Type

Oracle	ODP.NET and Microsoft Provider for Oracle
MySQL	Connector/Net and ADO.NET
Microsoft SQL Server	ADO.NET

- For complete visibility into the transaction path, make sure that you BTM-enable each IIS web server in the transaction path.
- To track and profile transactions to multiple web sites on an IIS web server, you need to add a separate *Microsoft IIS* component in eG Enterprise for each web site to be monitored.

2.4 Installing and Configuring the .NET Profiler on an IIS Web Server

To install the .NET profiler on the IIS web server, follow the steps below:

1. Login to the target IIS web server.
2. From the command-prompt, switch to the <EG_AGENT_INSTALL_DIR>\lib directory.
3. Now, issue the following command from that location:

   run SetupDotNetProfiler.bat

4. Once the batch file begins execution, the following messages will appear:

```
-----------------------------------------------------------------------
*** Setting up Registry values for the Profiler ***
-----------------------------------------------------------------------
Modifying the W3SVC service's Environment variables
Modifying the WAS service's Environment variables
```

5. If setup detects that a profiler (this can be an eG;s .NET profiler or any third-party profiler such as NewRelic) pre-exists on the host, then the following message will appear:

```
There is another profiler found in registry
-------------------------------------------------------------
COR_PROFILER=<71DA0A04-7777-4EC6-9643-7D28B46A8A41>
-------------------------------------------------------------
```

Setup will then prompt you to confirm whether you want to proceed with the installation or not.

```
Do you wish to continue setup <yes/no> : yes
```

If you specify **yes** here, then setup will try to overwrite the GUID presently registered with the COR_PROFILER registry key with the GUID of the eG Enterprise profiler. If the GUID is overwritten successfully, then a message to that effect will appear.

```
-----------------------------------------------------

RegValues are modified Successfully ***

-----------------------------------------------------
```

6. If setup is successful, then the following messages will appear:

```
Registering the Profiler.

Installing assembly in Cache...

Helper assemblies Installation Successful

Applying required permissions for eGurkha directory.
```

7. Finally, restart the IIS web server by issuing the **iisreset** command at the prompt. This will restart all the web sites and web applications on that IIS web server. If you would rather have the monitored web sites / web applications alone to be restarted, do the following:

   - Open the Internet Information Services (IIS) Manager (see Figure 2.6).

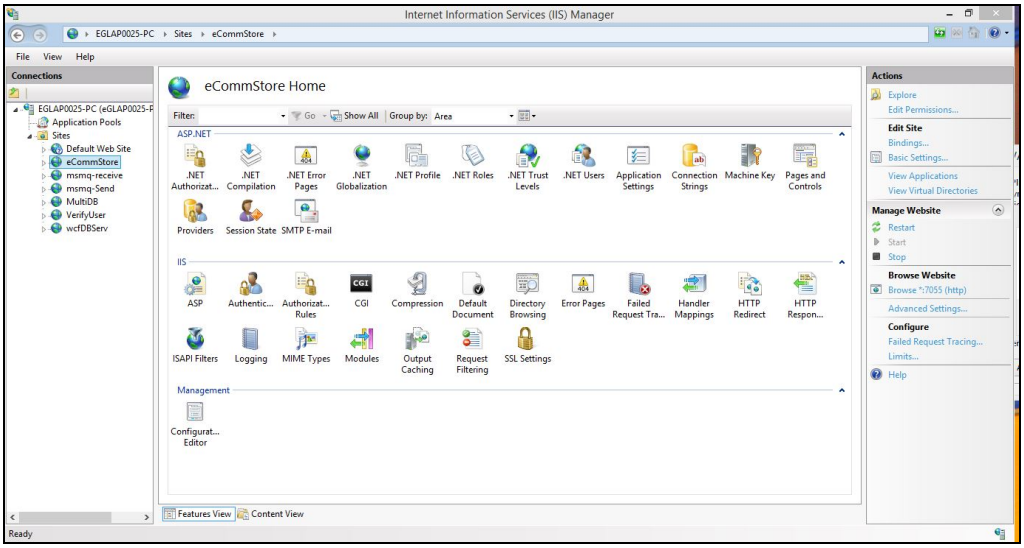


Figure 2.6: The IIS Manager console

- Expand the Sites node in the tree-structure in the left panel of Figure 2.6, and select the sub-node representing the web site / web application you are monitoring. Right-click on that sub-node, move your mouse pointer over **Manage Website** in the shortcut menu that appears, and pick the **Advanced Settings** option (see Figure 2.7).

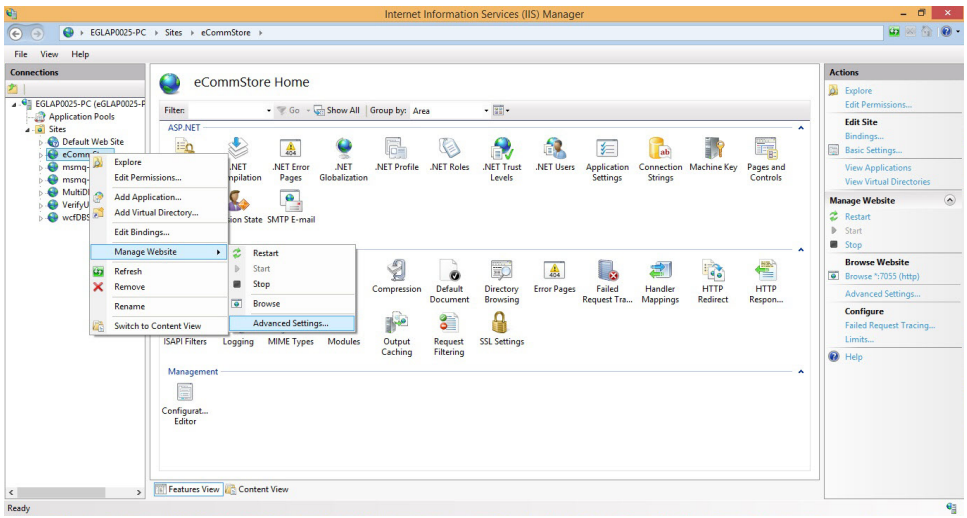


Figure 2.7: Selecting the Advanced Settings option

- Figure 2.8 will then appear. From Figure 2.8, you can figure out which **Application Pool** manages the target web site / web application.

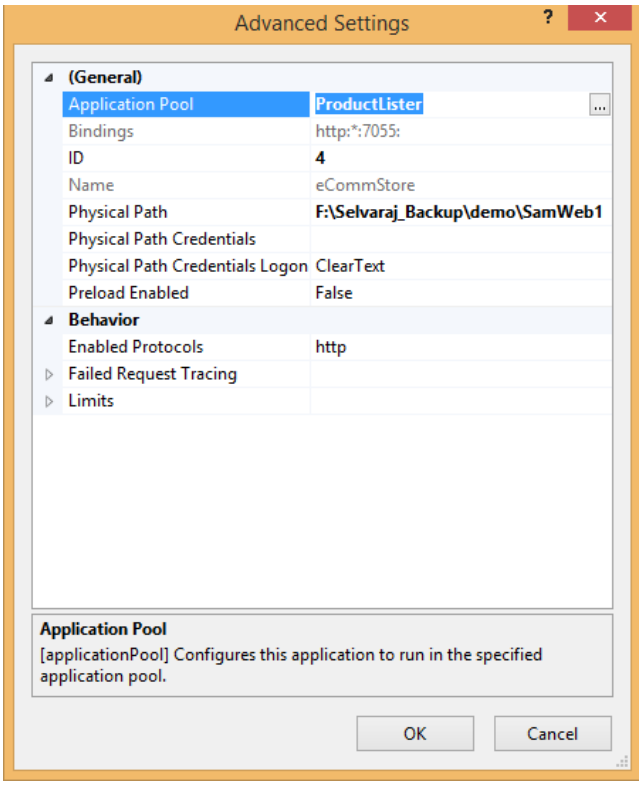


Figure 2.8: The Advanced Settings page

- Then, click the **Cancel** button in Figure 2.8 to return to the IIS Manager console.
- Next, click the Application Pools node in the tree-structure in the left panel of the IIS Manager console (see Figure 2.9). The right panel will then display all the **Application Pools** on that IIS web server. Browse the list to locate the **Application Pool** to which the target web site / web application belongs - i.e., the **Application Pool** displayed in Figure 2.8.

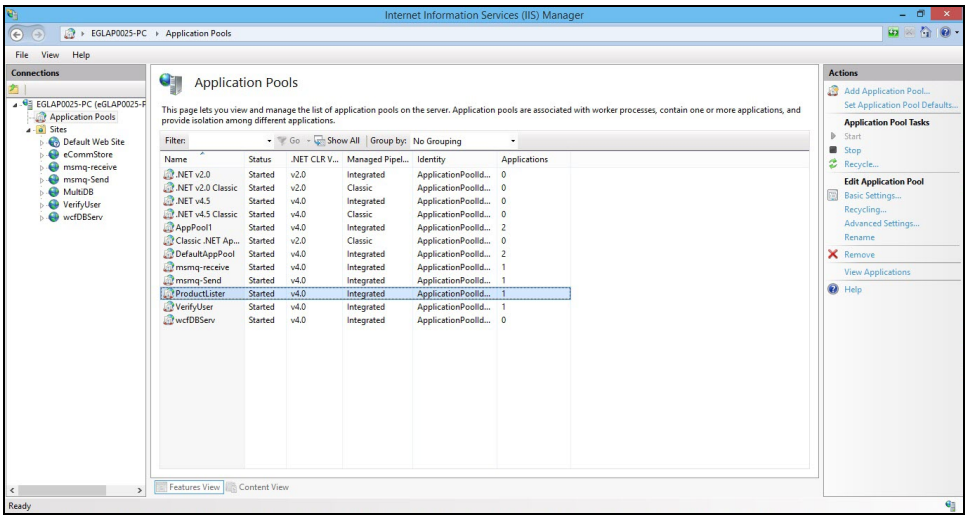


Figure 2.9: Locating the Application Pool to which the target web site/web application belongs

- Right-click on that **Application Pool** and select the **Recycle** option from the short-cut menu that appears (see Figure 2.10). Doing so will restart only those web sites / web applications managed by that application pool.

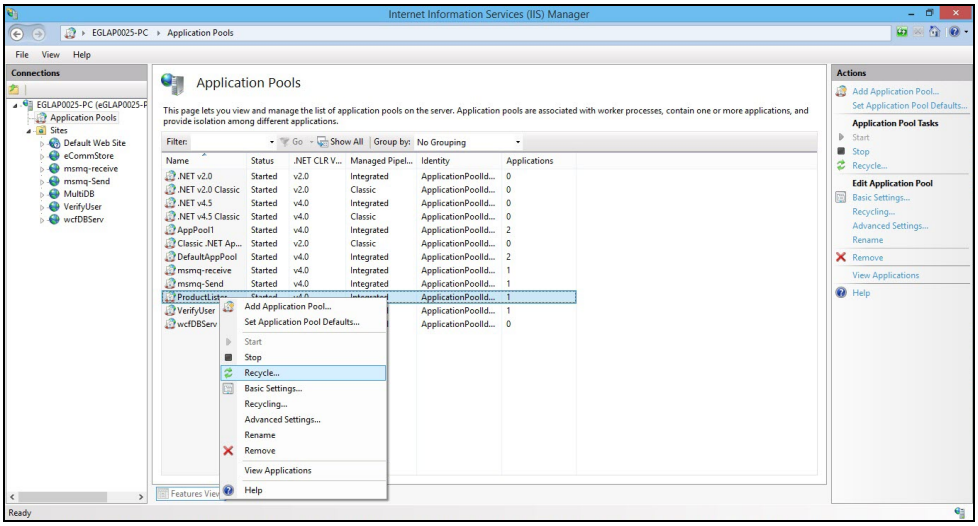


Figure 2.10: Recycling the application pool to which the target web site/web application belongs

2.5 Installing and Configuring the .NET Profiler for Microsoft SharePoint Running as a Service Inside IIS

To install the .NET profiler for Microsoft SharePoint that is running as a service inside IIS, follow the steps below:

1. Login to the target IIS web server.
2. Before attempting to install the profiler, you will have to add a new entry to the IIS registry. For that, follow the steps below:
   - First, go to the Registry Editor (regedit).
   - Under HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework, create a new DWORD "LoaderOptimization" with value 1.
   - Then, under HKEY_ LOCAL_ MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\, create a new DWORD "LoaderOptimization" with value 1.

     These configurations disable loader optimization assemblies, which may interfere with profiler operations.
   - Finally, restart the IIS web server.
3. Then, log back into the IIS web server. From the command-prompt, switch to the <EG_AGENT_ INSTALL_DIR>\lib directory.
4. Now, issue the following command from that location:

   run SetupDotNetProfiler.bat
5. Once the batch file begins execution, the following messages will appear:

```
-----------------------------------------------------------------
*** Setting up Registry values for the Profiler ***
-----------------------------------------------------------------
Modifying the W3SVC service's Environment variables
Modifying the WAS service's Environment variables
```

6. If setup detects that a profiler (this can be an eG;s .NET profiler or any third-party profiler such as NewRelic) pre-exists on the host, then the following message will appear:

```
There is another profiler found in registry
-----------------------------------------------------------
COR_PROFILER=<71DA0A04-7777-4EC6-9643-7D28B46A8A41>
-----------------------------------------------------------
```

Setup will then prompt you to confirm whether you want to proceed with the installation or not.

```
Do you wish to continue setup <yes/no> : yes
```

If you specify **yes** here, then setup will try to overwrite the GUID presently registered with the COR_PROFILER registry key with the GUID of the eG Enterprise profiler. If the GUID is overwritten successfully, then a message to that effect will appear.

```
-----------------------------------------------------

RegValues are modified Successfully ***

-----------------------------------------------------
```

7. If setup is successful, then the following messages will appear:

```
Registering the Profiler.

Installing assembly in Cache...

Helper assemblies Installation Successful

Applying required permissions for eGurkha directory.
```

8. Finally, restart the IIS web server by issuing the **iisreset** command at the prompt. This will restart all the web sites and web applications on that IIS web server. If you would rather have the monitored web sites / web applications alone to be restarted, do the following:

   - Open the Internet Information Services (IIS) Manager (see Figure 2.11).

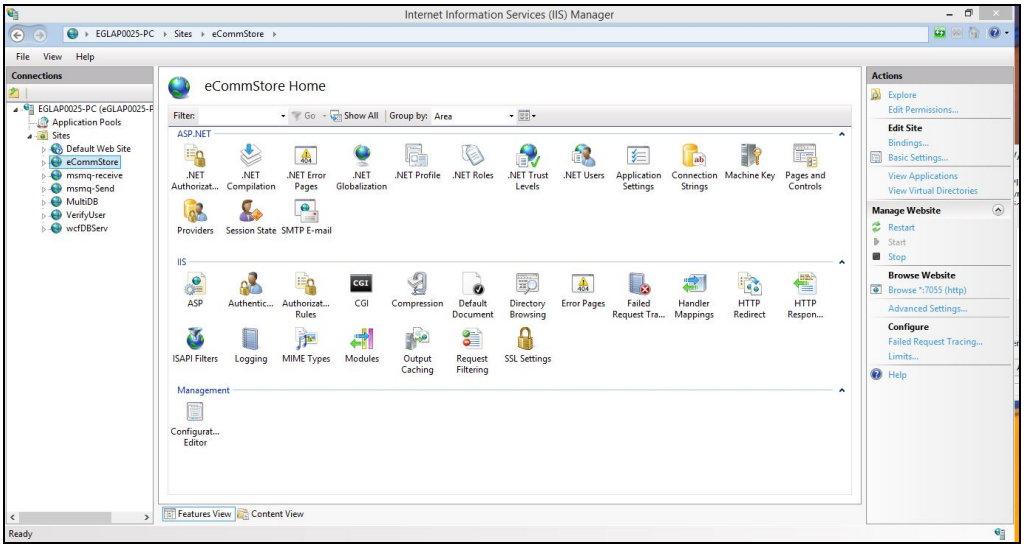


Figure 2.11: The IIS Manager console

- Expand the Sites node in the tree-structure in the left panel of Figure 2.11, and select the sub-node representing the web site / web application you are monitoring. Right-click on that sub-node, move your mouse pointer over **Manage Website** in the shortcut menu that appears, and pick the **Advanced Settings** option (see Figure 2.12).

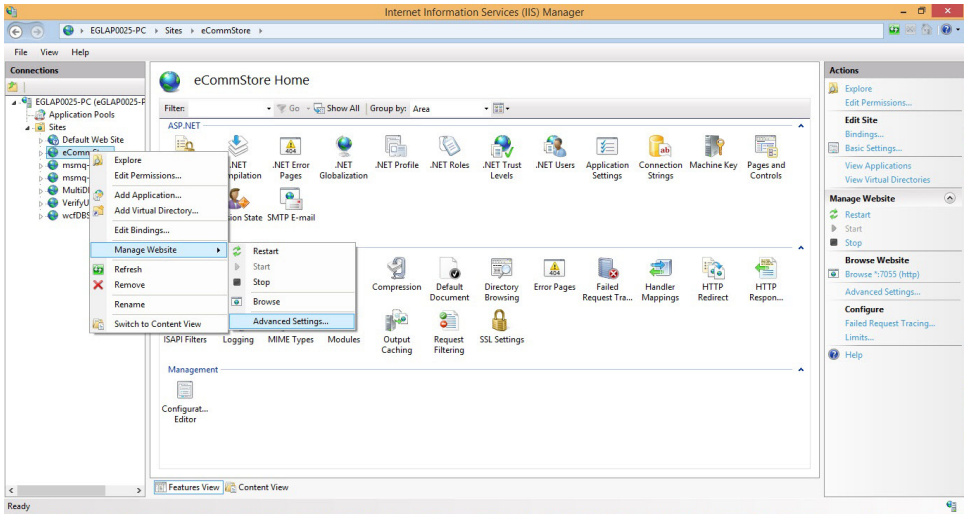


Figure 2.12: Selecting the Advanced Settings option

- Figure 2.13 will then appear. From Figure 2.13, you can figure out which **Application Pool** manages the target web site / web application.

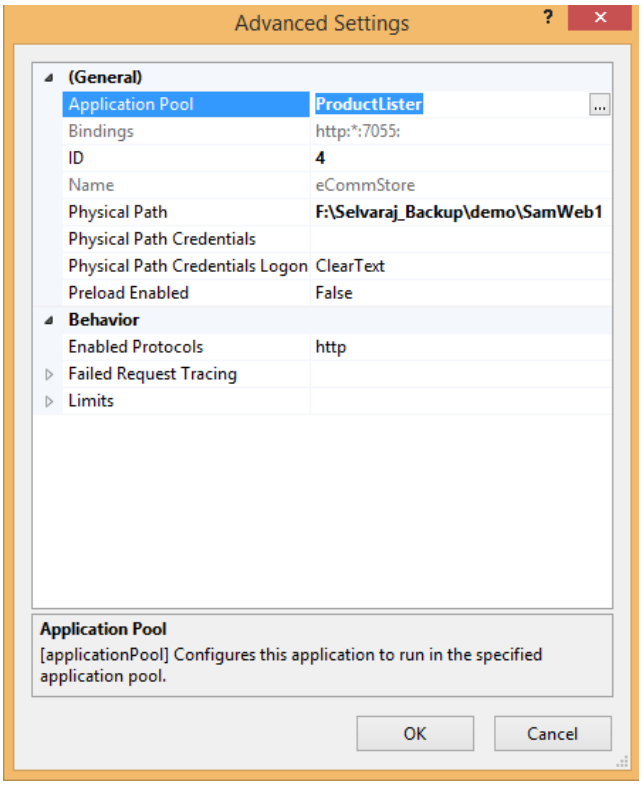


Figure 2.13: The Advanced Settings page

- Then, click the **Cancel** button in Figure 2.13 to return to the IIS Manager console.
- Next, click the Application Pools node in the tree-structure in the left panel of the IIS Manager console (see Figure 2.14). The right panel will then display all the **Application Pools** on that IIS web server. Browse the list to locate the **Application Pool** to which the target web site / web application belongs - i.e., the **Application Pool** displayed in Figure 2.13.

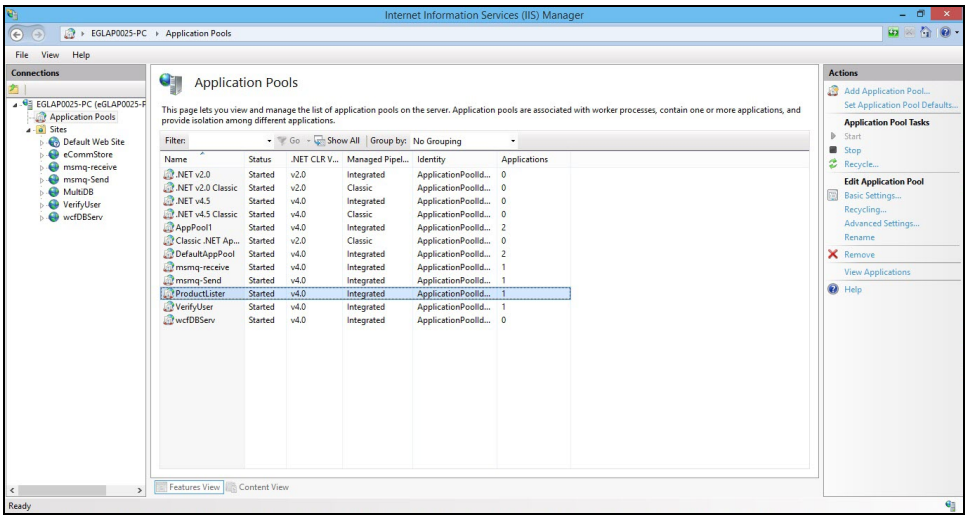


Figure 2.14: Locating the Application Pool to which the target web site/web application belongs

- Right-click on that **Application Pool** and select the **Recycle** option from the short-cut menu that appears (see Figure 2.15). Doing so will restart only those web sites / web applications managed by that application pool.

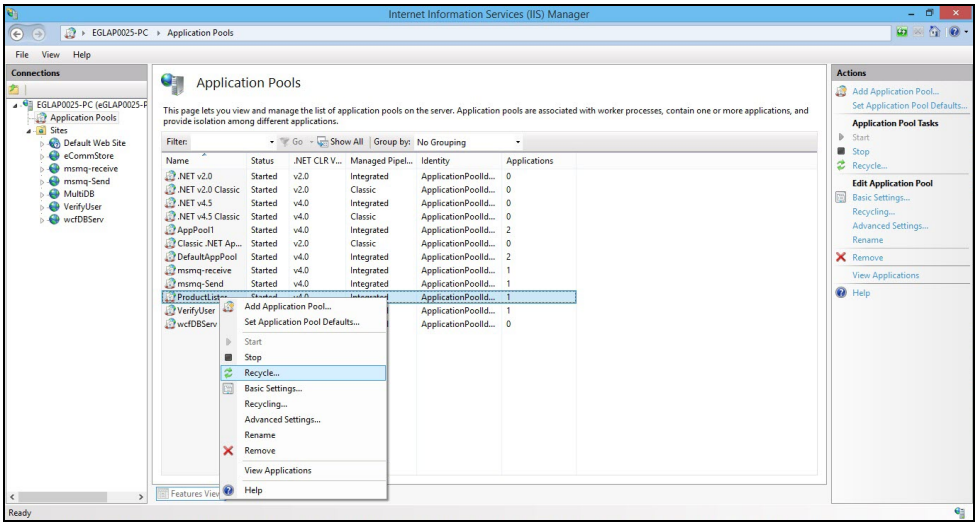


Figure 2.15: Recycling the application pool to which the target web site/web application belongs

2.6 .NET Business Transactions Test

The responsiveness of a transaction is the key determinant of user experience with that transaction; if response time increases, user experience deteriorates. To make users happy, a business transaction should be rapidly processed by each of the Microsoft IIS web servers in its path. Processing bottlenecks on a single server can slowdown/stall an entire business transaction or can cause serious transaction errors. This in turn can badly scar the experience of users. To avoid this,

administrators should promptly identify slow/stalled/errored transactions, isolate the IIS web server on which the slowness/error occurred, and uncover what caused the aberration on that server – is it owing to SQL queries executed by the server? Or is it because of external calls – eg., HTTP calls, web service calls, etc. - made by that node? The **.NET Business Transactions** test helps with this!

This test runs on a BTM-enabled IIS web server in an IT infrastructure and tracks all the transaction requests received by any web site that has been configured for monitoring on that server. The test then groups requests based on user-configured pattern specifications. For each transaction pattern, the test then computes and reports the average time taken by that server to respond to the transaction requests of that pattern to the configured web site. In the process, the test identifies the slow/stalled transactions of that pattern, and reports the count of such transactions and their responsiveness. Detailed diagnostics provided by the test accurately pinpoint the exact transaction URLs that are slow/stalled, the total round-trip time of each transaction, and also indicate when such transaction requests were received by that server. The slowest transaction in the group can thus be identified.

Moreover, to enable administrators to figure out if the slowness can be attributed to a bottleneck in SQL query processing, the test also reports the average time the transactions of each pattern took to execute SQL queries. If a majority of the queries are slow, then the test will instantly capture the same and notify administrators.

Additionally, the test promptly alerts administrators to error transactions of each pattern. To know which are the error transactions, the detailed diagnosis capability of the test can be used.

This way, the test effortlessly measures the performance of each transaction to a web site on a IIS web server, highlights transactions that are under-performing, and takes administrators close to the root-cause of poor transaction performance.

Note:

This test will report metrics only if the following pre-requisites are fulfilled:

- The eG .NET Business Transaction Monitor (a.k.a the .NET Profiler) should be installed and configured on the target IIS web server.
- The **IIS Management Scripts and Tools** feature should be installed and enabled on the target IIS web server.

Target of the Test : A BTM-enabled IIS web server

Agent deploying the test : An internal agent

Output of the test : One set of results for each grouped URL

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Website Name	Specify the name of the web site on the target IIS web server for which transaction monitoring is to be performed. **Note:** If this test is configured with a web site name, then all other web site-related tests of the target IIS web server will report metrics for this web site only.
Healthy URL Trace	By default, this flag is set to **No**. This means that eG Enterprise will not collect detailed diagnostics for those transactions that are healthy. If you want to enable the detailed diagnosis capability for healthy transactions as well, then set this flag to **Yes**.
Max Healthy URLs per Test Period	This parameter is applicable only if the Healthy URL Trace flag is set to 'Yes'**.** Here, specify the number of top-n transactions that should be listed in the detailed diagnosis of the *Healthy transactions* measure, every time the test runs. By default, this is set to 5*0*, indicating that the detailed diagnosis of the *Healthy transactions* measure will by default list the top-50 transactions, arranged in the descending order of their response times.
Max Slow URLs per Test Period	Specify the number of top-n transactions that should be listed in the detailed diagnosis of the *Slow transactions* measure, every time the test runs. By default, this is set to *10*, indicating that the detailed diagnosis of the *Slow transactions* measure will by default list the top-10 transactions, arranged in the descending order of their response times.
Max Stalled URLs per Test Period	Specify the number of top-n transactions that should be listed in the detailed diagnosis of the *Stalled transactions* measure, every time the test runs. By default, this is set to *10*, indicating that the detailed diagnosis of the *Stalled transactions* measure will by default list the top-10 transactions, arranged in the descending order of their response times.
Max Error URLs per Test Period	Specify the number of top-n transactions that should be listed in the detailed diagnosis of the *Error transactions* measure, every time the test runs. By default, this is set to *10*, indicating that the detailed diagnosis of the *Error transactions* measure will by default list the top-10 transactions, in terms of the number of errors they encountered.
Method Execution Cutoff (MS)	From the detailed diagnosis of slow/stalled/error transactions, you can drill down and perform deep execution analysis of a particular transaction. In this drill-down, the methods invoked by that slow/stalled/error transaction are listed in the order in which

Parameter	Description
	the transaction calls the methods. By configuring a method execution cutoff, you can make sure that methods that have been executing for a duration greater the specified cutoff are alone listed when performing execution analysis. For instance, if you specify *5* here, then the **Execution Analysis** window for a slow/stalled/error transaction will list only those methods that have been executing for over 5 milliseconds. This way, you get to focus on only those methods that could have caused the slowness, without being distracted by inconsequential methods. By default, the value of this parameter is set to *250 ms*.
SQL Execution Cutoff (MS)	Typically, from the detailed diagnosis of a slow/stalled/error transaction to a web site, you can drill down to view the SQL queries (if any) executed by that transaction from that node and the execution time of each query. By configuring a SQL Execution Cutoff, you can make sure that queries that have been executing for a duration greater the specified cutoff are alone listed when performing query analysis. For instance, if you specify *5* here, then for a slow/stalled/error transaction, the **SQL Queries** window will display only those queries that have been executing for over 5 milliseconds. This way, you get to focus on only those queries that could have contributed to the slowness. By default, the value of this parameter is set to *10 ms*.
Max Grouped URLs per Measure Period	This test groups URLs according to the 'URL Segments' specification. These grouped URLs will be the descriptors of the test. For each grouped URL, response time metrics will be aggregated across all transaction URLs in that group and reported. When monitoring web sites/web applications to which the transaction volume is normally high, this test may report metrics for hundreds of descriptors. If all these descriptors are listed in the **Layers** tab page of the eG monitoring console, it will certainly clutter the display. To avoid this, by default, the test displays metrics for a maximum of 50 descriptors – i.e., 50 grouped URLs alone – in the eG monitoring console, during every measure period. This is why, the Max Grouped URLs per Measure Period parameter is set to 50 by default. To determine which 50 grouped URLs should be displayed in the eG monitoring console, the eG BTM follows the below-mentioned logic: • Top priority is reserved for URL groups with error transactions. This means that eG BTM first scans URL groups for error transactions. If error transactions are found in 50 URL groups, then eG BTM computes the aggregated response time of each of the 50 groups, sorts the error groups in the descending order of their response time, and displays all these 50 groups alone as the descriptors of this test, in the sorted order.

Parameter	Description
	• On the other hand, if error transactions are found in only one / a few URL groups – say, only 20 URL groups – then, eG BTM will first arrange these 20 grouped URLs in the descending order of their response time. It will then compute the aggregated response time of the transactions in each of the other groups (i.e., the error-free groups) that were auto-discovered during the same measure period. These other groups are then arranged in the descending order of the aggregated response time of their transactions. Once this is done, eG BTM will then pick the top-30 grouped URLs from this sorted list. • In this case, when displaying the descriptors of this test in the **Layers** tab page, the 20 error groups are first displayed (in the descending order of their response time), followed by the 30 'error-free' groups (also in the descending order of their response time). At any given point in time, you can increase/decrease the maximum number of descriptors this test should support by modifying the value of the **MAX GROUPED URLS PER MEASURE PERIOD** parameter.
Max SQL Queries Per Transaction	Typically, from the detailed diagnosis of a slow/stalled/error transaction to a web site, you can drill down to view the SQL queries (if any) executed by that transaction from that node and the execution time of each query. By default, eG picks the first 500 SQL queries executed by the transaction, compares the execution time of each query with the SQL Execution Cutoff configured for this test, and displays only those queries with an execution time that is higher than the configured cutoff. This is why, the 'Max SQL Queries Per Transaction' parameter is set to 500 by default. To improve agent performance, you may want the **SQL EXECUTION CUTOFF** to be compared with the execution time of a less number of queries – say, 200 queries. Similary, to increase the probability of capturing more number of long-running queries, you may want the SQL Execution Cutoff to be compared with the execution time of a large number of queries – say, 1000 queries. For this, you just need to modify the 'Max SQL Queries Per Transaction' specification to suit your purpose.
Filtered URL Patterns	By default, this test does not track requests to the following URL patterns: **.ttf, *.otf, *.woff, *.woff2, *.eot, *.cff, *.afm, *.lwfn, *.ffil, *.fon, *.pfm, *.pfb, *.std, *.pro, *.xsf, *.jpg, *.jpeg, *.jpe, *.jif, *.jfif, *.jfi, *.jp2, *.j2k, *.jpf, *.jpx, *.jpm, *.jxr, *.hdp, *.wdp, *.mj2, *.webp, *.gif, *.png, *.apng, *.mng, *.tiff, *.tif, *.xbm, *.bmp, *.dib, *.svg, *.svgz, *.mpg, *.mpeg, *.mpeg2, *.avi, *.wmv, *.mov, *.rm, *.ram, *.swf, *.flv, *.ogg, *.webm, *.mp4, *.ts, *.mid, *.midi, *.rm, *.ram, *.wma, *.aac, *.wav, *.ogg, *.mp3, *.mp4, *.css, *.js, *.ico|/egurkha**

Parameter	Description
	If required, you can remove one/more patterns from this default list, so that such patterns are monitored, or can append more patterns to this list in order to exclude them from monitoring.
Show Cookies	An HTTP cookie is a small piece of data sent from a website and stored on the user's computer by the user's web browser while the user is browsing. Most commonly, cookies are used to provide a way for users to record items they want to purchase as they navigate throughout a website (a virtual "shopping cart" or "shopping basket"). To keep track of which user is assigned to which shopping cart, the server sends a cookie to the client that contains a unique session identifier (typically, a long string of random letters and numbers). Because cookies are sent to the server with every request the client makes, that session identifier will be sent back to the server every time the user visits a new page on the website, which lets the server know which shopping cart to display to the user. Another popular use of cookies is for logging into websites. When the user visits a website's login page, the web server typically sends the client a cookie containing a unique session identifier. When the user successfully logs in, the server remembers that that particular session identifier has been authenticated, and grants the user access to its services. If you want to view and analyze the useful information that is stored in such HTTP response cookies that a web server sends, then set this flag to **Yes**. By default, this flag is set to **No**, indicating that cookie information is not reported by default as part of detailed diagnostics.
Show Headers	HTTP headers allow the client and the server to pass additional information with the request or the response. A request header is a header that contains more information about the resource to be fetched or about the client itself. If you want the additional information stored in a request header to be displayed as part of detailed diagnostics, then set this flag to **Yes**. By default, this flag is set to **No** indicating that request headers are not displayed by default in the detailed diagnosis.
URL Segments	This test groups transaction URLs based on the URL segments count configured for monitoring and reports aggregated response time metrics for every group. Using this parameter, you can specify the number of URL segments based on which the transactions are to be grouped. URL segments are the parts of a URL (after the base URL) or path delimited by slashes. So if you had the URL: *http://www.eazykart.com/web/shopping/sportsgear/login.jsp*, then *http://www.eazykart.com* will be the base URL or domain, */web* will be the first URL segment, */shopping* will be the second URL segment, and */sportsgear* will be the third URL segment, and */login.jsp* will be the fourth URL segment. By default, this parameter is set to 3. This default setting, when applied to the sample URL provided above, implies that the eG agent will aggregate response time metrics to all transaction

Parameter	Description
	URLs under /web/shopping/sportsgear. Note that the base URL or domain will not be considered when counting URL segments. This in turn means that, if a web site receives transaction requests for the URLs such as http://www.eazykart.com/web/shopping/sportsgear/login.jsp, http://www.eazykart.com/web/shopping/sportsgear/jerseys.jsp, http://www.eazykart.com/web/shopping/sportsgear/shoes.jsp, http://www.eazykart.com/web/shopping/sportsgear/gloves.jsp, etc., then the eG agent will track the requests and responses for all these URLs, aggregate the results, and present the aggregated metrics for the descriptor /web/shopping/sportsgear. This way, the test will create different transaction groups based on each of the third-level URL segments – eg. /web/shopping/weddings, /web/shopping/holiday, /web/shopping/gifts etc. – and will report aggregated metrics for each group so created. If you want, you can override the default setting by providing a different URL segment number here. For instance, your specification can be just *2*. In this case, for the URL*tp://www.eazykart.com/web/shopping/login.jsp*, the test will report metrics for the descriptor *web/shopping.*
DD Frequency	Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is 1:1. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD Frequency.
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: • The eG manager license should allow the detailed diagnosis capability • Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measures reported by the test:

Measurement	Description	Measurement Unit	Interpretation
All transactions	Indicates the total number of requests received for transactions of this pattern during the last measurement period.	Number	By comparing the value of this measure across transaction patterns, you can identify the most popular transaction patterns. Using the detailed diagnosis of this measure, you can then figure out which specific transactions of that pattern are most requested. For the **Summary** descriptor, this measure will reveal the total number of transaction requests received by the target web site during the last measurement period. This is a good indicator of the transaction workload on that web site.
Avg response time	Indicates the average time taken by the transactions of this pattern to complete execution.	Secs	Compare the value of this measure across patterns to isolate the type of transactions that were taking too long to execute. You can then use the detailed diagnosis of the All transactions measure of that group to know how much time each transaction in that group took to execute. This will lead you to the slowest transaction. For the **Summary** descriptor, this measure will reveal the average responsiveness of all the transaction requests received by the target web siteduring the last measurement period. An abnormally low value for this measure for the Summary descriptor could indicate a serious processing bottleneck on the target web site.
Healthy transactions	Indicates the number of healthy transactions of this pattern.	Number	By default, this measure will report the count of transactions with a response time less than 4000 milliseconds. You can change this default setting by

Measurement	Description	Measurement Unit	Interpretation
			modifying the thresholds of the *Avg response time* measure using the eG admin interface. For the **Summary** descriptor, this measure will report the total number of healthy transactions on the target web site.
Healthy transactions percentage	Indicates what percentage of the total number of transactions of this pattern is healthy.	Percent	To know which are the healthy transactions, use the detailed diagnosis of this measure. For the **Summary** descriptor, this measure will report the overall percentage of healthy transactions on the target web site.
Slow transactions	Indicates the number of transactions of this pattern that were slow during the last measurement period.	Number	By default, this measure will report the number of transactions with a response time higher than 4000 milliseconds and lesser than 60000 milliseconds. You can change this default setting by modifying the thresholds of the *Avg response time* measure using the eG admin interface. A high value for this measure is a cause for concern, as too many slow transactions means that user experience with the web application is poor. For the **Summary** descriptor, this measure will report the total number of slow transactons on the target web site. This is a good indicator of the processing power of the target web site.
Slow transaction response time	Indicates the average time taken by the slow transactions of this pattern to execute.	Secs	For the **Summary** descriptor, this measure will report the average response time of all the slow transactions on the target web site.

Measurement	Description	Measurement Unit	Interpretation
Slow transactions percentage	Indicates what percentage of the total number of transactions of this pattern is currently slow.	Percent	Use the detailed diagnosis of this measure to know which precise transactions of a pattern are slow. You can drill down from a slow transaction to know what is causing the slowness. For the **Summary** descriptor, this measure will report the overall percentage of slow transactions on the monitored web site.
Error transactions	Indicates the number of transactions of this pattern that experienced errors during the last measurement period.	Number	A high value is a cause for concern, as too many error transactions to a web application can significantly damage the user experience with that application. For the **Summary** descriptor, this measure will report the total number of error transactons on the target web site. This is a good indicator of how error-prone the target web site is.
Error transactions response time	Indicates the average duration for which the transactions of this pattern were processed before an error condition was detected.	Secs	The value of this measure will help you discern if error transactions were also slow. For the **Summary** descriptor, this measure will report the average response time of all error transactions on the target web site.
Error transactions percentage	Indicates what percentage of the total number of transactions of this pattern is experiencing errors.	Percent	Use the detailed diagnosis of this measure to isolate the error transactions. You can even drill down from an error transaction in the detailed diagnosis to determine the cause of the error. For the **Summary** descriptor, this measure will report the overall percentage of transactions of this pattern on the target web site that is currently experiencing errors.
Stalled transactions	Indicates the number of transactions of this pattern that were stalled during the	Number	By default, this measure will report the number of transactions with a

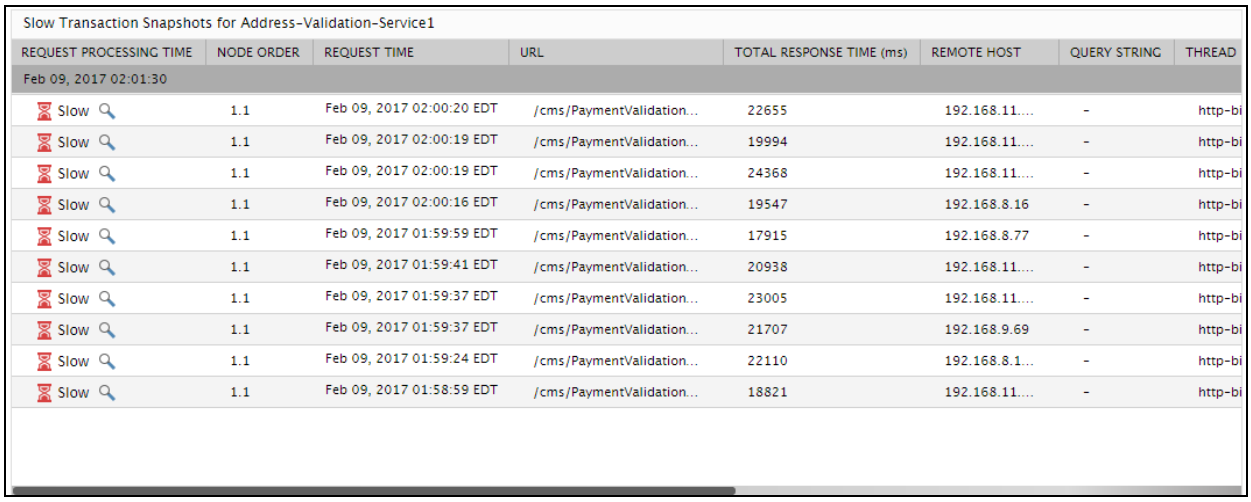
Measurement	Description	Measurement Unit	Interpretation
	last measurement period.		response time higher than 60000 milliseconds. You can change this default setting by modifying the thresholds of the *Avg response time* measure using the eG admin interface. A high value is a cause for concern, as too many stalled transactions means that user experience with the web application is poor. For the **Summary** descriptor, this measure will report the total number of stalled transactons on the target web site.
Stalled transactions response time:	Indicates the average time taken by the stalled transactions of this pattern to execute.	Secs	For the **Summary** descriptor, this measure will report the average response time of all stalled transactions on the target web site.
Stalled transactions percentage	Indicates what percentage of the total number of transactions of this pattern is stalling.	Percent	Use the detailed diagnosis of this measure to know which precise transactions of a pattern are stalled. You can drill down from a stalled transaction to know what is causing that transaction to stall. For the **Summary** descriptor, this measure will report the overall percentage of transactions of this pattern on the target web site that is stalling.
Slow SQL statements executed	Indicates the number of slow SQL queries that were executed by the transactions of this pattern during the last measurement period.	Number	For the **Summary** descriptor, this measure will report the total number of slow SQL queries executed by all transactions to the target web site.
Slow SQL statement time	Indicates the average execution time of the slow SQL queries that were run by the transactions of this pattern.	Secs	If there are too many slow transactions of a pattern, you may want to check the value of this measure for that pattern to figure out if query execution is slowing down the transactions. Use

Measurement	Description	Measurement Unit	Interpretation
			the detailed diagnosis of the *Slow transactions* measure to identify the precise slow transaction. Then, drill down from that slow transaction to confirm whether/not database queries have contributed to the slowness. Deep-diving into the queries will reveal the slowest queries and their impact on the execution time of the transaction.
Total transactions per minute	Indicates the number of transactions of this pattern that are executed per minute.	Number	
Error transactions per minute	Indicates the number of error transactions of this pattern that are executed per minute.	Number	A very low value is desired for this measure. Compare the value of this measure across transaction patterns to find that pattern of transactions that is experiencing errors frequently.

2.7 Detailed Diagnostics

By reporting detailed diagnostics on transaction responsiveness and errors, eG Enterprise not only points you to the slow/stalled/error transaction URLs, but also reveals what could be causing the slowness/errors.

2.7 reveals detailed diagnosis of the Slow transactions percentage measure of the **.NET Business Transactions** test.


Slow Transaction Snapshots for Address-Validation-Service1

REQUEST PROCESSING TIME	NODE ORDER	REQUEST TIME	URL	TOTAL RESPONSE TIME (ms)	REMOTE HOST	QUERY STRING	THREAD
Feb 09, 2017 02:01:30							
Slow	1.1	Feb 09, 2017 02:00:20 EDT	/cms/PaymentValidation...	22655	192.168.11....	-	http-bi
Slow	1.1	Feb 09, 2017 02:00:19 EDT	/cms/PaymentValidation...	19994	192.168.11....	-	http-bi
Slow	1.1	Feb 09, 2017 02:00:19 EDT	/cms/PaymentValidation...	24368	192.168.11....	-	http-bi
Slow	1.1	Feb 09, 2017 02:00:16 EDT	/cms/PaymentValidation...	19547	192.168.8.16	-	http-bi
Slow	1.1	Feb 09, 2017 01:59:59 EDT	/cms/PaymentValidation...	17915	192.168.8.77	-	http-bi
Slow	1.1	Feb 09, 2017 01:59:41 EDT	/cms/PaymentValidation...	20938	192.168.11....	-	http-bi
Slow	1.1	Feb 09, 2017 01:59:37 EDT	/cms/PaymentValidation...	23005	192.168.11....	-	http-bi
Slow	1.1	Feb 09, 2017 01:59:37 EDT	/cms/PaymentValidation...	21707	192.168.9.69	-	http-bi
Slow	1.1	Feb 09, 2017 01:59:24 EDT	/cms/PaymentValidation...	22110	192.168.8.1...	-	http-bi
Slow	1.1	Feb 09, 2017 01:58:59 EDT	/cms/PaymentValidation...	18821	192.168.11....	-	http-bi

Figure 2.16: The detailed diagnosis of the Slow transactions percentage measure of the .NET Business Transactions test

The detailed diagnosis reveals the individual transaction URLs in the grouped URL that users requested for, the total response time of each transaction, the client (remote host) from which each transaction request was received, the thread executing the transaction, and the query string of the transaction URL.

The per-transaction response time displayed in 2.7 includes the following:

- The total time for which the transaction request was processed by the target IIS web server and by other BTM-enabled IIS web servers in the transaction path thereafter, until the time the response for that transaction request was sent out by the target IIS web server;
- The time taken by external calls (SQL query / HTTP / WCF / Web Service) to other IIS web servers, applicaions, or backend servers in the transaction path;

Additionally, the overall experience of the users with each transaction – whether it is slow, stalled, or error - is also revealed in the **REQUEST PROCESSING TIME** column. Furthermore, the HTTP headers, cookies, the HTTP status code returned by the monitored node in response to the transaction request, and the HTTP method invoked by the transaction on that node are also revealed. The per-transaction statistics are also sorted in the descending order of the transaction response time, starting with the slowest transaction and ending with the healthiest one. In the event that the Avg response time of a grouped URL registers an abnormally high value, you can use these detailed metrics to quickly and accurately identify the exact transaction in the group that is significantly contributing to the poor user experience with the group.

Once a slow/stalled transaction is identified, the next question is what is causing the transaction to slowdown. Transaction responsiveness can be impacted by any of the following factors:

- An inefficient database query run by the target IIS web server or any other IIS web server in the transaction path;
- Issues in the .NET application code;
- A poorly responsiveness remote service call made by the target IIS web server or any other IIS web server in the transaction path;

With the help of illustrated examples, the links below describe how drill-downs from the detailed diagnostics enable accurate isolation of the root-cause of a transaction slowdown / errors in a transaction. Detailed Diagnostics Revealing that Slow .Net Processing is the Reason for a Slow transaction

2.7.1 Detailed Diagnostics Revealing that Slow .Net Processing is the Reason for Transaction Slowness

Let us consider the example of a .NET application, where the following transactions are slow.

REQUEST PROCESSING TIME	NODE ORDER	REQUEST TIME	URL	TOTAL RESPONSE TIME (ms)	REMOTE HOST	QUERY STRING	THREAD INFO	P
Mar 28, 2017 15:50:10								
Slow	1	Mar 28, 2017 15:49:33 IST	/samweb1.a...	6624	172.16.0...	-	592	
Slow	1	Mar 28, 2017 15:49:31 IST	/samweb1.a...	6451	172.16.0...	-	586	
Slow	1	Mar 28, 2017 15:49:33 IST	/samweb1.a...	5623	172.16.0...	-	582	
Slow	1	Mar 28, 2017 15:49:34 IST	/samweb1.a...	5398	172.16.0...	-	607	

Figure 2.17: Detailed diagnosis of the Slow transactions percentage measure

Let us focus on the first transaction in the slow transactions list of Figure 2.17. To zoom into the transaction, click on it. Figure 2.18 will then appear revealing the cross-application flow of that transaction.

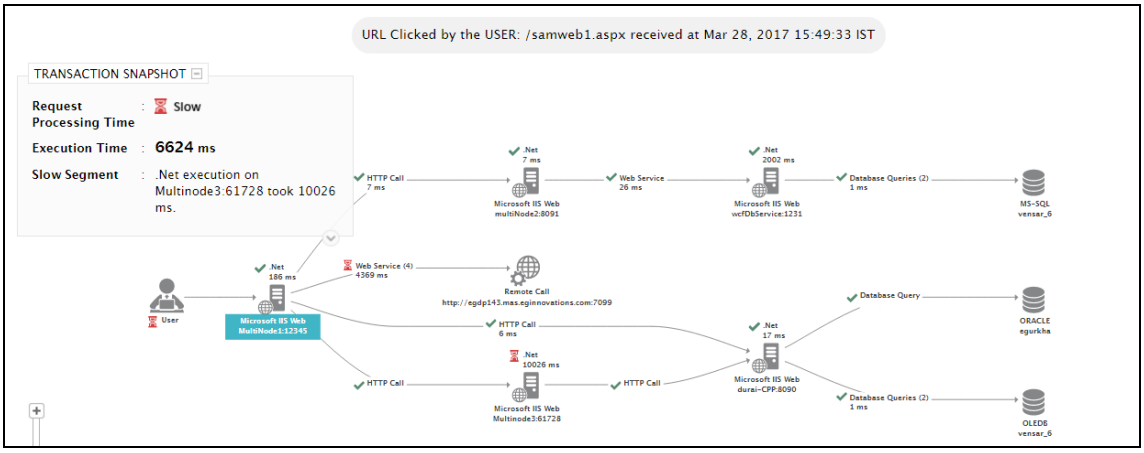


Figure 2.18: The cross-application flow of the slow transaction

This flow diagram clearly reveals the following:

- The IIS web servers and backends through which the transaction travelled;
- The time for which the transaction request was processed at each BTM-enabled IIS web server in the transaction path; **note that this time will not be computed for IIS web servers that are in the transaction path, but are not BTM-enabled and those that are BTM-enabled but are not managed by eG Enterprise;**
- The exit calls made by each BTM-enabled node to another node as part of the transaction's journey, the time consumed by each exit call,and the number of times each type of call was made; the following exit calls are supported by eG BTM:
  - Database Query
  - HTTP
  - Web service
  - WCF

Note:

- If a profiled node appears 'grayed out' in the cross-application transaction flow, it denotes that profiler could not collect detailed diagnostics for that node. The reasons for this could be either or both of the following:
  - Transaction responsiveness on the 'grayed out' node was either healthy or was only slightly slow, and hence, did not appear in the list of Top-N slow transactions.
  - Slow data transmission from eG agent to manager;
- If an IIS web server makes a call to a messaging server, then. in the transaction topology, that messaging server will be identified by the URL of the exact queue/topic that is managing the request. If an IIS web server makes a SQL query call to a database server, then the details displayed for that database server in the transaction topology depends upon whether/not that database server is managed by eG Enterprise. If the database server is not managed by eG Enterprise, then such a database server will be represented in the topology using the server type (whether Oracle, Microsoft SQL etc.) and the name of the database that was accessed by the SQL query. To know the IP and port number of the unmanaged database server, you can drill-down from the **Database queries** call in the topology. On the other hand, if the database server in question is being monitored by eG Enterprise, then such a server will

be represented in the topology using the server type, nick name, port number, and the database name. Additionally, the SID will be displayed in case of an Oracle database server, and the instance name will be displayed in case of an instance-based Microsoft SQL server.

- Sometimes, empty nodes – i.e., nodes without any details – will be visible in the cross-application transaction flow topology. Likewise, the time spent on certain external calls may also not be displayed in the topology. This is owing to inconsistencies in the collection of detailed diagnostics.

Using conventional color codes and intuitive icons, the transaction flow chart precisely pinpoints where the transaction slowed down. In the case of 2.7.1 above, from the color-coding it is clear that the problem is with the .Net layer of the IIS web server, *Multinode3:61728*. To zoom into the problem, click on that IIS web server in the transaction topology. 2.7.1 will then appear providing a detailed Execution Analysis of the .Net methods called by *Multinode3:61728*.

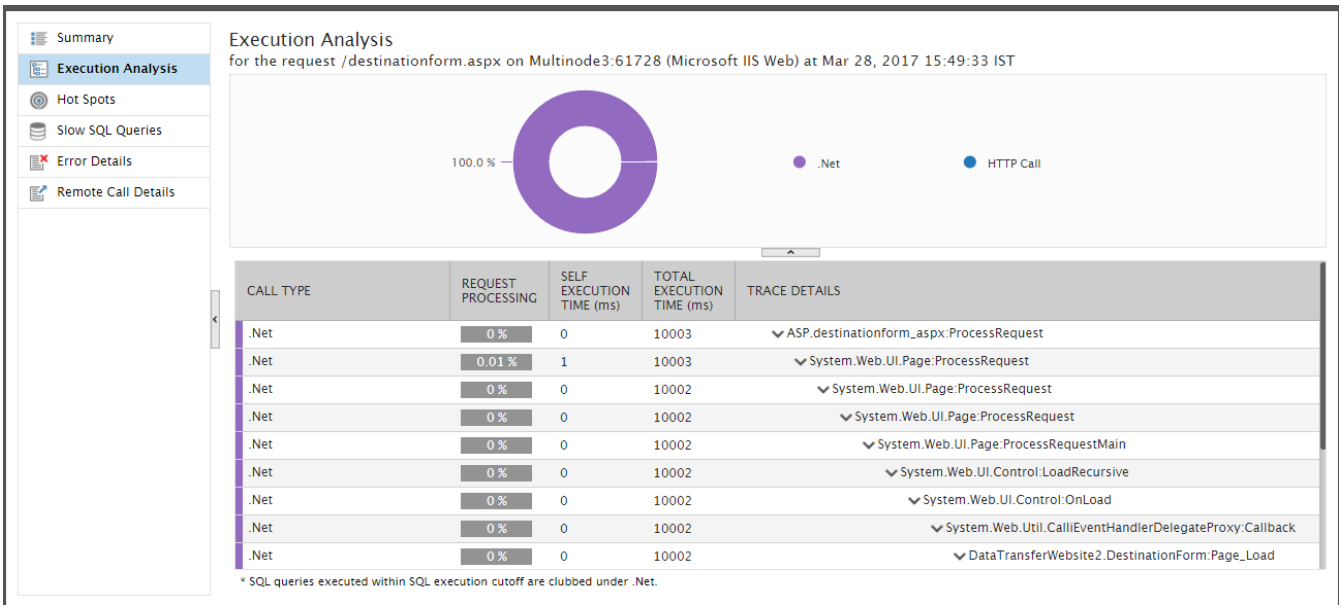


Figure 2.19: The call graph of the .Net transaction

As part of this analysis, a pie chart is presented that quickly reveals the percentage of time *Multinode3:61728* in our example spent processing the server's .Net code and making external HTTP/WCF/Web service calls. The table below the pie chart in Figure 2.19 lists the exact methods that performed .Net processing or made the remote calls. A quick look at this table reveals that the .Net method, ASP.destinationform_aspx:ProcessRequest invoked a series of child methods and external calls, which together took over 10000 milliseconds to execute. However, the method itself did not take any time to execute (self execution time)! Browsing the child methods called by the parent method reveals that the transaction spent over 90% of its time on the .Net call,

'DataTransferWebsite2.DestinationForm:Page_ Load'. This means that the 'sun.net.www.protocol..' is the method that is delaying the transaction.

This way, eG BTM enables you to diagnose the root-cause of slowness in your .Net calls using just a few mouse clicks!

Note:

By default, the table in 2.7.1 will not include the methods/functions within those classes that are associated with the basic .Net framework (on the target IIS web server). This is because, such functions are less likely to encounter issues. Moreover, when a transaction slows down, application experts will more often than not look for issues in the application code and not in the underlying .Net code. Also, by excluding such classes from its monitoring purview, eG Enterprise also reduces its processing overheads significantly.

If required, you can include these classes in the profiler's monitoring scope or can even exclude more classes from monitoring. For this, do the following:

1. Edit the **eG_DotNetBTM.ini** in the <EG_INSTALL_DIR>\manager\config directory on the eG manager host.
2. In the **[EXCLUDE]** section of the file, you will find a **CLASSES** parameter. By default, this parameter will be set to a semicolon-separated list of (.Net framework) class patterns to be excluded from monitoring. The default specification will be as follows:

```
[EXCLUDE]
CLASSES=System.*;Microsoft.*;AsyncInvocationWithFilters.*;
```

3. If you want the profiler to monitor any of the excluded files, then simply remove the corresponding file pattern from the semicolon-separated list. For instance, if you want the profiler to monitor all classes that begin with *System.*, then remove the *System.** pattern from the **CLASSES** listed under **[EXCLUDE]**. If this is done, then the specification will be as shown below:

```
[EXCLUDE]
CLASSES=Microsoft.*;AsyncInvocationWithFilters.*;
```

4. Likewise, if only a sub-set of the excluded class patterns are to be included in the profiler's monitoring scope, then use the **CLASSES** parameter in the **[INCLUDE]** section of the file. For instance, of all classes of the *System.** pattern, if you want only the *System.Security* classes to be monitored, then, your specification in the **[INCLUDE]** section will be as follows:

```
[INCLUDE]
CLASSES=Microsoft.CRM.*;System.Security.*;
```

5. Additional class patterns or classes can be appended to the **CLASSES** list under **[INCLUDE]** by separating each class pattern/class name by a semi-colon.
6. Finally, save the file.

2.8 Disabling/Uninstalling the eG .NET Profiler

To disable/uninstall the eG .NET Profiler, follow the steps below:

1. Login to the system hosting the eG .NET Profiler.
2. Go to the command prompt and switch to the <EG_INSTALL_DIR>\lib directory.
3. Issue the following command at the prompt to disable the profiler:

   uninstall_Profiler.bat
4. Once the batch file successfully executes, the eG .NET Profiler will get uninstalled. However, even after uninstallation, the .NET Profiler will continue to remain attached to the worker process that first loaded the profiler. To detach the profiler from that worker process, run IISRESET at the command prompt. Alternatively, wait till the worker process stops by itself.

Note:

If the eG agent on a profiled IIS web server is uninstalled, the eG .NET Profiler also gets automatically uninstalled. Here again, the profiler will continue to be associated with the worker process that originally loaded it. To detach the profiler from that worker process, you either have to run IISRESET at the command prompt after uninstalling the eG agent, or wait till the worker process stops by itself.

2.9 Performance Overhead of the eG .NET Business Transaction Monitor

eG BTM leaves a very minimal resource footprint on the application it monitors. Typically, it adds a meagre 2-5% to the application overhead.

Chapter 3: Troubleshooting

This chapter provides tips for easily and effectively troubleshooting issues that may surface when installing and/or working with the eG .NET Profiler.

3.1 Troubleshooting the Installation of the .NET Profiler

The profiler installation will fail under the following circumstances:

- The absence of VC++ Runtime on the host.
- The presence of multiple profiler GUIDs in the registry because of dirty installations of older profilers;
- Issues when accessing the Registry key;

The errors that the batch file will throw in each of the above situations and how to resolve such errors has been discussed below:

1. **The absence of VC++ Runtime on the profiler host**

   In this case, you will see the following error when running the **SetupDotNetProfiler.bat** file:

   ```
   ERROR: Pre-Requisite VC++ Runtime 2012 is not found
   ```

   You then have to exit the profiler setup, install the VC++ Runtime 2012, and then begin the profiler installation.

2. **The presence of multiple profiler GUIDs in the registry because of dirty installations of older profilers**

In this case, the following message will pop up when running the **SetupDotNetProfiler.bat** file:

```
There is another profiler found in registry
------------------------------------------------------------
COR_PROFILER=<71DA0A04-7777-4EC6-9643-7D28B46A8A41>
------------------------------------------------------------
```

Setup will then prompt you to confirm whether you want to proceed with the installation or not.

```
Do you wish to continue setup <yes/no> : yes
```

If you specify **yes** here, then setup will try to overwrite the GUID presently registered with the COR_PROFILER registry key with the GUID of the eG Enterprise profiler. If the GUID is overwritten successfully, then a message to that effect will appear.

```
-----------------------------------------------------
RegValues are modified Successfully ***
-----------------------------------------------------
```

However, sometimes, if multiple GUIDs are registered with the COR_PROFILER registry key, then setup may not be able to update the GUID. This can happen, if previously, many profiler installation attempts failed on the host or if many profilers were not installed / uninstalled properly. In this case, setup will automatically terminate with the following error message:

```
Multiple COR_PROFILER entries found in the Registry
Aborting the Setup
```

3. **Issue when accessing the registry key**

If setup is unable to access the COR_PROFILER registry key to write the eG profiler's GUID into it, the following error message will appear. Setup will automatically abort soon after.

```
ERROR in Accessing the Registry Key : 0x5
Setup Aborted
```

This can happen if setup is run by a user without *local administrator* privileges. In this case therefore, you will have to run setup again as a local administrator to ensure that it is successful.

3.2 Troubleshooting the Failure of the eG .NET Profiler to Profile and Measure Performance of .NET Transactions

This section discusses what to do to resolve some common problems that you may face when working with the eG .NET Profiler.

1. What are the very basic checks that need to be done when .NET BTM test for a website (say, SampleWebsite) is not reporting any metrics?

   - **IP / Port cross-check** – The IP and Port number of the “SampleWebsite” needs to be cross-checked while adding a component.

- **Website Name in Test Config** – The website name “SampleWebsite” (Case sensitive - as in the IIS server) needs to be specified in test configuration page of “.NET Business Transactions” Test.
- **Successful execution of SetupDotNetProfiler.bat** – Check if a “Setup Aborted” error message appeared when the SetupDotnetProfiler.bat was executed.
- **IISRESET** – Check if the IISRESET command was run after successful execution of batch file.
- **Initiate requests for website** – Ensure that the user hits the URL of the SampleWebsite.
- **Enable IIS Management Scripting & Tools** – Enable **IIS Management scripts and tools** from the Microsoft Windows features.
- **Check for the existence of other Profiling tools** – For this, follow the checks discussed in response to question 2 of the Section **Chapter 4** topic. If these checks reveal that some other profiler pre-exists, you will have to uninstall that profiler for the eG .NET profiler to work.

2. How to verify that the eG .NET BTM has been successfully installed?

   You can check the System Environment Variables and Event Logs to verify whether/not the eG .NET pofiler has been installed on a server.

   Checking System Environment Variables

   - Login to the system hosting the eG .NET BTM.
   - Open Windows Explorer and go to the *Computer* or *My Computer* node therein, depending upon the version of Windows in use. In some Windows versions, you can even type *This PC* in *Search*.
   - Right-click on the relevant node and select **Properties**.

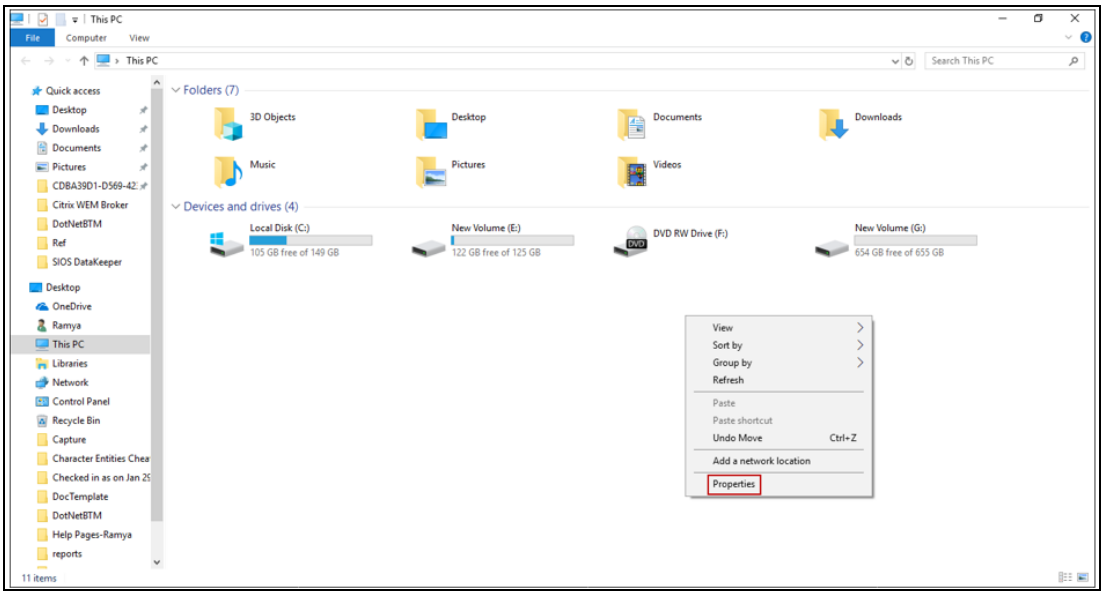


Figure 3.1: Selecting Properties option from shortcut menu

- Figure 3.2 will appear. Click on **Advanced Settings** in Figure 3.2.

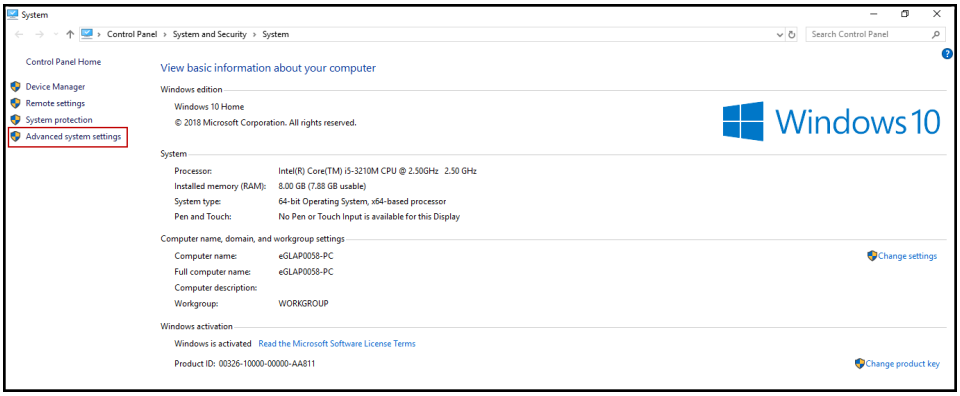


Figure 3.2: Clicking on Advanced Settings option

- Then Figure 3.3 appears. Under **Advanced** tab, click on the **Environment Variables** button.

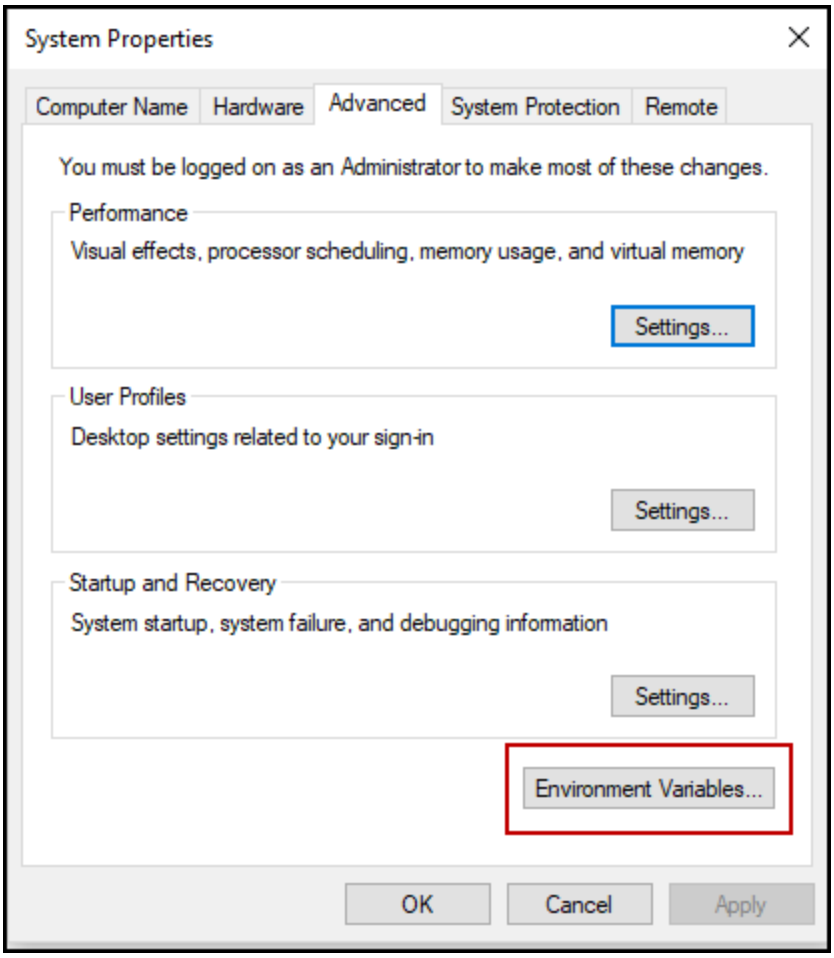


Figure 3.3: Clicking on the Environment Variables button

- In the list of **Environment Variables** that then appears, look for the **COR_PROFILER** and **COR_ENABLE_PROFILING** variables.

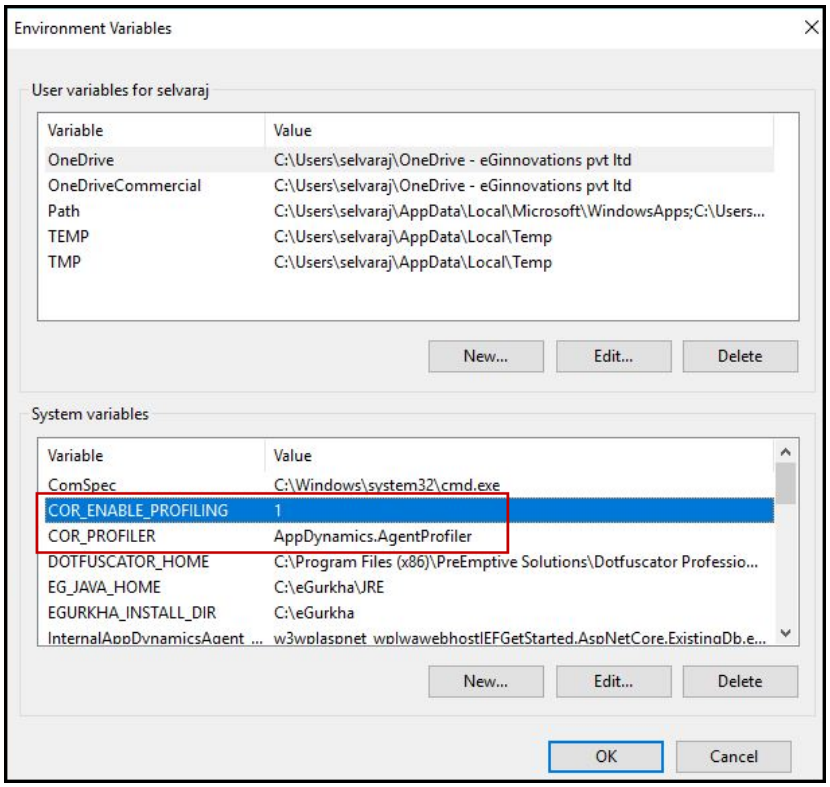


Figure 3.4: Checking the System Variables list for the COR_PROFILER and COR_ENABLE_ PROFILING variables

- If the **COR_ PROFILER** variable is set to the GUID, *947734AF- 7AE7- 41DD- BAE5- 5EA8E4AE89BB*, and the **COR_ENABLE_PROFILING** variable is set to 1, it means that the eG .NET profiler is installed on the server. If the **COR_PROFILER** variable is set to a different GUID, it means that some other third-party profiler is installed on the server.

Checking the Event Logs

- Open the Windows Event Viewer and check for *"NET Runtime"* information in Windows Logs > Application.
- Look for a message of the following format:

  The profiler was loaded successfully. Profiler CLSID: '{<GUID_of_Profiler>}'

- If a message of the above format exists, and the <*GUID_of_Profiler*> that is displayed as part of the message is *947734AF-7AE7-41DD-BAE5-5EA8E4AE89BB*, it means that the eG .NET profiler is installed on the server. On the other hand, if a different GUID is displayed as part of the message, it means that some other third-party profiler is installed on the server.

3. How to verify whether the eG .NET BTM module is successfully attached to a worker process (w3wp.exe)?

   - If you are using a **.NET Runtime version 2.0 (and above)**, then follow the steps below:

     - Open the Windows Event Viewer and check for *"NET Runtime"* information in Windows Logs > Application.
     - If you find the following message therein, it means that the attachment is successful.

       The profiler was loaded successfully. Profiler CLSID: '{947734AF-7AE7-41DD-BAE5-5EA8E4AE89BB}'

   - If you are using a **.NET Runtime version prior to v2.0**, then follow the steps below:

     - Download ProcessExplorer.exe and run in Administrator mode.
     - Press CTRL+ F and search for **eGCLRMonitor.**
     - Check if **w3wp.exe** of the web site's application pool contains eG .NET BTM. If so, it means that eG .NET BTM is successfully attached to the worker process.

4. What do I do if no messages are logged in .NetProfLogs or if the .NetProfLogs directory is empty?

   The eG .NET Profiler automatically creates log files in the .NetProfLogs directory, and logs each operation it performs and its status in these log files.

   If the .NetProflogs directory is empty or if no messages are logged in the log files in this directory, it could mean one/both of the following:

   - **The BTM module has not loaded properly**: To check this, follow the steps detailed in the response to question 3 above.
   - **The <EG_INSTALL_DIR> cannot be accessed by Authenticated Users:** Check whether *Authenticated Users* are allowed access to the <EG_INSTALL_DIR> that contains the .NetProfLogs directory. For that, do the following:

     - First, navigate to the <EG_INSTALL_DIR> in Windows Explorer.
     - Then, right-click on the <EG_INSTALL_DIR>, and select **Properties**.
     - In the **Properties** dialog box that then appears, click the **Security** tab page.
     - Check for the *Authenticated Users* group in the **Group or user names** list in the

Security tab page.

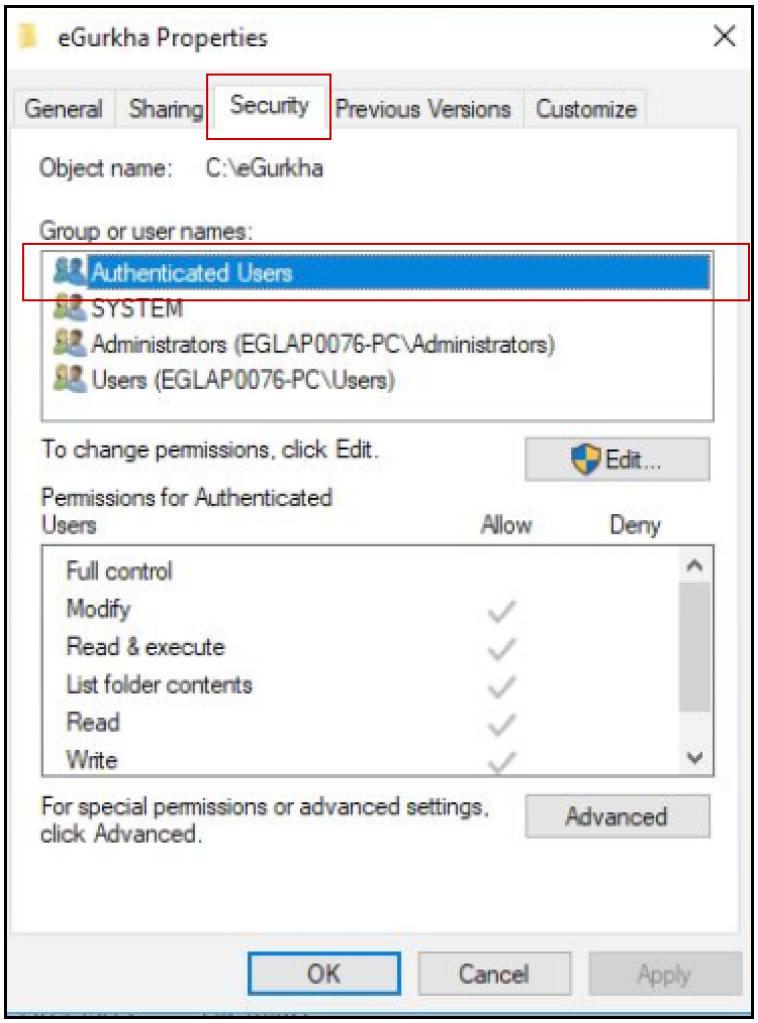


Figure 3.5: Checking whether/not the 'Authenticated Users ' group is listed in the Group or User names list

- If you do not find that group name in the **Group or User names** list, it means that the authenticated users cannot access the <EG_INSTALL_DIR> or its sub-directories. In this case, make sure that the **Authenticated Users** are allowed read/write access to the <EG_ INSTALL_DIR>.

5. What do I do to save web application when the web site crashes / throws ERROR in web page, after .NET BTM is setup?

- First, open <EG_ INSTALL_ DIR>\agent\config\eg_ DotnetServer.ini file and change **INSTRUMENTATION_LEVEL** to '2'. Then, run the IISRESET command.

- If the web site crashes even after the instrumentation level is changed, then run the **uninstall_Profiler.bat** in the <EG_INSTALL_DIR>\lib directory to uninstall the profiler. Once the profiler is uninstalled, run the IISRESET command
- If the web site crashes even after the profiler is uninstalled, please contact the eG development team.

6. How can I check if .NET BTM module is sending data to eG agent?

   Open the **AgentComm.log** file (in the <EG_INSTALL_DIR>\agent\logs\.NetProfLogs directory) and do the following:

   - Check for the following message in log file:

     "[ERROR] unable to Connect with server 10061"

     If the message is found, it indicates that the test has not started.

   - Check for the following message in log file:

     "REQ SWAPPED - 1 & REQ SENT: 0 "

     If the message is found, it indicates that the request was filtered/discarded, since the request does not pertain to the web site being monitored.

7. What if .Net BTM could not capture the requests even after all basic checks were performed and passed?

   Open the **AgentComm.log** file (in the <EG_INSTALL_DIR>\agent\logs\.NetProfLogs directory) and check if **REQ_SWAPPED** is '0' for the **w3wp.exe** Process ID of the web site being monitored, even after the user hits the URL of the web site. If so, it indicates that the requests were not captured. Typically, requests to the following page types will not be captured:

   - Pure HTML pages
   - Pure ASP pages
   - Pages with any other extensions

   This is because, the eG .NET BTM can trace the path of requests to only .NET web applications.

   Please contact the development team if the problem persists.

8. I have installed the eG .NET Profiler on a Microsoft Sharepoint portal, but it is not working. What do I do?

   Loader optimization assemblies in Sharepoint can interfere with profiler operations. If these assemblies are not disabled before you install the profiler, the profiler may not be able to profile transactions or measure their performance. Therefore, its imperative that you disable the loader optimization assemblies before BTM-enabling a Sharepoint portal/site. To achieve this, follow the steps below:

   - First, go to the Registry Editor (regedit).
   - Under HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework, create a new DWORD **"LoaderOptimization"** with value 1.
   - Then, under HKEY_ LOCAL_ MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\, create a new DWORD "**LoaderOptimization**" with value 1.

     These configurations disable loader optimization assemblies, which may interfere with profiler operations.

   - Finally, restart the IIS web server.

Chapter 4: Frequently Asked Questions (FAQ)

This chapter topic provides answers to some of the frequently asked questions related to the eG .NET Profiler.

1. What entries are written in registry during Profiler Setup?

   - The profiler related values are written to the “Environment” key in the following locations:

     Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WAS

     Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC

   - The “Environment” key should have the following values in it:

     COR_PROFILER={947734AF-7AE7-41DD-BAE5-5EA8E4AE89BB} **--> eG .NET BTM Module’s GUID**

     COR_ENABLE_PROFILING=1

2. How to confirm that no profiler is installed on the server?

   Any profiler that is installed on a server leaves a footprint in one/more of the following locations:

   - The Windows Registry
   - The System Environment Variables
   - The Windows Event Logs

   You will have to check each of these locations to determine whether/not any profiler has been installed on the server.

   Checking the Windows Registry

   - Login to the system where the eG .NET BTM is installed.
   - Look for **COR_PROFILER** and **COR_ENABLE_PROFILING** entries in the “Environment” key in the following registry locations:

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WAS

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC

- If no such entries are present, then it means that no profiler is installed on the server machine.

Checking System Environment Variables

- Open Windows Explorer and go to the *Computer* or *My Computer* node therein, depending upon the version of Windows in use. In some Windows versions, you can even type *This PC* in *Search*.
- Right-click on the relevant node and select **Properties**.

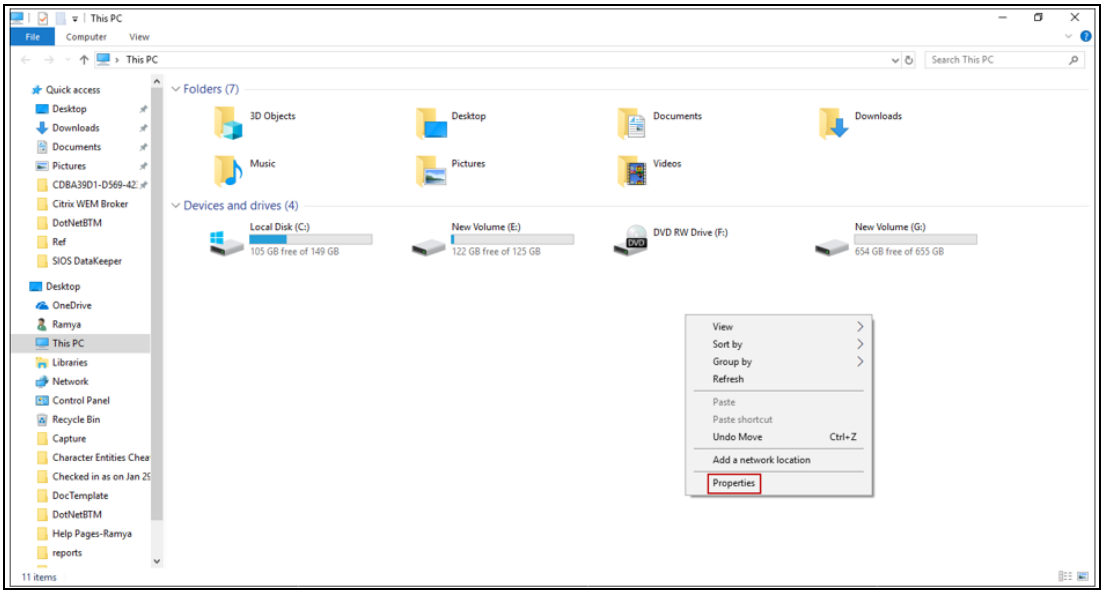


Figure 4.1: Selecting Properties option from shortcut menu

- Figure 4.2 will appear. Click on **Advanced Settings** in Figure 4.2.

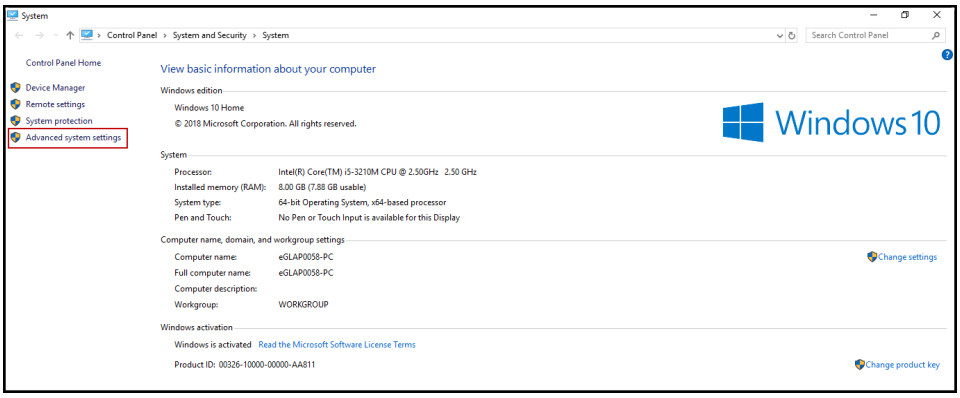


Figure 4.2: Clicking on Advanced Settings option

- Then Figure 4.3 appears. Under **Advanced** tab, click on the **Environment Variables** button.

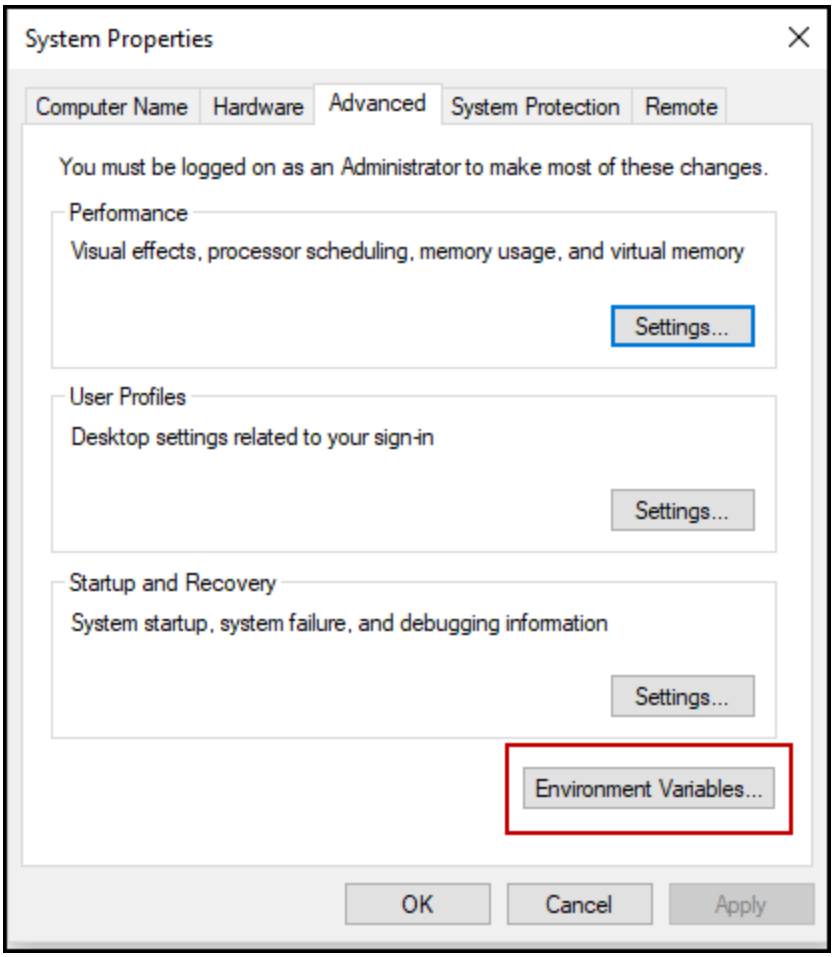


Figure 4.3: Clicking on the Environment Variables button

- In the list of **Environment Variables** that then appears, look for the **COR_PROFILER** and **COR_ENABLE_PROFILING** variables.

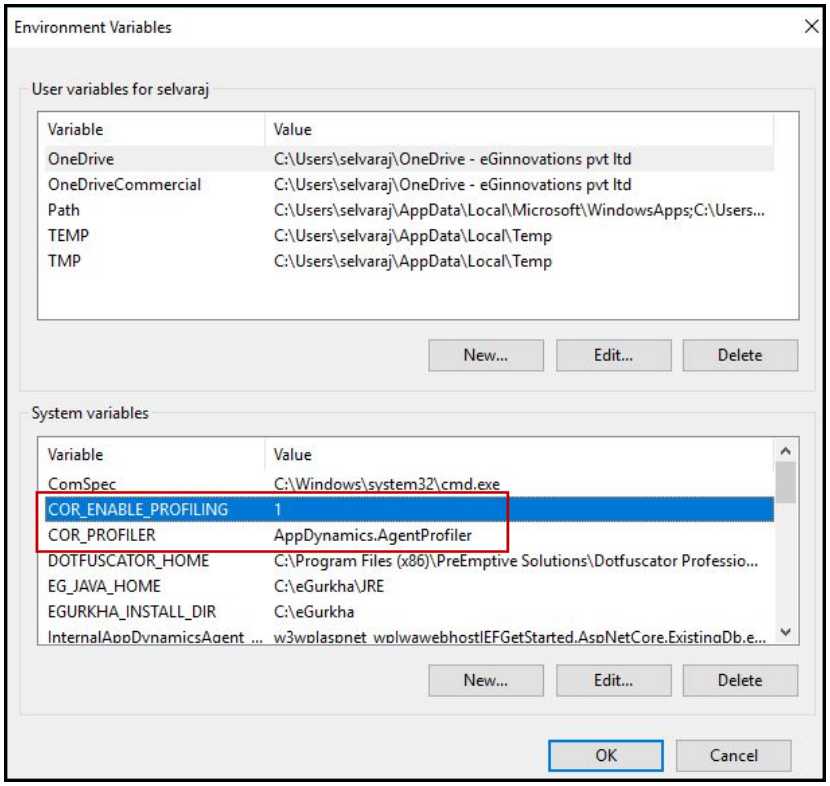


Figure 4.4: Checking the System Variables list for the COR_PROFILER and COR_ENABLE_PROFILING variables

- If you do not find the **COR_PROFILER** and **COR_ENABLE_PROFILING** variables displayed therein, it means that no profiler is installed on the server.

Checking the Event Logs

- Open the Windows Event Viewer and check for *"NET Runtime"* information in Windows Logs > Application.

  If you find the following message therein, it means that a profiler exists. If you do not find this message, it means that no profiler exists.

  The profiler was loaded successfully. Profiler CLSID: '{947734AF-7AE7-41DD-BAE5-5EA8E4AE89BB}'

3. Which VC++ runtime is required for profiler dlls?

VC++ redistributable for Visual Studio 2012 Runtime (64bit or 32 bit) needs to be installed for profiler to run.

4. What are the Healthy transactions and how to monitor them?

   Transactions with a response time that is less than the slow threshold (4000 ms) are considered as healthy transactions. Tracking of such transactions is disabled by default in the test configuration page of the .NET Transactions test. If you want such transactions to be monitored, then set the Healthy URL Trace flag of the .NET Transactions test to Yes.

5. Can eG monitor .NET client applications?

   No - .NET BTM is currently supported for web applications developed using .NET only.

6. How to remove the BTM Module after the agent is uninstalled from the machine?

   Run **uninstall_Profiler.bat** from the <EG_INSTALL_DIR>\lib directory to uninstall the profiler. Then, run the IISRESET command to remove the BTM Module.

7. What happens if the agent is stopped and the BTM Module is running?

   The .NET BTM continues to collect metrics for transactions that occur after the agent is stopped. However, such metrics will be discarded once the transactions end.

8. What should I do if the web server is running on load balancer mode?

   .NET BTM module must not be installed on the load balancer machine. The BTM module must be set up on the server where the web site is deployed and running.

9. What should I do for monitoring a transaction which spans both java and .NET web apps?

   BTM must be enabled for both the Java and .NET web apps to trace the complete path of the transaction.

10. How to identify whether the web server is using .NET CORE for runtime?

    Check if BTM module is attached to the worker. Then, check if there is a process named **"dotnet.exe"**. If it exists, then it means that the web server is using .NET CORE. For additional confirmation, cross-check with the customer.

11. How can I fetch the User Name and Business Context for the transactions?

There are four methods to fetch Username and Business Context. Refer to the Configuring User Name and Business Context topic for more details.

About eG Innovations

eG Innovations provides intelligent performance management solutions that automate and dramatically accelerate the discovery, diagnosis, and resolution of IT performance issues in on-premises, cloud and hybrid environments. Where traditional monitoring tools often fail to provide insight into the performance drivers of business services and user experience, eG Innovations provides total performance visibility across every layer and every tier of the IT infrastructure that supports the business service chain. From desktops to applications, from servers to network and storage, from virtualization to cloud, eG Innovations helps companies proactively discover, instantly diagnose, and rapidly resolve even the most challenging performance and user experience issues.

eG Innovations is dedicated to helping businesses across the globe transform IT service delivery into a competitive advantage and a center for productivity, growth and profit. Many of the world's largest businesses use eG Enterprise to enhance IT service performance, increase operational efficiency, ensure IT effectiveness and deliver on the ROI promise of transformational IT investments across physical, virtual and cloud environments.

To learn more visit www.eginnovations.com.

Contact Us

For support queries, email support@eginnovations.com.

To contact eG Innovations sales team, email sales@eginnovations.com.