Monitoring Microsoft IIS Web Server

eG Innovations Product Documentation

www.eginnovations.com

Total Performance Visibility

Table of Contents

Table of Figures

Chapter 1: Introduction

Web servers like Microsoft IIS servers are the heart of IT infrastructures in various domains - Healthcare, Banking, Trading, Logistics, etc. To ensure scalability and high performance, most web sites are being architected to use the multi-tier model - i.e., with the web server (IIS, Apache, etc.) functioning as the front-end, the middleware application server (J2EE, .Net based, etc.) that hosts the business logic functioning as the mid-tier, and a database server (SQL, Oracle, etc.) as the backend. In such architectures, the web server plays a pivotal role since all users to the other tiers are routed via the web server and hence, any slowdown or problem in the web server tier can adversely impact the end user experience.

The availability of a web site and the response time for user accesses to the site are the most critical metrics of web performance. Both these metrics may vary depending from one website to another and even from one transaction to another. For instance, one set of application components may come into play when a user logs in to an eBanking site, while a set of components may be invoked when a user transfers funds between his/her accounts. Consequently, a web monitoring solution must be able to report the availability and response time for individual user transactions to a web site.

Most web monitoring solutions rely on request emulation to monitor web transactions to a site. These request emulators generate synthetic requests periodically from one or more locations to the site and monitor the availability and response time for each transaction. This simple yet elegant solution provides the external perspective of the site.

The main limitations of a request emulation-only approach are:

1. This approach cannot be used to monitor the most critical transactions to a web site e.g., a user making a payment, a user registering to a web site, etc.,
2. Moreover, this approach mainly samples the functioning of the target environment. If a specific transaction is failing, say 10% of the time, the emulation approach only has a 10% chance of reporting the problem. Consequently, this approach is able to consistently detect and report problems only when they are severe enough to impact the end user performance. i.e., a request emulated approach only enables reactive monitoring.

The eG web monitor adopts a unique two-pronged approach for web transaction monitoring. The external agent uses request emulation to assess the user experience from different locations. By doing so, the external agent captures the effect of the network latency and the server-side processing time on the end user experience.

To quantify the server processing times for real user requests (not emulated requests), the eG internal agent deploys a proprietary web-adapter technology. This technology enhances web servers with the capability to track HTTP/HTTPS requests to a web server and the corresponding responses. For each transaction that is configured for monitoring, the web adapter analyzes the request URLs and responses to report various metrics relating to individual web transactions in real-time.

The monitoring is done in an implementation-independent manner, as a result of which eG agents are able to monitor Java (Servlets, EJB, JSPs) and other non-Java implementations (ASP, PHP, CGI, etc.) with equal felicity. Since it is able to monitor real-user transactions to web servers in real-time, eG Enterprise's web adapter technology enables the agents to proactively monitor and quantify all anomalies that may occur in a web infrastructure. The ability to offer real-time monitoring of 100% of the real user transactions, without the need for explicit, expensive logging is a key distinguishing feature of eG Enterprise's Web Server Monitor. This kind monitoring capacity supports Microsoft IIS web servers.

eG Enterprise offers specialized monitoring models for the IIS web servers. The statistics reported by these models, enable administrators to find accurate answers to the following performance queries:

External monitoring	• Is the web site available for user accesses from different locations? • What is the response time for user accesses to the site from different geographic locations? • Is a slowdown due to increased network latency or due to increased server-side processing?
Internal transaction monitoring	• How are the critical transactions of a web site functioning? • What is the request rate for each transaction? • What is the average response time for each transaction? • Are there many aborts for the transaction?
Web site monitoring	• What is the status of the different web sites hosted on a single web server? • Are there many errors occurring in the system? • Are the servers supporting the web infrastructure adequately sized? • Are there usage trends that need to be accounted for future

	capacity planning?
Bottleneck detection	Is an increase in server-side processing time due to the web server or due to the middleware application server or due to the database?
Capacity planning	• Is the load being effectively balanced across all the web servers? • Are the critical web server processes up and running?

This document engages you in an elaborate discussion on how eG Enterprise monitors the IIS web servers.

Chapter 2: Configuring IIS Web Server to work with eG Agent on Windows Environments

2.1 Configuring eG Agent to Monitor Web Transactions to Web Sites on IIS Web Server Operating on Windows 2000/2003

To enable the eG agent to monitor an IIS web server on Windows 2000/2003, follow the steps below:

- First, make sure 'logging' is enabled on these platforms;
- Next, make sure that the eG agent configuration is modified to support web transaction monitoring.

The sub-sections that follow will discuss each of these steps elaborately.

2.1.1 Enabling Logging on IIS Web Server

Logging triggers the creation of log files that track the URLs accessed on the IIS web server. The eG Enterprise suite requires these log files for monitoring the transactions to the web sites hosted on the IIS web server. In the absence of these log files, an eG agent will not be able to monitor web site transactions for Microsoft IIS web servers. Therefore, in order to enable the eG Enterprise suite to perform effective web transaction monitoring, logging must be enabled for the managed web sites.

2.1.2 Enabling Logging for Web Sites on Windows 2003

In the case of an IIS web server on Windows 2003, logging can be enabled using the procedure discussed below:

1. Open the **Internet Information Services (IIS) Manager** on the IIS web server host using the menu sequence Start -> Programs -> Administrative Tools -> Internet Information Services (IIS) Manager. Figure 2.1 will then appear.

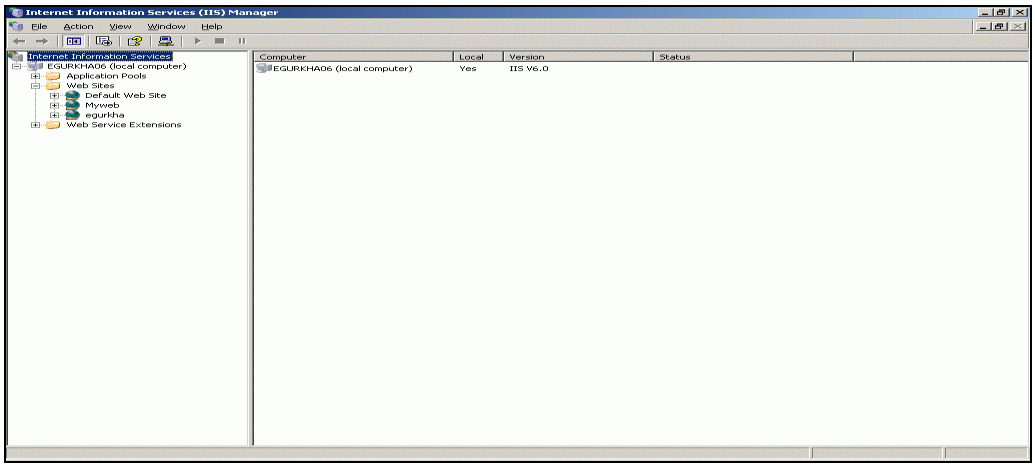


Figure 2.1: The IIS console

2. If all the web sites on the IIS web server are being monitored by eG, then you will have to enable logging for all. To achieve this, right-click on the **Web Sites** node in the tree structure on the left pane of Figure 2.1, and select **Properties** (see Figure 2.2) from the shortcut menu that appears.

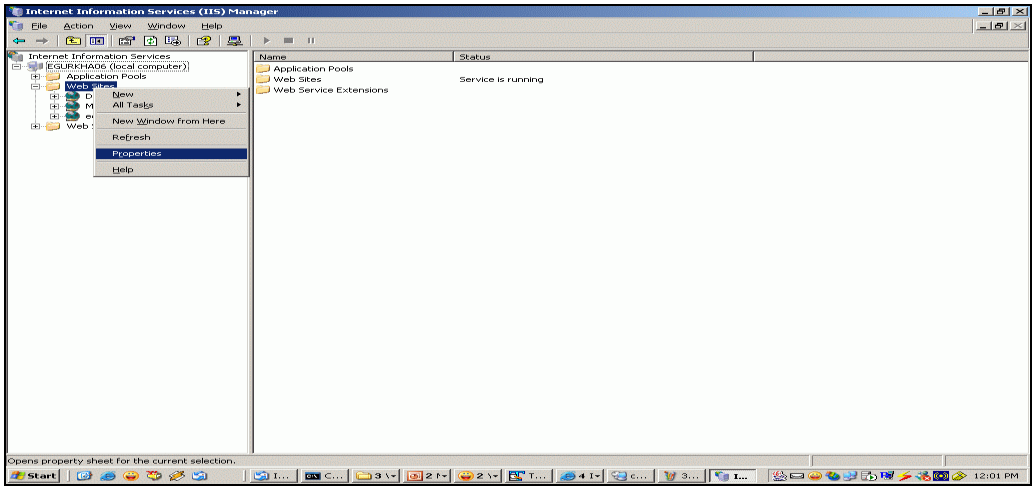


Figure 2.2: Selecting the Properties option from the shortcut menu of the Web Sites node (Windows 2003)

3. Next, click the **Web Site** tab of the **Properties** dialog box (see Figure 2.3) that appears, and ensure that the **Enable logging** check box is selected.

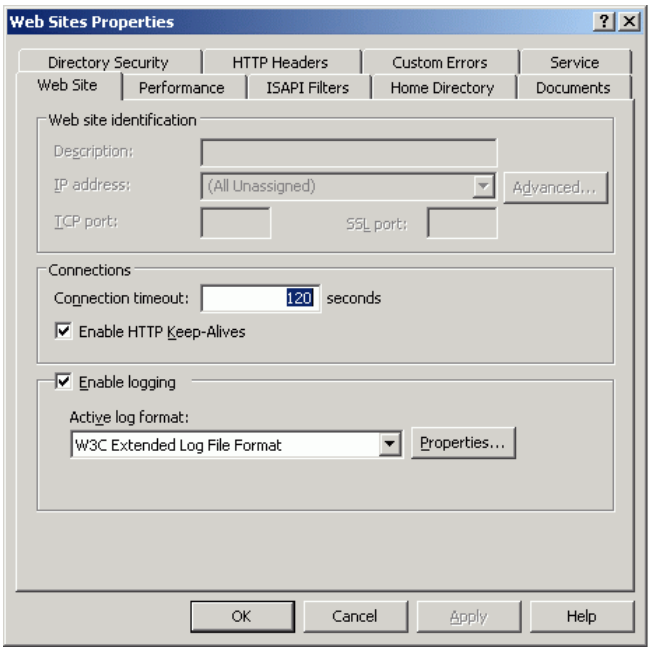


Figure 2.3: Enabling logging for all the web sites

4. Finally, click on the **Apply** and **OK** buttons to register the changes.
5. If only a few selected web sites on the IIS web server are being monitored by the eG Enterprise suite, then logging needs to be enabled for those specific sites only. To achieve this, right-click on the web site being monitored from the tree-structure in the left pane the IIS Manager, and select **Properties** from the shortcut menu (see Figure 2.4).

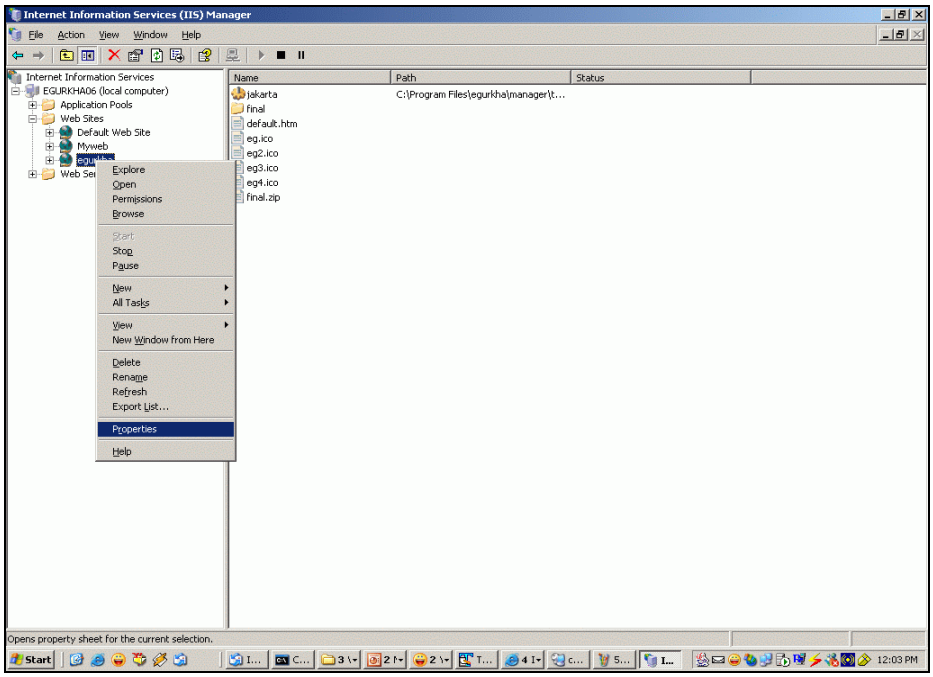


Figure 2.4: Selecting the Properties option for the egurkha web site

6. Next, select the **Web Site** tab from the **Properties** dialog box, and select the **Enable Logging** check box as depicted by Figure 2.5. Finally, click on the **Apply** button and then the **OK** button.

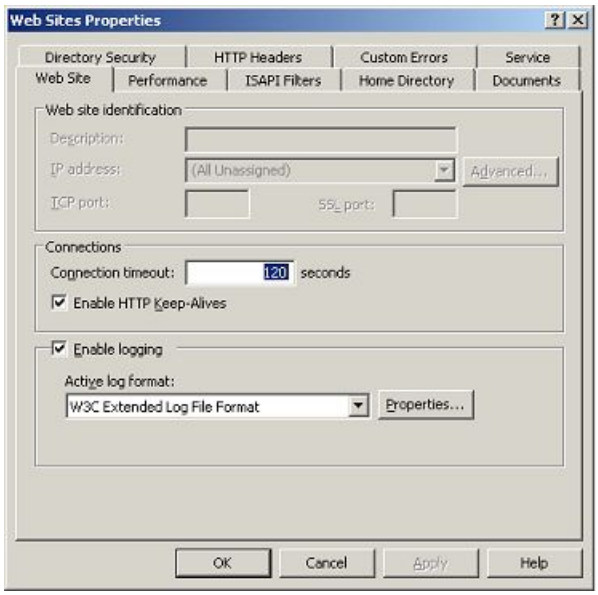


Figure 2.5: Enabling access logging for the egurkha web site

7. Enabling Logging for Web Sites on Windows 2000

In the case of an IIS web server on a Windows 2000 host, follow the steps below to enable logging for the web sites.

8. Open the **Internet Information Services** console on the IIS host using the menu sequence Start -> Programs -> Administrative Tools -> Internet Services Manager.
9. If all the web sites on the IIS web server are being monitored by eG, you will have to enable logging for all. To achieve this, right-click on the node representing the IIS host in the tree structure in the left pane of the IIS console (see Figure 2.6), and select **Properties** from the shortcut menu that appears.

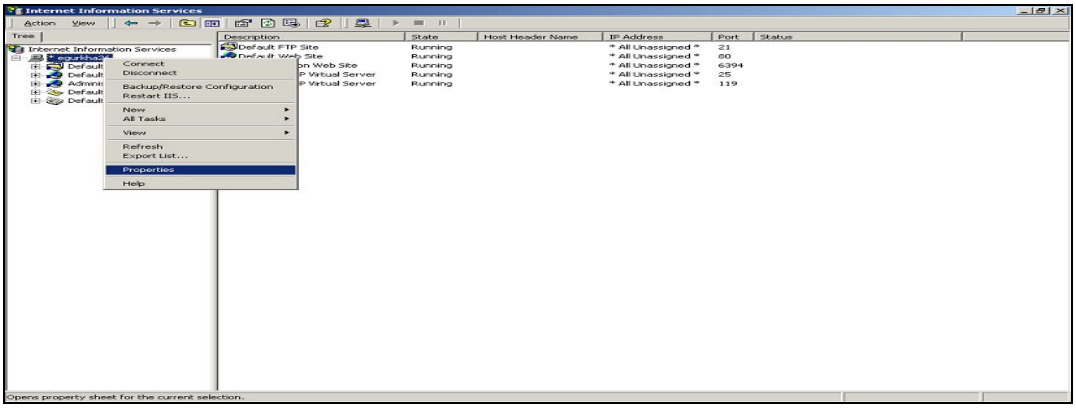


Figure 2.6: Selecting the Properties option from the shortcut menu of the IIS host node

10. Click on the **Edit** button in Figure 2.7 to modify the **Properties** of the IIS web server.

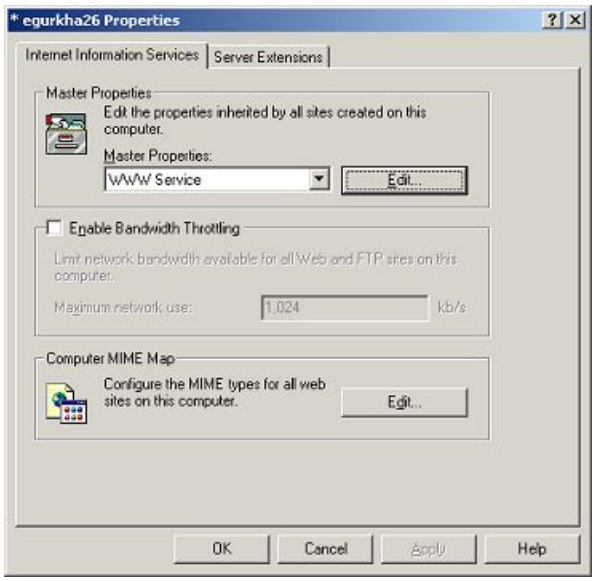


Figure 2.7: Clicking on the Edit button

11. Next, click the **Web Site** tab of the **Properties** dialog box (see Figure 2.8) that appears, and ensure that the **Enable Logging** check box is selected.

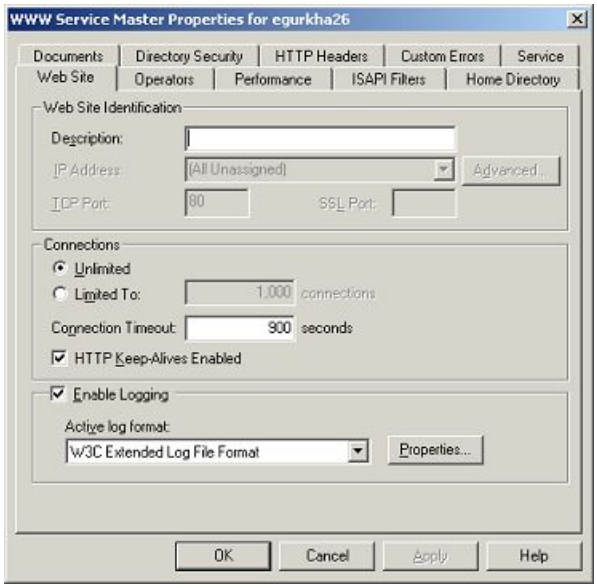


Figure 2.8: Selecting the 'Enable Logging' checkbox

12. Finally, click on the **Apply** and **OK** buttons to register the changes.

13. If only a few selected web sites on the IIS web server are being monitored by the eG Enterprise suite, then logging needs to be enabled for those specific sites only. To achieve this, right-click on the web site being monitored from the tree-structure in the left pane of the IIS console, and select **Properties** from the shortcut menu (see Figure 2.9).

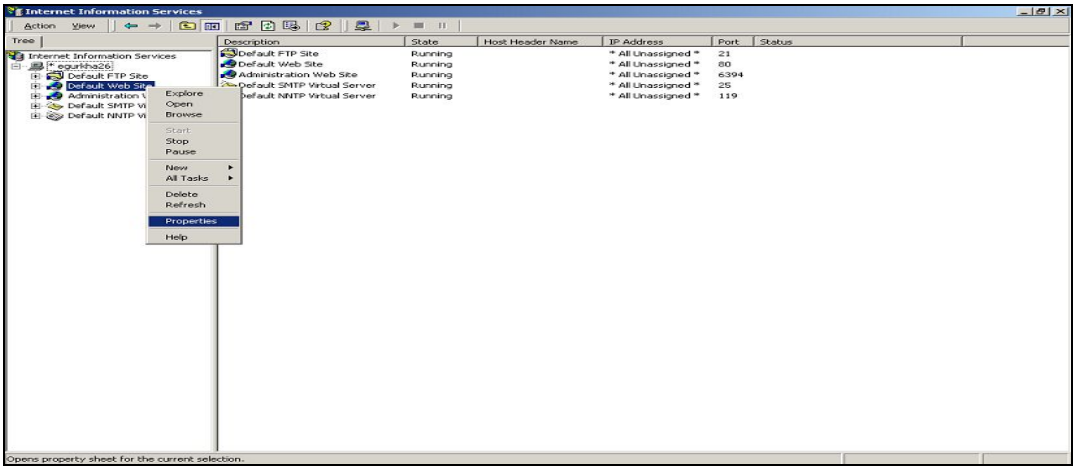


Figure 2.9: Selecting the Properties option for the Default web site

14. Next, select the **Web Site** tab from the **Properties** dialog box, and select the **Enable Logging**

check box as depicted by Figure 2.10. Finally, click on the **Apply** button and then the **OK** button.

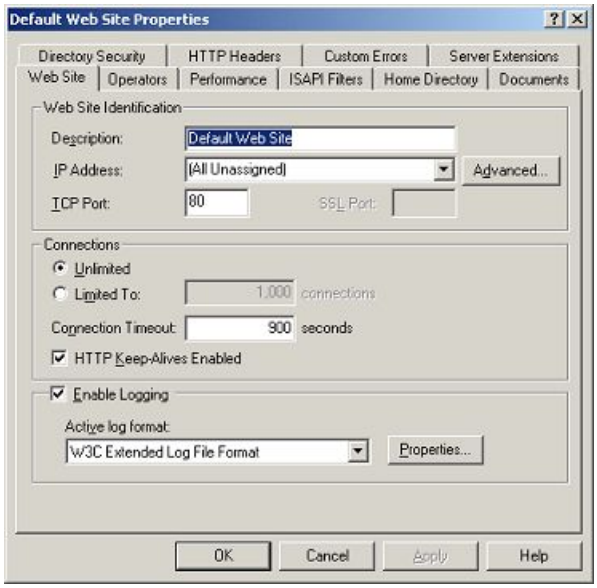


Figure 2.10: Enabling access logging for the Default web site

2.1.3 Modifying the eG Agent Configuration to Enable Web Transaction Monitoring

In order to monitor the web transactions to the web sites on an IIS web server, a specific filter needs to be installed on the IIS web server to track all requests to and from the web server. To achieve this, the eG agent on the IIS web server has to be modified. To do so, perform the steps given below:

1. Select **Uninstall Agent** from the options available under the eG Monitoring Suite -> eG Agent menu. The screen depicted by Figure 2.11 will appear. Here, select the **Modify** option and click the **Next >** button.

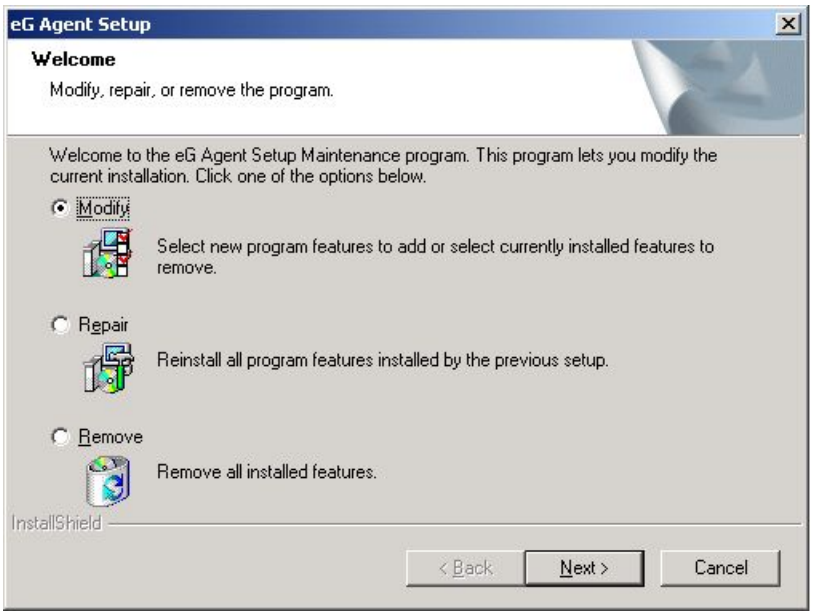


Figure 2.11: Modifying the agent configuration

2. If the eG agent setup program identifies an IIS server in the user environment, it expects the user to state if he/she wants to monitor this IIS server as depicted by Figure 2.12. If the user chooses **Yes**, the Setup installs a specific filter that will be used to track all requests to and from a web server. The default option is **No**.

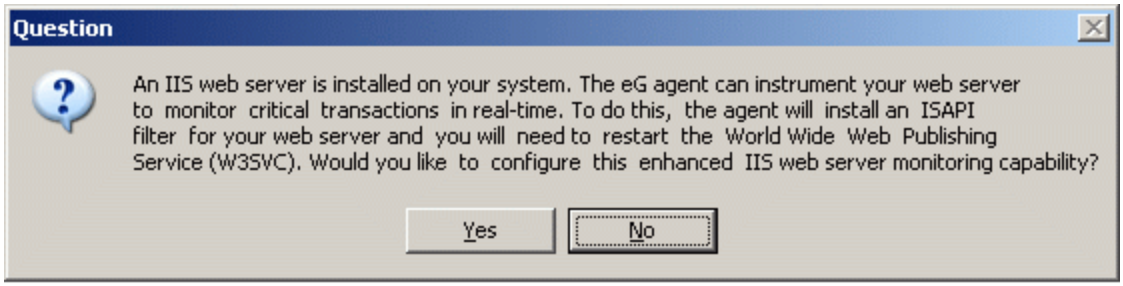


Figure 2.12: IIS web server monitoring

3. Then, restart the **World Wide Web Publishing Service**.
4. Configuring the eG Agent to Monitor an IIS Web Server Operating on Windows 2008

The eG agent can monitor an IIS web server operating on Windows 2008 only if the **Web Server** role is configured on the target Windows 2008 server.

Typically, for an IIS web server to function on a Windows 2008 server, a **Web Server Role** should be configured on the server. The **Web Server** role in Windows Server 2008 lets you share information with users on the Internet, an intranet, or an extranet. If such a role does not exist on a

Windows 2008 server, then, you cannot monitor the transactions to the IIS web server on that host; this is because, the ISAPI filter required for transaction monitoring cannot be installed on a Windows 2008 server without the **Web Server** role.

To configure this **Web Server** role on a Windows 2008 server, follow the steps detailed below:

1. Login to the Windows 2008 server as a local/domain administrator.
2. Open the **Server Manager** console by following the menu sequence, Start -> Programs -> Administrative Tools -> Server Manager (see Figure 2.13).

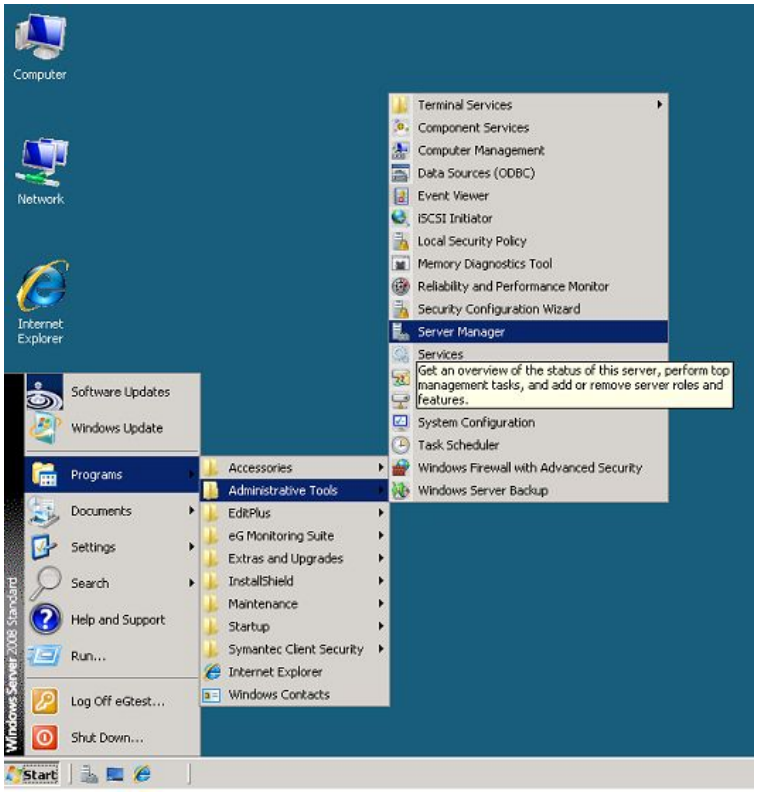


Figure 2.13: Opening the Server Manager

3. The **Server Manager** console then appears (see Figure 2.14).

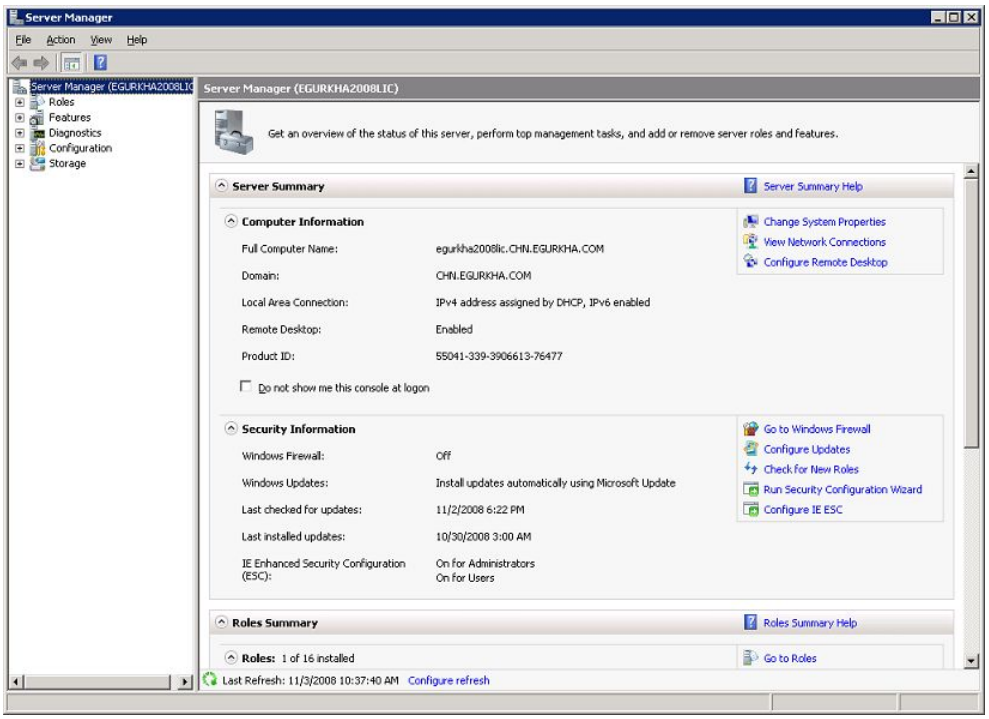


Figure 2.14: The Server Manager console

4. In the **Server Manager** console, click on the **Roles** node in the tree-structure in the left panel of the console. The information in the right-panel will change to display a **Roles Summary** and related details. To add a new role, click on the **Add Roles** option in the right panel of Figure 2.15.

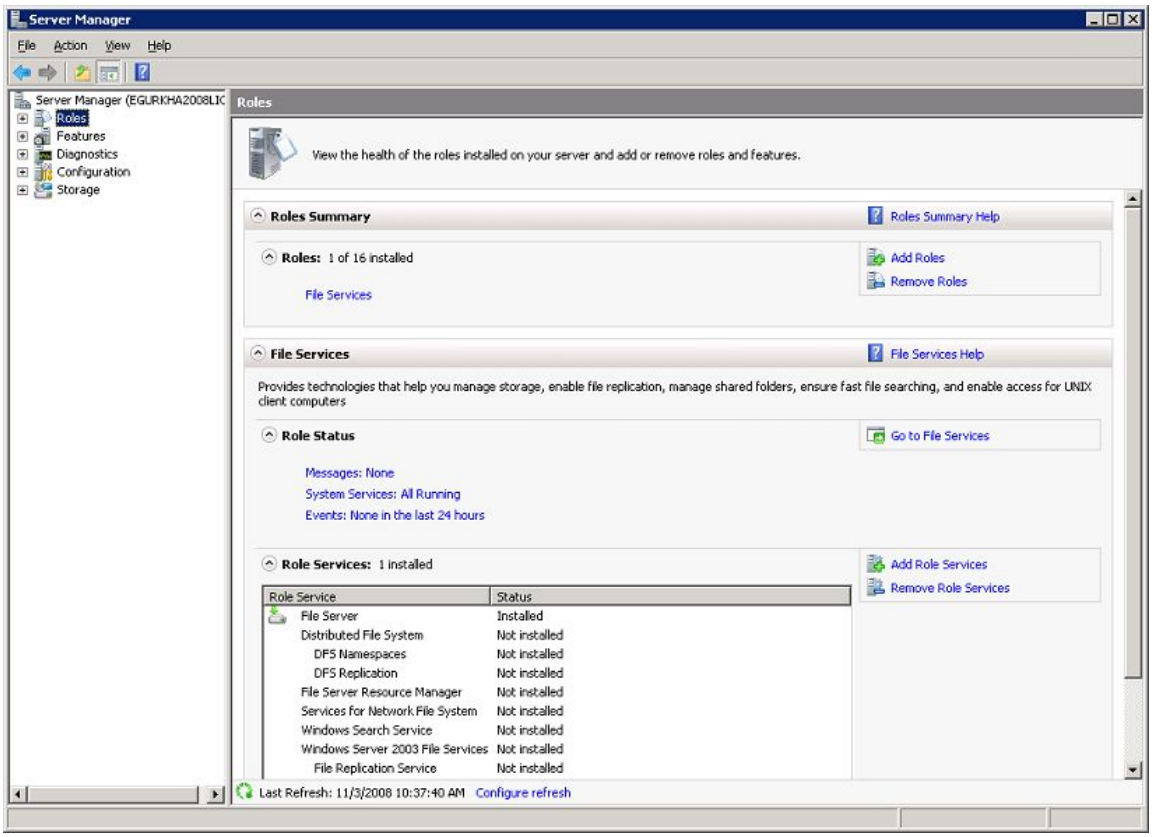


Figure 2.15: Clicking on the Roles node in the tree-structure

5. This will invoke the **Add Roles Wizard**. Click on the **Next** button in the welcome screen of Figure 2.16 to proceed with the role creation.

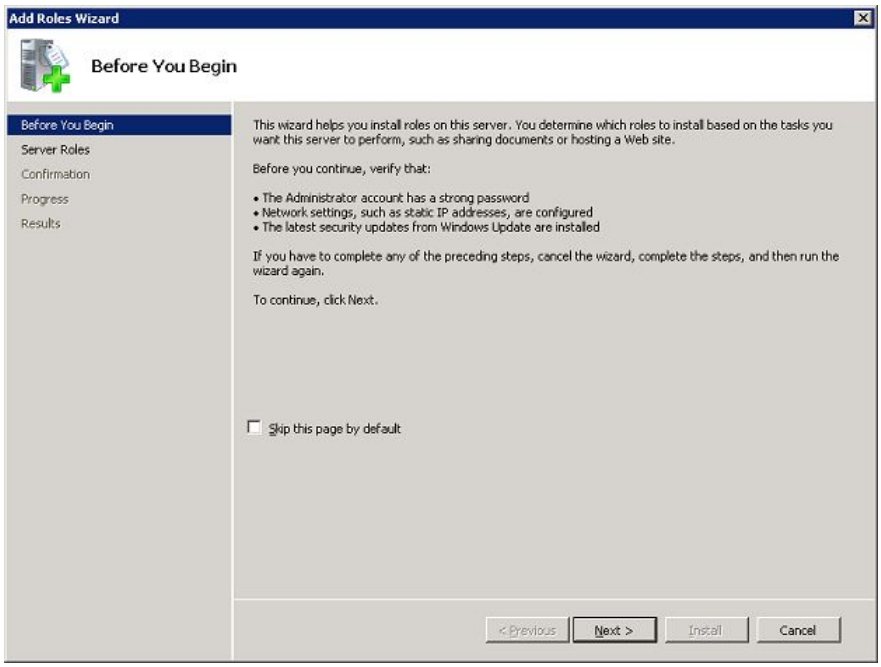


Figure 2.16: Clicking on the Next button in the welcome screen of the Add Roles Wizard

6. The next step of the wizard prompts you to pick one/more roles to install on the Windows 2008 server. Select the **Web Server (IIS)** role depicted by Figure 2.17 to install it. Then, click the **Next** button to proceed.

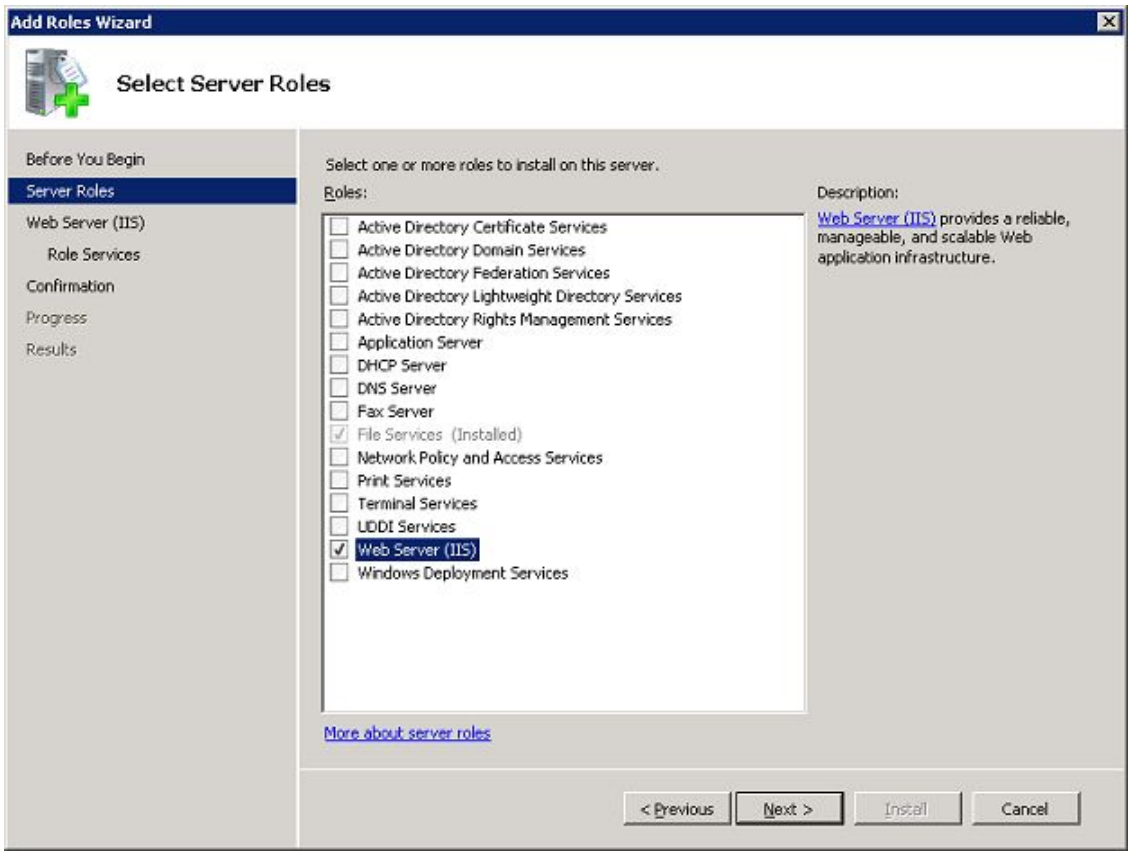


Figure 2.17: Selecting the Web Server (IIS) role

7. Then, when Figure 2.18 appears, click on the **Next** button to switch to the next step of the role installation.

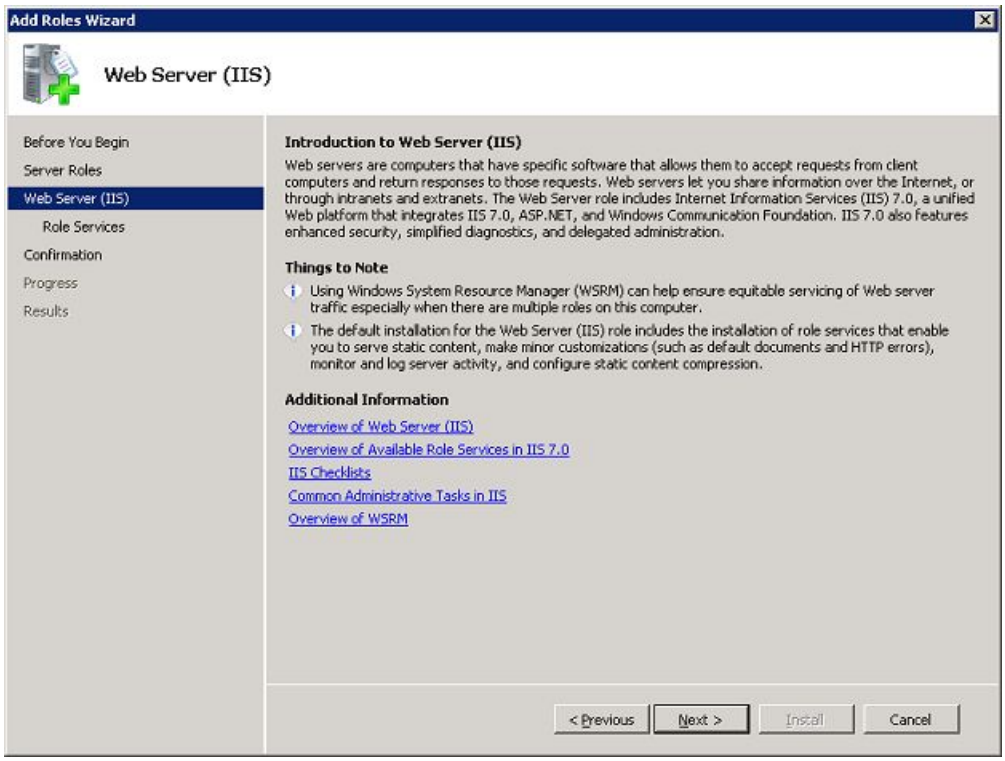


Figure 2.18: An introduction to the web server role

8. The next step will prompt you to choose the role services. Select all the listed services and click the **Next** button to proceed. **Make sure that the IIS Management Scripts and Tools feature in particular is installed and enabled for the 'Web Server' role**.

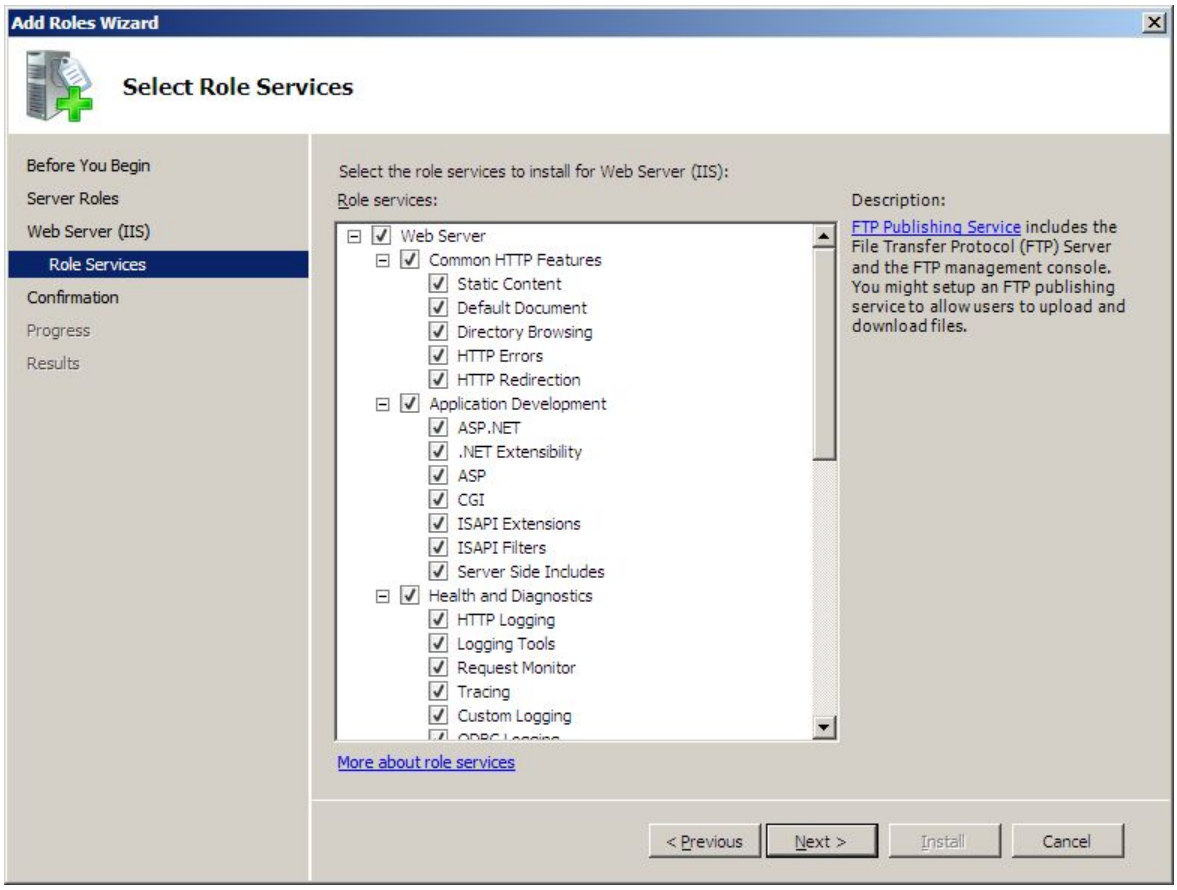


Figure 2.19: Selecting the required role services

9. The screen that appears subsequently provides a summary of your specifications. After reviewing your selections, you can confirm installation of the chosen web server role by clicking on the **Install** button in Figure 2.20.

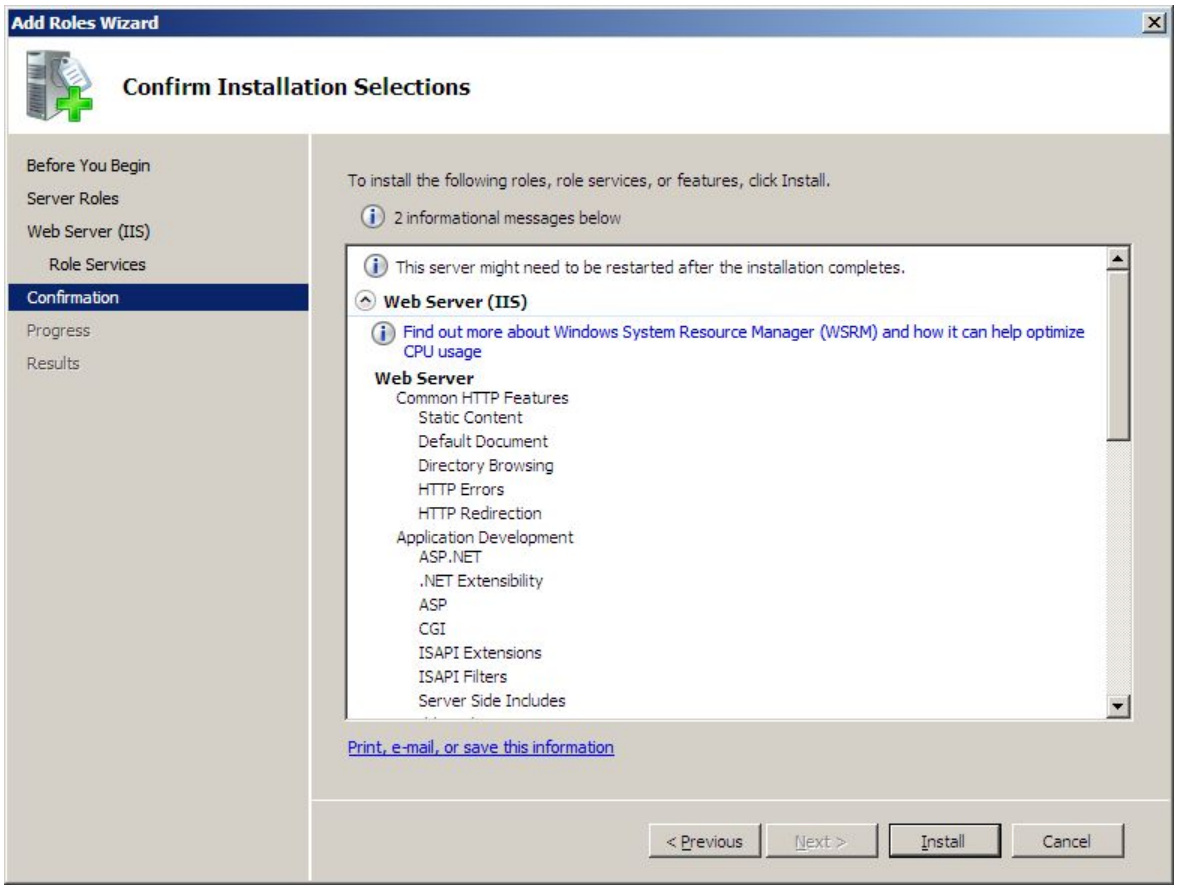


Figure 2.20: Installing the web server role

10. Once installation completes successfully, Figure 2.21 will appear confirming the success of the installation.

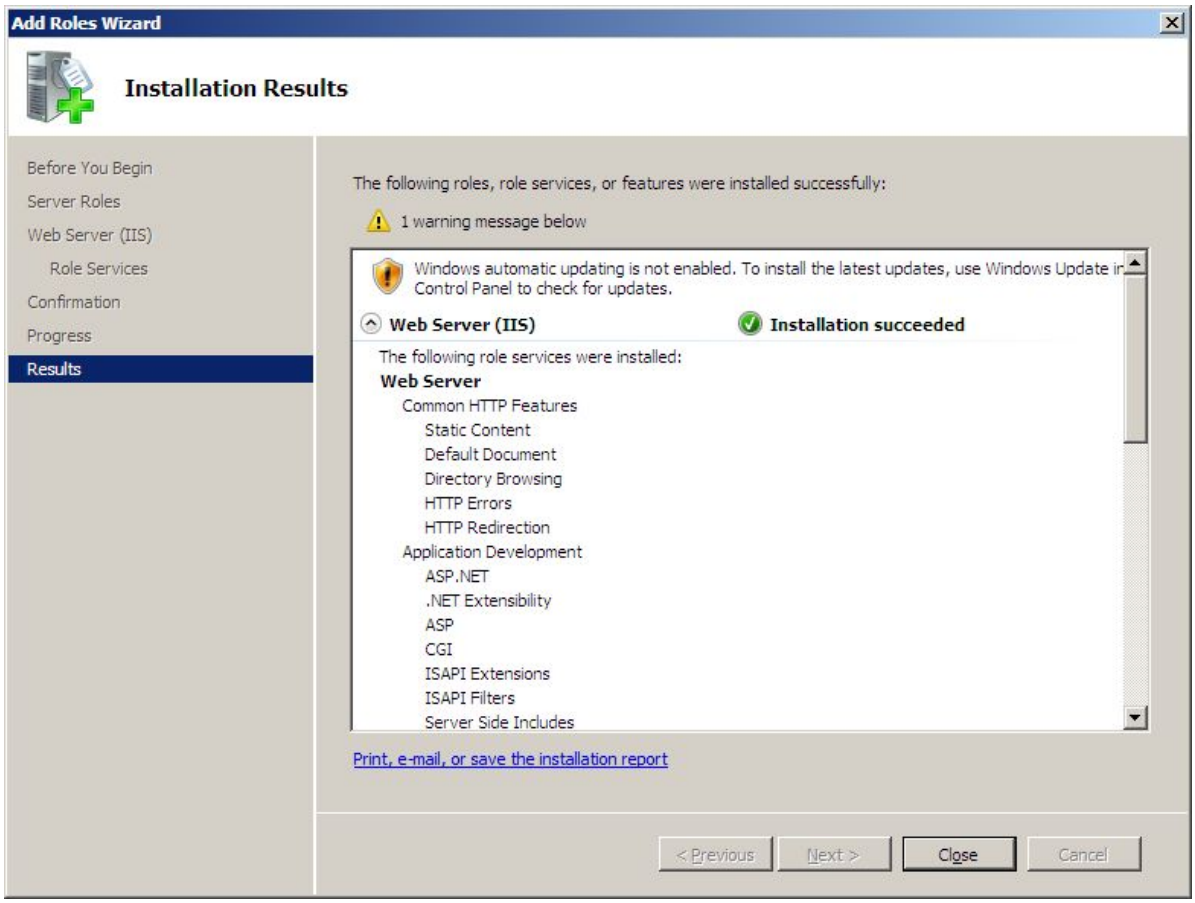


Figure 2.21: A message indicating that installation was successful

11. Click on the **Close** button in Figure 2.21 to close the wizard. Figure 2.22 will then appear displaying the newly installed role.

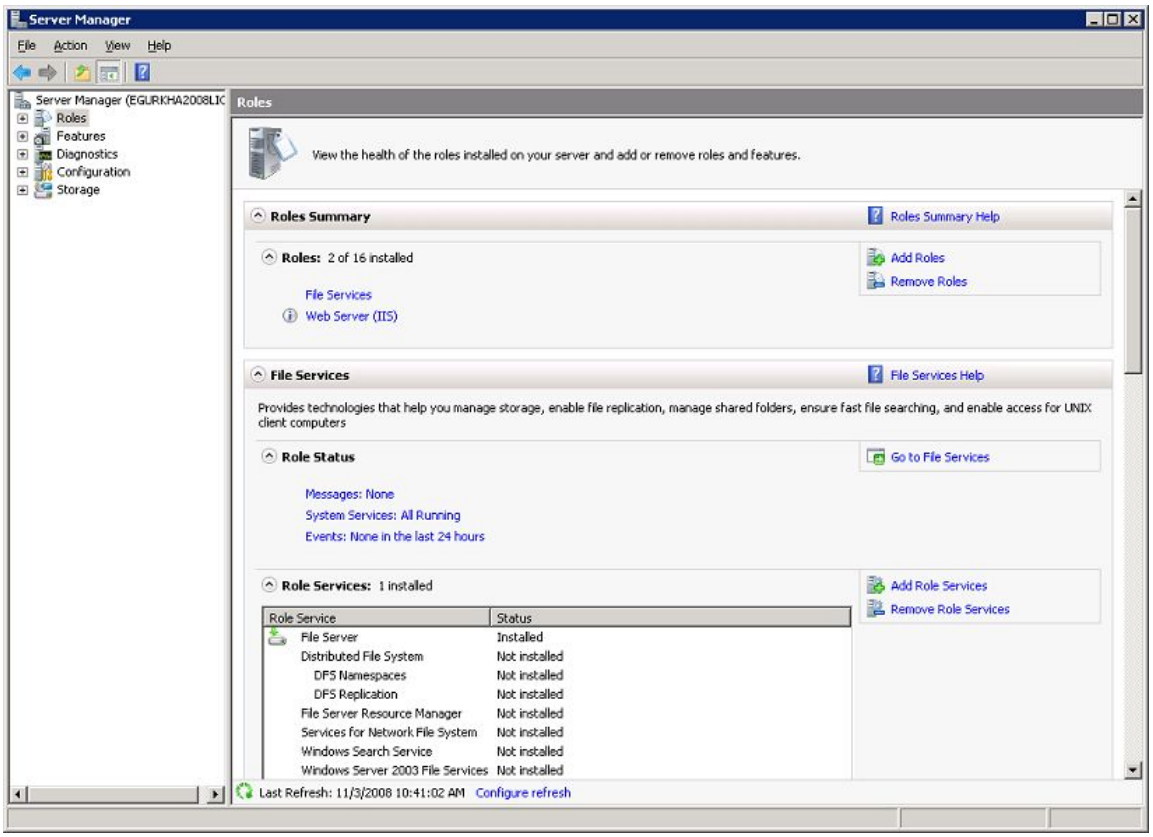


Figure 2.22: The Roles page in the right panel displaying the Web Server (IIS) role that was just installed

2.2 Configuring the eG Agent to Monitor an IIS Web Server Operating on Windows 2008

The eG agent can monitor an IIS web server operating on Windows 2008 only if the **Web Server** role is configured on the target Windows 2008 server.

Typically, for an IIS web server to function on a Windows 2008 server, a **Web Server Role** should be configured on the server. The **Web Server** role in Windows Server 2008 lets you share information with users on the Internet, an intranet, or an extranet. If such a role does not exist on a Windows 2008 server, then, you cannot monitor the transactions to the IIS web server on that host; this is because, the ISAPI filter required for transaction monitoring cannot be installed on a Windows 2008 server without the **Web Server** role.

To configure this **Web Server** role on a Windows 2008 server, follow the steps detailed below:

1. Login to the Windows 2008 server as a local/domain administrator.
2. Open the **Server Manager** console by following the menu sequence, Start -> Programs ->

Administrative Tools -> Server Manager (see Figure 2.23).

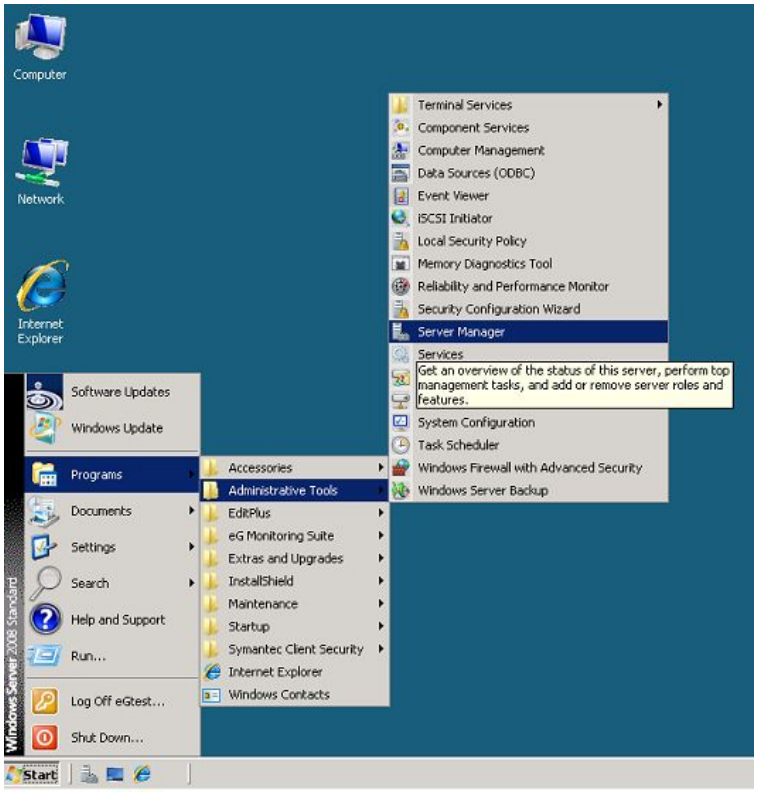


Figure 2.23: Opening the Server Manager

3. The **Server Manager** console then appears (see Figure 2.24).

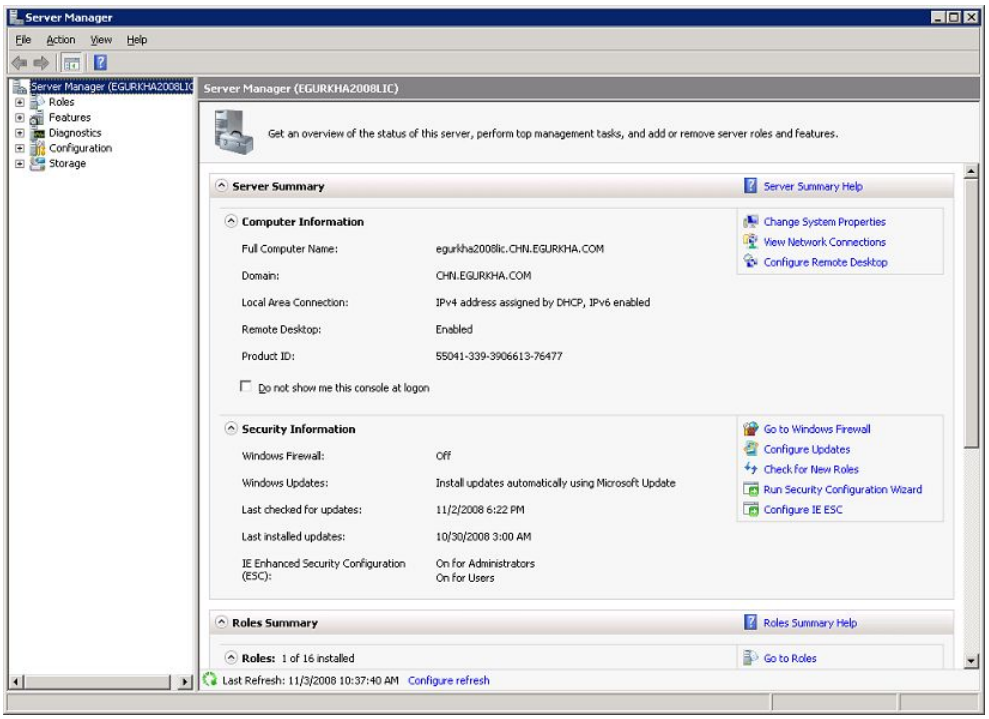


Figure 2.24: The Server Manager console

4. In the **Server Manager** console, click on the **Roles** node in the tree-structure in the left panel of the console. The information in the right-panel will change to display a **Roles Summary** and related details. To add a new role, click on the **Add Roles** option in the right panel of Figure 2.25.

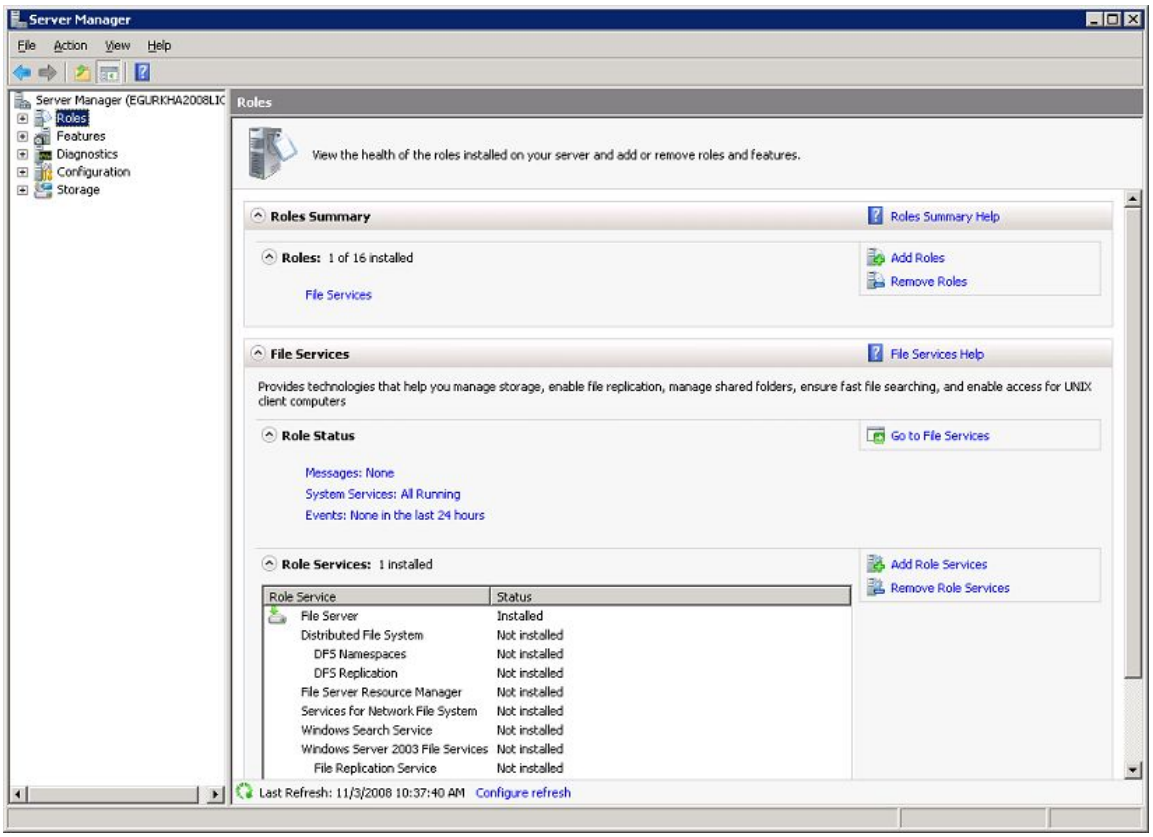


Figure 2.25: Clicking on the Roles node in the tree-structure

5. This will invoke the **Add Roles Wizard**. Click on the **Next** button in the welcome screen of Figure 2.26 to proceed with the role creation.

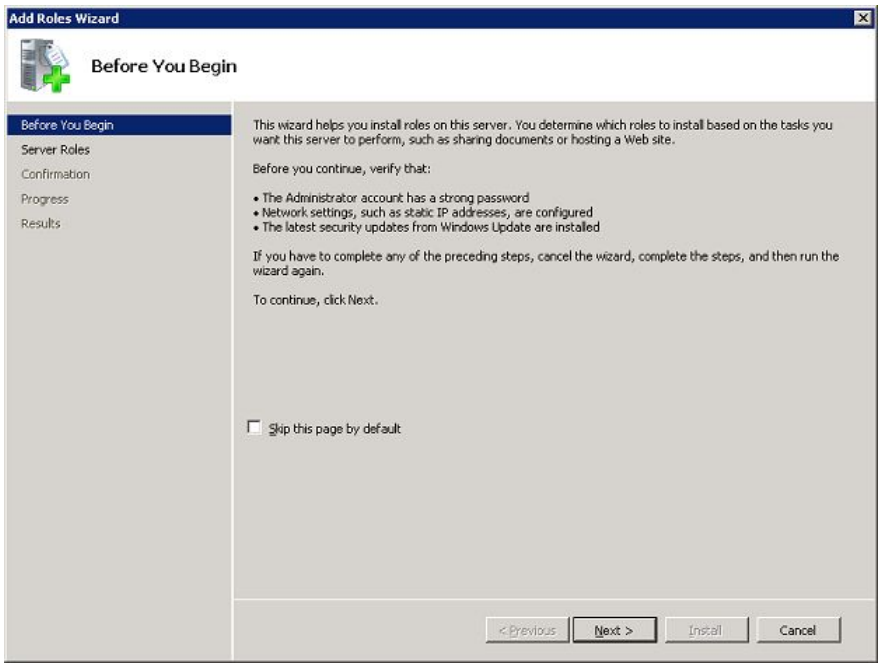


Figure 2.26: Clicking on the Next button in the welcome screen of the Add Roles Wizard

6. The next step of the wizard prompts you to pick one/more roles to install on the Windows 2008 server. Select the **Web Server (IIS)** role depicted by Figure 2.27 to install it. Then, click the **Next** button to proceed.

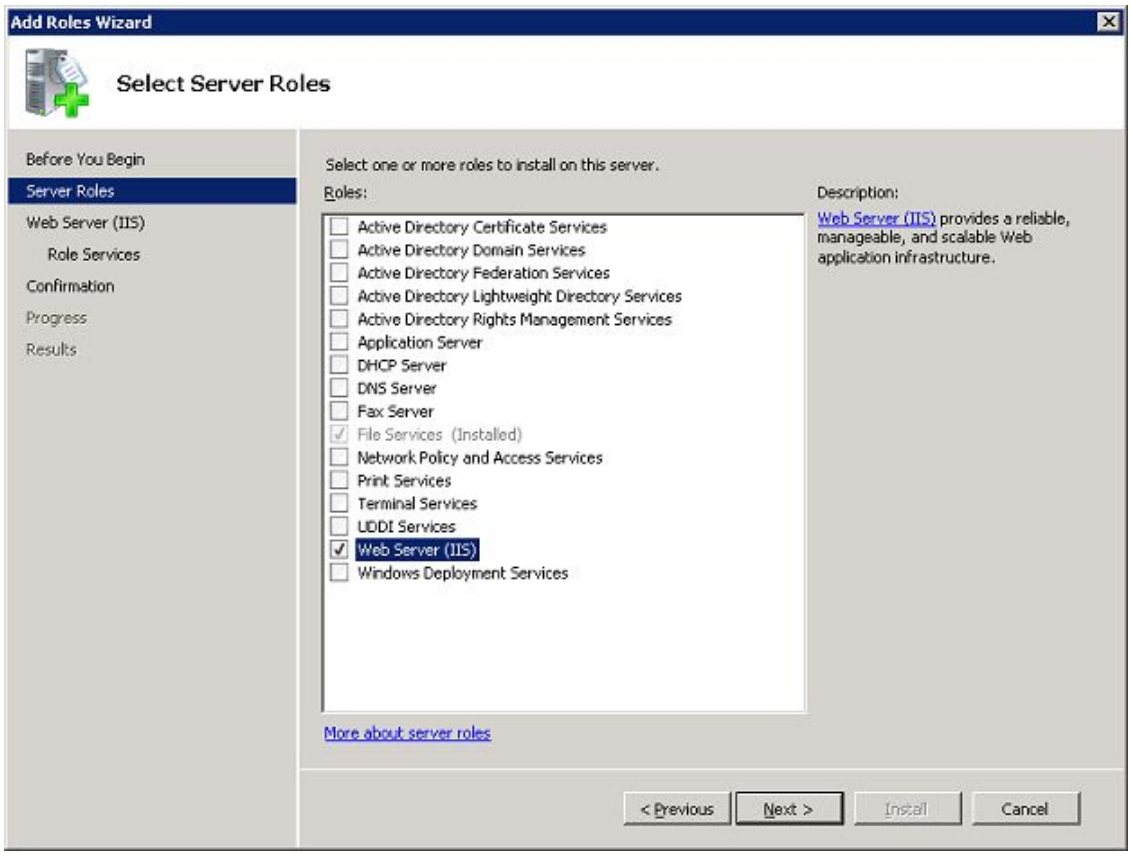


Figure 2.27: Selecting the Web Server (IIS) role

7. Then, when Figure 2.28 appears, click on the **Next** button to switch to the next step of the role installation.

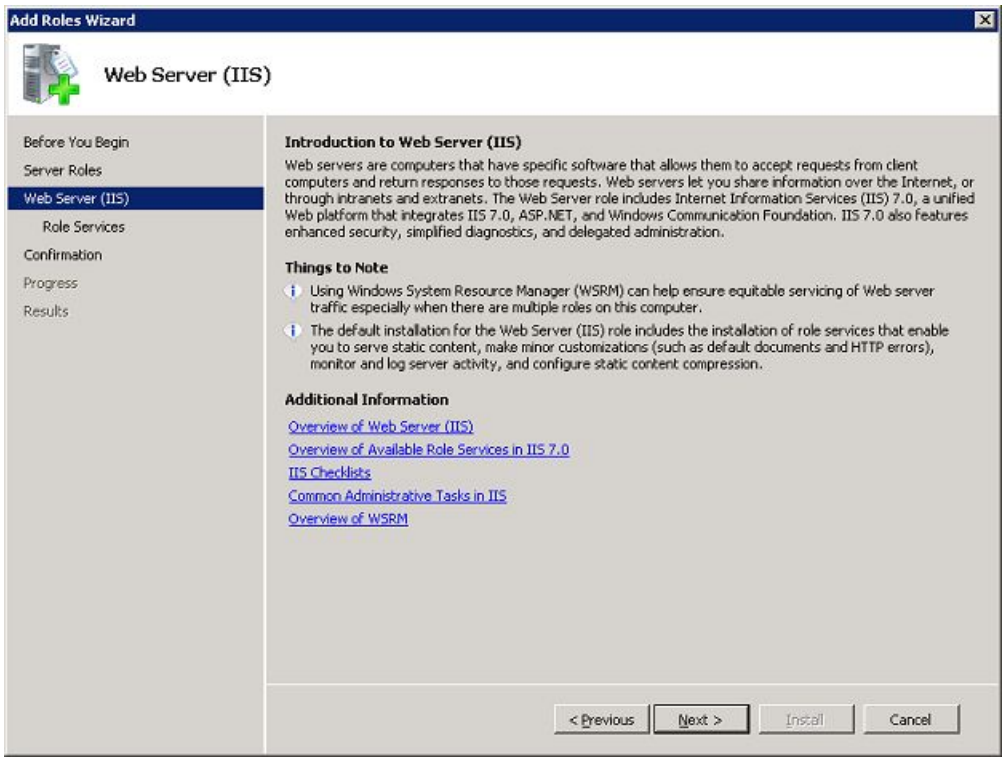


Figure 2.28: An introduction to the web server role

8. The next step will prompt you to choose the role services. Select all the listed services and click the **Next** button to proceed. **Make sure that the IIS Management Scripts and Tools feature in particular is installed and enabled for the 'Web Server' role.**

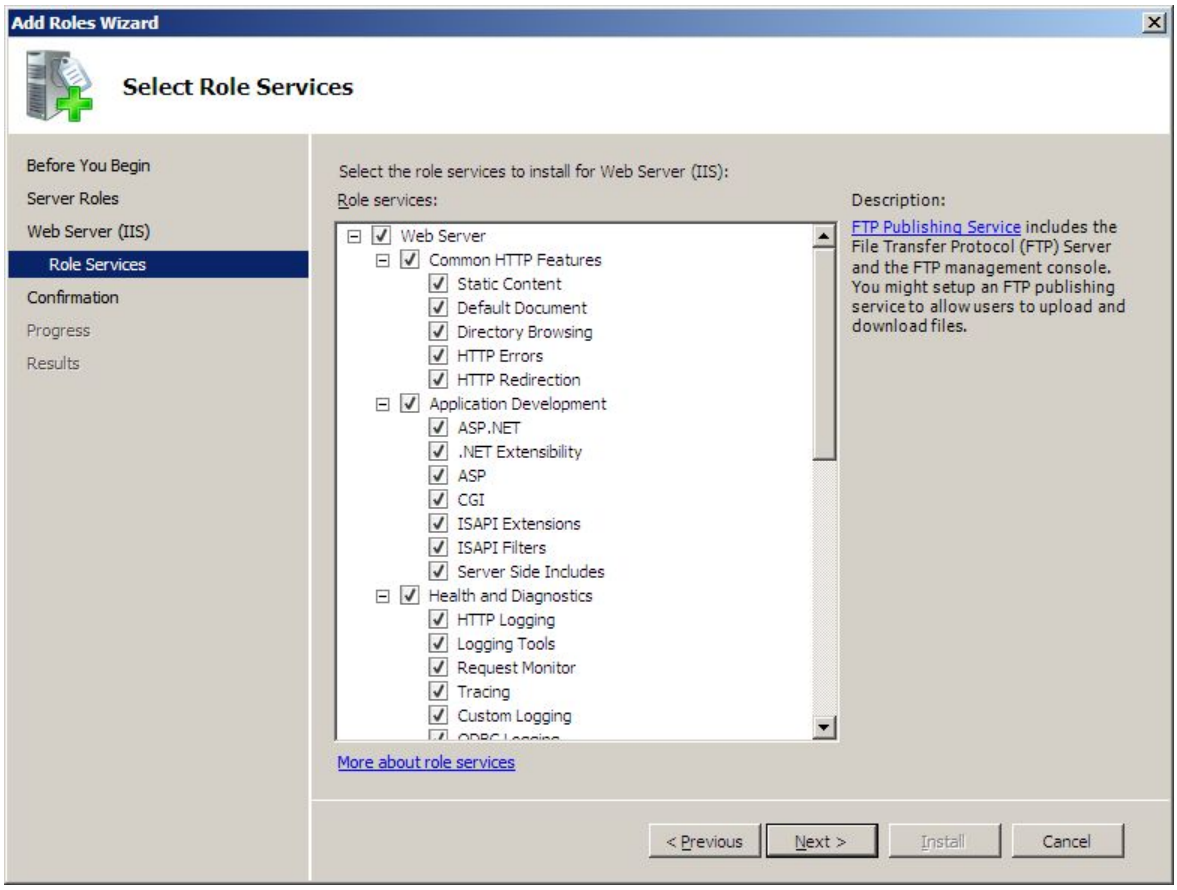


Figure 2.29: Selecting the required role services

9. The screen that appears subsequently provides a summary of your specifications. After reviewing your selections, you can confirm installation of the chosen web server role by clicking on the **Install** button in Figure 2.30.

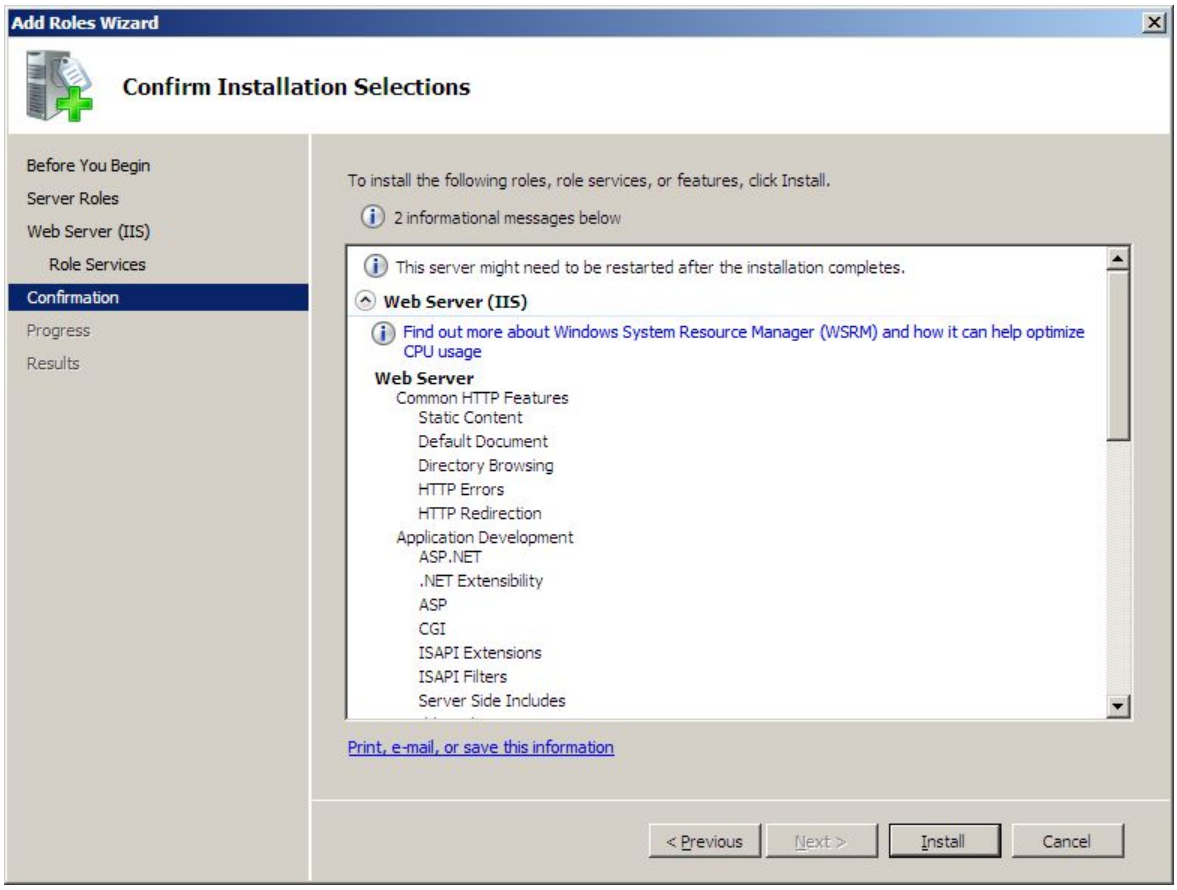


Figure 2.30: Installing the web server role

10. Once installation completes successfully, Figure 2.31 will appear confirming the success of the installation.

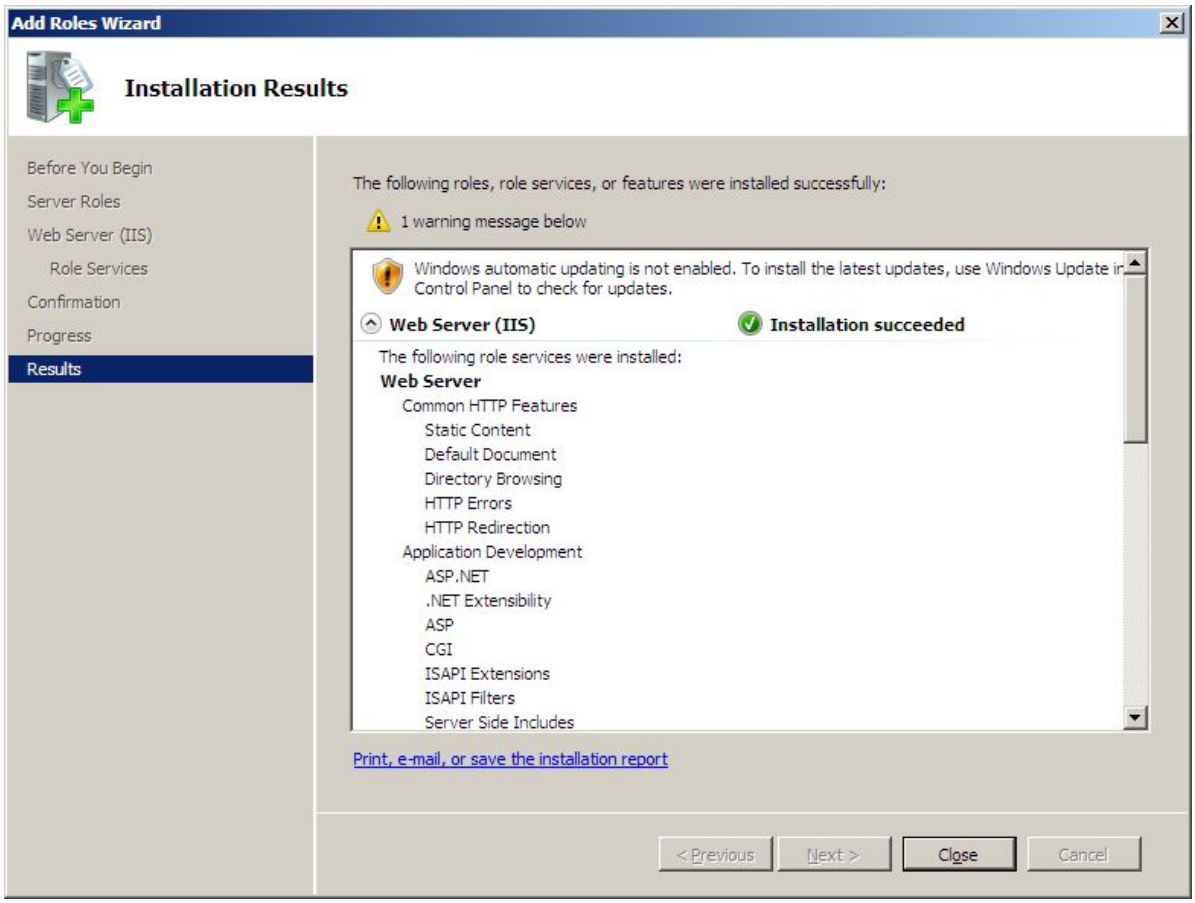


Figure 2.31: A message indicating that installation was successful

11. Click on the **Close** button in Figure 2.31 to close the wizard. Figure 2.32 will then appear displaying the newly installed role.

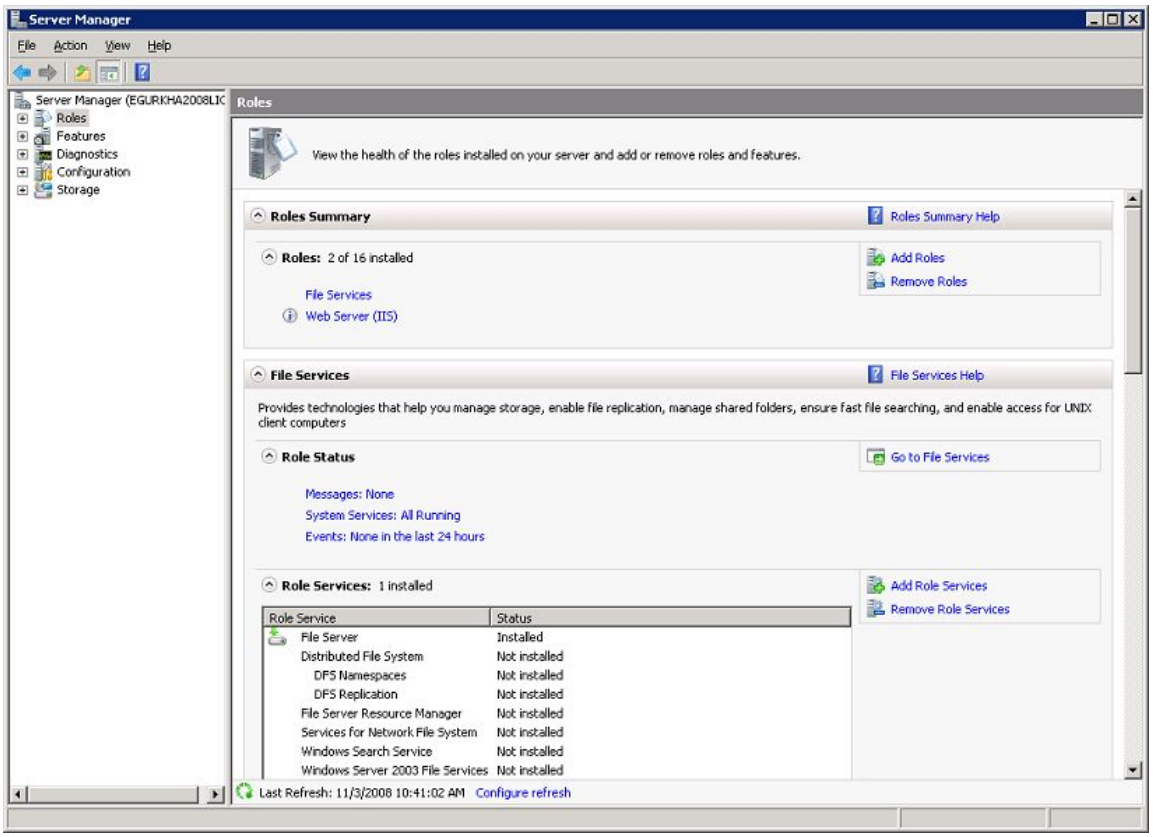


Figure 2.32: The Roles page in the right panel displaying the Web Server (IIS) role that was just installed

2.3 Configuring the eG Agent to Monitor the Web Transactions to Web Sites on an IIS Web Server Operating on Windows 2008

To perform web site transaction monitoring on an IIS web server executing on Windows 2008, you need to install and configure Advanced Logging on the target IIS web server, soon after you create the Web Server role on the Windows 2008 server.

IIS Advanced Logging is an extension for Internet Information Services (IIS) 7 that provides enhanced data collection and real-time server and client-side logging capabilities. It can be managed by using IIS Manager and other tools that can work with the IIS 7 configuration system.

The Advanced Logging feature supports complex Web and media delivery scenarios that demand flexibility and control. These scenarios may require custom logging fields, real-time access to data, greater control over what gets logged and when, extensibility for new sources of data, the ability to consolidate log data posted by clients and correlate it to server data, the option of sharing data from various sources and storing it in multiple logs, capturing system-state information, inclusion of canceled requests in logs, and even logging multiple times per request.

In order to monitor the web transactions to IIS 7 (that is bundled with the Windows 2008 server), the eG agent requires that the **Advanced Logging** be installed and configured on IIS 7. The steps in this regard have been discussed below:

1. Login to the IIS host.
2. Download the executable that installs the **Advanced Logging** feature from any of the following URLs, depending upon whether the IIS 7 installation is a 32-bit one or a 64-bit one:

32-bt/64-bit	URL
32-bit	HTTP://www.microsoft.com/downloads/en/details.aspx?FamilyID=4d110e78-95cb-4764-959c-b8afc33df496&displaylang=en
64-bit	HTTP://www.microsoft.com/downloads/en/details.aspx?FamilyID=793051A8-36A0-4342-BDFE-47A6B0E3488F

3. Once the download is complete, go to the directory to which the executable was downloaded and double-click on it.
4. Figure 2.33 will then appear. Accept the license by selecting the **I accept the terms in the License Agreement** check box, and click on the **Install** button to proceed with the installation.

Figure 2.33: Accepting the license agreement

5. Once the installation ends, Figure 2.34 will appear indicating the successful installation of the **Advanced Logging** feature. Click the **Finish** button to exit the wizard.

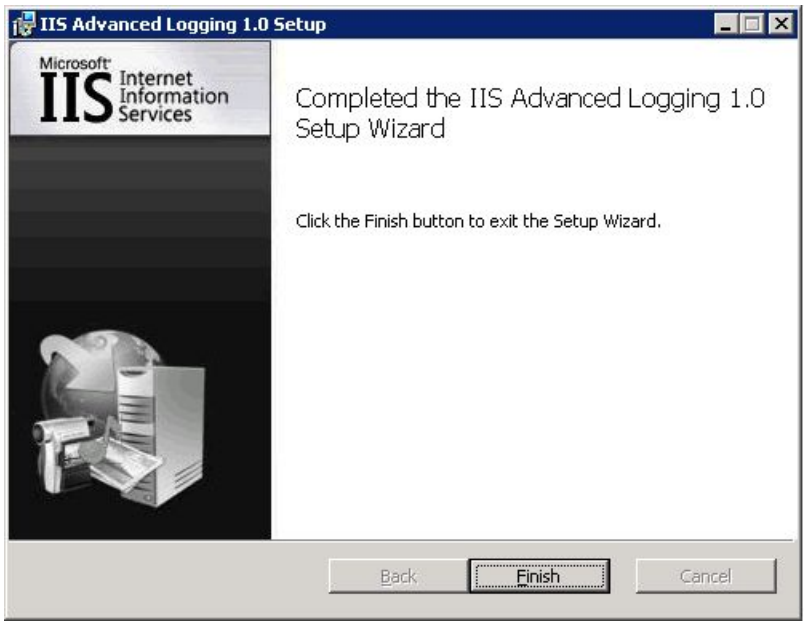


Figure 2.34: Finishing the installation

6. Next, proceed to configure the Advanced Logs. For that, first, open the **Internet Information Services (IIS) Manager** console using the menu sequence: Start -> Programs -> Administrative Tools -> Internet Information Services (IIS) Manager. Figure 2.35 will then appear.

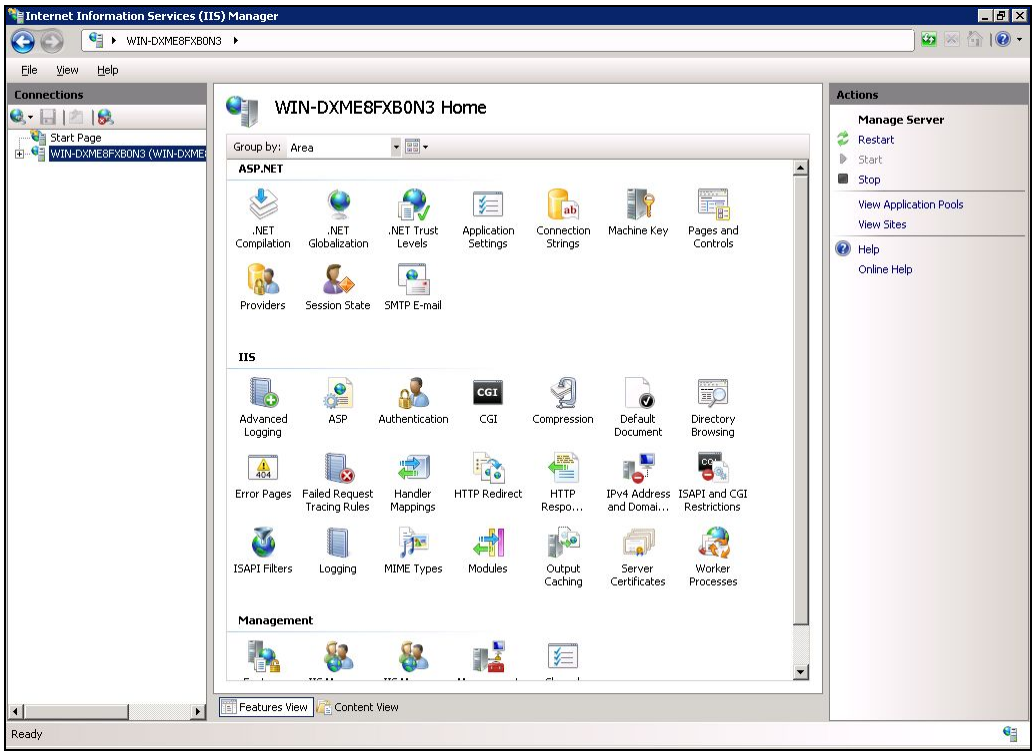


Figure 2.35: The Internet Information Services (IIS) Manager console

7. Click on the node representing the IIS web server host in the tree-structure in the left panel of the console. The right panel will change to display a variety of options. In the **IIS** section of the right panel, click on the **Advanced Logging** option. Figure 2.36 will then appear. In the **Actions** list in the right panel, click on the **Add Log Definition** option (as indicated by Figure 2.36) to add a new log definition.

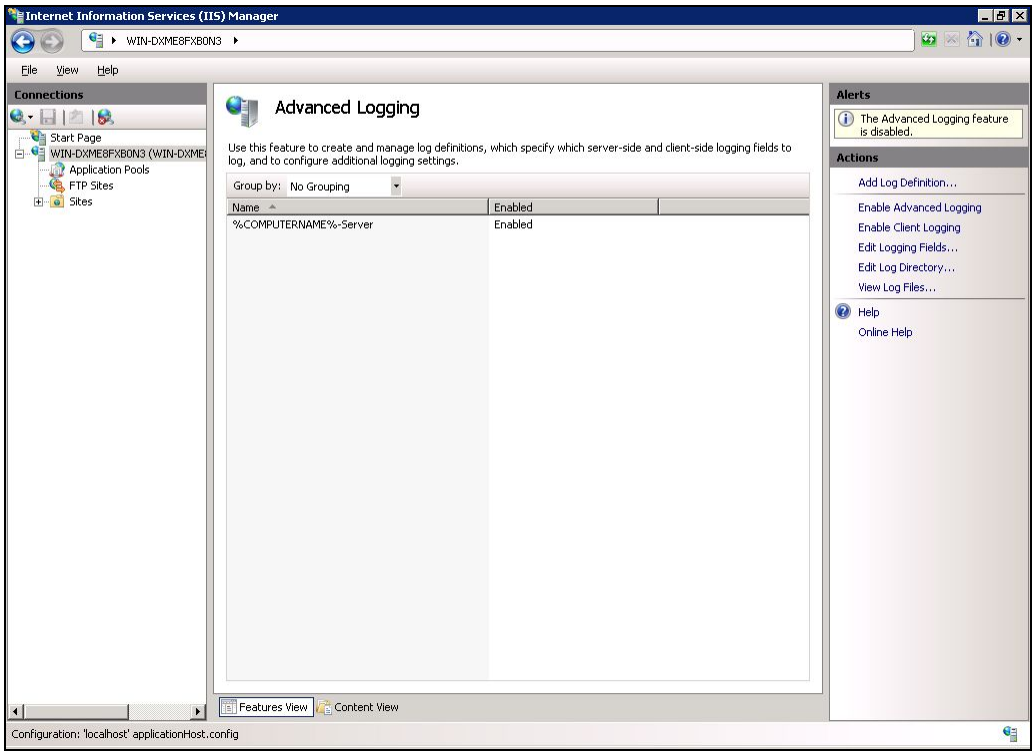


Figure 2.36: Viewing the list of log definitons that pre-exist

8. In the **Log Definition** page that appears, specify **WebAdapterFile** as the **Base file name**. Check the **Enabled** option, the **Publish real-time events** option, and the **Write to disk** sub-option.

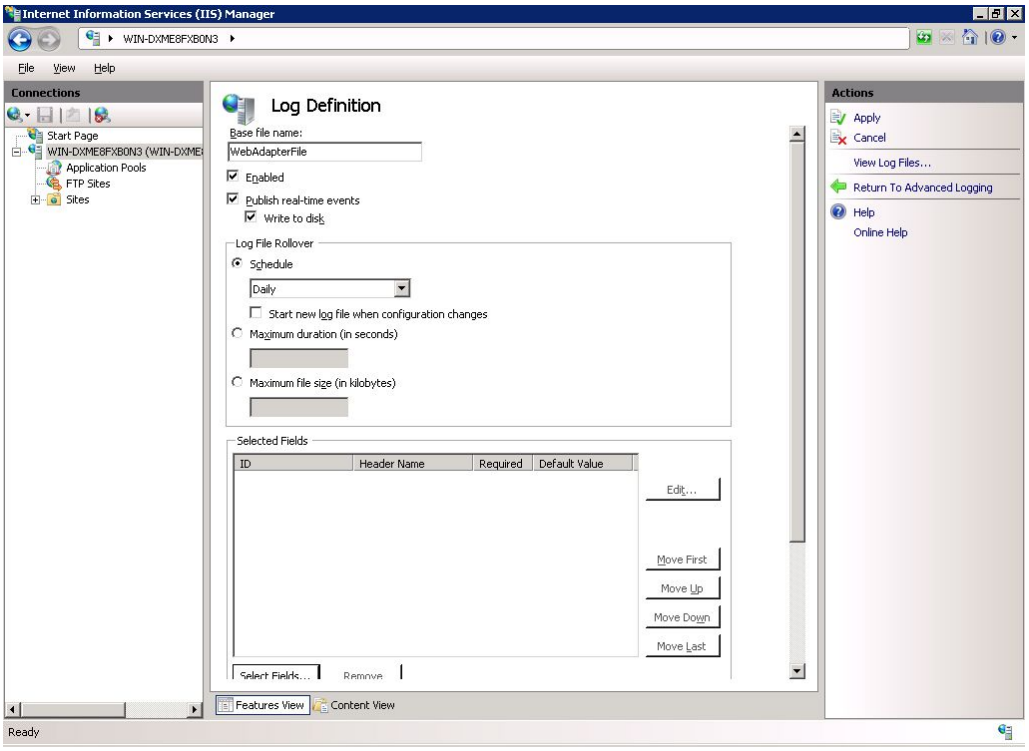


Figure 2.37: Adding a new log file definition

9. Then, click on the **Select Fields** button at the bottom of the **Log Definition** page to select he server-side and client-side logging fields to be logged in the specified log file. Doing so will invoke Figure 2.38, from which you wil have to select the following fields:

   - UserName
   - URI-Stem
   - URI-QueryString
   - Time-Local
   - Time Taken
   - Status
   - Server-IP
   - Server Port
   - Server Name
   - Site Name
   - CPU-utilization

- Bytes Sent
- Bytes Received
- Host
- Client Ip
- Date-local

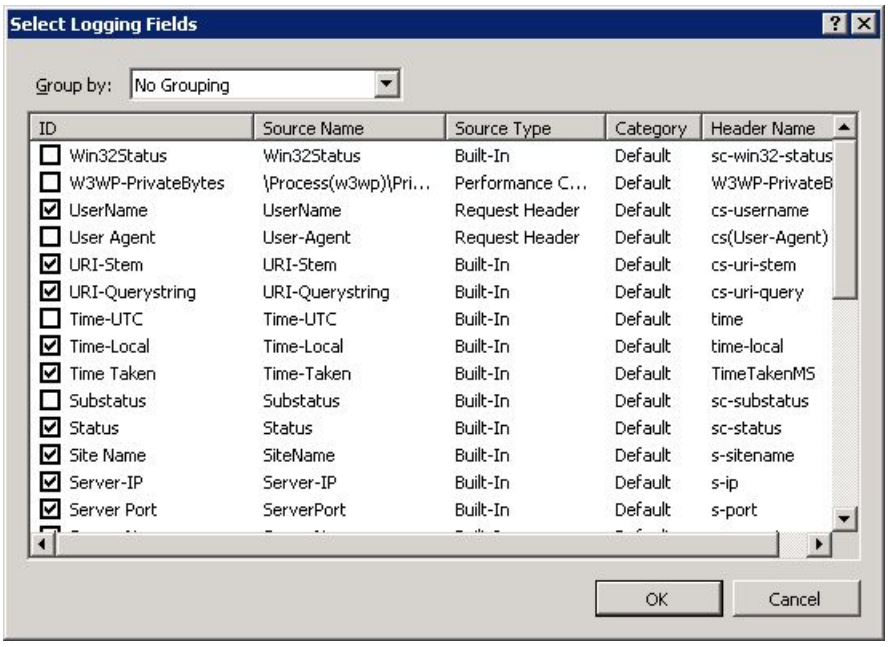


Figure 2.38: Selecting the logging fields to be logged

10. Click on the **OK** button in Figure 2.38 to confirm the selection. When this is done, the **Selected Fields** section of the **Log Definition** page will get updated with your selection (see Figure 2.39). Use the **Move First, Move Up**, **Move Down**, and **Move Last** buttons adjacent to your selection to re-arrange the sequence of the logging fields. The desired sequence is as follows:

- Time-Local
- Host
- Server-IP
- Server Port
- Status
- URI-stem

- URI-QueryString
- CPU-utilization
- Bytes Sent
- Bytes Received
- Time Taken
- Server Name
- Site Name
- User Name
- Client Ip
- Date-local

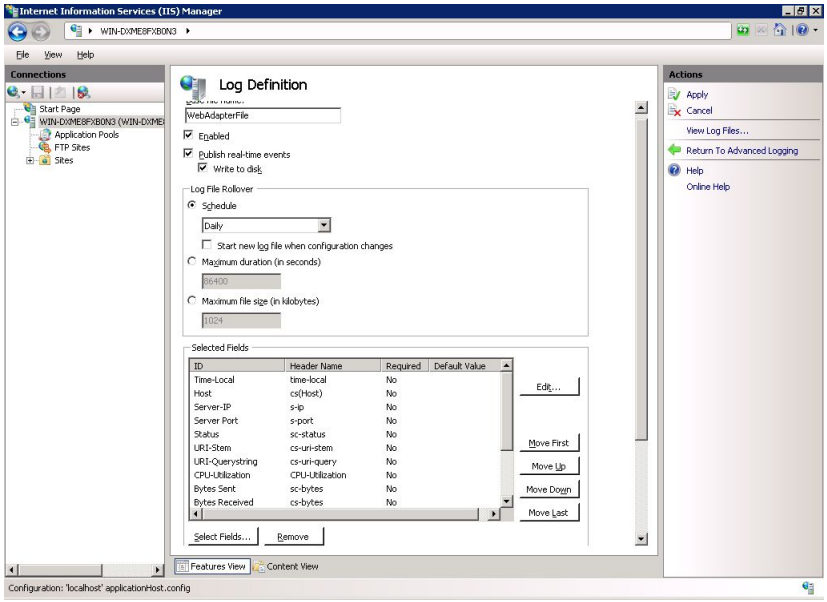


Figure 2.39: Re-arranging the sequence of the logging fields

11. Then, apply the changes by clicking on the **Apply** button indicated by Figure 2.39 above. Once the changes are saved, click on the **Return to Advanced Logging** option indicated by Figure 2.39 above. Figure 2.40 will then appear. In the right panel of Figure 2.40, you will find that the newly added **WebAdapterFile** is appended to the list of log file definitions that pre-exist.

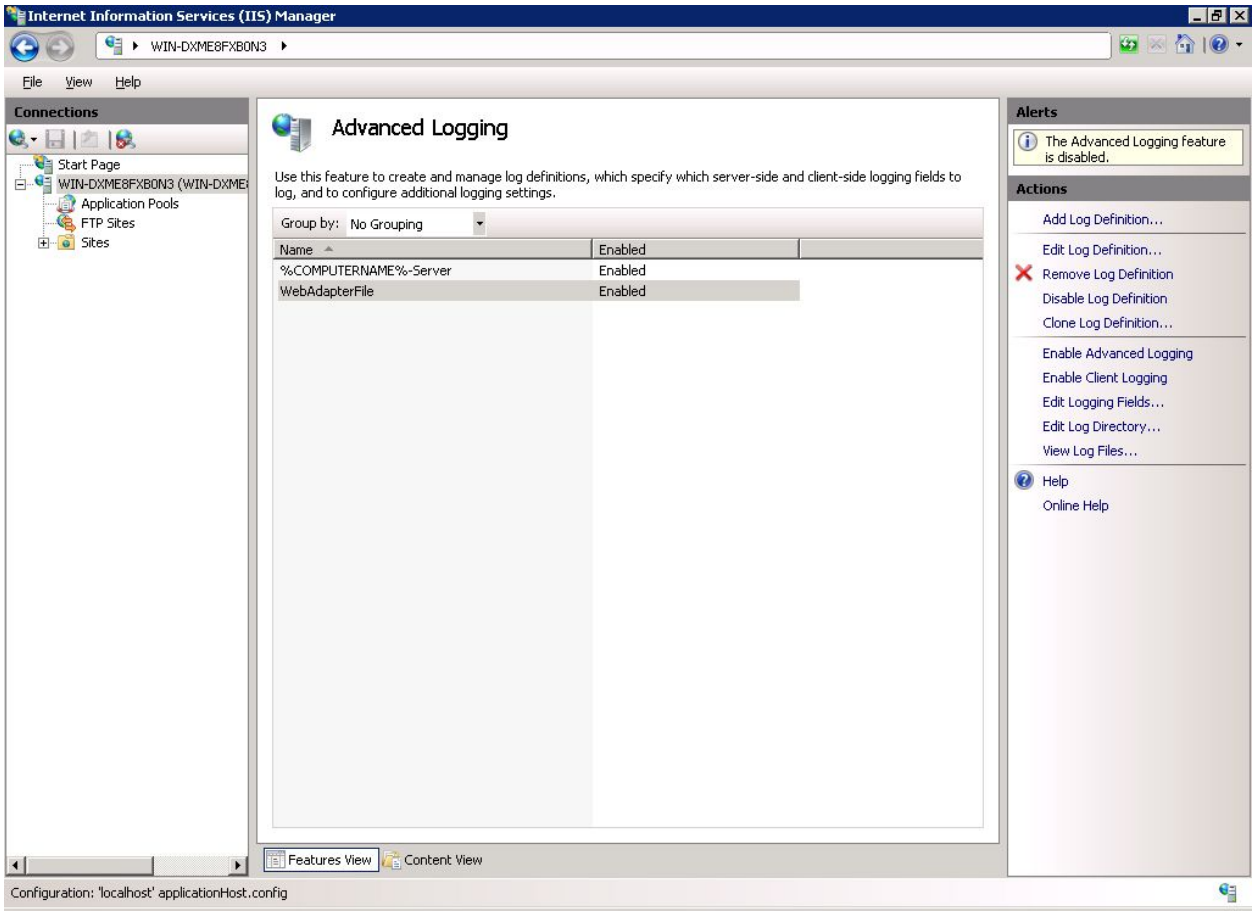


Figure 2.40: The newly added log definition displayed in the list of log files that pre-exist

12. Now, select the **WebAdapterFile** entry in Figure 2.40 and click on the **Edit Log Directory** option in the **Actions** list, as indicated by Figure 2.40. When Figure 2.41 appears, change the default values of the **Server log directory** and **Default site log directory** text boxes to **<EG_ INSTALL_ DIR>\agent\logs\IISAdvlogs** directory. Then, click the **OK** button therein.

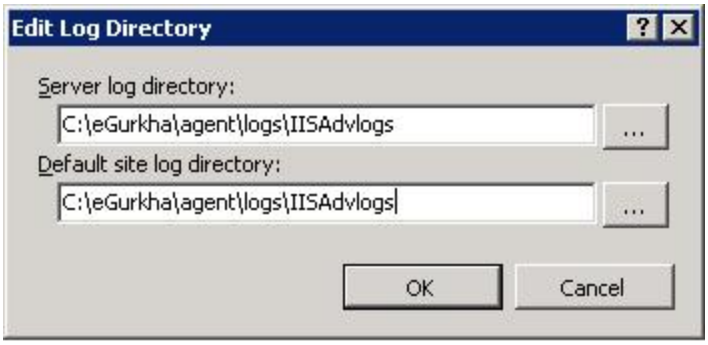


Figure 2.41: Changing the server log and default site log directories

13. You will then return to Figure 2.40. Select the **WebAdapterFile** entry yet again, and this time, click on the **View Log Files** option in the **Actions** list. This will invoke Figure 2.42, where all the log files saved to the **<EG_INSTALL_DIR>\agent\logs\IISAdvlogs** directory will be displayed.

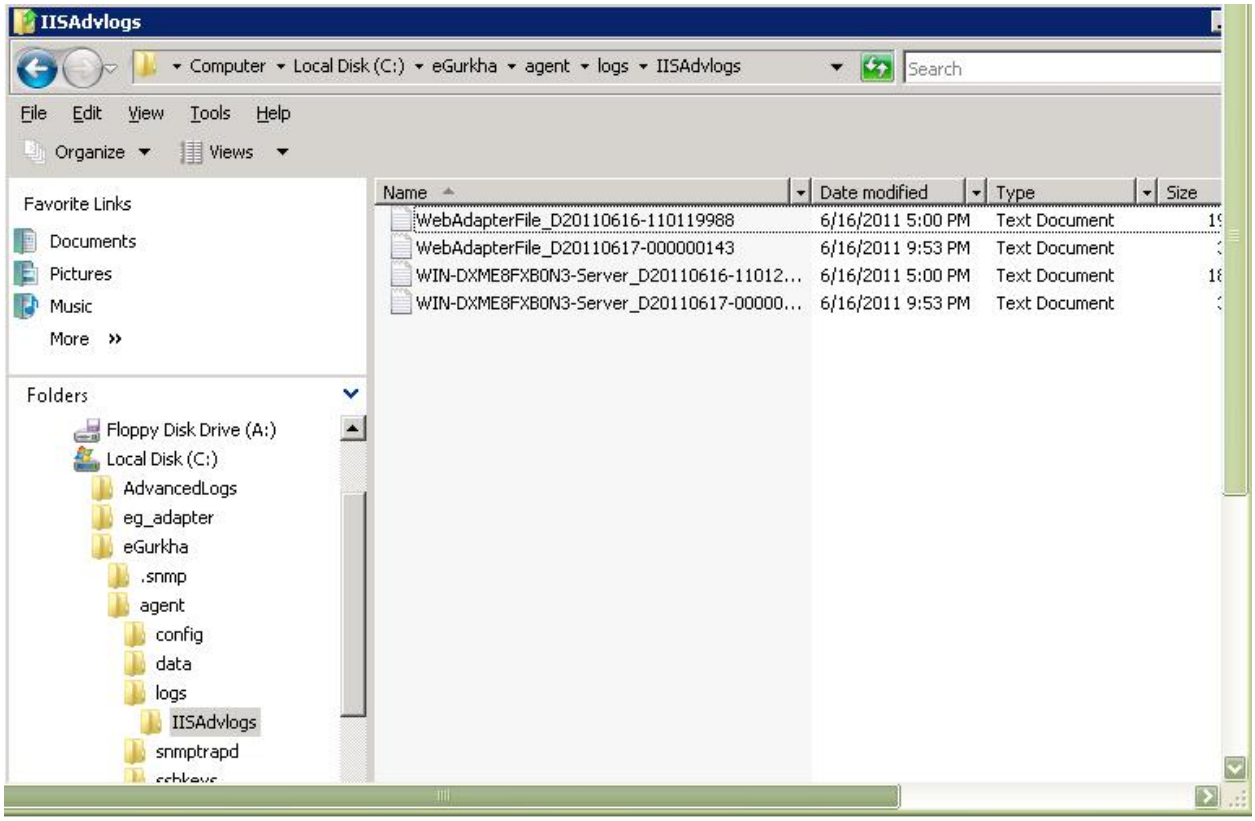


Figure 2.42: List of log files saved to the AdvancedLogs directory

14. To view a log file, click on any of the log files in the list of Figure 2.42. The chosen log file will then open in Notepad as depicted by Figure 2.43.

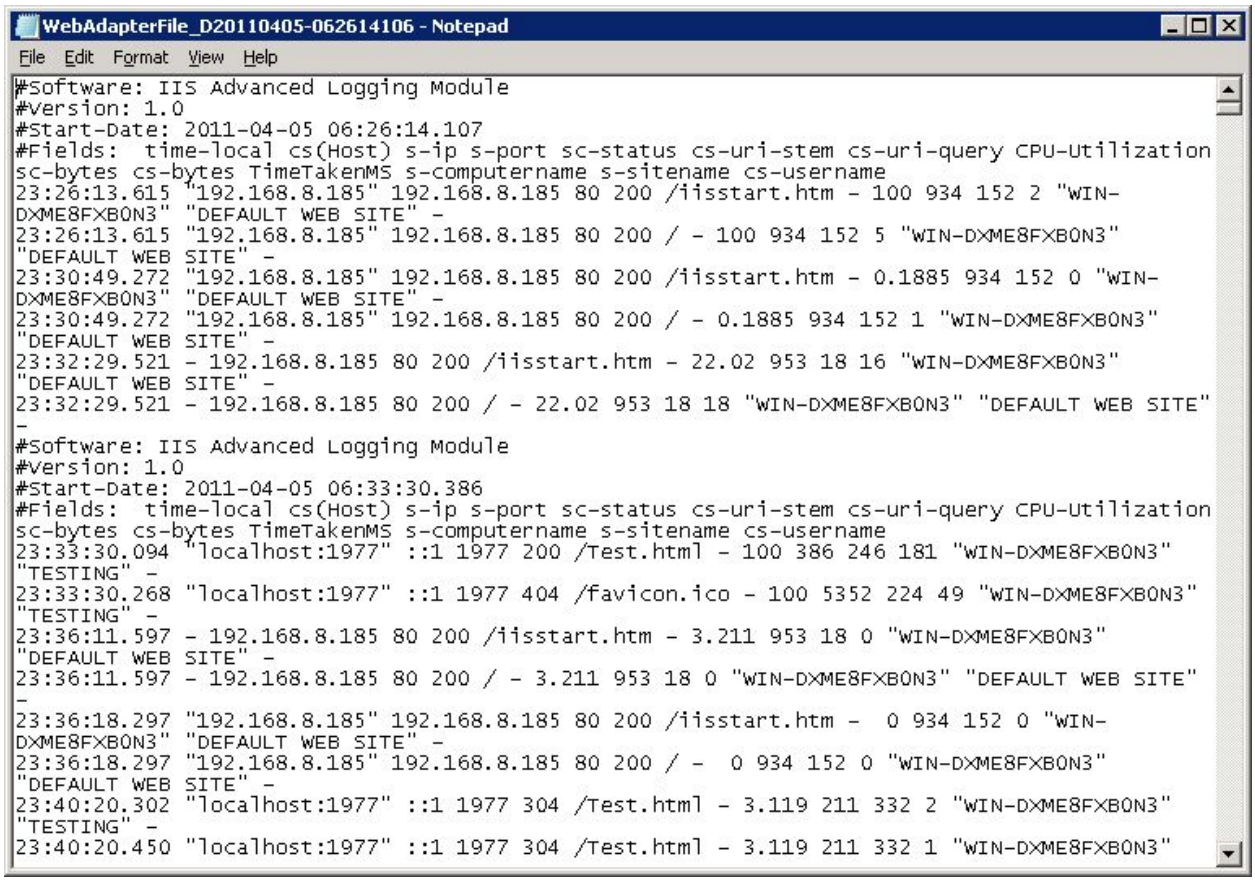


Figure 2.43: Viewing the log file

Chapter 3: Administering eG Manager to monitor IIS Web Server

After installation of eG agent, please follow the following steps to configure eG to monitor an IIS web server.

1. Login to eG user interface as an administrator.
2. If the IIS Web Server is already discovered, navigate to the **COMPONENTS – MANAGE / UNMANAGE** page following the menu Infrastructure -> Components -> Manage/Unmanage, to manage it.
3. On the other hand, if the IIS Web Server is yet to be discovered, then run the discovery procedure to get IIS Web servers discovered, or manually add the IIS Web server. To run the discovery, open the **START DISCOVERY** page using the Infrastructure -> Components -> Discovery menu sequence, and click the **Start** option under the Action node in the **DISCOVERY** tree.
4. To manually add the IIS Web Server, go to the **COMPONENTS** page through the Infrastructure -> Components -> Add/Modify menu sequence and then add the server as indicated by Figure 3.1.

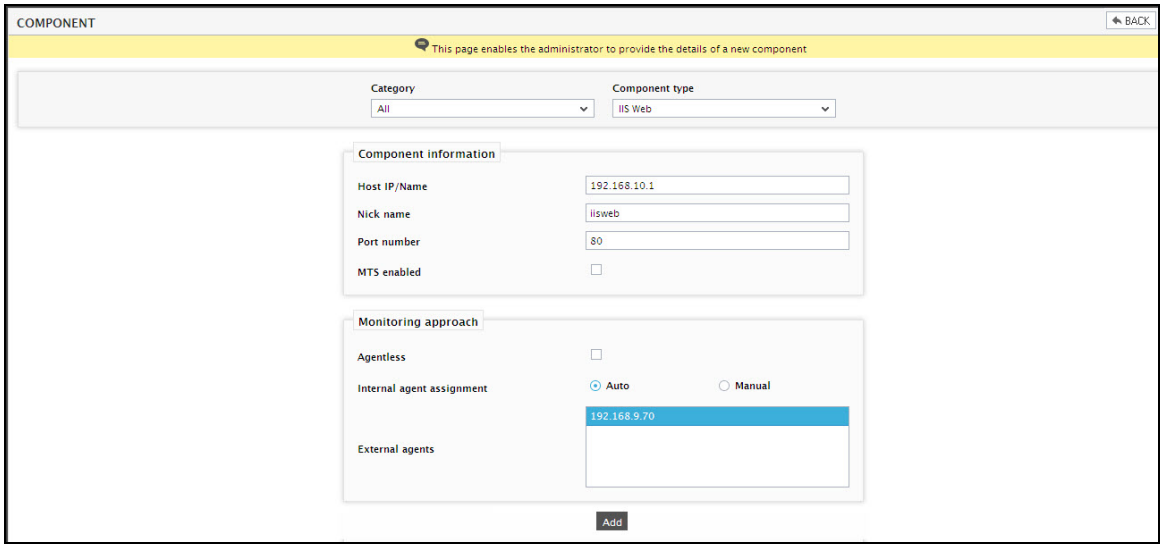


Figure 3.1: Adding an IIS web server

5. If a Microsoft Transaction server (MTS) is available on the target IIS web server, then, you can manage the MTS server along with your IIS web server by setting the **MTS enabled** flag to Yes.

This will automatically add a Microsoft Transaction server component, with the same IP-nickname as the IIS web server (see Figure 3.2)

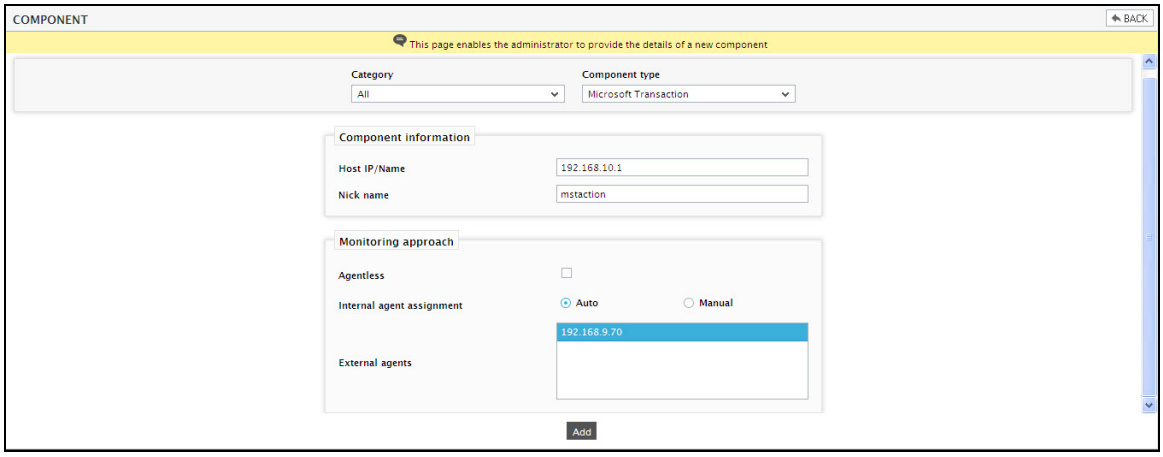


Figure 3.2: An MTS server being automatically added

6. Components added using the **COMPONENTS** page will automatically appear in the **MANAGED COMPONENTS** list of the **COMPONENTS - MANAGE/UNMANAGE** page (see Figure 3.3). Discovered servers, on the other hand, need to be managed manually using the **COMPONENTS - MANAGE/UNMANAGE** page (see Figure 3.3). For accessing this page, use the menu sequence Infrastructure -> Components -> Manage/Unmanage. The screen below shows all the IIS Web Servers discovered in a given range but not managed. Select the component-type that requires monitoring from the **Component type** list. To manage a particular component of the selected type, select the component from the **UNMANAGED COMPONENTS** list and click on the **Manage** button and finally, the **Update** button.

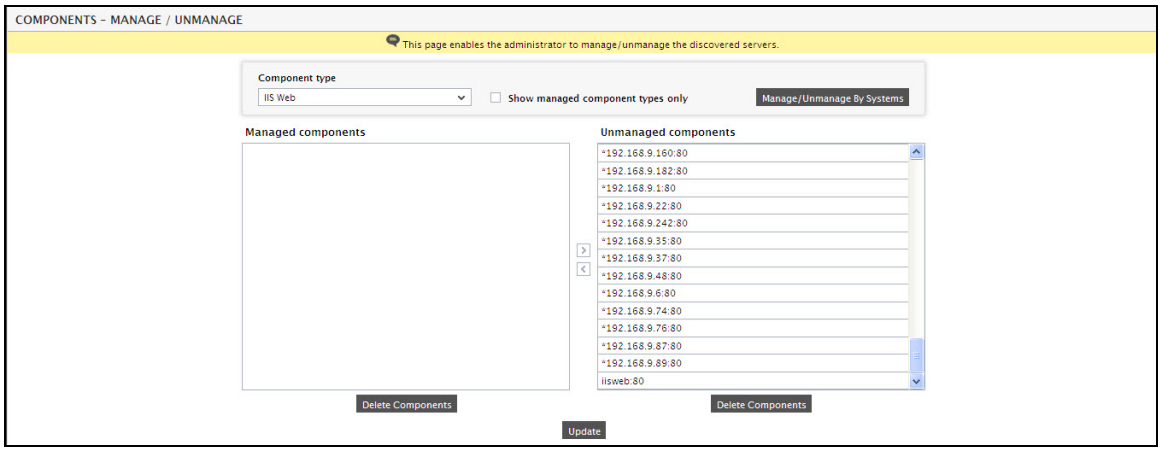


Figure 3.3: Viewing the list of unmanaged IIS web servers

7. After managing the web server, the screen would appear as shown below:

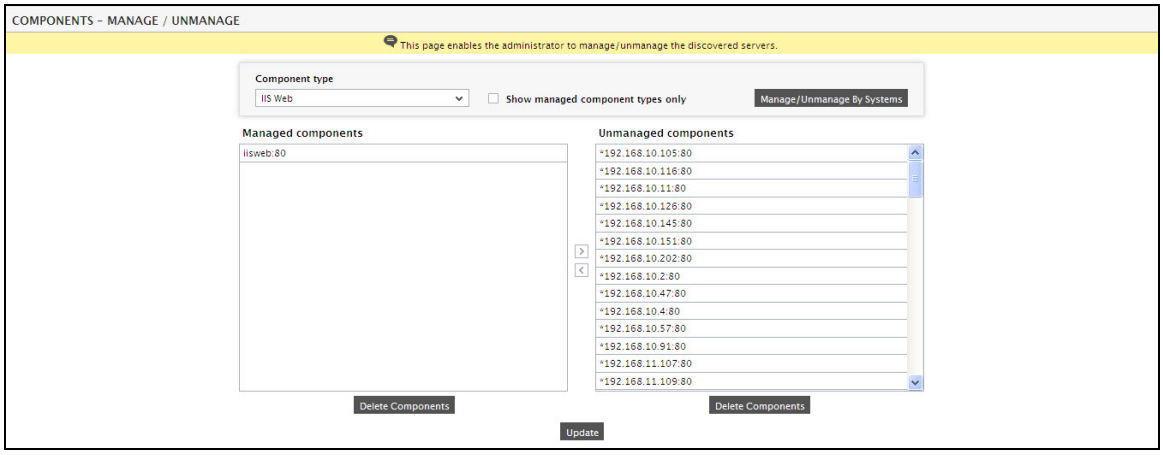


Figure 3.4: Managing an IIS web server

8. Then, proceed to configure web sites and related transactions for the IIS web server. Refer to Monitoring Apache Web Server document for a more elaborate discussion on how to configure web sites and transactions.

9. Once this is done, sign out of the eG administrative interface.

Chapter 4: Monitoring IIS Web Servers

Internet Information Services (IIS) is a powerful Web server that provides a highly reliable, manageable, and scalable Web application infrastructure for Windows servers.

eG Enterprise offers a specialized IIS web server model for monitoring an IIS web server. Figure 4.1 depicts the model used by the eG Enterprise suite to monitor an Internet Information Server.

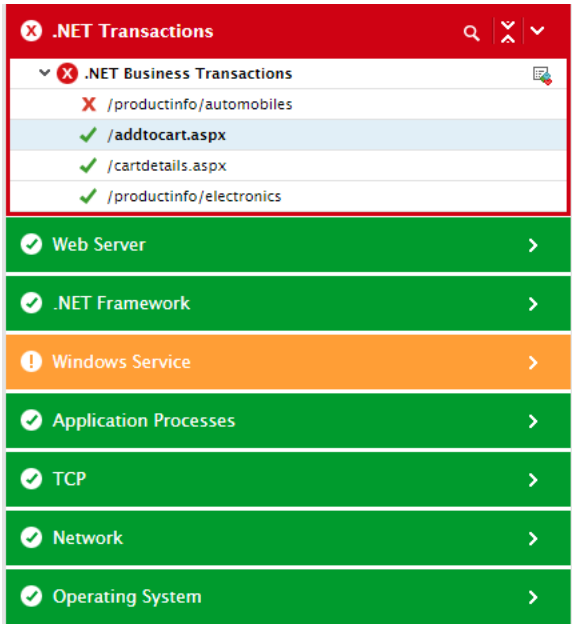


Figure 4.1: The different layers that eG Enterprise monitors for an IIS web server

The **Operating System**, **Network**, and **TCP** layers of an IIS web server model are similar to that of a Windows Generic server model. Since these tests have been dealt with in the *Monitoring Unix and Windows Servers* document, Section 3.1 focuses on the **Application Processes** layer.

Similarly, the **Web Site** and **Web Transactions** layers have also been described in detail in Chapter 2. Section 3.2 therefore, discusses the **Web Server** layer of the IIS web server.

Note:

- The eG agent is capable of monitoring IIS web servers (ver. 4, 5, 6, and 7) in an agent-based and an agentless manner; however, note that, in the agentless mode, the solution cannot monitor web transactions to web sites configured on the target IIS web server.

- To monitor an IIS web server on Windows 2008, you need to configure a **Web Server** role on the target Windows 2008 server. To know how to configure the **Web Server** role, refer to the *eG Installation Guide*.

4.1 The Application Processes Layer

The **Application Processes** layer of an IIS web server periodically checks whether the critical web server processes are running or not, and also reports the percentage of CPU and memory resources utilized by these processes. In addition, the layer also monitors the eventlogs on the IIS web server host from time to time, so that critical application and system-related errors are instantly detected.

Figure 4.2: The tests mapped to the Application Processes layer

For details on the **Processes** test and the **WindowsProcesses** test in Figure 3.2, please refer to the Monitoring Unix and Windows Servers document. The sections below discuss the **ApplicationEvents** and **SystemEvents** tests only.

4.1.1 Application Events Test

This test reports the statistical information about the application events generated by the target system.

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for the **FILTER** configured

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed

2. **HOST** - The host for which the test is to be configured
3. **PORT** – Refers to the port used by the EventLog Service. Here it is null.
4. **LOGTYPE** – Refers to the type of event logs to be monitored. The default value is *application*.
5. **POLICY BASED FILTER** - Using this page, administrators can configure the event sources, event IDs, and event descriptions to be monitored by this test. In order to enable administrators to easily and accurately provide this specification, this page provides the following options:
   - Manually specify the event sources, IDs, and descriptions in the **FILTER** text area, or,
   - Select a specification from the predefined filter policies listed in the **FILTER** box

   For explicit, manual specification of the filter conditions, select the **NO** option against the **POLICY BASED FILTER** field. This is the default selection. To choose from the list of pre-configured filter policies, or to create a new filter policy and then associate the same with the test, select the **YES** option against the **POLICY BASED FILTER** field.
6. **FILTER** - If the **POLICY BASED FILTER** flag is set to **NO**, then a **FILTER** text area will appear, wherein you will have to specify the event sources, event IDs, and event descriptions to be monitored. This specification should be of the following format: *{Displayname}:{event_sources_to_be_included}:{event_sources_to_be_excluded}:{event_IDs_to_be_included}:{event_IDs_to_be_excluded}:{event_descriptions_to_be_included}:{event_descriptions_to_be_excluded}*. For example, assume that the **FILTER** text area takes the value, *OS_events:all:Browse,Print:all:none:all:none*. Here:
   - *OS_events* is the display name that will appear as a descriptor of the test in the monitor UI;
   - *all* indicates that all the event sources need to be considered while monitoring. To monitor specific event sources, provide the source names as a comma-separated list. To ensure that none of the event sources are monitored, specify none.
   - Next, to ensure that specific event sources are excluded from monitoring, provide a comma-separated list of source names. Accordingly, in our example, *Browse* and *Print* have been excluded from monitoring. Alternatively, you can use *all* to indicate that all the event sources have to be excluded from monitoring, or *none* to denote that none of the event sources need be excluded.
   - In the same manner, you can provide a comma-separated list of event IDs that require monitoring. The *all* in our example represents that all the event IDs need to be considered while monitoring.
   - Similarly, the *none* (following *all* in our example) is indicative of the fact that none of the event IDs need to be excluded from monitoring. On the other hand, if you want to instruct

the eG Enterprise system to ignore a few event IDs during monitoring, then provide the IDs as a comma-separated list. Likewise, specifying *all* makes sure that all the event IDs are excluded from monitoring.

- The *all* which follows implies that all events, regardless of description, need to be included for monitoring. To exclude all events, use *none*. On the other hand, if you provide a comma-separated list of event descriptions, then the events with the specified descriptions will alone be monitored. Event descriptions can be of any of the following forms - *desc**, or *desc*, or **desc**,or *desc**, or *desc1*desc2*, etc. *desc* here refers to any string that forms part of the description. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters.
- In the same way, you can also provide a comma-separated list of event descriptions to be excluded from monitoring. Here again, the specification can be of any of the following forms: *desc**, or *desc*, or **desc**,or *desc**, or *desc1*desc2*, etc. *desc* here refers to any string that forms part of the description. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters. In our example however, none is specified, indicating that no event descriptions are to be excluded from monitoring. If you use *all* instead, it would mean that all event descriptions are to be excluded from monitoring.

By default, the **FILTER** parameter contains the value: *all:all:none:all:none:all:none*. Multiple filters are to be separated by semi-colons (;).

Note:

The event sources and event IDs specified here should be exactly the same as that which appears in the Event Viewer window.

On the other hand, if the **POLICY BASED FILTER** flag is set to **YES**, then a **FILTER** list box will appear, displaying the filter policies that pre-exist in the eG Enterprise system. A filter policy typically comprises of a specific set of event sources, event IDs, and event descriptions to be monitored. This specification is built into the policy in the following format:

{Policyname}:{event_sources_to_be_included}:{event_sources_to_be_excluded}:{event_IDs_to_be_included}:{event_IDs_to_be_excluded}:{event_descriptions_to_be_included}:{event_descriptions_to_be_excluded}

To monitor a specific combination of event sources, event IDs, and event descriptions, you can choose the corresponding filter policy from the **FILTER** list box. Multiple filter policies can be so selected. Alternatively, you can modify any of the existing policies to suit your needs, or create a new filter policy. To facilitate this, a **Click here** link appears just above the test configuration section, once the **YES** option is chosen against **POLICY BASED FILTER**. Clicking on the **Click**

link leads you to a page where you can modify the existing policies or create a new one (refer to page Section). The changed policy or the new policy can then be associated with the test by selecting the policy name from the **FILTER** list box in this page.

7. **USEWMI** - The eG agent can either use WMI to extract event log statistics or directly parse the event logs using event log APIs. If the **USEWMI** flag is **YES**, then WMI is used. If not, the event log APIs are used. This option is provided because on some Windows 2000 systems (especially ones with service pack 3 or lower), the use of WMI access to event logs can cause the CPU usage of the WinMgmt process to shoot up. On such systems, set the **USEWMI** parameter value to **NO**.
8. **STATELESS ALERTS** - Typically, the eG manager generates email alerts only when the state of a specific measurement changes. A state change typically occurs only when the threshold of a measure is violated a configured number of times within a specified time window. While this ensured that the eG manager raised alarms only when the problem was severe enough, in some cases, it may cause one/more problems to go unnoticed, just because they did not result in a state change. For example, take the case of the EventLog test. When this test captures an error event for the very first time, the eG manager will send out a **CRITICAL** email alert with the details of the error event to configured recipients. Now, the next time the test runs, if a different error event is captured, the eG manager will keep the state of the measure as **CRITICAL**, but will not send out the details of this error event to the user; thus, the second issue will remain hidden from the user. To make sure that administrators do not miss/overlook critical issues, the eG Enterprise monitoring solution provides the **stateless alerting** capability. To enable this capability for this test, set the **STATELESS ALERTS** flag to **Yes**. This will ensure that email alerts are generated for this test, regardless of whether or not the state of the measures reported by this test changes.
9. **EVENTS DURING RESTART** - By default, the **EVENTS DURING RESTART** flag is set to **Yes**. This ensures that whenever the agent is stopped and later started, the events that might have occurred during the period of non-availability of the agent are included in the number of events reported by the agent. Setting the flag to **No** ensures that the agent, when restarted, ignores the events that occurred during the time it was not available.
10. **DDFORINFORMATION** – eG Enterprise also provides you with options to restrict the amount of storage required for event log tests. Towards this end, the **DDFORINFORMATION** and **DDFORWARNING** flags have been made available in this page. By default, both these flags are set to **Yes**, indicating that by default, the test generates detailed diagnostic measures for information events and warning events. If you do not want the test to generate and store detailed measures for information events, set the **DDFORINFORMATION** flag to **No**.
11. **DDFORWARNING** – To ensure that the test does not generate and store detailed measures for

warning events, set the **DDFORWARNING** flag to **No**.

12. **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**.
13. 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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Application errors:	This refers to the number of application error events that were generated.	Number	A very low value (zero) indicates that the system is in a healthy state and all applications are running smoothly without any potential problems. An increasing trend or high value indicates the existence of problems like loss of functionality or data in one or more applications. Please check the Application Logs in the Event Log Viewer for more details.
Application	This refers to the	Number	A change in the value of this

Measurement	Description	Measurement Unit	Interpretation
information count:	number of application information events generated when the test was last executed.		measure may indicate infrequent but successful operations performed by one or more applications. Please check the Application Logs in the Event Log Viewer for more details.
Application warnings:	This refers to the number of warnings that were generated when the test was last executed.	Number	A high value of this measure indicates application problems that may not have an immediate impact, but may cause future problems in one or more applications. Please check the Application Logs in the Event Log Viewer for more details.

The filter policy for the ApplicationEvents test and SystemEvents test typically comprises of a specific set of event sources, event IDs, and event descriptions to be monitored. This specification is expressed by the eG Enterprise system in the following format:

{Policyname}:{event_sources_to_be_included}:{event_sources_to_be_excluded}:{event_IDs_to_be_ included}: {event_ IDs_ to_ be_ excluded}: {event_ descriptions_ to_ be_ included}: {event_ descriptions_to_be_excluded}

To add a new policy, do the following:

1. Click on the **Click here** hyperlink available just above the test configuration of the ApplicationEvents test or SystemEvents test (see Figure 4.3).

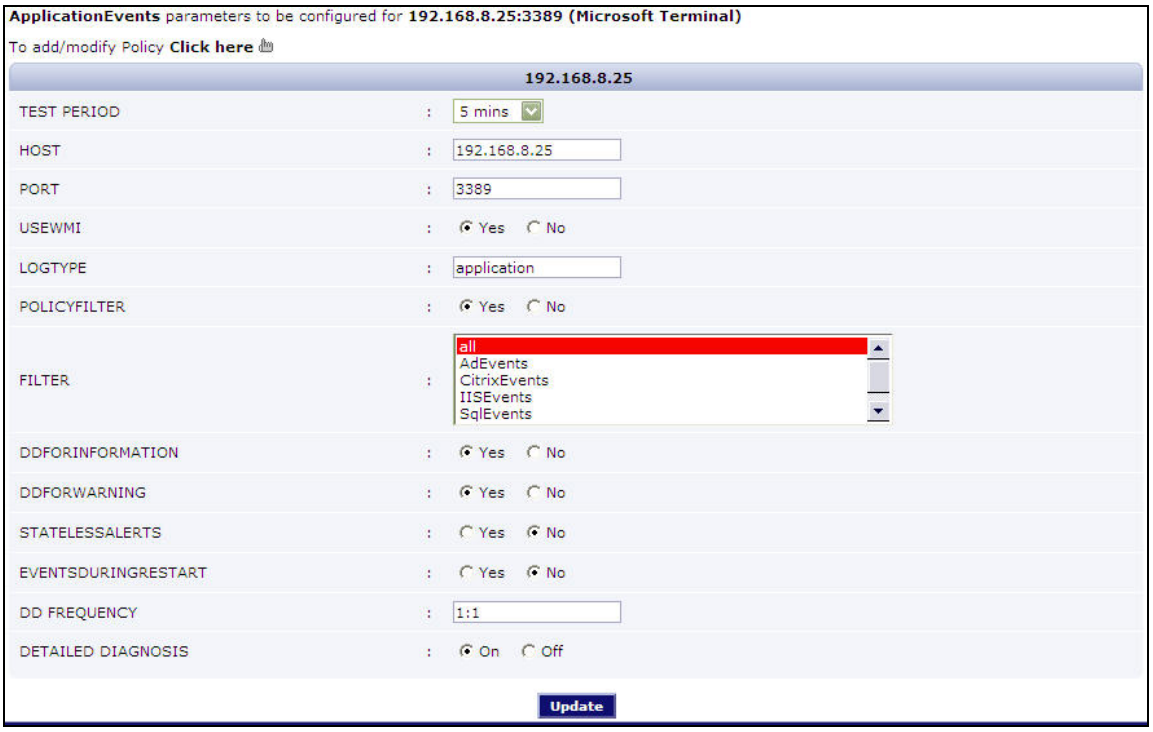


Figure 4.3: Configuring an ApplicationEvents test

2. Figure 4.4 will then appear listing the policies that pre-exist.

Figure 4.4: List of policies

3. To view the contents of a policy, click on the **View** button against the policy name. While a policy can be modified by clicking on the **Modify** button, it can be deleted using the **Delete** button. The default policy is *all*, which can only be viewed and *not modified* or *deleted*. The specification contained within this policy is: *all:none:all:none:all:none*.

4. To create a new policy, click on the **Add New Policy** button in Figure 4.4. Doing so invokes Figure 4.5, using which a new policy can be created.

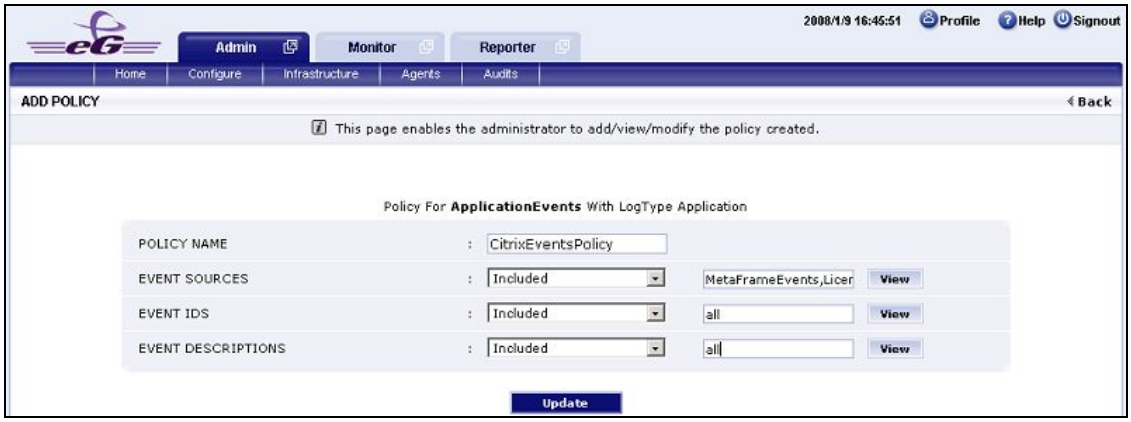


Figure 4.5: Adding a new filter policy

5. In Figure 4.5, first, provide a unique name against **POLICY NAME**.
6. To include one/more event sources for monitoring, select **Included** from the **EVENT SOURCES** drop-down list, and then specify a comma-separated list of event sources in the adjacent text box. If you require more space to specify the event sources, click on the **View** button next to the text box. This will invoke an **EVENT SOURCES INCLUDED** text area (see Figure 4.6), wherein the specification can be provided more clearly and comfortably.

Figure 4.6: Viewing the text area

7. To exclude specific event sources from monitoring, select **Excluded** from the **EVENT SOURCES** drop-down list, and then specify a comma-separated list of event sources to be excluded in the adjacent text box. If you require more space to specify the event sources, click on the **View** button next to the text box. This will invoke an **EVENT SOURCES EXCLUDED** text area, wherein the

specification can be provided more clearly and comfortably.

Note:

At any given point in time, you can choose to either **Include** or **Exclude** event sources, but you cannot do both. If you have chosen to include event sources, then the eG Enterprise system automatically assumes that no event sources need be excluded. Accordingly, the *{event_sources_to_be_excluded}* section of the filter format mentioned above, will assume the value *none*. Similarly, if you have chosen to exclude specific event sources from monitoring, then the *{event_sources_to_be_included}* section of the format above will automatically take the value *all*, indicating that all event sources except the ones explicitly excluded, will be included for monitoring.

8. In the same way, select **Included** from the **EVENT IDS** list and then, provide a comma-separated list of event IDs to be monitored. For more space, click on the **View** button next to the text box, so that an **EVENT IDS INCLUDED** text area appears.
9. If you, on the other hand, want to exclude specific event IDs from monitoring, then first select **Excluded** from the **EVENT IDS** list box, and then provide a comma-separated list of event IDs to be excluded. For more space, click on the **View** button next to the text box, so that an **EVENT IDS EXCLUDED** text area appears.

   Note:

   At any given point in time, you can choose to either **Include** or **Exclude** event IDs, but you cannot do both. If you have chosen to include event IDs, then the eG Enterprise system automatically assumes that no event IDs need be excluded. Accordingly, the *{event_IDs_to_be_excluded}* section of the filter format mentioned above, will assume the value *none*. Similarly, if you have chosen to exclude specific event IDs from monitoring, then the *{event_IDs_to_be_included}* section of the format above will automatically take the value *all*, indicating that all event IDs except the ones explicitly excluded, will be included for monitoring.

10. Likewise, select **Included** from the **EVENT DESCRIPTIONS** list and then, provide a comma-separated list of event descriptions to be monitored. For more space, click on the **View** button next to the text box, so that an **EVENT DESCRIPTIONS INCLUDED** text area appears.
11. For excluding specific event descriptions from monitoring, first select **Excluded** from the **EVENT DESCRIPTIONS** list box, and then provide a comma-separated list of event descriptions to be excluded. For more space, click on the **View** button next to the text box, so that an **EVENT DESCRIPTIONS EXCLUDED** text area appears.

    Note:

At any given point in time, you can choose to either **Include** or **Exclude** event descriptions, but you cannot do both. If you have chosen to include event descriptions, then the eG Enterprise system automatically assumes that no event descriptions need be excluded. Accordingly, the *{event_descriptions_to_be_excluded}* section of the filter format mentioned above, will assume the value *none*. Similarly, if you have chosen to exclude specific event descriptions from monitoring, then the *{event_ descriptions_ to_ be_ included}* section of the format above will automatically take the value *all*. This indicates that all event descriptions except the ones explicitly excluded, will be included for monitoring.

12. Finally, click the **Update** button.
13. The results of the configuration will then be displayed as depicted by Figure 4.7

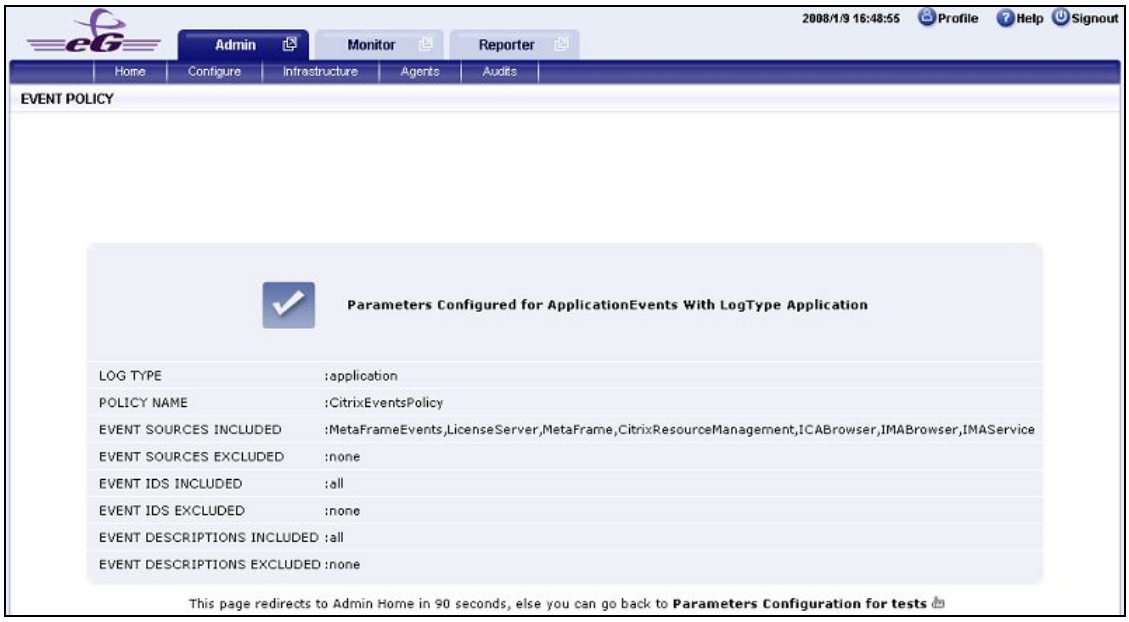


Figure 4.7: Results of the configuration

4.1.2 System Events Test

This test reports the statistical information about the system events generated by the target system.

Target of the test : An IIS web server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the **FILTER** configured.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** - The host for which the test is to be configured
3. **PORT** – Refers to the port used by the EventLog Service. Here it is null.
4. **LOGTYPE** – Refers to the type of event logs to be monitored. The default value is *application*.
5. **POLICY BASED FILTER** - Using this page, administrators can configure the event sources, event IDs, and event descriptions to be monitored by this test. In order to enable administrators to easily and accurately provide this specification, this page provides the following options:

   - Manually specify the event sources, IDs, and descriptions in the **FILTER** text area, or,
   - Select a specification from the predefined filter policies listed in the **FILTER** box

   For explicit, manual specification of the filter conditions, select the **NO** option against the **POLICY BASED FILTER** field. This is the default selection. To choose from the list of pre-configured filter policies, or to create a new filter policy and then associate the same with the test, select the **YES** option against the **POLICY BASED FILTER** field.

6. **FILTER** - If the **POLICY BASED FILTER** flag is set to **NO**, then a **FILTER** text area will appear, wherein you will have to specify the event sources, event IDs, and event descriptions to be monitored. This specification should be of the following format: *{Displayname}:{event_sources_to_be_included}:{event_sources_to_be_excluded}:{event_IDs_to_be_included}:{event_IDs_to_be_excluded}:{event_descriptions_to_be_included}:{event_descriptions_to_be_excluded}* . For example, assume that the **FILTER** text area takes the value, *OS_events:all:Browse,Print:all:none:all:none*. Here:

   - *OS_events* is the display name that will appear as a descriptor of the test in the monitor UI;
   - *all* indicates that all the event sources need to be considered while monitoring. To monitor specific event sources, provide the source names as a comma-separated list. To ensure that none of the event sources are monitored, specify *none*.
   - Next, to ensure that specific event sources are excluded from monitoring, provide a comma-separated list of source names. Accordingly, in our example, *Browse* and *Print* have been excluded from monitoring. Alternatively, you can use *all* to indicate that all the event sources have to be excluded from monitoring, or *none* to denote that none of the event sources need be excluded.
   - In the same manner, you can provide a comma-separated list of event IDs that require monitoring. The *all* in our example represents that all the event IDs need to be considered

while monitoring.

- Similarly, the *none* (following *all* in our example) is indicative of the fact that none of the event IDs need to be excluded from monitoring. On the other hand, if you want to instruct the eG Enterprise system to ignore a few event IDs during monitoring, then provide the IDs as a comma-separated list. Likewise, specifying *all* makes sure that all the event IDs are excluded from monitoring.
- The *all* which follows implies that all events, regardless of description, need to be included for monitoring. To exclude all events, use *none*. On the other hand, if you provide a comma-separated list of event descriptions, then the events with the specified descriptions will alone be monitored. Event descriptions can be of any of the following forms - *desc**, or *desc*, or **desc**,or *desc**, or *desc1*desc2*, etc. *desc* here refers to any string that forms part of the description. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters.
- In the same way, you can also provide a comma-separated list of event descriptions to be excluded from monitoring. Here again, the specification can be of any of the following forms: *desc**, or *desc*, or **desc**,or *desc**, or *desc1*desc2*, etc. *desc* here refers to any string that forms part of the description. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters. In our example however, none is specified, indicating that no event descriptions are to be excluded from monitoring. If you use *all* instead, it would mean that all event descriptions are to be excluded from monitoring.

By default, the **FILTER** parameter contains the value: *all:all:none:all:none:all:none*. Multiple filters are to be separated by semi-colons (;).

Note:

The event sources and event IDs specified here should be exactly the same as that which appears in the Event Viewer window.

On the other hand, if the **POLICY BASED FILTER** flag is set to **YES**, then a **FILTER** list box will appear, displaying the filter policies that pre-exist in the eG Enterprise system. A filter policy typically comprises of a specific set of event sources, event IDs, and event descriptions to be monitored. This specification is built into the policy in the following format:

{Policyname}:{event_sources_to_be_included}:{event_sources_to_be_excluded}:{event_IDs_to_be_included}:{event_IDs_to_be_excluded}:{event_descriptions_to_be_included}:{event_descriptions_to_be_excluded}

To monitor a specific combination of event sources, event IDs, and event descriptions, you can

FILTER list box. Multiple filter policies can be so selected. Alternatively, you can modify any of the existing policies to suit your needs, or create a new filter policy. To facilitate this, a **Click here** link appears just above the test configuration section, once the **YES** option is chosen against **POLICY BASED FILTER**. Clicking on the **Click here** link leads you to a page where you can modify the existing policies or create a new one (refer to page list box. Multiple filter policies can be so selected. Alternatively, you can modify any of the existing policies to suit your needs, or create a new filter policy. To facilitate this, a **Click here** link appears just above the test configuration section, once the **YES** option is chosen against **POLICY BASED FILTER**. Clicking on the **Click here** link leads you to a page where you can modify the existing policies or create a new one (refer to page Section). The changed policy or the new policy can then be associated with the test by selecting the policy name from the **FILTER** list box in this page.

7. **USEWMI** - The eG agent can either use WMI to extract event log statistics or directly parse the event logs using event log APIs. If the **USEWMI** flag is **YES**, then WMI is used. If not, the event log APIs are used. This option is provided because on some Windows 2000 systems (especially ones with service pack 3 or lower), the use of WMI access to event logs can cause the CPU usage of the WinMgmt process to shoot up. On such systems, set the **USEWMI** parameter value to **NO**.
8. **STATELESS ALERTS** - Typically, the eG manager generates email alerts only when the state of a specific measurement changes. A state change typically occurs only when the threshold of a measure is violated a configured number of times within a specified time window. While this ensured that the eG manager raised alarms only when the problem was severe enough, in some cases, it may cause one/more problems to go unnoticed, just because they did not result in a state change. For example, take the case of the EventLog test. When this test captures an error event for the very first time, the eG manager will send out a **CRITICAL** email alert with the details of the error event to configured recipients. Now, the next time the test runs, if a different error event is captured, the eG manager will keep the state of the measure as **CRITICAL**, but will not send out the details of this error event to the user; thus, the second issue will remain hidden from the user. To make sure that administrators do not miss/overlook critical issues, the eG Enterprise monitoring solution provides the **stateless alerting** capability. To enable this capability for this test, set the **STATELESS ALERTS** flag to **Yes**. This will ensure that email alerts are generated for this test, regardless of whether or not the state of the measures reported by this test changes.
9. **EVENTS DURING RESTART** - By default, the **EVENTS DURING RESTART** flag is set to **Yes**. This ensures that whenever the agent is stopped and later started, the events that might have occurred during the period of non-availability of the agent are included in the number of events reported by the agent. Setting the flag to **No** ensures that the agent, when restarted, ignores the events that occurred during the time it was not available.

10. **DDFORINFORMATION** – eG Enterprise also provides you with options to restrict the amount of storage required for event log tests. Towards this end, the **DDFORINFORMATION** and **DDFORWARNING** flags have been made available in this page. By default, both these flags are set to **Yes**, indicating that by default, the test generates detailed diagnostic measures for information events and warning events. If you do not want the test to generate and store detailed measures for information events, set the **DDFORINFORMATION** flag to **No**.
11. **DDFORWARNING** – To ensure that the test does not generate and store detailed measures for warning events, set the **DDFORWARNING** flag to **No**.
12. **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**.
13. 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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
System errors:	This refers to the number of system error events generated during the last execution of the test.	Number	A very low value (zero) indicates that the system is in healthy state and all Windows services and low level drivers are running without any potential problems. An increasing trend or a high value indicates the existence of problems

Measurement	Description	Measurement Unit	Interpretation
			such as loss of functionality or data in one or more Windows services and low level drivers. Please check the Application Logs in the Event Log Viewer for more details.
System information messages:	This refers to the number of service-related and driver-related information events that were generated during the test's last execution.	Number	A change in value of this measure may indicate infrequent but successful operations performed by one or more applications. Please check the Application Logs in the Event Log Viewer for more details.
System warnings:	This refers to the number of service-related and driver-related warnings generated in the during the test's last execution.	Number	A high value of this measure indicates problems that may not have an immediate impact, but may cause future problems in one or more Windows servers and low level drivers. Please check the Application Logs in the Event Log Viewer for more details.

Note:

The **STATELESS ALERTING** capability is currently available for the following tests alone, by default:

- EventLog test
- ApplicationEventLog test
- SystemEventLog test
- ApplicationEvents test
- SystemEvents test
- SecurityLog test

If need be, you can enable the **stateless alerting** capability for other tests. To achieve this, follow the steps given below:

- Login to the eG manager host.
- Edit the **eg_specs.ini** file in the **<EG_INSTALL_DIR>\manager\config** directory.
- Locate the test for which the **Stateless Alarms** flag has to be enabled.
- Insert the entry, **-statelessAlerts yes**, into the test specification as depicted below:

```
EventLogTest::$hostName:$portNo=$hostName, -auto, -host $hostName -port $portNo -eventhost $hostIp -eventsrc all -excludedSrc none -useWmi yes -statelessAlerts yes -ddFreq 1:1 -rptName $hostName, 300
```

- Finally, save the file.
- If need be, you can change the status of the **statelessAlerts** flag by reconfiguring the test in the eG administrative interface.

Once the **stateless alerting capability** is enabled for a test (as discussed above), you will find that everytime the test reports a problem, the eG manager does the following:

- Closes the alarm that pre-exists for that problem;
- Sends out a normal alert indicating the closure of the old problem;
- Opens a new alarm and assigns a new alarm ID to it;
- Sends out a fresh email alert to the configured users, intimating them of the new issue.

In a redundant manager setup, the secondary manager automatically downloads the updated **eg_specs.ini** file from the primary manager, and determines whether the stateless alerting capability has been enabled for any of the tests reporting metrics to it. If so, everytime a threshold violation is detected by such a test, the secondary manager will perform the tasks discussed above for the problem reported by that test. Similarly, the primary manager will check whether the stateless alert flag has been switched on for any of the tests reporting to it, and if so, will automatically perform the above-mentioned tasks whenever those tests report a deviation from the norm.

Note:

- Since alerts will be closed after every measurement period, alarm escalation will no longer be relevant for tests that have **statelessAlerts** set to **yes**.
- For tests with **statelessAlerts** set to **yes**, **statelessAlerts** will apply for all measurements of that test (i.e., it will not be possible to only have one of the measurements with stateless alerts and others without).

- If **statelessAlerts** is set to **yes** for a test, an alarm will be opened during one measurement period (if a threshold violation happens) and will be closed prior to the next measurement period. This way, if a threshold violation happens in successive measurement periods, there will be one alarm per measurement period. This will reflect in all the corresponding places in the eG Enterprise system. For example, multiple alerts in successive measurement periods will result in multiple trouble tickets being opened (one for each measurement period). Likewise, the alarm history will also show alarms being opened during a measurement period and closed during the next measurement period.

4.2 The .NET Framework Layer

The tests mapped to the .NET Framework layer report on the health of the CLR (Common Language Runtime) of the ASP .NET framework that the IIS web server supports, and that of the AppDomains in which the .NET applications run.

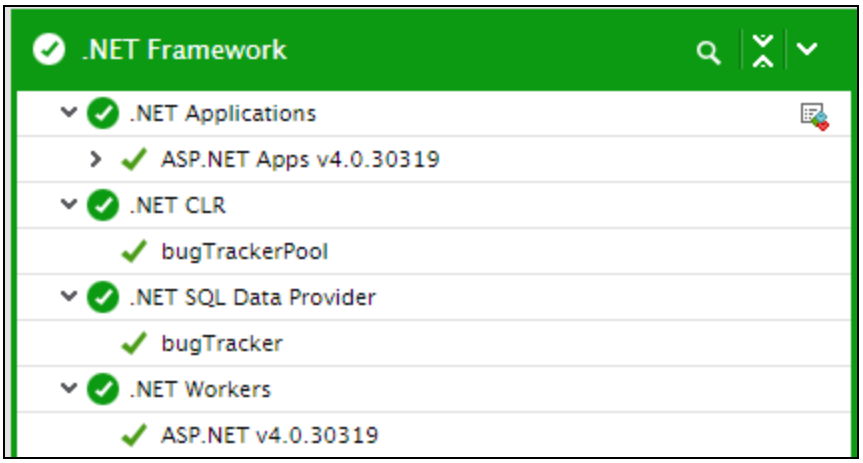


Figure 4.8: The tests mapped with the .NET Framework layer

4.2.1 ASP.Net SQL Data Provider Test

A data provider in the .NET Framework serves as a bridge between an application and a data source. A .NET Framework data provider enables you to return query results from a data source, execute commands at a data source, and propagate changes in a DataSet to a data source.

The .Net Data Provider for SQL Server allows you to connect to Microsoft SQL Server 7.0, 2000, and 2005 databases, and perform the above-mentioned operations. This test reports many useful metrics that shed light on the health of the interactions between the ASP .Net sever and the SQL server.

Target of the test : The ASP .Net server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for the ASP .Net server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Hard connects:	Indicates the number of actual connections per second that are being made to a database server.	Connects/Sec	
Hard disconnects:	Indicates the number of actual disconnects per second that are being made to a database server.	Disconnects/Sec	
Active connection pool groups:	Indicates the number of currently active connection pool groups.	Number	The value of this measure is controlled by the number of unique connection strings that are found in the AppDomain.

Measurement	Description	Measurement Unit	Interpretation
Active connection pools:	Indicates the number of connection pools that are currently active.	Number	When a connection is first opened, a connection pool is created based on matching criteria that associates the pool with the connection string in the connection. Each connection pool is associated with a distinct connection string. If the connection string is not an exact match to an existing pool when a new connection is opened, a new pool is created. Connections are pooled per process, per application domain, per connection string, and, when integrated security is used, per Windows identity. When using Windows Authentication (integrated security), both the Active connection pool groups and Active connection pools measures are significant. The reason is that connection pool groups map to unique connection strings. When integrated security is used, connection pools map to connection strings and additionally create separate pools for individual Windows identities. For example, if Fred and Julie, each within the same AppDomain, both use the connection string "Data Source=MySqlServer;Integrated Security=true", a connection pool group is created for the connection string, and two additional pools are created, one for Fred and one for Julie. If John and Martha use a connection string with an identical SQL Server login, "Data Source=MySqlServer;User Id=lowPrivUser;Password=Strong?Password", then only a single pool is created for the lowPrivUser identity.
Active connections:	Indicates the number of connections	Number	

Measurement	Description	Measurement Unit	Interpretation
	that are currently in use.		
Free connections:	Indicates the count of unused connections.	Number	Ideally, the value of this measure. A very low value indicates excessive connection usage.
Inactive connection pools:	Indicates the number of connection pools that have had no recent activity and are waiting to be disposed.	Number	
Inactive connection pool groups:	Indicates the number of inactive connection pool groups that were waiting to be deactivated i.e., to be pruned.	Number	
Non- pooled connections:	Indicates the number of active connections that are not using any of the	Number	

Measurement	Description	Measurement Unit	Interpretation
	connection pools.		
Pooled connections:	Indicates the number of connections that are managed by the connection pooler.	Number	
Reclaimed connections:	Indicates the number of connections that have been reclaimed through garbage collection where Close or Dispose was not called by the application.	Number	Not explicitly closing or disposing connections hurts performance.
Waiting connections:	Indicates the number of connections that are currently awaiting completion of an action and are therefore unavailable for use by any	Number	

Measurement	Description	Measurement Unit	Interpretation
	other application.		
Soft connects:	Indicates the rate at which connections are pulled from the connection pool.	Connects/Sec	
Soft disconnects:	Indicates the rate at which connections are returned to the connection pool.	Disconnects/Sec	

4.2.2 ASP .Net Oracle Data Provider Test

A data provider in the .NET Framework serves as a bridge between an application and a data source. A .NET Framework data provider enables you to return query results from a data source, execute commands at a data source, and propagate changes in a DataSet to a data source.

The Oracle Data Provider for .NET (ODP.NET) features optimized data access to the Oracle database from a .NET environment. ODP.NET allows developers to take advantage of advanced Oracle database functionality, including Real Application Clusters, XML DB, and advanced security. The data provider can be used from any .NET language, including C# and Visual Basic .NET.

This test reports many useful metrics that shed light on the health of the interactions between the ASP .Net sever and the Oracle database server.

Target of the test : The ASP .Net server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for the ASP .Net server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Hard connects:	Indicates the number of actual connections per second that are being made to a database server.	Connects/Sec	
Hard disconnects:	Indicates the number of actual disconnects per second that are being made to a database server.	Disconnects/Sec	
Active connection pool groups:	Indicates the number of currently active connection pool groups.	Number	The value of this measure is controlled by the number of unique connection strings that are found in the AppDomain.
Active connection pools:	Indicates the number of currently	Number	When a connection is first opened, a connection pool is created based on matching criteria that

Measurement	Description	Measurement Unit	Interpretation
	active connection pools.		associates the pool with the connection string in the connection. Each connection pool is associated with a distinct connection string. If the connection string is not an exact match to an existing pool when a new connection is opened, a new pool is created. Connections are pooled per process, per application domain, per connection string, and, when integrated security is used, per Windows identity. When using Windows Authentication (integrated security), both the Active connection pool groups and Active connection pools measures are significant. The reason is that connection pool groups map to unique connection strings. When integrated security is used, connection pools map to connection strings and additionally create separate pools for individual Windows identities. For example, if Fred and Julie, each within the same AppDomain, both use the connection string "Data Source=MySqlServer;Integrated Security=true", a connection pool group is created for the connection string, and two additional pools are created, one for Fred and one for Julie. If John and Martha use a connection string with an identical SQL Server login, "Data Source=MySqlServer;User Id=lowPrivUser;Password=Strong?Password", then only a single pool is created for the lowPrivUser identity.
Active connections:	Indicates the number of connections that are currently in	Number	

Measurement	Description	Measurement Unit	Interpretation
	use.		
Free connections:	Indicates the count of unused connections.	Number	Ideally, the value of this measure. A very low value indicates excessive connection usage.
Inactive connection pools:	Indicates the number of connection pools that have had no recent activity and are waiting to be disposed.	Number	
Inactive connection pool groups:	Indicates the number of inactive connection pool groups that were waiting to be deactivated i.e., to be pruned.	Number	
Non- pooled connections:	Indicates the number of active connections that are not using any of the connection pools.	Number	

Measurement	Description	Measurement Unit	Interpretation
Pooled connections:	Indicates the number of connections that are managed by the connection pooler.	Number	
Reclaimed connections:	Indicates the number of connections that have been reclaimed through garbage collection where Close or Dispose was not called by the application.	Number	Not explicitly closing or disposing connections hurts performance.
Waiting connections:	Indicates the number of connections that are currently awaiting completion of an action and are therefore unavailable for use by any other application.	Number	

Measurement	Description	Measurement Unit	Interpretation
Soft connects:	Indicates the rate at which connections are pulled from the connection pool.	Connects/Sec	
Soft disconnects:	Indicates the rate at which connections are returned to the connection pool.	Disconnects/Sec	

4.2.3 ASP SQL Clients Test

This test reports metrics pertaining to client connections to the ASP .Net server.

Target of the test : The ASP .Net server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for the ASP .Net server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Connection pool	Indicates the number of	Number	If the connection pool maxes out

Measurement	Description	Measurement Unit	Interpretation
size:	connection pools that have been created.		while new connection requests are still coming in, you willl see connection requests refused, apparently at random. The cure in this case is simply to specify a higher value for the Max Pool Size property.
Number of connections:	Indicates the number of connections currently in the pool.	Number	
Pooled connections:	Indicates the number of connections that have been pooled.	Number	
Pooled connections peak:	Indicates the highest number of connections that have been used.	Number	If the value of this measure is at the Max Pool Size value, and the value of the Failed connects measure increases while the application is running, you might have to consider increasing the size of the connection pool.
Failed connects:	Indicates the number of connection attempts that have failed.	Number	If the connection pool maxes out while new connection requests are still coming in, you willl see connection requests refused, apparently at random. The cure in this case is simply to specify a higher value for the Max Pool Size property.

4.2.4 ASP .Net Workers Test

This test reports statistics pertaining to the performance of the worker process of the ASP .Net server.

Target of the test : The ASP .Net server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for the ASP .Net server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Application restarts:	The number of application restarts.	Number	In a perfect world, the application domain will and should survive for the life of the process. Even if a single restart occurs, it is a cause for concern because proactive and reactive restarts cause automatic recycling of the worker process. Moreover, restarts warrant recreation of the application domain and recompilation of the pages, both of which consume a lot of time. To investigate the reasons for a restart, check the values set in the processModel configuration.
Applications running:	The number of applications currently running.	Number	
Requests current:	The number of requests currently handled by the ASP.NET ISAPI. This includes those that are queued , executing, or waiting to be written to the client.	Number	

Measurement	Description	Measurement Unit	Interpretation
Request execution time:	The number of seconds taken to execute the last request.	Number	In version 1.0 of the framework, the execution time begins when the worker process receives the request, and stop when the ASP.NET ISAPI sends HSE_ REQ_DONE_WITH_SESSION to IIS. In version 1.1 of the framework, execution begins when the HttpContext for the request is created, and stop before the response is sent to IIS. The value of this measure should be stable. Any sudden change from the previous recorded values should be notified.
Requests queued:	The number of requests currently queued.	Number	When running on IIS 5.0, there is a queue between inetinfo and aspnet_wp, and there is one queue for each virtual directory. When running on IIS 6.0, there is a queue where requests are posted to the managed ThreadPool from native code, and a queue for each virtual directory. This counter includes requests in all queues. The queue between inetinfo and aspnet_wp is a named pipe through which the request is sent from one process to the other. The number of requests in this queue increases if there is a shortage of available I/O threads in the aspnet_wp process. On IIS 6.0 it increases when there are incoming requests and a shortage of worker threads.
Requests rejected:	The number of rejected	Number	Requests are rejected when one of

Measurement	Description	Measurement Unit	Interpretation
	requests		the queue limits is exceeded. An excessive value of this measure hence indicates that the worker process is unable to process the requests due to overwhelming load or low memory in the processor.
Requests wait time:	The number of seconds that the most recent request spent waiting in the queue, or named pipe that exists between inetinfo and aspnet_wp. This does not include any time spent waiting in the application queues.	Secs	
Worker processes running:	The current number of aspnet_ wp worker processes	Number	Every application executing on the .NET server corresponds to a worker process. Sometimes, during active or proactive recycling, a new worker process and the worker process that is being replaced may coexist. Under such circumstances, a single application might have multiple worker processes executing for it. Therefore, if the value of this measure is not the same as that of Applications running, then it calls for closer examination of the reasons behind the occurence.
Worker process restarts:	The number of aspnet_ wp process restarts in the machine	Number	Process restarts are expensive and undesirable. The values of this metric are dependent upon the process model configuration settings, as well as unforeseen

Measurement	Description	Measurement Unit	Interpretation
			access violations, memory leaks, and deadlocks.

4.2.5 ASP .Net Applications Test

NET introduces the concept of an application domain, or AppDomain. Like a process, the AppDomain is both a container and a boundary. The .NET runtime uses an AppDomain as a container for code and data, just like the operating system uses a process as a container for code and data. As the operating system uses a process to isolate misbehaving code, the .NET runtime uses an AppDomain to isolate code inside of a secure boundary. An AppDomain belongs to only a single process, but a single process can hold multiple AppDomains.

An AppDomain is relatively cheap to create (compared to a process), and has relatively less overhead to maintain than a process. For these reasons, an AppDomain is a great solution for the ISP who is hosting hundreds of applications. Each application can exist inside an isolated AppDomain, and many of these AppDomains can exist inside of a single process.

From this, it can be inferred that to understand how well an application is performing, it is necessary to monitor the AppDomain in which that application exists. This involves the following:

- Tracking the requests to the AppDomain, measuring the time taken by the AppDomain to service the requests, and keeping an eye on the pending requests count on the AppDomain;
- Understanding how the AppDomain uses its cache and isolating irregularities in cache usage or sizing;
- Monitoring the session load on the AppDomain and measuring the efficiency of the AppDomain in handling this load;
- Capturing errors encountered by the AppDomain.

The **ASP .Net Applications** test performs all of the above for every AppDomain on the monitored ASP .Net server. This way, the test proactively alerts administrators to potential request processing bottlenecks in any AppDomain, promptly identifies the AppDomain that is utilizing its cache ineffectively, instantly captures errors (if any) in the AppDomain and brings them to the notice of administrators, and rapidly notifies administrators if any AppDomain is overloaded with sessions or is abandoning/timing out sessions at a brisk pace.

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every ASP .Net application/application domain on a monitored ASP .Net server

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Cache total entries	The current number of entries in the cache (both User and Internal) of this AppDomain/application.	Number	
Cache hit ratio	The current hit-to-miss ratio of all cache requests (both user and internal) to the cache of this AppDomain/application.	Percent	Physical I/O takes a significant amount of time, and also increases the CPU resources required. The server configuration should therefore ensure that the required information is available on the memory. A low value of this measure indicates that physical I/O is greater.
Cache turnover rate	The number of additions and removals per second (both user and internal) to the cache of this AppDomain/application.	Cached/Sec	A high turnover rate indicates that items are being quickly added and removed, which can be expensive.
Cache API entries	The number of entries currently in the user cache of this AppDomain/application.	Number	
Cache user hit ratio	Total hit-to-miss ratio of requests to the user cache of this AppDomain/application .	Percent	A high value of this measure is indicative of the good health of the server.
Cache user turnover	The number of additions and	Cached/Sec	A high turnover rate indicates that

Measurement	Description	Measurement Unit	Interpretation
rate	removals per second to the user cache of this AppDomain/application.		items are being quickly added and removed, which can be expensive.
Output cache entries	The number of entries currently in the Output Cache of this AppDomain/application.	Number	
Output cache hit ratio	The total hit-to-miss ratio of requests to the Output Cache of this AppDomain/application.	Percent	A high value of this measure is a sign of good health.
Output cache turnover rate	The number of additions and removals per second to the Output Cache of this AppDomain/application.	Cached/Sec	Output caching allows you to store dynamic page and user control responses on any HTTP 1.1 cache-capable device in the output stream, from the originating server to the requesting browser. On subsequent requests, the page or user control code is not executed; the cached output is used to satisfy the request Sudden increases in the value of this measure are indicative of backend latency.
Compilation total	The total number of compilations that have taken place during the lifetime of the current Web server process. This occurs when a file with a .aspx, .asmx, asax,.ascx, or .ashx extension or code-behind source files are dynamically compiled on the server.	Number	
Processing errors	The rate at which configuration and parsing errors occur on this AppDomain/application.	Errors/Sec	A consistent increase in the value of this measure could prove to be fatal for the application domain.

Measurement	Description	Measurement Unit	Interpretation
Compilation errors	The rate at which compilation errors occur on this AppDomain/application.	Errors/Sec	The response is cached, and this counter increments only once until recompilation is forced by a file change.
Runtime errors	The rate at which run-time errors occur on this AppDomain/application.	Errors/Sec	
Unhandled runtime errors	The rate of unhandled runtime exceptions on this AppDomain/application.	Errors/Sec	A consistent increase in the value of this measure could prove to be fatal for the application domain. This measure however, does not include the following: • Errors cleared by an event handler (for example, by Page_Error or Application_Error) • Errors handled by a redirect page • Errors that occur within a try/catch block
Requests executing	The number of requests currently executing on this AppDomain/application.	Number	This measure is incremented when the HttpRuntime begins to process the request and is decremented after the HttpRuntime finishes the request.
Requests app queue	The number of requests currently in the application request queue of this AppDomain/application.	Number	A steady increase in the value of this measure could indicate a request processing bottleneck in the AppDomain.
Requests not found	The number of requests that did not find the required resource on this AppDomain/application.	Number	Ideally, the value of this measure should be 0.
Requests not authorized	The number of requests to this AppDomain/application that failed due to unauthorized access.	Number	Values greater than 0 indicate that proper authorization has not been provided, or invalid authors are trying to access a particular resource.

Measurement	Description	Measurement Unit	Interpretation
Requests timed out	The number of requests to this AppDomain/application that timed out.	Number	Ideally, the value of this measure should be 0.
Requests succeeded	The rate at which requests to this AppDomain/application succeeded.	Requests/Sec	
Request rate	Indicates the number of requests executed per second by this AppDomain/application.	Number	This represents the current throughput of the application.
Pipeline instances	Indicates the number of active pipeline instances for this AppDomain/application.	Number	Since only one execution thread can run within a pipeline instance, this number gives the maximum number of concurrent requests that are being processed for a given application. Ideally, the value of this measure should be low.
Number of errors	Indicates the total sum of all errors that occurred during the execution of HTTP requests by this AppDomain/application.	Number	This measure should be kept at 0 or a very low value.
SQL connections	Indicates the number of connections to the SQL Server used by session state.	Number	An unusually high value may indicate a sudden increase in sessions to the SQL Server.
State server connections	Indicates the number of connections to the StateServer used by session state.	Number	An unusually high value may indicate a sudden increase in sessions to the StateServer.
Abandoned ASPNet application sessions	Indicates the number of sessions on this AppDomain/application that have been explicitly abandoned during the last measurement period.	Number	Ideally, the value of this measure should be 0.

Measurement	Description	Measurement Unit	Interpretation
Active ASPNet application sessions	Indicates the currently active sessions on this AppDomain/application.	Number	This is a good indicator of the current session load on the AppDomain/application.
Timedout ASPNet application sessions	Indicates the number of sessions on this AppDomain/application that timed out during the last measurement period.	Number	Ideally, the value of this measure should be 0.
ASPNet application sessions	Indicates the total number of sessions on this AppDomain/application during the last measurement period.	Number	
Request execution time	The number of seconds this AppDomain/application took to execute the last request.	Number	In version 1.0 of the framework, the execution time begins when the worker process receives the request, and stop when the *ASP.NET ISAPI* sends *HSE_REQ_DONE_WITH_SESSION* to IIS. In version 1.1 of the framework, execution begins when the HttpContext for the request is created, and stop before the response is sent to IIS. The value of this measure should be stable. Any sudden change from the previous recorded values should be notified.
Request execution time	Indicates the time for which the most recent request i.e., the last request was waiting in the queue.	Seconds	
Error rate	Indicates the rate at which errors occurred while the request was being processed.	Errors/sec	

4.2.6 ASP .NET CLR Test

The Common Language Runtime (CLR), the virtual machine component of Microsoft's .NET framework, manages the execution of .NET programs. A process known as just-in-time (JIT) compilation converts compiled code into machine instructions which the computer's CPU then executes. The CLR provides additional services including memory management, type safety, exception handling, garbage collection, security and thread management. All programs written for the .NET framework, regardless of programming language, are executed by the CLR.

This is why, if an application running on the .NET framework slows down, the first place that an administrator should look for hints is the CLR. Some of the common causes of poor program/application are excessive CLR heap usage, ineffective garbage collection by CLR, class loading failures, thread contentions on the CLR, and expensive run time checks by CLR. To be able to capture such anomalies on-the-fly, it is imperative that administrators closely monitor the performance of the CLR. The **ASP .NET CLR** test helps administrators do just that! The test auto-discovers the worker processes running in the ASP .NET server, and for every process, reports the following:

- The count of CLR exceptions that were thrown by that worker process;
- The heap memory usage of the worker process; in the event of excessive usage, you can also take a look at the garbage collection-related statistics reported by this test for that worker process to determine whether/not ineffective GC is causing heap memory usage to soar.
- The count of classes loaded and class loading failures;
- The rate of thread contentions in the worker process;
- The count of waiting threads;
- The time spent in performing runtime Code Access Security (CAS) checks;
- The count of runtime checks performed.

These metrics provide useful pointers to the source of performance degradations experienced by applications.

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every worker process of every application pool on the monitored ASP .Net server

First-level descriptor: Application pool

Second-level descriptor: Worker process

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.
4. **WEBSITE NAME** - By default, this parameter is set to *none*. This implies that the test monitors all web sites, by default. If you want the test to monitor a specific web site alone, then specify that web site name here.

   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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Clr exceptions:	The total number of managed exceptions thrown per second.	Exceptions/Sec	Exceptions are very costly and can severely degrade your application performance. A high value of this measure is therefore an indicator of potential performance issues.
Heap mem usage:	The number of bytes committed by managed objects. This is the sum of the large object heap and the generation 0, 1, and 2 heaps.	MB	
Gen 0 collections:	The rate at which the generation 0 objects (youngest; most recently allocated) are garbage collected (Gen 0 GC) since the	Collections/Sec	

Measurement	Description	Measurement Unit	Interpretation
	start of the application.		
Gen 1 collections:	The rate at which the generation 1 objects have been garbage collected since the start of the application. Objects that survive are promoted to generation 2.	Collections/Sec	
Gen 2 collections:	The number of seconds taken to execute the last request.	Number	The number of times generation 2 objects have been garbage collected since the start of the application. Generation 2 is the highest, thus objects that survive collection remain in generation 2. Gen 2 collections can be very expensive, especially if the size of the Gen 2 heap is huge.
Time in gc:	% Time in GC is the percentage of elapsed time that was spent in performing a garbage collection (GC) since the last GC cycle.	Percent	This measure is usually an indicator of the work done by the Garbage Collector on behalf of the application to collect and conserve memory. This measure is updated only at the end of every GC and the measure reflects the last observed value; its not an average.
Classes loaded:	Indicates the cumulative number of classes loaded in all assemblies since the start of this worker process.	Number	A class is essentially the blueprint for an object. It contains the definition for how a particular object will be instantiated at runtime, such as the properties and methods that will be exposed publicly by the object and any internal storage structures.

Measurement	Description	Measurement Unit	Interpretation
Current classes loaded:	Indicates the current number of classes loaded in all Assemblies.	Number	An unusually high value may indicate a sudden increase in classes which loaded on to this .NET application.
Rate of assemblies:	The rate at which Assemblies were loaded.	Assembles/Sec	Also known as Managed DLLs, assemblies are the fundamental unit of deployment for the .NET platform. The .NET Framework itself is made up of a number of assemblies, including mscorlib.dll, among others. The assembly boundary is also where versioning and security are applied. An assembly contains Intermediate Language generated by a specific language compiler, an assembly manifest (containing information about the assembly), type metadata, and resources. If the Assembly is loaded as domain-neutral from multiple AppDomains then this counter is incremented once only. Assemblies can be loaded as domain-neutral when their code can be shared by all AppDomains or they can be loaded as domain-specific when their code is private to the AppDomain. This counter is not an average over time; it displays the difference between the values observed in the last two samples divided by the duration of the sample interval.
Rate of classes	This rate at which the	Classes/Sec	This counter is not an average over

Measurement	Description	Measurement Unit	Interpretation
loaded:	classes loaded in all Assemblies.		time; it displays the difference between the values observed in the last two samples divided by the duration of the sample interval.
Rate of load failures:	The rate of load failures on this worker process.	Failures/Sec	This counter is not an average over time; it displays the difference between the values observed in the last two samples divided by the duration of the sample interval. These load failures could be due to many reasons like inadequate security or illegal format.
Current appdomains:	The number of AppDomains currently loaded in this worker process.	Number	AppDomains (application domains) provide a secure and versatile unit of processing that the CLR can use to provide isolation between applications running in the same process.
Current assemblies:	The number of assemblies currently loaded on this worker process.	Number	If the Assembly is loaded as domain-neutral from multiple AppDomains then this counter is incremented once only. Assemblies can be loaded as domain-neutral when their code can be shared by all AppDomains or they can be loaded as domain-specific when their code is private to the AppDomain.
Loader heap size:	The size of the memory committed by the class loader.	MB	Committed memory is the physical memory for which space has been reserved on the disk paging file.
Load failures:	The number of classes that have failed to load during the last measurement period.	Number	These load failures could be due to many reasons like inadequate security or illegal format.

Measurement	Description	Measurement Unit	Interpretation
Appdomains loaded:	The number of AppDomains loaded during the last measurement period.	Number	
Num assemblies:	The number of assemblies loaded during the last measurement period.	Number	An assembly in ASP.NET is a collection of single-file or multiple files. The assembly that has more than one file contains either a dynamic link library (DLL) or an EXE file. The assembly also contains metadata that is known as assembly manifest. The assembly manifest contains data about the versioning requirements of the assembly, author name of the assembly, the security requirements that the assembly requires to run, and the various files that form part of the assembly. The biggest advantage of using ASP.NET Assemblies is that developers can create applications without interfering with other applications on the system.
Current logical threads:	The number of current managed thread objects in this worker process. This measure maintains the count of both running and stopped threads.	Number	
Current physical threads:	The number of native operating system threads created and	Number	

Measurement	Description	Measurement Unit	Interpretation
	owned by the common language runtime to act as underlying threads for managed thread objects. This measure does not include the threads used by the runtime in its internal operations.		
Current recognized threads:	The number of threads that are currently recognized by the runtime. These threads are associated with a corresponding managed thread object.	Number	
Contention rate:	The rate at which threads in the runtime attempt to acquire a managed lock unsuccessfully.	Rate/Sec	
Current queue length:	The total number of threads that are currently waiting to acquire a managed lock in the application.	Number	
Queue length rate:	Indicates the rate at which threads are waiting to acquire some lock.	Threads/Sec	
Recognized	Indicates the number	Threads/Sec	The recognized threads have a

Measurement	Description	Measurement Unit	Interpretation
threads rate:	of threads per second that have been recognized by the CLR.		corresponding .NET thread object associated with them. These threads are not created by the CLR; they are created outside the CLR but have since run inside the CLR at least once. Only unique threads are tracked; threads with the same thread ID re-entering the CLR or recreated after thread exit are not counted twice.
Queue length peak:	Indicates the total number of threads that waited to acquire some managed lock during the last measurement period.	Number	A high turnover rate indicates that items are being quickly added and removed, which can be expensive.
Recognized threads:	Indicates the total number of threads that have been recognized by the CLR during the last measurement period.	Number	The recognized threads have a corresponding .NET thread object associated with them. These threads are not created by the CLR; they are created outside the CLR but have since run inside the CLR at least once. Only unique threads are tracked; threads with the same thread ID re-entering the CLR or recreated after thread exit are not counted twice.
Contention threads:	Indicates the total number of times threads in the CLR have attempted to acquire a managed lock unsuccessfully.	Number	Managed locks can be acquired in many ways; by the lock statement in C# or by calling System.Monitor.Enter or by using MethodImplOptions.Synchronized custom attribute.
Time in runtime checks:	Indicates the percentage of elapsed time spent in	Percent	If this counter is high, revisit what is being checked and how often. The application may be executing

Measurement	Description	Measurement Unit	Interpretation
	performing runtime Code Access Security (CAS) checks during the last measurement period.		unnecessary stack walk depths. Another cause for a high percentage of time spent in runtime checks could be numerous linktime checks.
Stack walk depth:	Indicates the depth of the stack during that last measurement period.	Number	
Link time checks:	Indicates the total number of linktime Code Access Security (CAS) checks during the last measurement period.	Number	The value displayed is not indicative of serious performance issues, but it is indicative of the health of the security system activity.
Runtime checks:	Indicates the total number of runtime CAS checks performed during the last measurement period.	Number	A high number for the total runtime checks along with a high stack walk depth indicates performance overhead.

4.3 The Web Server Layer

Besides the **WebServer**, **Http**, and **HttpPost** tests that have already been dealt with in Chapter 2 of this document, the **Web Server** layer of an IIS web server (see Figure 4.9), is additionally mapped to an **IIS Web Transactions** test, an **IISWebCache** test, an **IISWebSites** test, an **IIS ApplicationPools** test, a **Windows Process Activation Service** test, and a **Worker Processes Statistics** test.

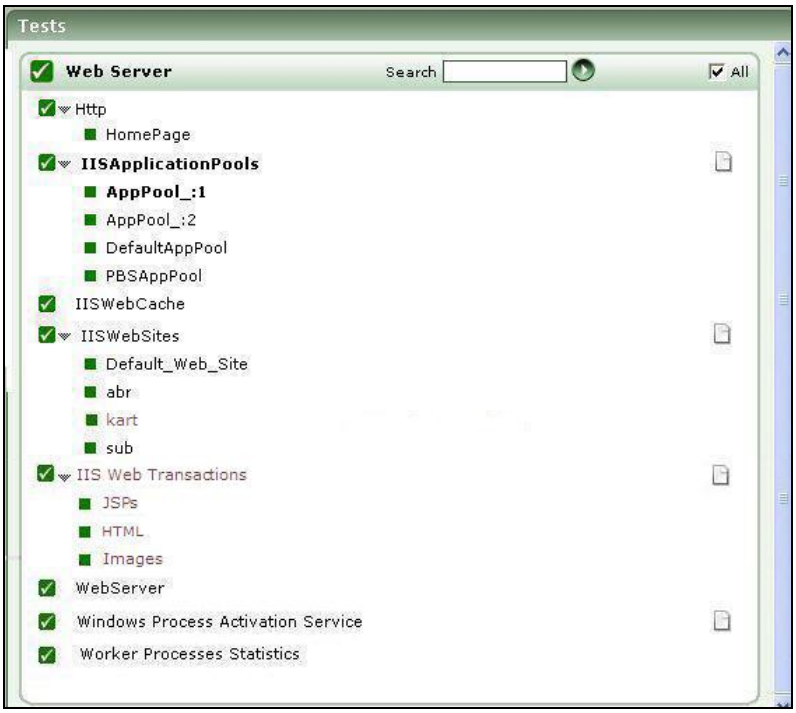


Figure 4.9: Tests associated with the Web Server layer of the IIS web server

4.3.1 Web Server Test

This internal test complements the measurements made from an external perspective by the Http test. It collects various statistics that measure the request load on the web server and its responsiveness to the load. The statistics collected by this test are detailed below:

Target of the test : A web server instance

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every web server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed .
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Connections:	Rate of connections to the web server.	Conns/Sec	An increase or decrease in connection rate can represent a change in user workload
Requests:	Rate of requests to the web server during the last measurement period.	Reqs/Sec	With the advent of HTTP/1.1, multiple requests can be transmitted over the same TCP connection. The ratio of requests per connection can provide an idea of the effectiveness of the HTTP 1.1 protocol.
Data transmitted:	Rate at which the data was transmitted by the server during the last measurement period.	KB/Sec	A large increase in the data transmission rate can be indicative of an increase in the popularity of one or more web sites hosted on the server.
Data received:	Rate at which the data was received by the server during the last measurement period.	KB/Sec	An increase in this value is indicative of an increase in user requests to the server.
Errors:	Percentage of error responses from the server during the last measurement period.	Percent	Percentage of responses with a 400 or 500 status code.
400 errors:	Percentage of responses with a status code in the range 400-499 during the last measurement period.	Percent	A high value indicates a number of missing/error pages.
Current requests:	Number of server threads/processes currently in use for serving requests (this measurement is not	Number	If a majority of the server threads/processes are in use simultaneously, this may be indicative of a server bottleneck.

Measurement	Description	Measurement Unit	Interpretation
	available for Apache web servers).		

Note:

The WebServer test will report metrics for an IIS web server executing on a Windows 2008 host, only if the **IIS Management Scripts and Tools** feature is installed on that host, and the **Web Server** role you create enables this feature. If the aforesaid feature is not enabled for the **Web Server** role, then remove the role and re-create it with the feature enabled.

4.3.2 IIS Web Sites Test

Typically, the Web Site Transactions test monitors only those web sites (on an IIS web server) for which transactions have been explicitly configured using the eG administrative interface. Therefore, if an IIS web server hosts multiple sites, and if transactions have been configured only for one of them, then the Web Site Transactions test will report statistics pertaining to that one site alone. Performance issues in other web sites, will hence go undetected and consequently, unresolved. Using this test, you can monitor each of the web sites hosted by an IIS web server, regardless of whether or not transactions have been configured for them.

Target of the test : A web site supported by a web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every web site monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - the port to which the specified **HOST** listens.
4. **WEBSITE NAME** - By default, this parameter is set to *none*. This implies that the test monitors all web sites, by default. If you want the test to monitor a specific web site alone, then specify that web site name here.

   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.

5. **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**.

6. **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.

Measurements made by the test

Measurement	description	Measurement Unit	Interpretation
Availability:	Indicates whether the web site is currently available or not.	Percent	The value 0 indicates that the website is not running, the value 100 indicates that it is up and running.
Data receive rate:	Indicates the rate at which the data is received by the web server.	KBytes/Sec	An increase in this value is indicative of an increase in user requests to the web server.
Data transmit rate:	Indicates the rate at which the data is transmitted by the web server.	KBytes/Sec	A large increase in the data transmission rate can be indicative of an increase in the popularity of one or more web sites hosted on the server.
Connection rate:	Indicates the rate of connections to the web server.	Conns/Sec	An increase or decrease in connection rate can represent a change in user workload.

Measurement	description	Measurement Unit	Interpretation
Current requests:	Indicates the number of server threads/ processes that are currently in use for serving requests.	Number	If a majority of the server threads/processes are in use simultaneously, this may be indicative of a server bottleneck.
Requests serviced rate:	Indicates the rate at which all HTTP Requests are issued.	Requests/Sec	
Current anonymous users:	Indicates the number of users who currently have an anonymous connection.	Number	
Current non-anonymous users:	Indicates the number of users who currently have non-anonymous connection.	Number	
Get requests:	Indicates the rate at which HTTP requests using the GET method are made.	Requests/Sec	
Post requests:	Indicates the rate at which HTTP requests using the POST method are made.	Requests/Sec	
Head requests:	Indicates the rate at which HTTP requests using the HEAD method are issued.	Requests/Sec	
Maximum connections:	Indicates the maximum number of simultaneous connections established in the last measurement period.	Number	
Not found requests:	Indicates the rate of errors due to requests that could not be satisfied by the web server because the requested document could not found.	Requests/Sec	A high value indicates a number of missing/error pages.
Requests serviced:	Indicates the number of	Number	

Measurement	description	Measurement Unit	Interpretation
	HTTP requests serviced, currently.		
Applications configured to this site	Indicates the number of applications configured to this site.	Number	Use the detailed diagnosis of this measure to know which applications are configured to this site.
Application pools assigned to this site	Indicates the number of application pools assigned to this site.	Number	Use the detailed diagnosis of this measure to know which application pools are assigned to this site.

The detailed diagnosis of the Applications configured to this site measure lists the applications that are configured to the monitored site and the application pool to which every application belongs.

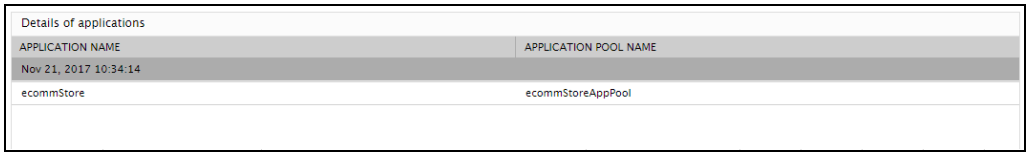


Figure 4.10: Detailed diagnosis revealing applications configured to the monitored site

The detailed diagnosis of the Application pools assigned to this site measure reveals the application pools that are assigned to the monitored site and the status of each pool.

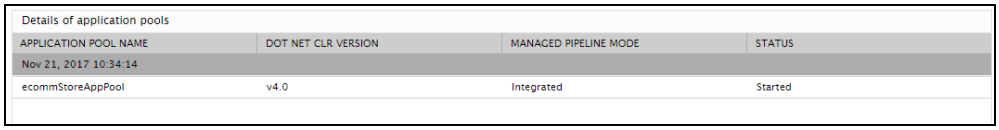


Figure 4.11: Detailed diagnosis revealing the details of application pools assigned to the monitored site

4.3.3 IIS Web Cache Test

This test monitors the IIS web server's web cache and reports critical performance metrics pertaining to it.

Note that this test will not work on Windows 2000.

Target of the test : A web site supported by a web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every web site monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** - The host for which the test is to be configured
3. **PORT** - the port to which the specified **HOST** listens

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
File cache memory usage:	Indicates data used for the user- mode file cache.	KB	An unusually high value may indicate a sudden increase of memory usage in file cache.
Files cached:	Indicates the number of files, the content of which is in the user- mode cache.	Number	
Metadata cached:	Indicates the metadata information blocks currently in the user-mode cache.	Number	
URIs cached:	Indicates the number of URI information blocks in the user-mode cache.	Number	
File cache hits:	Indicates the number of successful lookups in the user- mode file cache during the last measurement period.	Number	A consistent decrease in the value of this measure is indicative of an increase in direct disk accesses. This is a cause for concern.
File cache misses:	Indicates the number of unsuccessful lookups in the user-mode file cache during the last measurement period.	Number	A consistent increase in the value of this measure is indicative of an increase in direct disk accesses. This is a cause for concern.
File cache hit ratio:	Indicates the ratio between the file cache	Percent	A very low percentage indicates that files requested are being

Measurement	Description	Measurement Unit	Interpretation
	hits and file cache misses.		directly accessed from the disk and not from the file cache.

4.3.4 Application Pool Workers Test

An application pool is a group of one or more URLs that are served by a worker process or a set of worker processes. Application pools set boundaries for the applications/web sites they contain, which means that applications/web sites that are running outside a given application pool cannot affect the applications/web sites in the application pool. On the other hand, a single resource-intensive application/web site within a pool can affect overall pool performance, and eventually impact web server performance adversely! This poses a huge management challenge, particularly in MSP environments, where a single web server may host multiple web sites – one each for every MSP customer. A web server slowdown in such environments often leaves administrators flummoxed, as they struggle to fathom what problem in which web site of which application pool is responsible for the slowdown. This is where the **Application Pool Workers** test helps!

By periodically monitoring the overall status, requests, resource usage, open handles, and I/O activity in each application pool on an IIS web server, the test quickly and accurately points you to the problematic application pool and the nature of its problem – has the application pool stopped? is a web site/application in the pool utilizing resources excessively? is any handle leak detected in the pool? is there any I/O processing bottleneck in the pool? is any web site/applicationn in the pool responding slowly to user requests? You can even quickly drill down to the exact web site/application that is causing this problem, using the detailed diagnosis capability of the test. In addition, the test also captures and reports operational issues with the worker processes that service requests to an application pool, thereby shedding light on the contribution of worker processes to slowdowns/outages that a web server suffers.

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every application pool on the IIS web server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** - The host for which the test is to be configured.

3. **PORT** - The port to which the specified **HOST** listens.
4. **WEBSITE NAME** - By default, this parameter is set to *none*. This implies that the test monitors all web sites, by default. If you want the test to monitor a specific web site alone, then specify that web site name here.

   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.
5. **SLOW REQUEST CUT OFF** - If a request is processed for a period equal to or more than the duration (in second) specified here, then such a request is counted as a slow request. By default, the slow request cut off is 4. This means that, by default, the test will report all requests that take 4 seconds or more to be serviced as slow requests. You can override this default setting by specified a different cut-off value here.
6. SHOW WAITING THREAD DD -
7. **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**.
8. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Pool status	Indicates the current		The values that this measure can

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td>status of this application pool.</td><td></td><td>report and their corresponding numeric values have been listed in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Stopped</td><td>0</td></tr><tr><td>Started</td><td>1</td></tr></table>
If this measure reports the value Sopped, then you need to make sure that the application pool is started. If a pool has stopped, then you can use the detailed diagnosis of this measure to identify the applications/web sites that are impacted as a result of this pool stopping.
This measure is reported for IIS v6 (and above) only.
Note:
By default, the measure reports the Measure Values listed in the table above to indicate the status of the application pool. The graph of this measure however represents the same using the numeric equivalents only.</td></tr>
<tr><td>Has pool been restarted?</td><td>Indicates whether this pool is restarted or not.</td><td></td><td>The values that this measure can report and their corresponding numeric values have been listed in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>No</td><td>0</td></tr><tr><td>Yes</td><td>1</td></tr></table></td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
			This measure is reported for IIS v7 (and above) only. **Note:** By default, the measure reports the **Measure Value**s listed in the table above to indicate the restart status of the application pool. The graph of this measure however represents the same using the numeric equivalents only.
Number of processes	Indicates the number of worker processes that are currently servicing requests to this application pool.	Number	A zero value in this measure indicates that the application pool has crashed. Use the detailed diagnosis of this measure to know the pid of every worker process that is handling requests to this pool and the command line argument revealing the name of the application/web site in the application pool, which is accessed by that worker process. In addition, the detailed diagnosis also displays the websites associated with the worker process, resource allocations (private data, pool paged data, non pool paged data), the resource usage (CPU and memory), open handle count, thread count, and I/O activity of each application in the pool that is accessed by every worker process. This way, resource-intensive applications in the pool can be isolated. This measure is reported for IIS v6 (and above) only.
CPU utilization	Indicates the percentage CPU utilization of this application pool.	Percent	A higher value indicates excessive CPU utilization. Use the detailed diagnosis of this measure to know the pid of every worker process that is handling

Measurement	Description	Measurement Unit	Interpretation
			requests to this pool and the command line argument revealing the name of the application/web site in the application pool, which is accessed by that worker process. In addition, the detailed diagnosis also displays the websites associated with the worker process, resource allocations (private data, pool paged data, non pool paged data), the resource usage (CPU and memory), open handle count, thread count, and I/O activity of each application in the pool that is accessed by every worker process. This way, resource-intensive applications in the pool can be isolated. This measure is reported for IIS v6 (and above) only.
Memory utilization	Indicates the percentage of total memory utilized by this application pool.	Percent	If the value of this measure consistently increases, it indicates a memory bottleneck. Use the detailed diagnosis of this measure to know the pid of every worker process that is handling requests to this pool and the command line argument revealing the name of the application/web site in the application pool, which is accessed by that worker process. In addition, the detailed diagnosis also displays the websites associated with the worker process, resource allocations (private data, pool paged data, non pool paged data), the resource usage (CPU and memory), open handle count, thread count, and I/O activity of each application in the pool that is accessed by every worker process. This way, resource-intensive applications in the pool can be isolated.

Measurement	Description	Measurement Unit	Interpretation
			This measure is reported for IIS v6 (and above) only.
Number of threads	Indicates the number of threads that are currently active in this application pool.	Number	This is an indicator of the workload on the pool. This measure is reported for IIS v6 (and above) only.
Number of handles	Indicates the number of handles currently opened by this application pool.	Number	A sudden/consistent increase in the value of this measure could indicate a handle leak in the pool. This measure is reported for IIS v6 (and above) only.
Page faults	Indicates the rate of page faults happening in this application pool.	Faults/Sec	This measure is reported for IIS v6 (and above) only.
Private data	Indicates the amount of data that this application pool has currently allocated and cannot be shared with other application pools.	MB	A gradual growth in this measure indicates a memory leak in the websites that are running in this application pool. This measure is reported for IIS v6 (and above) only.
Pool paged data	Indicates the amount of memory currently allocated from the paged pool to this application pool. The Paged Pool is an area of the System's virtual memory that is limited in size and used for various system related functions.	MB	A gradual growth in this measure indicates a memory leak in the websites that are running in the application pool. This measure is reported for IIS v6 (and above) only.
Pool non paged data	Indicates the amount of memory currently allocated from the non-paged pool to this application pool. The Non-	MB	A gradual growth in this measure indicates a memory leak in the websites that are running in the application pool. Running out of space in the nonpaged pool often results in a Blue Screen.

Measurement	Description	Measurement Unit	Interpretation
	Paged Pool is an area of the System's virtual memory that is limited in size and used for kernel and device driver functions.		This measure is reported for IIS v6 (and above) only.
I/O reads	Indicates the rate at which the worker process(es) servicing requests to this application pool is reading data.	KBytes/Sec	These measures are reported for IIS v6 (and above) only. Use the detailed diagnosis of these measures to know the pid of every worker process that is handling requests to this pool and the command line argument revealing the name of the application/web site in the application pool, which is accessed by that worker process. In addition, the detailed diagnosis also displays the websites associated with the worker process, resource allocations (private data, pool paged data, non pool paged data), the resource usage (CPU and memory), open handle count, thread count, and I/O activity of each application in the pool that is accessed by every worker process. This way, resource-intensive applications in the pool can be isolated. A very low or a consistent dip in the value of these measures is a cause for concern, as it could indicate a bottleneck when processing read requests to the application pool. You can compare the value of these measures across pools, to identify that application pool that is the slowest in read request processing.
I/O read operations	Indicates the rate at which the worker process(es) serving requests to this pool is performing I/O reads.	Operations/Sec	
I/O writes	Indicates the rate at which the worker process(es) serving requests to this	KBytes/Sec	These measures are reported for IIS v6 (and above) only.

Measurement	Description	Measurement Unit	Interpretation
	pool is writing data.		Use the detailed diagnosis of these measures to know the pid of every worker process that is handling requests to this pool and the command line argument revealing the name of the application/web site in the application pool, which is accessed by that worker process. In addition, the detailed diagnosis also displays the websites associated with the worker process, resource allocations (private data, pool paged data, non pool paged data), the resource usage (CPU and memory), open handle count, thread count, and I/O activity of each application in the pool that is accessed by every worker process. This way, resource-intensive applications in the pool can be isolated. A very low or a consistent dip in the value of these measures is a cause for concern, as it could indicate a bottleneck when processing write requests to the application pool. You can compare the value of these measures across pools, to identify that application pool that is the slowest in write request processing.
I/O write operations	Indicates the rate at which the worker process(es) serving requests to this pool is issuing write I/O commands.	Operations/Sec	
Pool uptime during the last measure period	Indicates the length of time, in seconds, that this application pool has been running since it was started.	Secs	
Current process	Indicates the number of worker processes currently servicing requests to this application pool.	Number	This measure is reported for IIS v7 (and above) only.

Measurement	Description	Measurement Unit	Interpretation
Recent failure	Indicates the number of times the worker processes associated with this pool failed during the rapid-fail protection interval.	Number	Rapid Fail Protection is a feature in IIS 7, which is enabled by default. This feature checks for a specified number of hard failures in a given time period - 5 failures within 5 minutes by default. If a hard failure that occurs meets this default setting, then the Application Pool will crash and does not automatically restart. It would be good practice to set the hard failure setting as the threshold for this measure; this way, you can be proactively alerted to a potential application pool crash. This measure is reported for IIS v7 (and above) only.
Pool recycle	Indicates the number of times this application pool was recycled since Windows Process Activation Service (WAS) started.	Number	If you have a problematic application and you cannot easily correct the code that causes the problems, you can limit the extent of these problems by periodically recycling the worker process that services the application. In addition to recycling an application pool on demand when problems occur, you can configure an application pool to recycle a worker process for the following reasons: • At a scheduled time • After an elapsed time • After reaching a number of requests • After reaching a virtual memory threshold • After reaching a used memory threshold

Measurement	Description	Measurement Unit	Interpretation
			This measure is reported for IIS v7 (and above) only.
Startup failure	Indicates the number of times that Windows Process Activation Service (WAS) failed to start a worker process that serves requests to this application pool.	Number	Ideally, the value of this measure should be 0. This measure is reported for IIS v7 (and above) only.
Total failure	Indicates the number of times the worker processes handling requests to this pool have crashed since the application pool was started.	Number	Ideally, the value of this measure should be 0. This measure is reported for IIS v7 (and above) only.
Ping failure	Indicates the number of times the Windows Process Activation Service (WAS) did not receive a response to ping messages sent to a worker process that serves requests to this application pool.	Number	IIS periodically pings a worker process to determine its responsiveness. A high value of this measure indicates that very often the worker process is too busy to respond to these ping requests. This may sometimes force IIS to terminate the worker process. Since the worker process clings on to areas of memory, the termination attempt may fail. As a result, errors/delays may occur when users attempt to hit specific web pages on the IIS web server. To avoid this, you can either disable the Ping, or increase the Ping Maximum Response Time. This measure is reported for IIS v7 (and above) only.
Shutdown failure	Indicates the number of times that Windows Process Activation Service (WAS) failed to	Number	This measure is reported for IIS v7 (and above) only.

Measurement	Description	Measurement Unit	Interpretation
	shut down a worker process servicing requests to this application pool.		
Initialized threads	Indicates the number of threads that are currently in an initialized state in this application pool.	Number	
Ready threads	Indicates the number of threads that are currently in the 'Ready' state in this application pool.	Number	A Ready thread wants to use a processor but is waiting because none is free. A high value for this measure is therefore indicative of a potential processor contention, probably caused by an overload or a processing bottleneck in the application pool.
Running threads	Indicates the numberof threads that are currently in the 'Running' state in this application pool.	Number	This measure is a good indicator of the current workload on the application pool.
Threads in STANDBY state:	Indicates the number of threads that are currently in the **STANDBY** state in this application pool.	Number	A STANDBY thread is about to use a processor.
Terminated threads	Indicates the number of threads that are currently in a **TERMINATED** state in this application pool.	Number	A low value is desired for this measure.
Threads in WAIT state	Indicates the number of threads that are in the WAIT state in this application pool.	Number	A waiting thread has no use for a processor because it is waiting for a peripheral operation or a resource to become free. A high value for this measure could hint at a potential resource contention.
Threads in TRANSITION state	Indicates the number of threads that are currently in the TRANSITION state	Number	A thread in transition is waiting for a resource in order to execute, such as waiting for its execution stack to be

Measurement	Description	Measurement Unit	Interpretation
	in this pool.		paged in from a disk.
Threads in UNKNOWN state	Indicates the number of threads that are currently in the UNKNOWN state in this application pool.	Number	Ideally, the value of this measure should be 0.
Percentage of running threads	Indicates the percentage of threads that are currently in the RUNNING state in this application pool.	Percent	A high value is desired for this measure, as it indicates that that the application pool has adequate active threads to process its requests.
Actively processing request threads	Indicates the number of threads actively processing requests in the worker process that is serving requests to this pool.	Number	
Requests being processed	Indicates the total number of requests that this application pool is processing currently.	Number	This is a good indicator of the current workload of an application pool. To view the details of active requests, use the detailed diagnosis of this measure.
Slow requests	Indicates the number of requests to this pool that are slow.	Number	Ideally, the value of this measure should be 0. A non-zero value denotes that requests to one/more web sites/applications in this pool have violated the **SLOW REQUEST CUT OFF** configured for this test. Compare the value of this measure across pools to know which pool contains the slowest web sites/applications. To know which are the slow web sites/applications in the pool, use the detailed diagnosis of this measure.
Current file cache memory usage	Indicates the amount of memory used by user-	MB	

Measurement	Description	Measurement Unit	Interpretation
	mode file cache of the worker process that is serving requests to this pool.		
Current files cached	Indicates the current number of files whose contents are present in the user-mode file cache of the worker process that is serving requests to this pool.	Number	
Requests rate	Indicates the number of HTTP requests/sec being processed by the worker process that is serving requests to this pool.	Requests/Sec	
HTTP requests served	Indicates the total number of HTTP requests served by the worker process that is serving requests to this pool.	Number	
Available threads to process requests	Indicates the total number of threads available to process requests in the worker process that is serving requests to this pool.	Number	By closely monitoring the thread usage over time, you can proactively capture when the worker process runs out of idle threads, and promptly take corrective measures, so as to avert any processing bottlenecks.
Max threads from thread pool	Indicates the maximum number of threads to which this thread pool can grow.	Number	

Use the detailed diagnosis of the *Number of processes* measure to know the **PID** of every worker process that is handling requests to this pool and the command line argument revealing the name of the application/web site in the application pool, which is accessed by that worker process. In addition, the detailed diagnosis also displays the websites associated with the worker process, resource allocations (private data, pool paged data, non pool paged data), the resource usage (CPU

and memory), open handle count, thread count, and I/O activity of each application in the pool that is accessed by every worker process. This way, resource-intensive applications in the pool can be isolated.

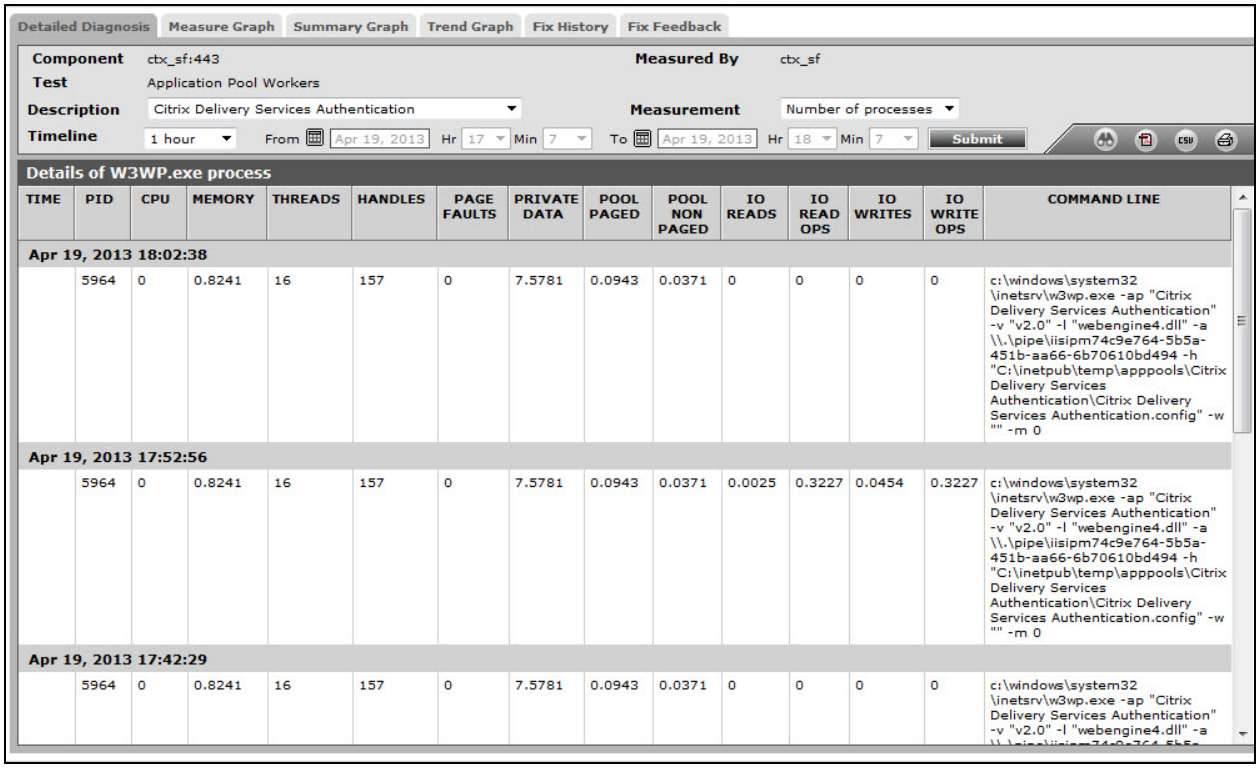


Figure 4.12: The detailed diagnosis of the Number of processes measure of the Application Worker Processes test

The detailed diagnosis of the Requests being processed measure reveals the details of the active requests to the application pool. The request URL, the type of request (Verb), the stage, module name, the web site to which the request pertains, the worker process that processed the request, the client from which the request originated, and the time taken to process the request are reported as part of the detailed diagnostics. From this information, administrators can proactively detect a potential request processing bottleneck, and can initiate preventive measures immediately. Additionally, administrators can also figure out if many requests to a web site are slowing down, and thus isolate problematic web sites.

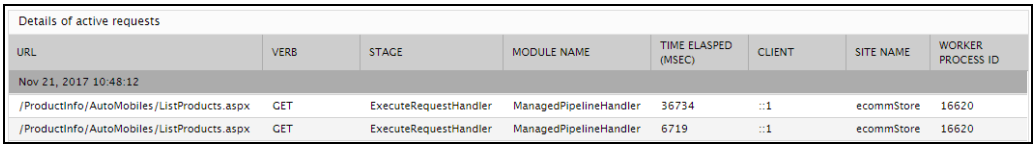
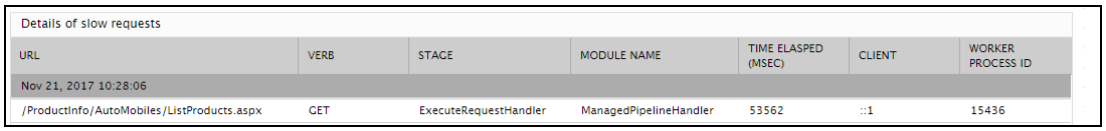


Figure 4.13: Detailed diagnosis of the Requests being processed measure

The detailed diagnosis of the Slow requests measure lists the request URLs that were processed slowly by the web site. Since the list is sorted in the descending order of the Total Elapsed time - i.e., processing time - a quick look at the list will reveal to you the slowest request. The type of request (GET or POST), the stage, and module name are also displayed as part of the diagnostics to aid effective troubleshooting.



Details of slow requests

URL	VERB	STAGE	MODULE NAME	TIME ELASPED (MSEC)	CLIENT	WORKER PROCESS ID
Nov 21, 2017 10:28:06						
/ProductInfo/AutoMobiles/ListProducts.aspx	GET	ExecuteRequestHandler	ManagedPipelineHandler	53562	::1	15436

Figure 4.14: The detailed diagnosis of the Slow requests measure

4.3.5 IIS Application Pools Test

An Application Pool can contain one or more applications and provides a level of isolation between different Web applications. For example, if you want to isolate all the Web applications running in the same computer, you can do this by creating a separate application pool for every Web application and placing them in their corresponding application pool. Because each application pool runs in its own worker process, errors in one application pool will not affect the applications running in other application pools.

The IIS Application Pools test discovers all the application pools configured on an IIS Web server and monitors their status. **Note that this test will work on IIS v. 6.0 and above only.**

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every web site monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** - The host for which the test is to be configured
3. **PORT** - the port to which the specified **HOST** listens
4. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Application pool status:	Indicates the current status of this application pool.	Boolean	If this measure returns the value 1, it indicates that the application pool has started or is starting. While the value 0 for this measure indicates that the application pool has stopped or is stopping, the value -1 indicates that the pool is in an 'Unknown' state. If this measure reports the value 0 or -1, then you need to make sure that the application pool is started. If a pool has stopped, then you can use the detailed diagnosis of this measure to identify the applications that are impacted as a result of this pool stopping.

4.3.6 IIS Web Server Test

This internal test complements the measurements made from an external perspective by the Http test. It collects various statistics that measure the request load on the web server and its responsiveness to the load. The statistics collected by this test are detailed below:

Target of the test : A web server instance

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every web server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** - The host for which the test is to be configured
3. **PORT** - the port to which the specified **HOST** listens

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Connections:	Rate of connections to the web server.	Conns/Sec	An increase or decrease in connection rate can represent a change in user workload
Requests:	Rate of requests to the web server during the last measurement period.	Reqs/Sec	With the advent of HTTP/1.1, multiple requests can be transmitted over the same TCP connection. The ratio of requests per connection can provide an idea of the effectiveness of the HTTP 1.1 protocol.
Data transmitted:	Rate at which the data was transmitted by the server during the last measurement period.	KB/Sec	A large increase in the data transmission rate can be indicative of an increase in the popularity of one or more web sites hosted on the server.
Data received:	Rate at which the data was received by the server during the last measurement period.	KB/Sec	An increase in this value is indicative of an increase in user requests to the server.
Errors:	Percentage of error responses from the server during the last measurement period.	Percent	This is the percentage of errors that occurred due to requests that couldn't be satisfied by the server because the requested document could not be found.

Measurement	Description	Measurement Unit	Interpretation
400 errors:	Percentage of responses with a status code in the range 400-499 during the last measurement period.	Percent	A high value indicates a number of missing/error pages.
Current requests:	Number of server threads/processes currently in use for serving requests (this measurement is not available for Apache web servers).	Number	If a majority of the server threads/processes are in use simultaneously, this may be indicative of a server bottleneck.

Note:

The Web Server test will report metrics for an IIS web server executing on a Windows 2008 host, only if the **IIS Management Scripts and Tools** feature is installed on that host, and the **Web Server** role you create enables this feature. If the aforesaid feature is not enabled for the **Web Server** role, then remove the role and re-create it with the feature enabled.

4.3.7 HTTP Errors Test

Some errors that occur in IIS are automatically handled by the HTTP API instead of being passed back to IIS for handling. This behavior occurs because the frequency of such errors might otherwise flood an event log or an application handler. The errors captured by the HTTP API are logged in *httperr.log* files that are by default created in the **%SystemRoot%\System32\LogFiles\HTTPERR** folder on the IIS host. These log files provide a wealth of information on the following:

- **Responses to clients**: The HTTP API sends an error response to a client, for example, a 400 error that is caused by a parse error in the last received request. After the HTTP API sends the error response, it closes the connection.
- C**Connection time-outs**: The HTTP API times out a connection. If a request is pending when the connection times out, the request is used to provide more information about the connection in the error log.
- **Orphaned requests**: A user-mode process stops unexpectedly while there are still queued requests that are routed to that process. The HTTP API logs the orphaned requests in the error log.

Moreover, specific error types are also designated by **Reason Phrase** strings that always appear as the last field of each error line in the log file.

By periodically parsing these log files and reading the errors/reasons captured by the same, administrators can significantly shorten troubleshooting cycles and rapidly resolve problems. This is where the **HTTP Errors** test helps. This test, at configured intervals, parses the HTTP error log file, reads the reason phrases logged therein, and reports the number of errors that occurred due to each reason. Detailed diagnostics provided by the test also shed more light on the nature of these errors. This way, the test simplifies problem identification, analysis, and resolution.

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for the IIS web server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** - The host for which the test is to be configured
3. **PORT** - the port to which the specified **HOST** listens
4. **LOG FILE PATH** – The **httperr.log** files are by default created in the **%SystemRoot%\System32\LogFiles\HTTPERR** folder on the target server. Since this test, by defaut, looks for the **httperr.log** files in the aforesaid directory only, the **LOG FILE PATH** parameter is set to **none** by default. However, if you have configured these log files to be created in a different directory in your environment, then you will have to explicitly specify the full path to these log files in the **LOG FILE PATH** text box. For instance, your path specification can be: **C:\LogFiles\HttpErrors**
5. **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**.
6. 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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Application offline:	Indicates the number of errors logged in the log file with AppOffline as the Reason Phrase.	Number	If the Reason Phrase is AppOffline, it implies that a service unavailable error occurred (an HTTP error 503). The service is not available because application errors caused the application to be taken offline. The detailed diagnosis of this measure reveals the complete details of errors with AppOffline as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response

Measurement	Description	Measurement Unit	Interpretation
			to which the error occurred, and the reason for the error.
Busy application pool processes:	Indicates the number of errors logged in the log file with AppPoolTimer as the Reason Phrase.	Number	If the Reason Phrase is AppPoolTimer, it implies that a service unavailable error occurred (an HTTP error 503), because the application pool process was too busy to handle the request. The detailed diagnosis of this measure reveals the complete details of errors with AppPoolTimer as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.
Application shutdown:	Indicates the number of errors logged in the log	Number	If the Reason Phrase is AppShutdown, it implies that a service unavailable error occurred

Measurement	Description	Measurement Unit	Interpretation
	file with AppShutdown as the Reason Phrase.		(an HTTP error 503), because application shut down automatically in response to administrator policy. The detailed diagnosis of this measure reveals the complete details of errors with AppShutdown as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.
Bad requests:	Indicates the number of errors logged in the log file with BadRequest as the Reason Phrase.	Number	If the Reason Phrase is BadRequest, it implies that a parse error occurred while processing a request. The detailed diagnosis of this measure reveals the complete

Measurement	Description	Measurement Unit	Interpretation
			details of errors with BadRequest as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.
Connections reset by client:	Indicates the number of errors logged in the log file with Client_Reset as the Reason Phrase.	Number	If the Reason Phrase is Client_ Reset, it implies that the connection between the client and the server was closed before the request could be assigned to a worker process. The most common cause of this behavior is that the client prematurely closes its connection to the server. The detailed diagnosis of this measure reveals the complete details of errors with Client_Reset as the Reason Phrase. Such

Measurement	Description	Measurement Unit	Interpretation
			details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.
Connections abandoned by application pool:	Indicates the number of errors logged in the log file with Connection_ Abandoned_ By_ AppPool as the Reason Phrase.	Number	If the Reason Phrase is Connection_ Abandoned_ By_ AppPool, it implies that a worker process from the application pool has quit unexpectedly or orphaned a pending request by closing its handle. The detailed diagnosis of this measure reveals the complete details of errors with Connection_ Abandoned_ By_ AppPool as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address

Measurement	Description	Measurement Unit	Interpretation
			and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.
Connections abandoned by request queue:	Indicates the number of errors logged in the log file with Connection_ Abandoned_ By_ ReqQueue as the Reason Phrase.	Number	If the Reason Phrase is Connection_ Abandoned_ By_ ReqQueue, it implies that a worker process from the application pool has quit unexpectedly or orphaned a pending request by closing its handle. This is specific to Windows Vista and later versions and to Windows Server 2008 and later versions. The detailed diagnosis of this measure reveals the complete details of errors with Connection_ Abandoned_By_ReqQueue as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the

Measurement	Description	Measurement Unit	Interpretation
			version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.
Connections dropped:	Indicates the number of errors logged in the log file with Connection_ Dropped as the Reason Phrase.	Number	If the Reason Phrase is Connection_ Dropped, it implies that the connection between the client and the server was closed before the server could send its final response packet. The most common cause of this behavior is that the client prematurely closes its connection to the server. The detailed diagnosis of this measure reveals the complete details of errors with Connection_ Dropped as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes,

Measurement	Description	Measurement Unit	Interpretation
			the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.
Connections limit has reached:	Indicates the number of errors logged in the log file with ConnLimit as the Reason Phrase.	Number	If the Reason Phrase is ConnLimit, it implies that a a service unavailable error occurred (an HTTP error 503), because the site level connection limit has been reached or exceeded. The detailed diagnosis of this measure reveals the complete details of errors with ConnLimit as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly

Measurement	Description	Measurement Unit	Interpretation
			figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.
Connections refused:	Indicates the number of errors logged in the log file with Connections_ Refused as the Reason Phrase.	Number	If the Reason Phrase is Connections_ Refused, it implies that the kernel NonPagedPool memory has dropped below 20MB and http.sys has stopped receiving new connections. The detailed diagnosis of this measure reveals the complete details of errors with Connections_ Refused as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.

Measurement	Description	Measurement Unit	Interpretation
Internal errors:	Indicates the number of errors logged in the log file with Internal as the Reason Phrase.	Number	If the Reason Phrase is Internal, it implies that an internal server error occurred (an HTTP error 500). The detailed diagnosis of this measure reveals the complete details of errors with Internal as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.
Application request limit reached:	Indicates the number of errors logged in the log file with QueueFull as the Reason Phrase.	Number	If the Reason Phrase is QueueFull, it implies that a service unavailable error occurred (an HTTP error 503), because the application request queue is full. The detailed diagnosis of this measure reveals the complete details of errors with QueueFull as

Measurement	Description	Measurement Unit	Interpretation
			the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.
Connections expired:	Indicates the number of errors logged in the log file with Connection_ Dropped_ List_ Full as the Reason Phrase.	Number	If the Reason Phrase is Connection_Dropped_List_Full, it implies that the list of dropped connections between clients and the server is full. This is specific to Windows Vista and later versions and to Windows Server 2008 and later versions. The detailed diagnosis of this measure reveals the complete details of errors with Connection_ Dropped_List_Full as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected

Measurement	Description	Measurement Unit	Interpretation
			client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.
URL processing errors:	Indicates the number of errors logged in the log file with URL as the Reason Phrase.	Number	If the Reason Phrase is URL, it implies that a parse error occurred while processing a URL. The detailed diagnosis of this measure reveals the complete details of errors with URL as the Reason Phrase. Such details include, the date and time of the error, the IP address and port of the affected client, the IP address and port of the affected server, the version of the protocol being used, the verb state that the last request that is parsed passes, the URL requested and any query that is associated with it, the protocol status of the response to the request, the reason phrase, and the request queue name. From

Measurement	Description	Measurement Unit	Interpretation
			these details, you can instantly figure out when the error occurred, the client and server affected by the error, the request in response to which the error occurred, and the reason for the error.

4.3.8 HTTP Service Request Queues Test

By monitoring request queues to every IIS application pool, administrators can identify those application pools that have too many requests pending and those with a high request rejection rate. This is exactly what the **HTTP Service Request Queues** test does. This test auto-discovers the application pools and reports the length of request queues and rejection rate of requests in queue for each application pool. This way, the test sheds light on those application pools that suffer from processing pains.

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every application pool on Exchange

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Requests arrival rate	Indicates the rate at which requests are arriving in the request queue of this application pool.	Reqs/Sec	A high rate could indicate a probable request overload on an application pool.

Measurement	Description	Measurement Unit	Interpretation
Requests in queue	Indicates the current number of requests in the queue of this application pool.	Number	If this value increases consistently, it could indicate that the application pool is not processing requests quickly. Compare the value of this measure across pools to identify the pool with a processing bottleneck.
Requests rejected from queue	Indicates the number of requests in this application pool's queue that were rejected.	Number	A non-zero value is desired for this measure.
Cache hit rate	Indicates the rate of cache hits from the queue of this application pool.	Hits/sec	
Request rejection rate	Indicates the rate at which this application pool rejected requests in the queue.	Reqs/Sec	Compare the value of this measure across pools to know which pool rejects queued requests frequently.
Maximum queue item age	Indicates the age of the oldest request in the queue.	Seconds	

4.3.9 IIS Web Transactions Test

Whenever users to your mission-critical web sites (on an IIS web server) or web services (overlying an IIS web server) complain of slowdowns, the first step to troubleshooting these issues is identifying which specific transaction(s) to the web sites is causing the slowdown. For this, you will have to periodically monitor the responsiveness of key transactions to your web sites, so as to quickly and accurately identify slow transactions. The **IIS Web Transactions** test helps you achieve this. This test monitors user-configured transaction patterns, captures the response time of each configured pattern in real-time, compares these actuals with the desired levels of responsiveness (which is also configurable), and thus isolates and proactively alerts you to slow transactions.

Note:

To make sure that this test reports measures, do the following:

- Configure a **Web Server** role on the target Windows 2008 server.
- Install and configure **Advanced Logging** for the IIS web server operating on Windows 2008

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every **NAME** configured

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** - The host for which the test is to be configured
3. **PORT** - The port to which the specified **HOST** listens
4. **NAME, PATTERN, THRESHOLD VALUE** - Using these text boxes, you can configure the URL patterns for the transactions to be monitored by this test. You can group a set of related/similar patterns under a common head - a display name for this group head can be provided in the **NAME** text box; this display name will appear as a descriptor of the test in the eG monitoring console. In the **PATTERN** text box, you can configure the transaction patterns that are to be grouped under the specified **NAME**. Multiple transaction patterns can be separated by a semi-colon. Wild card characters can be used while configuring these patterns. For example, to monitor all transactions with the extension **html** or **jsp**, your **PATTERN** specification should be: **.html;*.jsp*. In the **THRESHOLD VALUE** text box, specify the response time value (in seconds), which should be violated, for a transaction to be counted as a 'slow transaction'. All transaction patterns grouped under a given **NAME** will be governed by the **THRESHOLD VALUE** specified.

   You can, if you so need, configure multiple display **NAME**s, **PATTERN**s, and **THRESHOLD VALUE**s for monitoring. To enable you to configure the multiple values easily, the eG administrative interface provides a special page. To access this page, click on the **Click here** hyperlink, which appears just above the parameters of this test in the **TEST CONFIGURATION PAGE**. Refer to the Section **4.3.9.1** to know how to use this special page.

5. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Request rate:	Indicates the rate of requests to this transaction.	Requests/Sec	
Requests processed:	Indicates the number of requests to this transaction during the last measurement period.	Requests	
Successful responses:	Indicates the percentage of responses with the response code of 200.	Percent	Typically, successful requests are indicated using the response code 200 - 299.
Redirects:	Indicates the percentage of responses with the response code of 300.	Percent	300 responses could indicate page caching on the client browsers. Alternatively 300 responses could also indicate redirection of requests. A sudden change in this value could indicate a problem condition.
Responses with Status 400:	Indicates the percentage of responses with the response code of 400.	Percent	A high value indicates a number of missing/error pages.
Responses with Status 500:	Indicates the percentage of responses with the response code of 500.	Percent	Since responses with a status code of 500- 600 indicate server side processing errors, a high value reflects an error condition.

Measurement	Description	Measurement Unit	Interpretation
Min response time:	Indicates the minimum response time of this transaction.	Secs	
Max response time:	Indicates the maximum response time of this transaction.	Secs	
Avg. response time:	Indicates the average response time for all requests to this transaction.	Secs	A low value is desired for this measure. A sudden or steady increase in this value could indicate a problem with the transaction.
Data transmitted:	Indicates the total data transmitted by this transaction.	KB	
Avg. data transmitted:	Indicates the data transmitted by this transaction on an average.	KB	
Slow responses:	Indicates the number of requests to this transaction with a response time greater than the specified threshold value.	Number	Ideally, the value of this measure should be 0. A non-zero value indicates the existence of one/more slow transactions. Comparing the value of this measure across transactions will quickly point you to the transactions that are very slow. Using the detailed diagnosis of this measure you can view the complete details of the slow transactions.
Slow response ratio:	Indicates the percentage of requests to this transaction that were	Percent	A very low value is desired for this measure.

Measurement	Description	Measurement Unit	Interpretation
	serviced slowly - i.e., the percentage of requests with a response time greater than the specified threshold value.		When users complaint of slowdowns while trying to access a web site (on the monitored IIS web server), you may want to compare the value of this measure across transactions to know which transaction is the slowest, and hence could be causing the slowdown.
Avg. response time for slow responses:	Indicates the average response time of the slow requests to this transaction.	Secs	
Slow response threshold:	Indicates the threshold value configured for this transaction.	Secs	

Using the detailed diagnosis of the **Number of slow requests** measure you can know when the slow transactions occurred, identify the users who initiated the slow transactions, the IP address from which the transaction requests were received, and the URL that was accessed to execute the transaction. The response time registered by the transaction whenever it occurred will also be displayed, along with the data transmitted and received. With the help of this information, you can analyze why the transactions experienced slowdowns and initiate corrective action.

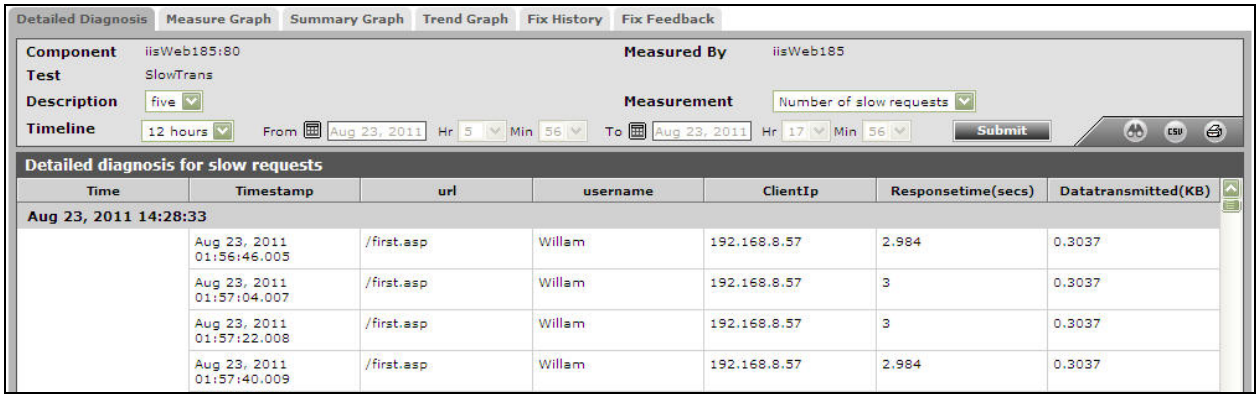


Figure 4.15: The detailed diagnosis of the Number of slow requests measure

Note:

Sometimes, the **Advanced Logging** module of IIS may not be able to log the username information related to web transactions. In such cases, no values will be displayed in the **username** column of the detailed diagnosis of the **IIS Web Transactions** test (see Figure 3.9). The workaround for this issue is provided below:

a. Login to the IIS host.

b. Open the **C:\Windows\System32\inetsrv\config\applicationHost.config** file in a text editor.

c. Locate the following line in the file:

```
<field id="UserName" sourceName="UserName" sourceType="RequestHeader"
logHeaderName="cs-username" category="Default" loggingDataType="TypeLPCSTR" />
```

d. Change the **sourceType** value to "BuiltIn" instead of "RequestHeader" as shown below:

```
<field id="UserName" sourceName="UserName" sourceType="BuiltIn" logHeaderName="cs-
username" category="Default" loggingDataType="TypeLPCSTR" />
```

e. Save the file.

4.3.9.1 Configuring Multiple URL Patterns for Monitoring

To enable administrators to easily configure the **Slow Transactions** test with multiple **NAME**s, **PATTERN**s, and **THRESHOLD VALUE**s, the eG administrative interface offers a special page. To access this page, you need to click on the **Click here** hyperlink, which will be available just about the parameters of this test in the **TEST CONFIGURATION** page (see Figure 4.16).

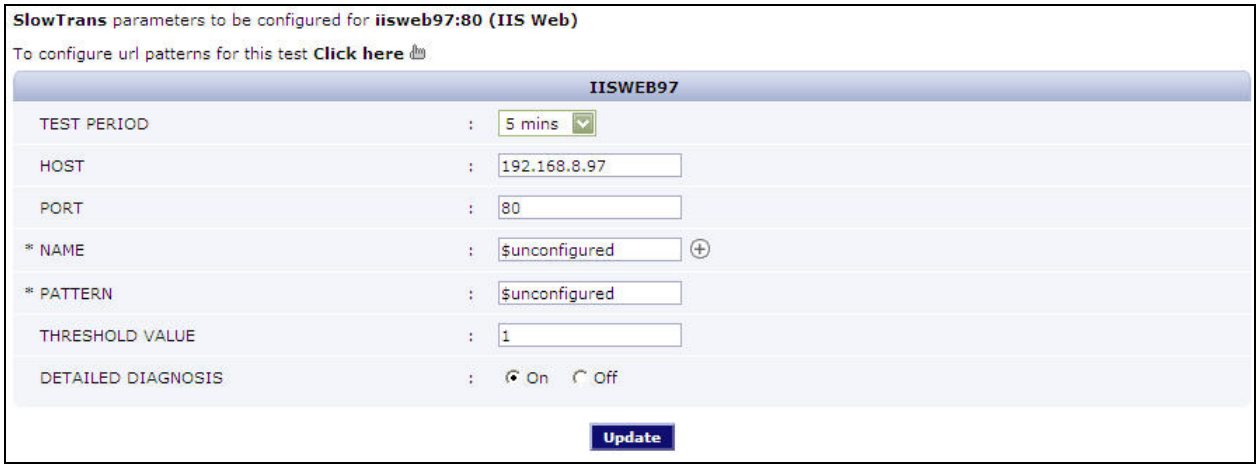


Figure 4.16: Configuring the Slow Transactions test

Doing so will invoke Figure 4.17, using which multiple URL patterns can be configured

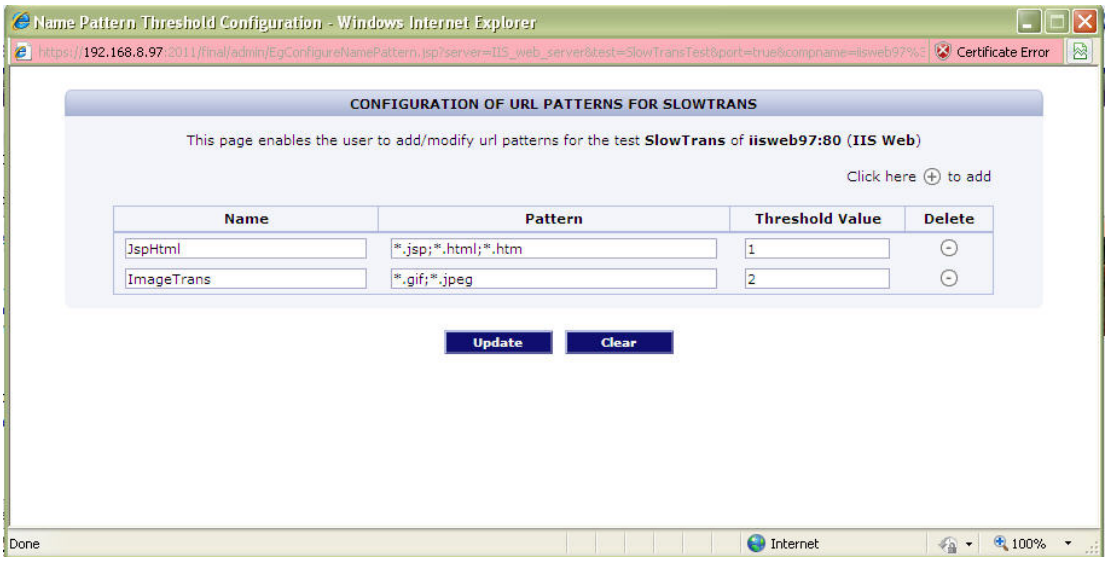


Figure 4.17: Configuring multiple URL patterns

To do so, follow the steps below:

a. Provide a display name for related/similar URL patterns in the **NAME** text box. This display name will appear as a descriptor of the test in the eG monitoring console.

b. In the **PATTERN** text box, you can configure the transaction patterns that are to be grouped under the specified **NAME**. Multiple transaction patterns can be separated by a semi-colon. Wild card characters can be used while configuring these patterns. For example, to monitor all transactions with the extension **html** or **jsp**, your **PATTERN** specification should be: **.html;*.jsp*.

c. In the **THRESHOLD VALUE** text box, specify the response time value (in seconds), which should be violated, for a transaction to be counted as a 'slow transaction'. All transaction patterns grouped under a given **NAME** will be governed by the **THRESHOLD VALUE** specified.

d. To add another specification, click on the encircled '+' button in Figure 4.17. This will invoke another row, where you can provide another **NAME**, associate a set of **PATTERN**s with that **NAME**, and configure a **THRESHOLD VALUE** for those patterns.

e. To delete a specification, simply click on the encircled '-' button at the end of every row in Figure 4.17.

f. To clear all your specifications at one shot, click on the **Clear** button in Figure 4.17.

g. To save the changes, click on the **Update** button.

h. You will then return to the **TEST CONFIGURATION PAGE**, where you can view all your specifications (see Figure 4.18).

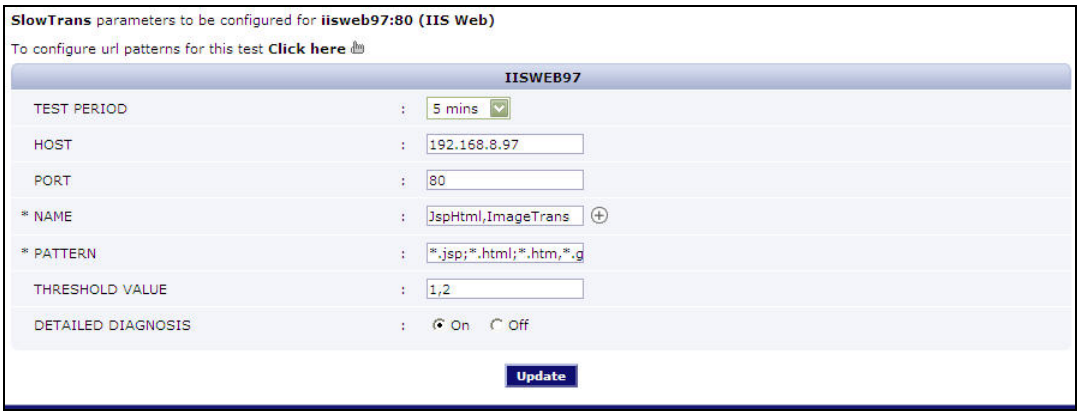


Figure 4.18: The Slow Transactions test configured ith multiple URL patterns

4.3.10 Windows Process Activation Service Test

The Windows Process Activation Service (WAS) manages application pool configuration and the creation and lifetime of worker processes for HTTP and other protocols. The World Wide Web Publishing Service (W3SVC) and other services depend on WAS.

This test monitors the Windows Process Activation Service and reports useful statistics revealing how well the service manages the worker processes and how healthy these worker processes are. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every worker process on the IIS web server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** - The host for which the test is to be configured
3. **PORT** - The port to which the specified **HOST** listens
4. **WEBSITE NAME** - By default, this parameter is set to *none*. This implies that the test monitors all web sites, by default. If you want the test to monitor a specific web site alone, then specify that web site name here.

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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Active listener channels:	Indicates the number of currently active listener channels in the worker process.	Number	
Active protocol handlers:	Indicates the number of currently active protocol handlers in the worker process.	Number	
Health ping reply latency:	Indicates the average time taken by worker process to reply to last health ping.	Millisecs	You can monitor and improve application pool health by having the Windows Process Activation Service (WAS) ping an application pool's worker process at set intervals. A delayed response from the worker process or the lack of any response might mean that the worker process does not have a thread to respond to the ping request, or that it is hanging for some other reason. Based on the results of the ping request, WAS can flag a worker process as unhealthy and shut it down. By default, worker process pinging is enabled. You may have to adjust the ping interval and the ping

Measurement	Description	Measurement Unit	Interpretation
			response time to gain access to timely information about application pool health without triggering false unhealthy conditions, for example, instability caused by an application.
Total requests served:	Indicates the total number of requests served by the worker process.	Number	This counter is only meaningful when request based recycling is enabled for the application pool.

4.3.11 Worker Processes Statistics Test

This test exposes how well the worker processes on an IIS web server are handling the HTTP requests to the server. Using the metrics reported you can understand the extent of load handled by the worker processes and can determine whether the worker processes have enough threads for processing requests.

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every worker process on the IIS web server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** - The host for which the test is to be configured
3. **PORT** - The port to which the specified **HOST** listens

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Active requests:	Indicates the current number of requests being processed by the	Number	

Measurement	Description	Measurement Unit	Interpretation
	worker processes.		
Active threads:	Indicates the number of threads actively processing requests in the worker process.	Number	
Current file cache memory usage:	Indicates the amount of memory used by user-mode file cache.	MB	
Current files cached:	Indicates the current number of files whose contents are present in the user- mode file cache.	Number	
Requests rate:	Indicates the number of HTTP requests/sec being processed by the worker process.	Requests/Sec	
Total requests served:	Indicates the total number of HTTP requests served by the worker process.	Number	
Total threads:	Indicates the total number of threads available to process requests in the worker process.	Number	By closely monitoring the thread usage over time, you can proactively capture when the worker process runs out of idle threads, and promptly take corrective measures, so as to avert any processing bottlenecks.

4.3.12 ASP Test

The ASP test monitors the performance of an IIS web server for servicing requests to Active Server Pages (ASPs). This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the desired

Component type, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every IIS web server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST -** The host for which the test is to be configured
3. **PORT -** The port to which the specified **HOST** listens

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
ASP requests:	Indicates the rate of requests for ASP pages during the last measurement period.	Reqs/Sec	
Transactions for ASP pages:	Indicates the rate of transactions started per second	Trans/Sec	
Error rate:	Indicates the rate of errors during ASP processing.	Errors/Sec	
Data received:	Indicates the rate of data being received by the web server for ASP processing,	Kbytes/Sec	
Data transmitted:	Indicates the rate of data being transmitted by the web server in response to ASP requests during	Kbytes/Sec	

Measurement	Description	Measurement Unit	Interpretation
	the last measurement period.		
Execution time:	Indicates the execution time of the last request in seconds. To determine where the bottleneck is, compare the execution time with the wait time.	Secs	If the wait time is low, but execution time is high, this indicates that the application logic in the ASP pages could be causing the high execution time. On the other hand, if wait time is high and execution time is low, this indicates a server-side bottleneck (e.g., due to too few processing threads).
Wait time for ASP execution:	Indicates the amount of time that the last request had to wait for service.	Secs	During ideal operation, the wait time should be near 0. If this value is high, it indicates that many requests are being queued for ASP processing.
Current executions:	Indicates the number of requests being currently executed by the ASP engine.	Number	
Current queue length:	Indicates the number of requests that are currently waiting for service.	Number	Ideally, this metric should be 0. A consistent non-zero value is an indicator of a server-side processing bottleneck.
Failed requests:	Indicates the number of requests that failed during the last measurement period. This includes authorization failures and rejections.	Number	
Rejected requests:	Indicates the number of requests that were	Number	

Measurement	Description	Measurement Unit	Interpretation
	rejected during the last measurement period.		
Pending transactions:	Indicates the number of transactions awaiting processing during the last measurement period.	Number	
Transaction commits:	Indicates the number of transactions that were committed during the last measurement period.	Number	
Transaction aborts:	Indicates the number of transactions that were aborted during the last measurement period.	Number	A very high value of this measure could indicate a server-processing bottleneck.

4.3.13 W3 WP Pools Test

Unlike IIS on Windows 2000, where all web sites operate within a logical default application pool, multiple application pools can be physically configured on IIS web servers on Windows 2003. Typically, each of these application pools will contain a web site. Problems in a web site will therefore, adversely impact the performance of the corresponding application pool, sometimes causing the pool to crash. In order to prevent such adversities, you can use the W3WPPool test to closely observe application pool performance (for IIS web servers on Windows 2003) in terms of resource usage, and promptly report abnormalities (if any). Using the statistics gathered by this test, you not only get to identify the problematic application pool, but also the erroneous web site.

This test is disabled by default, and can be enabled when monitoring an IIS web server on Windows 2003. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every web site monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed .
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.
4. **APPLICATION POOL NAME** - Provide a comma-separated list of application pools to be monitored. The default value is "all", indicating that all available application pools will be monitored by default. To know the application pools on the IIS web server in question, do the following:
   - On the IIS host, follow the menu sequence: Start -> Programs -> Administrative Tools -> Internet Information Services (IIS) Manager.
   - When the IIS Manager opens, expand the node that corresponds to the local host in the tree-structure in the left pane of the manager.
   - An Application Pools sub-node will appear. Expand this sub-node to view the complete list of application pools on the IIS web server.
5. 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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of processes:	Indicates number of w3wp.exe processes that are currently	Number	A zero value in this measure indicates that the application pool is crashed.

Measurement	Description	Measurement Unit	Interpretation
	running in the application pool.		
CPU utilization:	Indicates the percentage CPU utilization of the application pool.	Percent	A higher value indicates excessive CPU utilization.
Memory utilization:	Indicates the percentage of total memory utilized by the application pool.	Percent	If the value of this measure consistently increases, it indicates a memory bottleneck.
Number of threads:	Indicates the number of threads that are currently active in the application pool.	Number	This is an indicator of the workload on the pool.
Number of handles:	Indicates the number of handles currently opened by the application pool.	Number	
Page faults:	Indicates the rate of page faults happening in the application pool.	Faults/Sec	
Private data:	Indicates the amount of data that the application pool has currently allocated and cannot be shared with other application pools.	MB	A gradual growth in this measure indicates a memory leak in the websites that are running in the application pool.
Pool paged data:	Indicates the amount of memory currently allocated from the paged pool. The Paged Pool is an area of the	MB	A gradual growth in this measure indicates a memory leak in the websites that are running in the application pool.

Measurement	Description	Measurement Unit	Interpretation
	System's virtual memory that is limited in size and used for various system related functions.		
Pool non paged data:	Indicates the amount of memory currently allocated from the non-paged pool. The Non-Paged Pool is an area of the System's virtual memory that is limited in size and used for kernel and device driver functions.	MB	A gradual growth in this measure indicates a memory leak in the websites that are running in the application pool. Running out of space in the nonpaged pool often results in a Blue Screen.
I/O reads:	Indicates the rate at which the W3WP.exe process is reading data from I/O operations.	KBytes/Sec	
I/O read operations:	Indicates the rate at which the W3WP.exe process is issuing read I/O operations.	Operations/Sec	
I/O writes:	Indicates rate at which the W3WP.exe process is writing data into I/O operations.	KBytes/Sec	
I/O write operations:	Indicates the rate at which the W3WP.exe process is issuing write I/O operations	Operations/Sec	

4.3.14 Web Site Traffic Test

The **Web Site** layer executes a WebSiteTest that interfaces with the eG web adapter to collect statistics related to the usage of every web site configured on a web server. One of the key constraints of this test is that three critical usage metrics, namely, **Aborts**, **300 responses**, **500 errors**, will not be available for web servers executing on Windows environments. Therefore, for an IIS web server naturally, the eG agent will not be able to report these measures. In order to ensure that the above-mentioned error-related statistics are available to the IIS web servers also, eG Enterprise offers a separate **Web Site Traffic test**; this test executes only on IIS/IIS SSL web servers to monitor the traffic to and from each of the web sites on these servers, the errors that occurred during web site accesses, etc. This test reads the log files on the monitored IIS/IIS SSL web servers to extract important metrics pertaining to the web site usage. These log files are created only if **logging is explicity enabled** on the monitored IIS/IIS SSL web server. To enable logging, do the following:

1. On the IIS web server host, open the **Internet Information Services (IIS) Manager** using the following sequence of steps: Start -> Settings -> Control Panel -> Administrative Tools -> Internet Information Services (IIS) Manager.

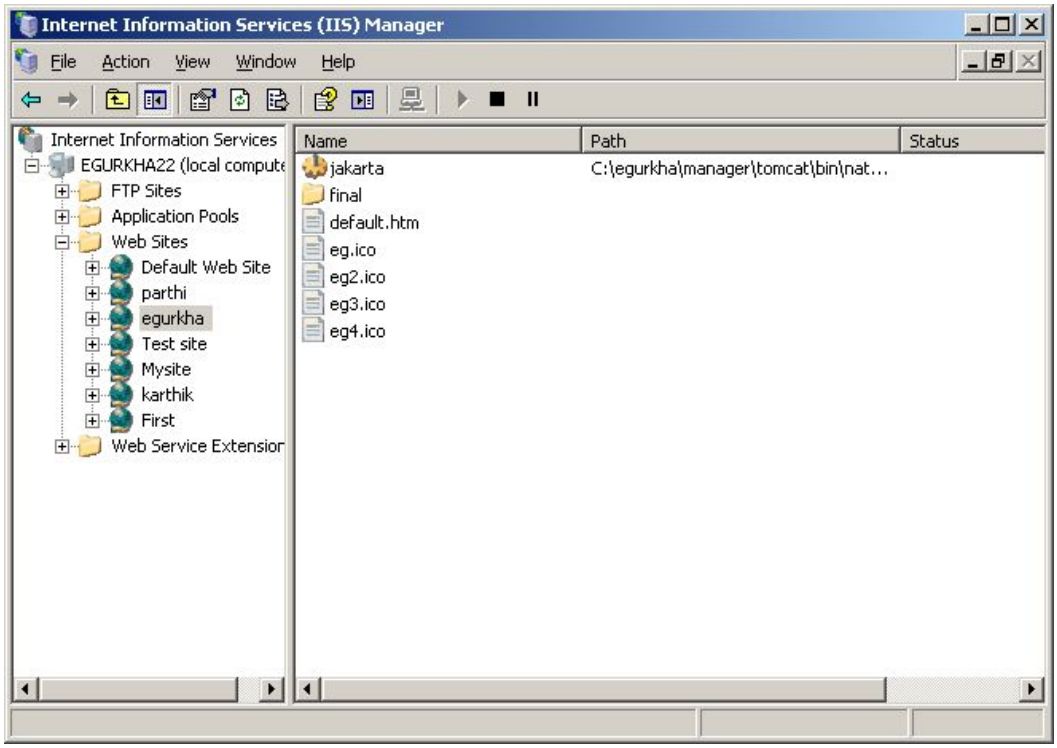


Figure 4.19: The IIS Manager console

2. In the tree-structure in the left pane of the console (see Figure 4.19), expand the node representing the local host, and then, the **Web Sites** node within; the complete list of web sites configured on the IIS web server will appear here.
3. To enable logging for a particular web site, select that web-site from under the **Web Sites** node, right-click on it, and then, select the **Properties** option from the shortcut menu (see Figure 4.20).

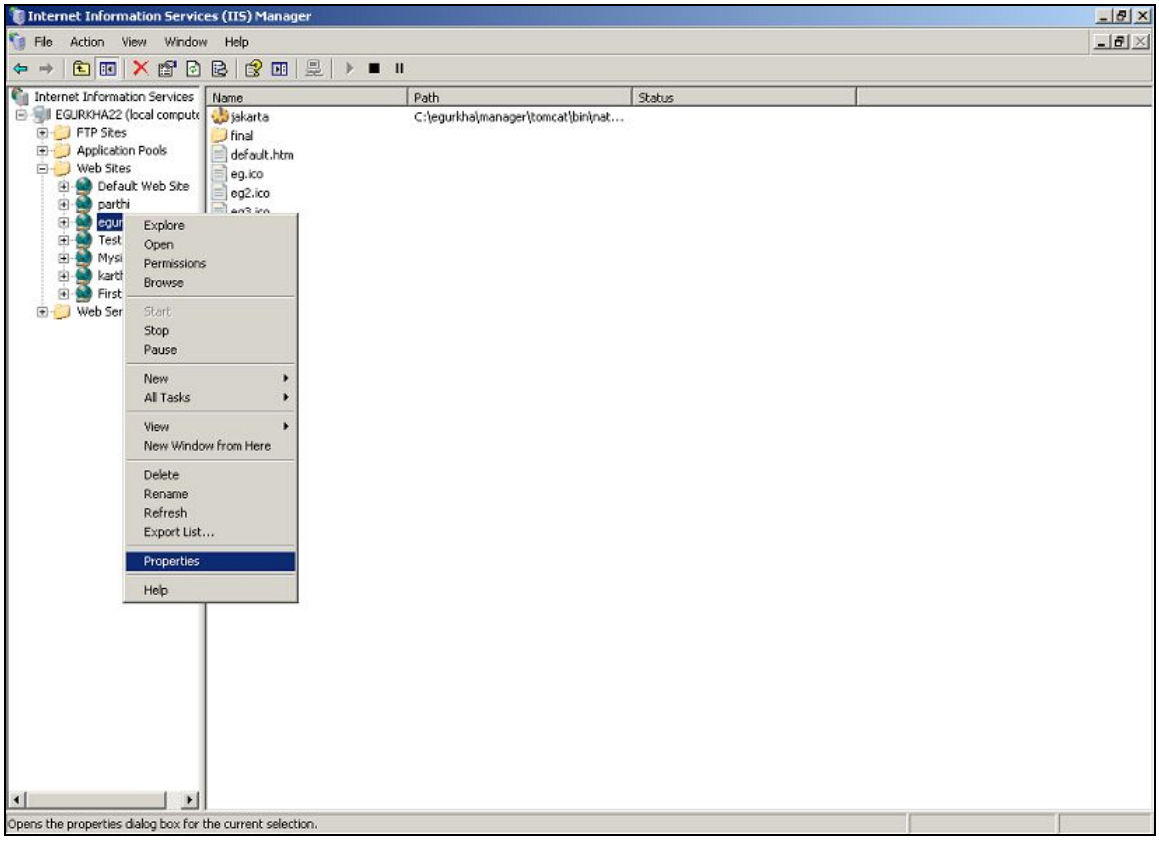


Figure 4.20: Selecting the Properties option of the web site

4. In the **Web Site** tab of the **Properties** dialog box that appears (see Figure 4.21), ensure that the **Enable Logging** check box is selected. Also, make sure that **W3C Extented Log File Format** is chosen as the **Active log format**.

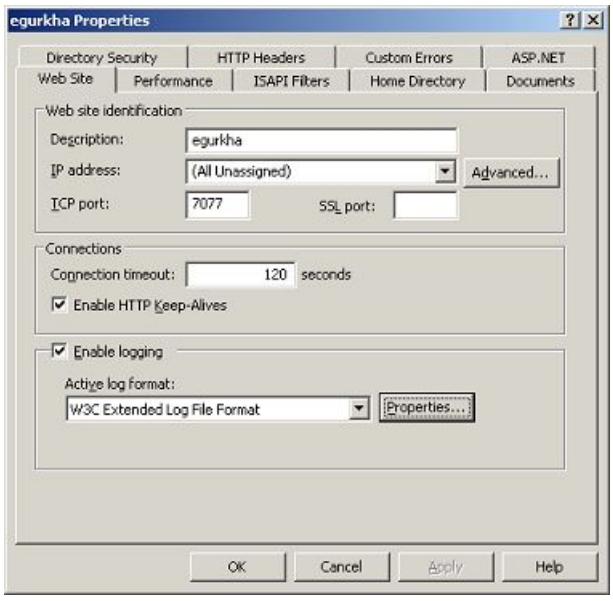


Figure 4.21: The Properties of a web site on a web server

5. Next, click on the **Properties** button adjacent to the **Active log format** chosen. Figure 4.22 then appears, displaying the **General** settings of the **W3C Extented Log File Format**. In the **Log file directory** text box here, specify the directory to which the log file should be written; by default, **C:\Windows\system32\LogFiles** is displayed therein.

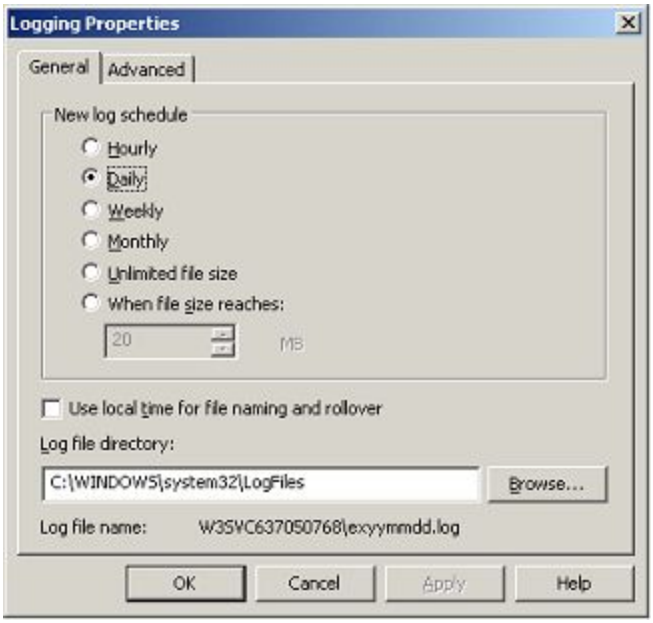


Figure 4.22: The General settings of the Active log format

6. Click on the **Advanced** tab in Figure 4.22 to indicate the type of statistics to be logged into the log file. In Figure 4.23 that appears, ensure that the **Win32 Status**, **Bytes Sent**, and **Bytes Received** check boxes are selected.

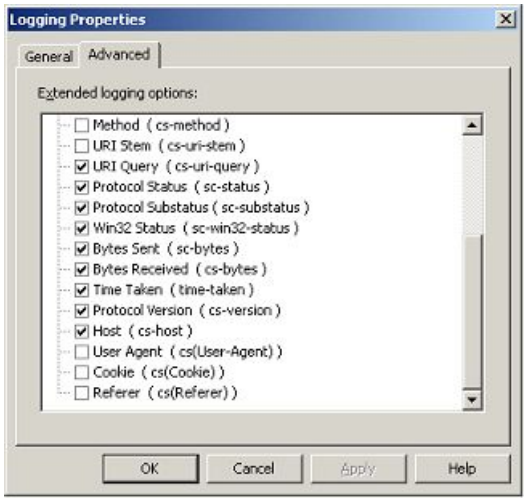


Figure 4.23: Selecting the information to be logged

7. Finally, click the **Apply** and **OK** buttons to register the changes.
8. Repeat this procedure for every web site to be monitored.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A web server instance

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every web site configured for monitoring on the target IIS/IIS SSL web server

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed .
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.
4. **LOGFILEPATH** - Specify the full path to directory on the target IIS/IIS SSL web server that

stores the log files; here, you need to specify the same directory that is provided in the **Log file directory** text box of Figure 4.22.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Requests handled	Indicates the number of requests currently received by this web site.	Number	
Data transmitted (MB)	Indicates the amount of data currently sent by this web site.	MB	A large increase in the data transmission can be indicative of an increase in the popularity of one or more web sites hosted on the server.
Data received (MB)	Indicates the amount of data currently received by this web site.	MB	An increase in this value is indicative of an increase in user requests to the server.
200 responses	Indicates the percentage of responses with the status code 200- 299 during the last measurement period.	Percent	Typically, successful requests are indicated using the response code 200 - 299.
300 responses	Percentage of responses with a status code in the 300- 399 range during the last measurement period.	Percent	300 responses could indicate page caching on the client browsers. Alternatively 300 responses could also indicate redirection of requests. A sudden change in this value could indicate a problem condition.
400 errors:	Percentage of responses with a status code in the range 400-499 during the last measurement period.	Percent	A high value indicates a number of missing/error pages.
500 errors	Percentage of	Percent	Since responses with a status

Measurement	Description	Measurement Unit	Interpretation
	responses with a status code in the range 500-599 during the last measurement period.		code of 500-600 indicate server side processing errors, a high value reflects an error condition.

4.4 The Web Site Layer

The **Web Site** layer is specific to each web site hosted on a web server. An internal WebSite test shown in Figure 4.24 provides a comprehensive view of the states of the individual web sites supported by a web server.

Figure 4.24: The WebSite test tracks the health of the Web Site layer

4.4.1 Web Site Test

For each web site configured for monitoring on the target web server, this test reports the load on the web site and how well the site processes the load.

Target of the test : A web site supported by a web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every web site monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed .
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Connections:	Rate of connections to the web site.	Conns/Sec	An increase or decrease in the connection rate can represent a change in the user workload.
Requests:	Rate of requests to the web site.	Reqs/Sec	With the advent of HTTP/1.1, multiple requests can be transmitted over the same TCP connection. The ratio of requests per connection can provide an idea of the effectiveness of the HTTP 1.1 protocol.
Data transmitted:	Rate at which the data is transmitted by the web site in response to user requests.	KB/Sec	A large increase in the data transmission rate can be indicative of an increase in the popularity of a web site hosted on the server.
Data received:	Rate at which the data is received by the web site.	KB/Sec	An increase in this value is indicative of an increase in user requests to the web site.
Errors:	Percentage of error responses from the web site.	Percent	Percentage of responses with a 400 or 500 status code.
400 errors:	Percentage of responses with a status code in the range 400-500.	Percent	An unexpected increase in the percentage of responses with status codes in the range 400-499 can indicate a sudden problem at the server.
Current requests:	Number of server threads/processes currently in use for serving requests for a web site (this measurement is not available for Apache web servers).	Number	A majority of the server threads/processes being used simultaneously to serve requests for a web site may be indicative of a server bottleneck caused by the web site.

4.5 The Web Transactions Layer

One of the unique capabilities of the eG Enterprise suite is its ability to monitor individual web transactions.

To monitor web site transactions to an IIS web server executing on Windows 2003, you need to do the following:

- Enable 'logging' on the target IIS web server;
- Modify the eG agent configuration to support web transaction monitoring

Both these procedures have been detailed in the *eG Installation Guide*.

To monitor web site transactions to the IIS web server executing on Windows 2008 (or above), you need to install and configure **Advanced Logging** on the target IIS web server, soon after configuring a **Web Server** role on the server. This procedure has also been detailed in the *eG Installation Guide*.

Figure 4.25: The tests that map to the Web Transactions layer of a web site.

4.5.1 Web Transactions Test

The Web Transactions test tracks the state of the individual transactions of a web site.

Note:

This test will not report metrics for web sites operating on IIS web server v8.x.

Target of the test : A web site supported by a web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for each transaction supported by a web site

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed .
2. **HOST** - The host for which the test is to be configured.
3. **PORT** - The port to which the specified **HOST** listens.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Requests:	Rate of requests for a specific transaction.	Reqs/Sec	An increase or decrease in request rate for a specific transaction can represent a change in the user workload.
Errors:	Percentage of error-filled responses from the web site for a specific transaction.	Percent	Percentage of responses for the transaction that report a 400 or 500 status code.
Data transmitted:	Rate at which the data is transmitted by the web site in response to user requests for a specific transaction.	KB/Sec	A large increase in the data transmission rate can be indicative of an increase in the popularity of a specific transaction. Alternatively, a sudden increase in data rate may also indicate a change in the characteristics of a transaction.
Avg response time:	The average time taken by the web site to respond to requests for a specific transaction, measured in seconds. Only requests for which successful responses are received are considered while computing the average response time.	Secs	An increase in response time is a major cause for user dissatisfaction with e- business sites. By correlating an increase in response time with the other metrics collected by the agent per transaction, an operator can diagnose the reason (s) for the deterioration in performance.

4.6 The .NET Transactions Layer

This layer will appear only if a .NET Profiler is installed and configured on the IIS web server to trace the path and monitor the performance of .NET business transactions.

A .NET Business Transactions test is mapped to this layer.

Figure 4.26: The test mapped to the .NET Transactions layer

To know more about the **.NET Business Transactions** test, refer to the *.NET Business Transaction Monitoring* document. The **IIS Web Transactions by Site** test helps administrators to track occurrence of any errors during performing transactions via each website.

4.6.1 IIS Web Transactions By Site Test

Whenever users encounter errors or complain of slowdowns during performing business transactions via your mission-critical web sites (on an IIS web server), administrators may want to know if the issue is caused from the web sites that perform the transactions. For this, administrators will have to periodically monitor the web sites, so as to quickly and accurately identify the web sites that process the requests slowly and return maximum number of erratic responses. The **IIS Web Transactions By Sites** test helps you achieve this.

This test auto-discovers the web sites on the IIS web server, and reports the rate at which each web site is receiving requests, the total number of requests processed and the number of requests that are successfully processed by each web site. In the process, this test also captures the response codes returned by each web site in real-time. By analyzing these codes, administrators can identify the web site that is encountering more number of errors which in turn will slow down the transactions.

Note:

This test will report metrics only when,

- Each web site configured on the IIS web server is logging in a separate log file, and
- File format of the log files is W3C.

Target of the test : An IIS web server

Agent deploying the test : An internal agent;

Outputs of the test : One set of results for every website monitored.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** - The host for which the test is to be configured
3. **PORT** - The port to which the specified **HOST** listens
4. **WEBSITE NAME** - By default, this parameter is set to *none*. This implies that the test monitors all web sites, by default. If you want the test to monitor a specific web site alone, then specify that web site name here.

   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.

5. **SHOW TOTAL REQUESTS DD** - By default, this flag is set to **No**. This implies that by default, detailed metrics will not be available for the *Total Requests processed* measure of this test. To enable detailed diagnosis for this measure, you can set this flag to **Yes**.
6. **SHOW SUCCESS REQUESTS DD** - By default, this flag is set to **No**. This implies that by default, detailed metrics will not be available for the *Requests processed successfully* measure of this test. To enable detailed diagnosis for this measure, you can set this flag to **Yes**.
7. **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.
8. **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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Requests	Indicates the rate of requests to this web site.	Requests/Sec	Compare the value of this measure across the web sites to identify the web site that is receiving more number of requests.
Errors	Indicates the percentage of error requests to this web site.	Percent	Compare the value of this measure across the web sites to identify the web site that is receiving more number of errors.
Aborts	Indicates the percentage of abort requests to this web site.	Percent	
Data transmitted	Indicates the rate at which the data is transmitted to this web site.	KB/sec	
Avg response time	Indicates the average time taken by this web site for responding to the requests it is receiving.	Seconds	Compare the value of this measure across the web sites to determine which web site is slow in responding to the requests.
Total requests processed	Indicates the total number of requests that are currently processed by this web site.	Number	This measure is a good indicator of current load on each web site. The detailed diagnosis of this measure reveals the following details for each web site: • IP address of the client • Name, IP address and port of the server • Site Name

Measurement	Description	Measurement Unit	Interpretation
			• Amount of data sent and received in KB • Time taken to process the requests in seconds • Host number • Protocol version • Method type • Http status • Win32 status • Time stamp at which the log was created • Date and time of local host • User Agent • Cookie • Referrer • Protocol SubStatus
Requests processed successfully	Indicates the number of requests that are successfully processed by this web site.	Number	Ideally, a high value is desired for this measure. This measure is a good indicator of current load on each website. The detailed diagnosis of this measure reveals the following details for each web site: • IP address of the client • Name, IP address and port of the server

Measurement	Description	Measurement Unit	Interpretation
			• Site Name • Amount of data sent and received in KB • Time taken to process the requests in seconds • Host number • Protocol version • Method type • Http status • Win32 status • Time stamp at which the log was created • Date and time of local host • User Agent • Cookie • Referrer • Protocol SubStatus
Total redirection requests	Indicates the total number of requests that are redirected from this web site.	Number	A site returns 300 series response codes when it redirects requests to other pages. Redirection is performed for various reasons. For example, when the client browser may have to request a different page on the server or to repeat the request by using a proxy server. These codes helps the client to ensure that a redirection to a different resource or URL should take place to

Measurement	Description	Measurement Unit	Interpretation
			complete the requests and access the desired resource.
Moved permanently	Indicates the number of requests that are permanently moved from this web site.	Number	When a request is permanently moved from the site, it returns the HTTP status code 301. This action is performed when the requested site has been permanently moved to a new URL.
Object moved	Indicates the number of requests that are temporarily moved from this web site.	Number	A 302 Found message is an HTTP response status code indicating that the requested resource has been temporarily moved to a different URL. Since the location or current redirection directive might be changed in the future, a client that receives a 302 response code should continue to use the original URL for future requests.
Not modified	Indicates the number of 304 redirection response codes returned by this web site.	Number	The HTTP 304 response code indicates that there is no need to retransmit the requested resources. It is an implicit redirection to a cached resource. This means that the requested resource is already in the cache and the resource has not been modified since it was cached. Therefore, the client can use the cached copy of the resource, instead of downloading it from the server. This happens when the request method is safe, like a GET or a HEAD request, or when the request is conditional and uses a If-None-Match or a If-Modified-Since header.
Temporary redirect	Indicates the number of 307 response codes currently returned by this web site.	Number	The HTTP 307 Temporary Redirect status response code indicates that the resource requested has been temporarily moved to the URL given

Measurement	Description	Measurement Unit	Interpretation
			by the Location headers.
Total 4xx requests	Represents the total number of HTTP 4xx (client error) codes currently returned by this web site.	Number	The 4xx HTTP status codes indicate that an error occurred and the client browser appears to be at fault. For example, the client browser may have requested a page that does not exist or may not have provided valid authentication information.
Bad requests	Indicates the number of HTTP 400 (bad requests) codes currently returned by this web site.	Number	The web site returns the HTTP 400 code when the request could not be understood by the server due to malformed syntax. This implies that the client should not repeat the request without modifications.
Total access denied requests	Indicates the total number of requests that are currently denied access to this web site.	Number	
Logon failed	Indicates the number of requests that are currently denied access to this web site due to login failure.	Number	The login failure error occurs when the logon attempt is unsuccessful, probably because of a user name or password that is not valid.
Logon failed due to server configuration	Indicates the number of login attempts to this web site failed due to lack of authentication to server configuration.	Number	Ideally, the value of this measure should be zero. The HTTP 401.2 status code indicates a problem in the authentication configuration settings on the server.
Authorization failed requests	Indicates the number of requests to this web site failed due to authorization failures.	Number	Ideally, the value of this measure should be zero. Authorization errors will be reported when the requests are not authenticated by ACL on resource or filter or ISAPI/CGI application.
Forbidden errors	Indicates the number of forbidden errors currently encountered by this web	Number	Ideally, the value of this measure should be zero.

Measurement	Description	Measurement Unit	Interpretation
	site.		The website encounters forbidden errors for various reasons, for example, when the website receives too many requests from the same client.
Not found errors	Indicates the number of 404 errors returned by this web site.	Number	Ideally, the value of this measure should be zero. The 404 error is reported when the server has not found anything matching the Request-URL. This error code is commonly used when the server does not wish to reveal exactly why the request has been refused, or when no other response is applicable.
Method not allowed	Indicates the number of 405 response codes returned by this web site.	Number	The 405 Method Not Allowed response code indicates that the method specified in the Request-Line is known by the origin server but is not supported by the target resource. To avoid this, administrators should include the response MUST an Allow header containing a list of valid methods for the requested resource.
Client browser not accept the MIME type	Indicates the number of 406 errors encountered by this web site.	Number	Ideally, the value of this measure should be zero. The HTTP 406 (Not Acceptable client) error code indicates that the server cannot produce a response matching the list of acceptable values defined in the request's proactive content negotiation headers, and that the server is unwilling to supply a default representation.
Request timed out	Indicates the number of 408 error codes returned by this web site.	Number	Ideally, the value of this measure should be zero. The 408 Request Timeout error

Measurement	Description	Measurement Unit	Interpretation
			indicates that the request sent to the website server (e.g. a request to load a web page) took longer than the website's server was prepared to wait. In other words, your connection with the website "timed out".
Precondition failed	Indicates the number of 412 errors encountered by this web site.	Number	Ideally, the value of this measure should be zero. The precondition given in one or more of the request-header fields evaluated to false when it was tested on the server. This response code allows the client to place preconditions on the current resource metainformation (header field data) and thus prevent the requested method from being applied to a resource other than the one intended.
Total internal server errors	Indicates the total number of server related errors encountered by this web site.	Number	Ideally, the value of this measure should be zero. The server error messages are reported for a wide variety of server-side errors. For more information on the errors, you can refer event viewer logs and find out the reason why the errors occur.
Module or ISAPI errors	Indicates the number of 500.0 errors encountered by this web site.	Number	Ideally, the value of this measure should be zero. The 500.0 error occurs when the ISAPIModule module is missing from the modules list for the website. ISAPI filters (modules) are libraries loaded by the IIS web server. Every incoming request and outgoing response passes through the filters, and they are free to perform any handling or translation they wish. They can be used for

Measurement	Description	Measurement Unit	Interpretation
			authentication, content transformation, logging, compression, and myriads of other uses.
Application is shutting down errors	Indicates the number of 500.11 errors encountered by this web site.	Number	Ideally, the value of this measure should be zero. The website reports 500.11 HTTP error code when an application on the server is shutting down, and thus the received request cannot be processed by the website.
Application is busy restarting errors	Indicates the number of 500.12 errors encountered by this web site.	Number	Ideally, the value of this measure should be zero. This error occurs when you tried to load an ASP page while IIS server was in the process of restarting the application. This error message will disappear when you refresh the page.
Web server busy errors	Indicates the number of 500.13 errors encountered by this web site.	Number	Ideally, the value of this measure should be zero.
Internal ASP errors	Indicates the number of 500.100 errors encountered by this web site.	Number	Ideally, the value of this measure should be zero. These errors occur when a dynamic-link library (DLL) that is required by the Microsoft Data Access Components is not registered. To alleviate this kind of errors, administrators should register the DLL before the errors cause serious impacts on the key transactions.
Service unavailable errors	Indicates the number of 503 errors encountered by this web site.	Number	Ideally, the value of this measure should be zero. The 503 errors occur when the application pool/service is stopped or mismatch in user identity settings is recorded. Administrators can fix these

Measurement	Description	Measurement Unit	Interpretation
			issues by restarting the stopped application pool/service and updating the user account settings.

Chapter 5: Troubleshooting

If the eG agent does not report any measures pertaining to the transactions that have been configured for an IIS web server, then restart the World Wide Web (WWW) Publishing service. To achieve this, do the following:

1. Select the **Services** option from the Start -> Programs -> Administrative Tools menu (see Figure 5.1).

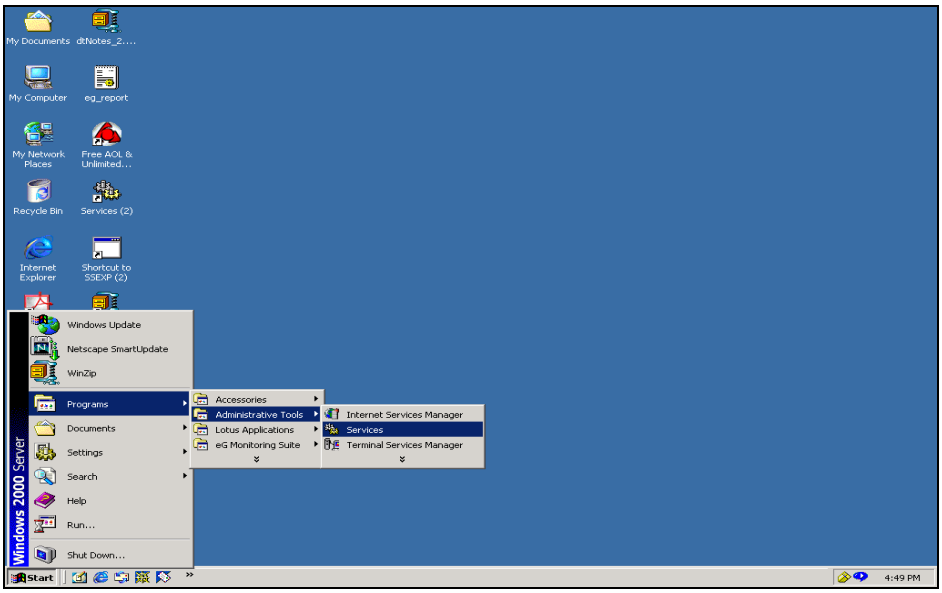


Figure 5.1: Selecting the Services option from the Administrative Tools menu

2. From the right pane of the window that appears, select **World Wide Web Publishing Service**, right-click on it, and then, select **Stop** from the shortcut menu that appears to stop the service (see Figure 5.2).

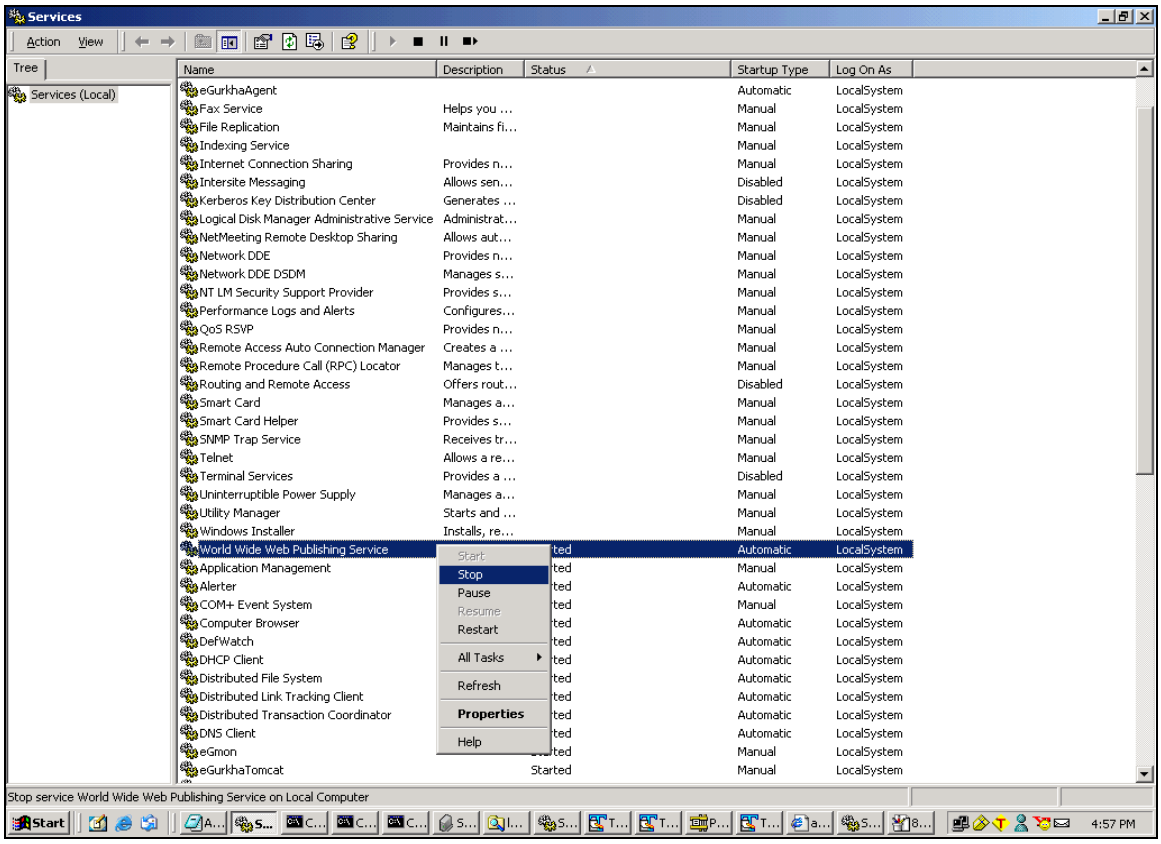


Figure 5.2: Stopping the World Wide Web Publishing Service

3. Now, to start the service, select it and right-click on it again. Then, from the shortcut menu, select **Start** (see Figure 5.3).

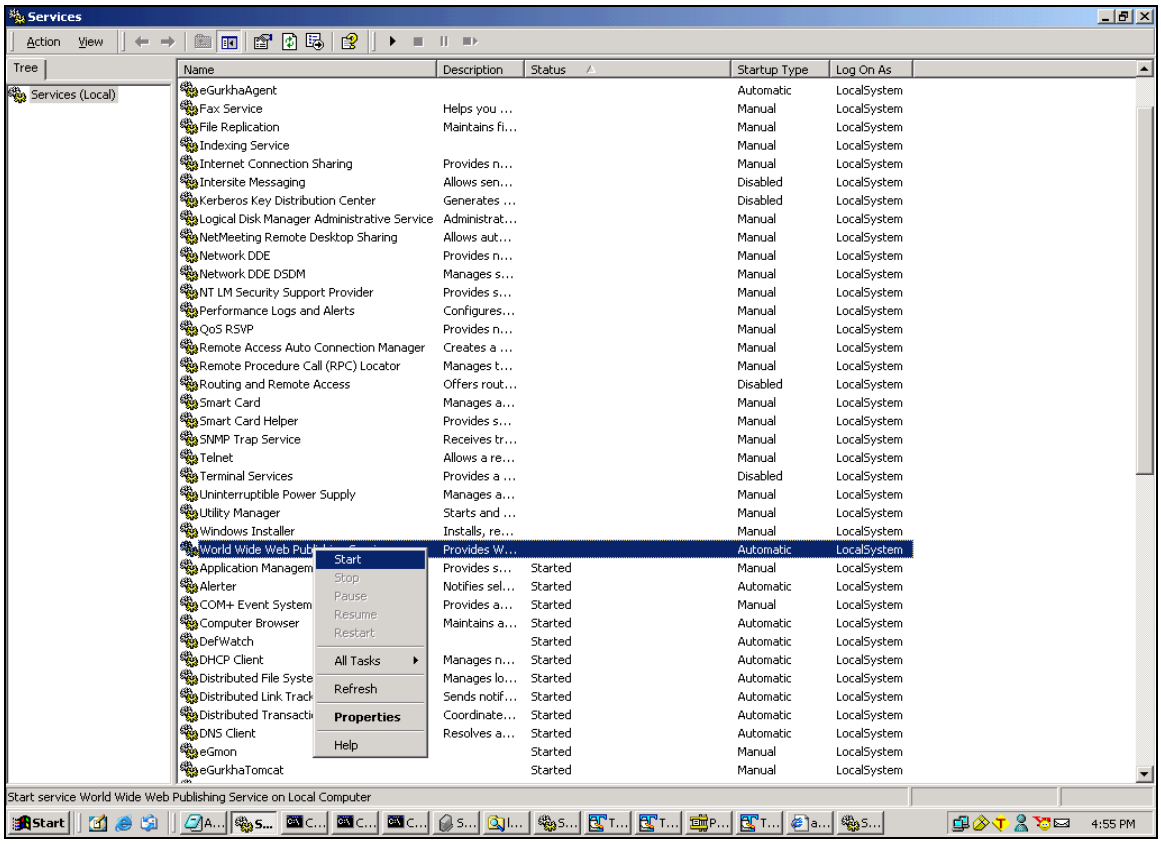


Figure 5.3: Starting the service

Now, log into the monitor interface to check whether the transactions are being monitored. If measures are still not been reported, then, do the following:

4. Select the **Internet Services Manager** option from the Start -> Programs -> Administrative Tools menu (see Figure 5.4).

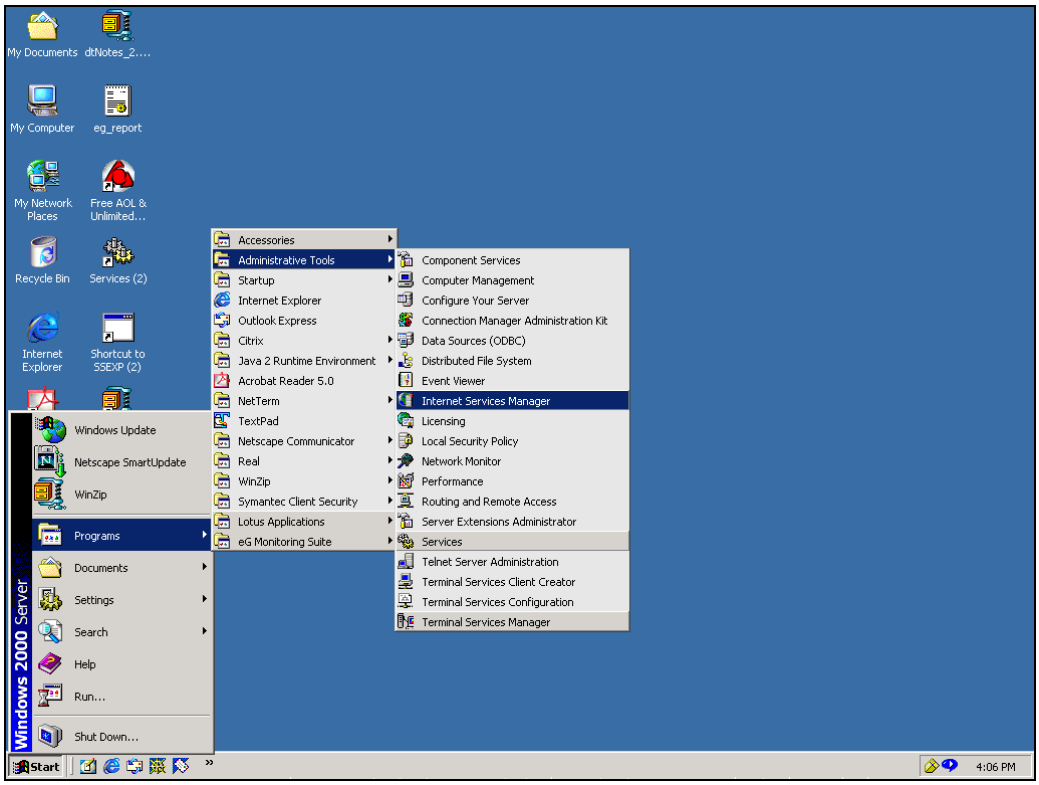


Figure 5.4: Selecting the Internet Services Manager option on Windows 2000

5. If the IIS web server executes on a Windows 2000 host, then, from the left pane of the **Internet Information Services** window that appears, select the IIS web server's host, right-click on it and choose the **Properties** option (see Figure 5.5). In case of the Windows 2003 host on the other hand, expand the node representing the IIS web server's host in the left pane, right-click on the **Web Sites** sub-node within, and pick the **Properties** option (see Figure 5.6).

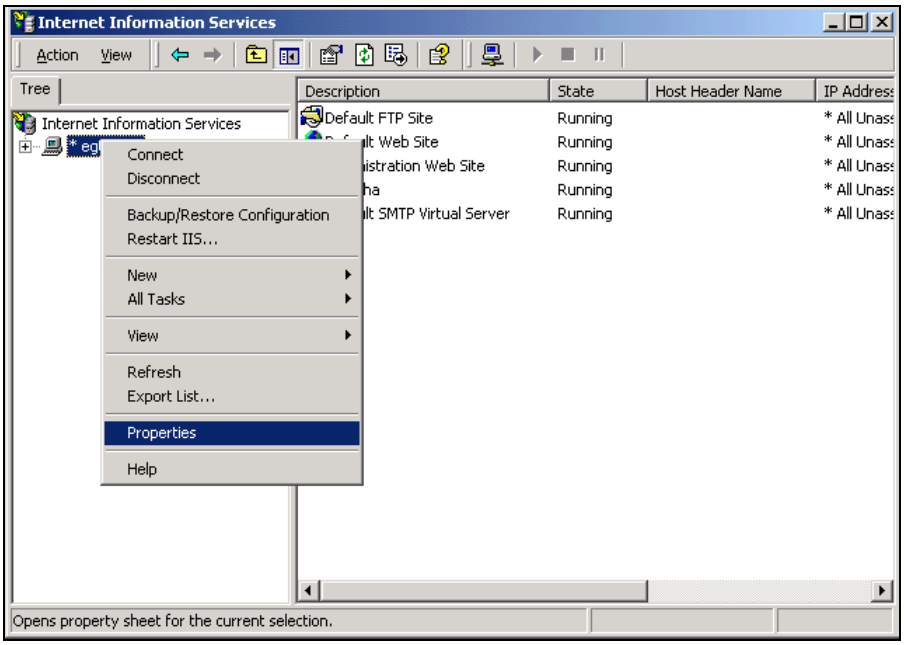


Figure 5.5: Editing the properties of the IIS web server's host (in IIS console on Windows 2000)

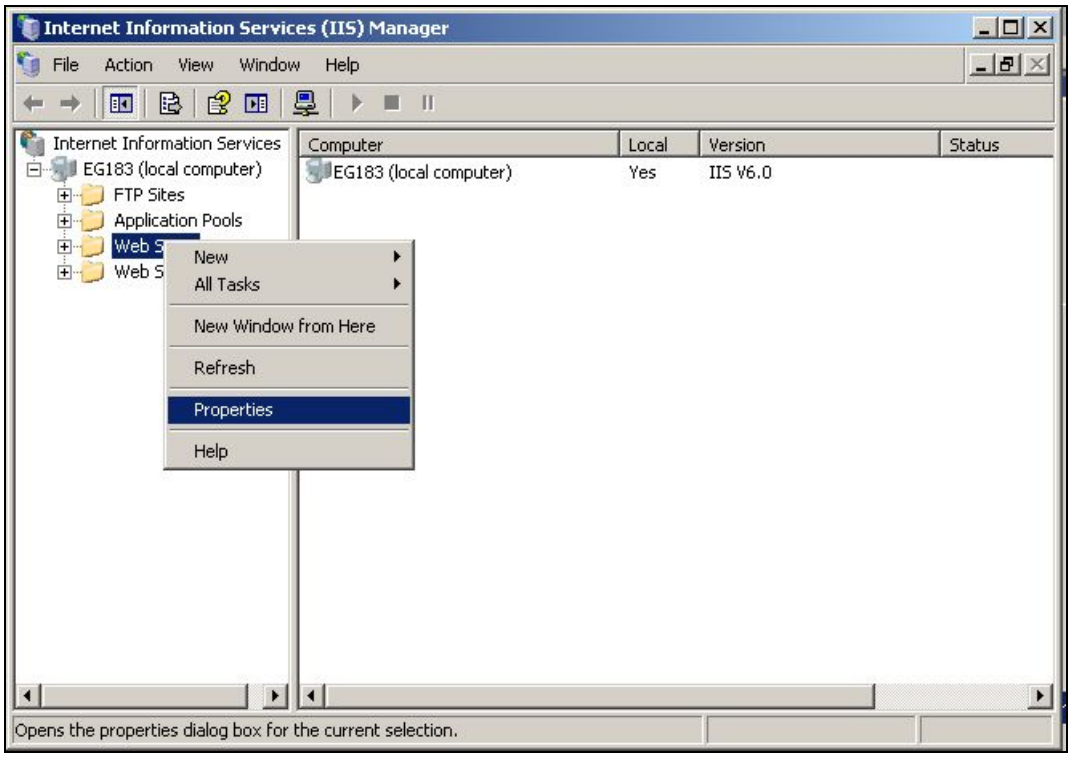


Figure 5.6: Picking the Properties option from the Web Sites tab (in the IIS console on Windows 2003)

6. On a Windows 2000 host, selecting the web server host's **Properties** will lead you to Figure 5.7.

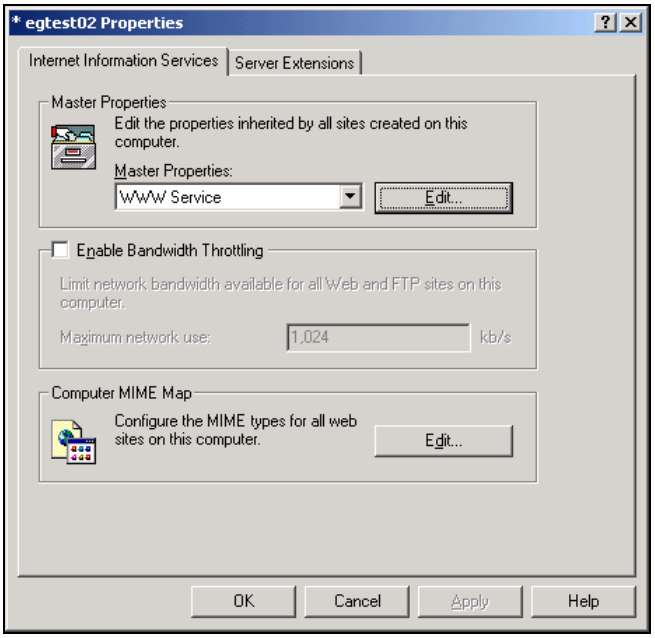


Figure 5.7: The Properties dialog box

As indicated by Figure 5.7, select **WWW Service** from the **Master Properties** list and click the **Edit** button to edit the properties of the selected service. Doing so will result in the display of a dialog box containing many tab pages. Click on the **ISAPI Filters** tab page (see Chapter 5).

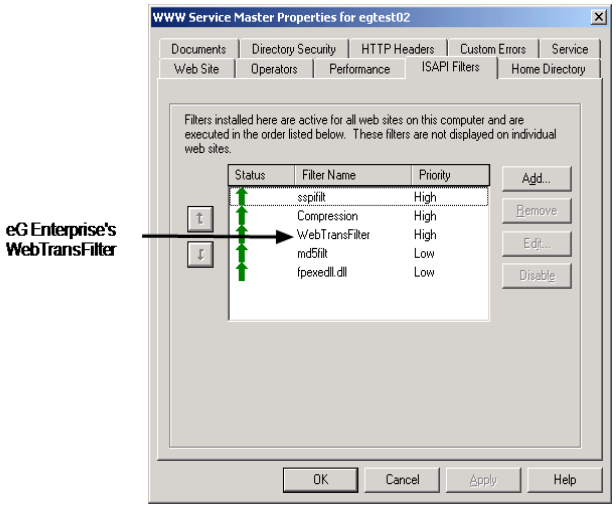


Figure 5.8: Viewing the status of the ISAPI filters

7. On a Windows 2003 host, selecting the **Properties** option of the **Web Sites** node will lead you

to a **Web Sites Properties** dialog box. Click on the **ISAPI Filters** tab page in that dialog box, and look for the **WebTransFilter** therein.

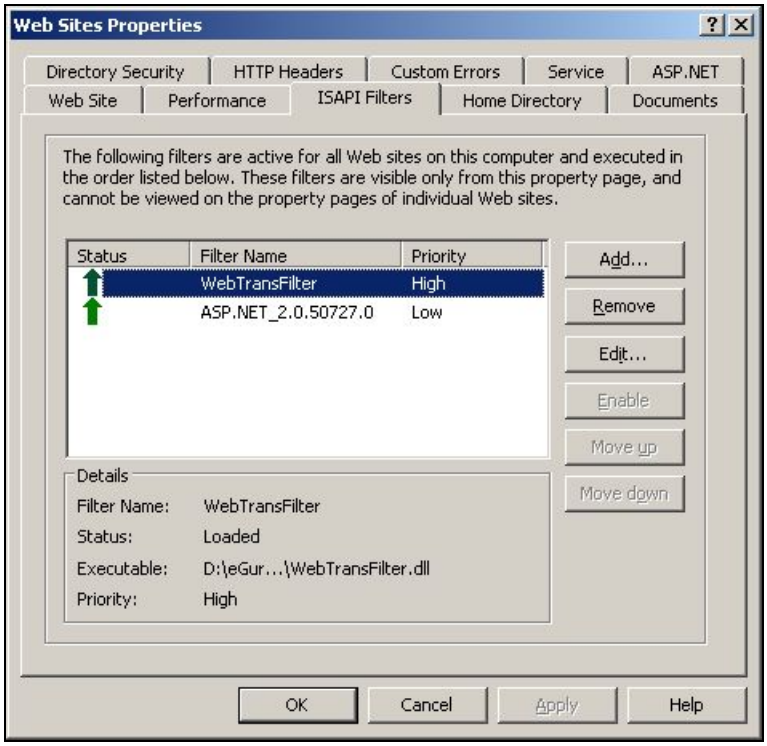


Figure 5.9: The Web Site Properties dialog box

8. Check the status of the WebTransFilter listed in the tab. Transaction monitoring in web servers is governed by this filter. The status of this filter has to be **GOOD** (indicated by an up arrow in green color) (see Figure 5.9), for the eG agent to perform transaction monitoring effectively. If the status of the filter is **BAD** (represented by a down arrow in red color) or **UNKNOWN** (indicated by a down arrow in blue color), then, you might have to reload the filter. For that, first, select the filter in Figure 5.9 and click the **Remove** button alongside it to remove it. Then, click the **Add** button. Doing so will result in the display of a screen wherein the **Filter Name** and the path to the filter **Executable** has to be specified (see Figure 5.10).

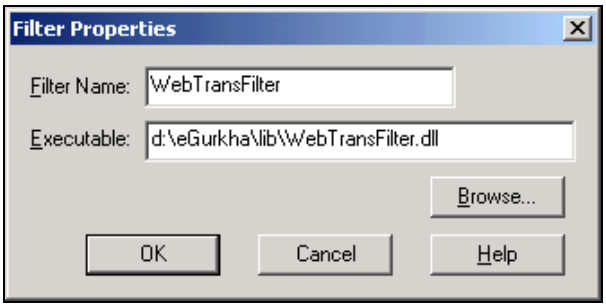


Figure 5.10: Adding the filter

9. The WebTrans filter will be available in the **<EG_HOME_DIR>/lib** directory. Specify the same against the **Executable** text box and then, click the **OK** button to register the changes.
10. Clicking on the **OK** button will take you back to the dialog box depicted by Figure 5.9. Click on the **OK** button in the dialog box and then, on the **OK** button in Figure 5.7.
11. Once the filter is loaded, restart the WWW service once again by following the procedure discussed previously.

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.