Monitoring Oracle WebLogic Application Server

eG Innovations Product Documentation

www.eginnovations.com

Total Performance Visibility

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.

Table of Contents

Table of Figures

Chapter 1: Introduction

BEA WebLogic Server is a fully-featured, standards-based application server providing the foundation on which an enterprise can build reliable, scalable, and manageable applications. With its comprehensive set of features, compliance with open standards, multi-tiered architecture, and support for component-based development, WebLogic Server provides the underlying core functionality necessary for the development and deployment of business-driven applications. Any issue with the functioning of the WebLogic server, if not troubleshooted on time, can rupture the very core of these business-critical applications, causing infrastructure downtime and huge revenue losses. This justifies the need for continuously monitoring the external availability and internal operations of the WebLogic server.

eG Enterprise provides two distinct models for monitoring WebLogic servers - the WebLogic model and the WebLogic (6/7/8) model. As the names suggest, the WebLogic 6/7/8 model can be used to monitor the WebLogic server version 6, 7, and 8, and the WebLogic model can be used for monitoring WebLogic 9.0 (and above).

Regardless of the model used, the metrics obtained enable administrators to find answers to the following persistent performance questions:

Server monitoring	• Is the WebLogic process running? • Is the memory usage of the server increasing over time? • Is the server's request processing rate unusually high?
JVM monitoring	• Is the JVM heap size adequate? • Is the garbage collection tuned well or is the JVM spending too much time in garbage collection?
Thread monitoring	• Are the WebLogic servers execute queues adequately sized? • Are there too many threads waiting to be serviced, thereby causing slow response time?
Security monitoring	• How many invalid login attempts have been made to the WebLogic server? • Are these attempts recurring?
JMS monitoring	• Are there many pending messages in the messaging server? • Is the message traffic unusually high?

Connector monitoring	• What is the usage pattern of connections in a connector pool?
Cluster monitoring	• Are all the WebLogic servers in the cluster currently available? • Is the load being balanced across the cluster?
Transaction monitoring	• How many user transactions are happening? • Are there too many rollbacks occurring?
Servlet monitoring	• Which servlet(s) are being extensively accessed? • What is the average invocation time for each servlet?
EJB Pool monitoring	• Are there adequate numbers of beans in a bean pool? • How many beans are in use? Are there any clients waiting for a bean?
EJB Cache monitoring	• Is the cache adequately sized or are there too many cache misses? • What is the rate of EJB activations and passivations?
EJB Lock monitoring	• Is there contention for locks? • How many beans are locked? • How many attempts have been made to acquire a lock for each bean?
JDBC Connection monitoring	• Are all the JDBC pools available? • Is each pool adequately sized? • What are the peak usage times and values? • How many connection leaks have occurred?
JDBC call monitoring	• How many JDBC calls have been made? • What was average response time of those calls? • What are the queries that take a long time to execute?

The sections that will follow discuss each of these models in great detail.

Chapter 2: How to Monitor Oracle WebLogic Application Server Using eG Enterprise?

This chapter elaborates on the procedures involved in configuring and monitoring an Oracle WebLogic Application server (for versions 6.x and higher). Configuring and Monitoring Oracle WebLogic Application Server Version 6/7/8

The sub-sections that follow will discuss how to configure and monitor the Oracle WebLogic servers ver. 6/7/8.

2.1 Pre-requisites for Monitoring Oracle WebLogic Application Server Version 6/7/8

To monitor the Oracle WebLogic Application Server Version 6/7/8, a set of pre-requisites should be fulfilled. These requirements are discussed in the following sections;

2.1.1 Configuring an Oracle WebLogic Application Server 6 (with SP 2) to work with the eG Agent

The eG agent package contains a web archive file called **egurkha_6_2.war**. On Unix environments, this file will be available in the /opt/egurkha/lib location. On Windows environment, this file will be available in the <EG_HOME_DIR>\lib directory. This file has to be installed on the WebLogic 6.0 server with SP2, which has to be monitored. Prior to deploying this war file, make sure that you follow the steps below:

- Login to the system hosting the eG agent that is monitoring the WebLogic server.
- Copy the **egurkha_6_2.war** file from the /opt/egurkha/lib directory to any other directory on the eG agent host.
- Rename the war file as **egurkha.war**.
- Next, follow the steps given below to deploy the war file on the WebLogic server version 6.x:

1. Go to the following URL: *http:*\\<*WebLogic_server_IP*>:<*WebLogicPort*>/*console*.
2. Now, an authentication dialog (see Figure 2.1) that requires a username and password, will appear. By default, the username is system. Specify the same password you gave during the installation of the Weblogic server. If your environment is configured differently, please contact your WebLogic administrator to find out the username and password to use to login to the WebLogic console.

Figure 2.1: Specifying user name and password in the authentication dialog box

3. Upon successful authentication, the following WebLogic 6.1 Administration console will appear (see Figure 2.2).

Figure 2.2: The Administration Console of the ORACLE WebLogic server 6.x

- From the right panel, choose the **Install Applications** link under the **Getting Started** group. The following screen will then appear:

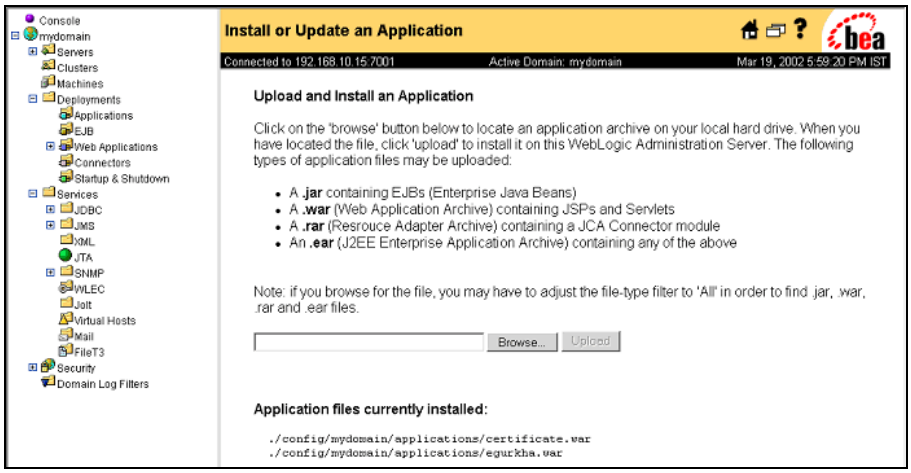


Figure 2.3: Screen where you can install or update an Application

5. Alternately, the following procedure can also be adopted to access the screen above.

   - Expand the **Deployments** node in the tree-structure in the left panel, and then, select **Web Applications**. A screen depicted by Figure 2.4 below will come up:

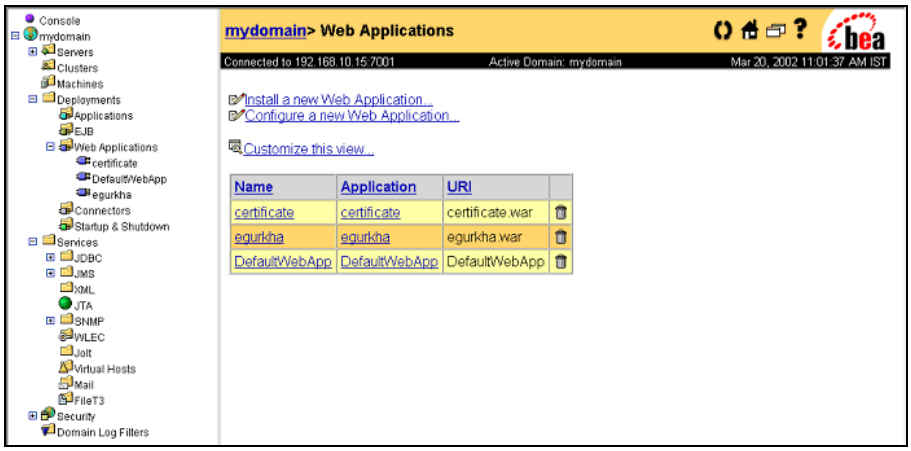


Figure 2.4: Viewing the Web Applications that have been deployed

7. From the right panel, choose the **Install a new Web Application** option, which will bring the screen represented by Figure 2.3 to light.

8. Next, using the **Browse** button in the right panel, select the **egurkha.war** file (see Figure 2.5) from the local directory in which it resides and then, click on the **Upload** button.

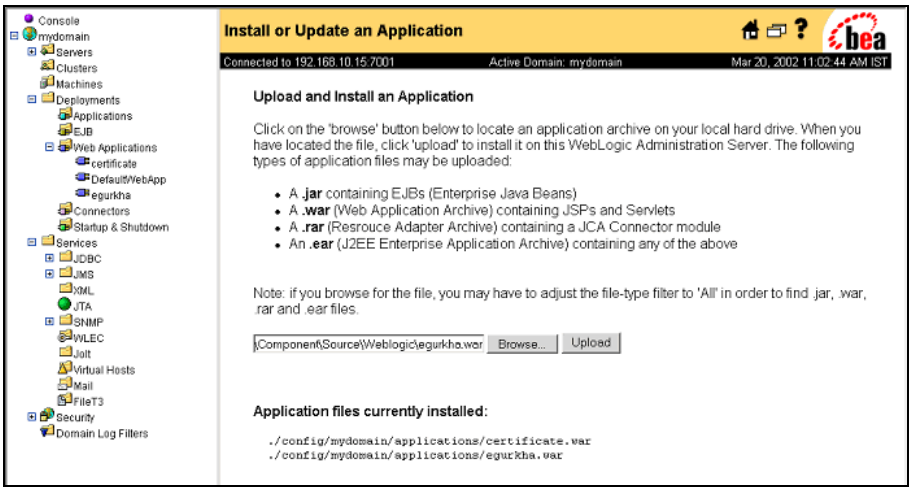


Figure 2.5: Specifying the location of the "egurkha.war" file

Note:

By default, the **egurkha.war** file will be searched for in the local disk of the system from which the browser is launched. Therefore, if you are trying to access the WebLogic console from a system that does not host an eG agent, then, before attempting to deploy the **egurkha.war** file, do the following:

- Login to any agent host in your environment.
- Copy the **egurkha_wl6_sp2.war** file in the <EG_AGNT_INSTALL_DIR>\lib directory (on Windows; on Unix, this will be the /optegurkha/lib directory) of the agent host to any other directory on that host.
- Rename the **egurkha_wl6_sp2.war** file as **egurkha.war** file.
- Copy the **egurkha.war** file to the local host from which the browser is launched.

9. Doing so will reveal the following screen, which informs you that the file has been successfully uploaded.

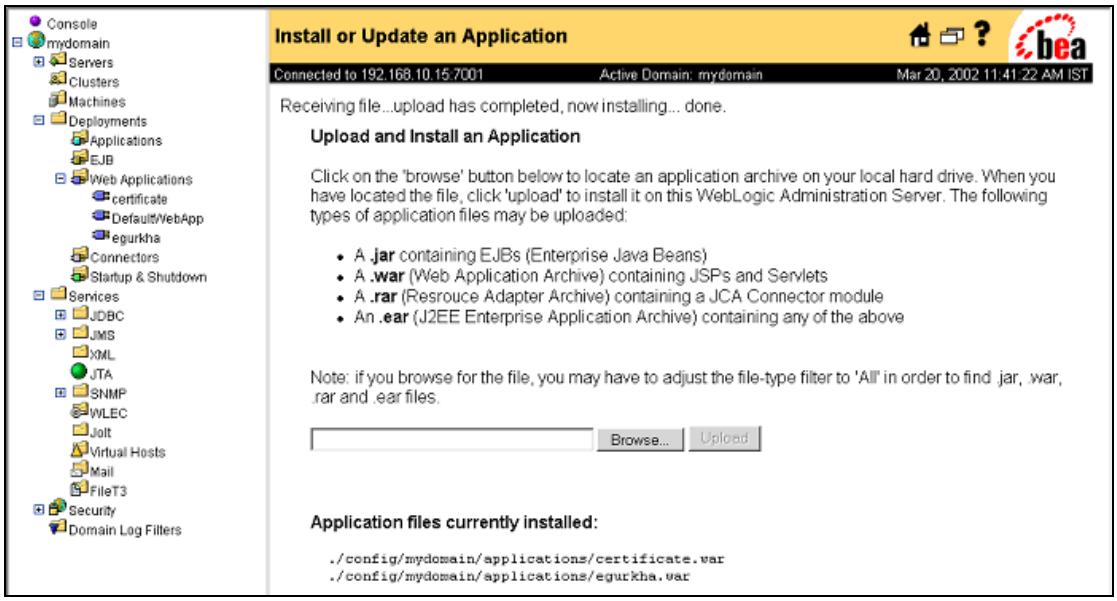


Figure 2.6: Screen informing you that egurkha.war file has been successfully uploaded

10. Once this is done, eG would appear as a new application under the Web Applications category. Now, the WebLogic server is ready to be used with eG.

11. If multiple instances of WebLogic are executing, then you can associate the **egurkha** application with specific instances by first selecting the **egurkha** application from the tree structure in the left pane of Figure 2.6. From the tabs listed in the right pane, select the **Targets** tab. The **Available** list in Figure 2.7 will display all the existing WebLogic instances that are available for selection, and the **Chosen** list will display the WebLogic server instances that have already been chosen for association. To associate the **egurkha** application with a WebLogic server instance, select an instance from the **Available** list and click on the -> button therein. The selection will automatically move to the **Chosen** list. Similarly, multiple instances can be added to the **Chosen** list. Finally, click on the **Apply** button to apply the changes.

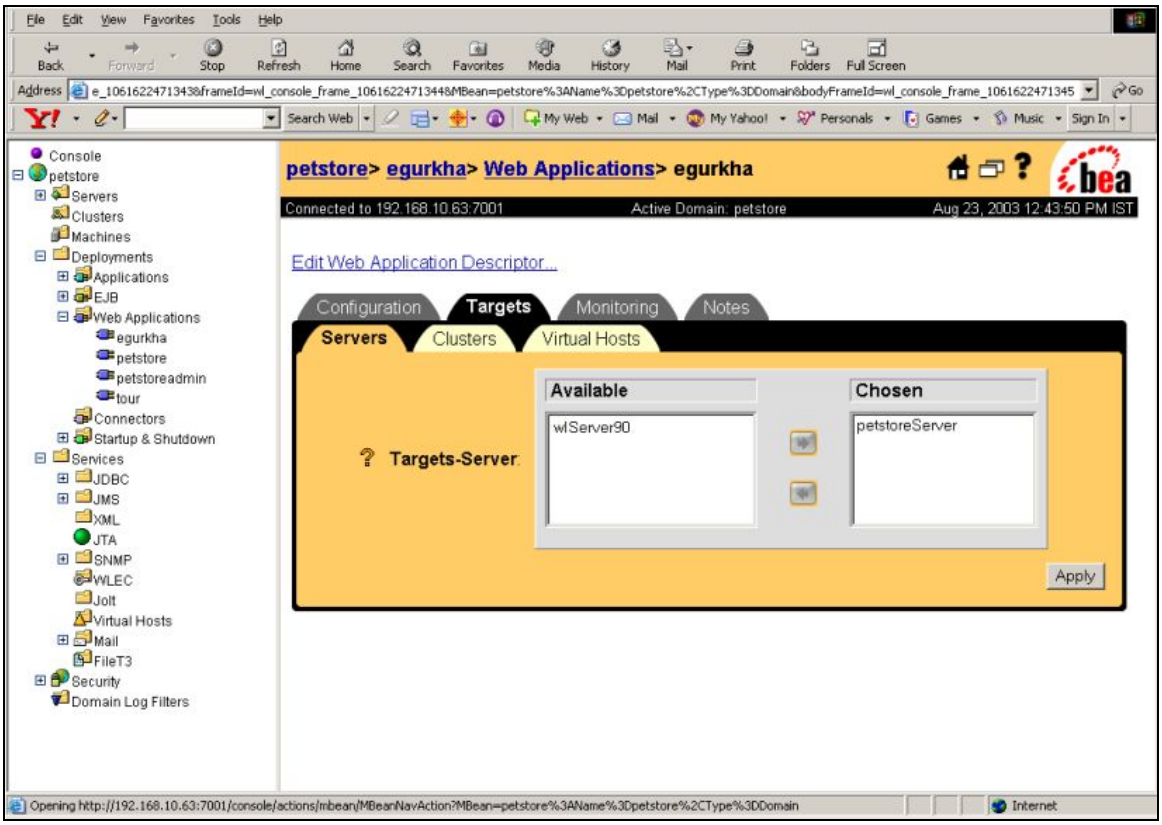


Figure 2.7: Associating the egurkha application with specific targets

2.1.2 Configuring a ORACLE WebLogic Application Server 6.x or WebLogic Server 7.x

The eG agent package contains a web archive file called **egurkha7x.war**. On Unix environments, this file will be available in the **/opt/egurkha/lib** location. On Windows environment, this file will be available in the **<EG_HO8ME_DIR>\lib directory**. Prior to monitoring a WebLogic Server 6.x or a WebLogic Server 7.x, make sure that this file is deployed on the target WebLogic server to enable the eG agent to monitor it.

Before deploying the war file, make sure that you follow the steps below:

- Login to the system hosting the eG agent that is monitoring the WebLogic server 6.x/7.x.
- Copy the **egurkha7x.**war file from the /opt/egurkha/lib directory to any other directory on the eG agent host.
- Rename the war file as **egurkha.war**.

You can deploy the **egurkha.war** file on WebLogic Server 6.x by following the instructions discussed in Section **2.1.1** above. To deploy the egurkha7x.war file on a ORACLE WebLogic application server 7.x that requires monitoring, do the following:

1. Go to the following URL: *http:*\\<*WebLogic_server_IP*>:<*WebLogicPort*>/*console*
2. Now, an authentication dialog (see Figure 2.8) that requires a username and password, will appear. Specify the same password you gave during the installation of the Weblogic server. If your environment is configured differently, please contact your WebLogic administrator to find out the username and password to use to login to the WebLogic console.

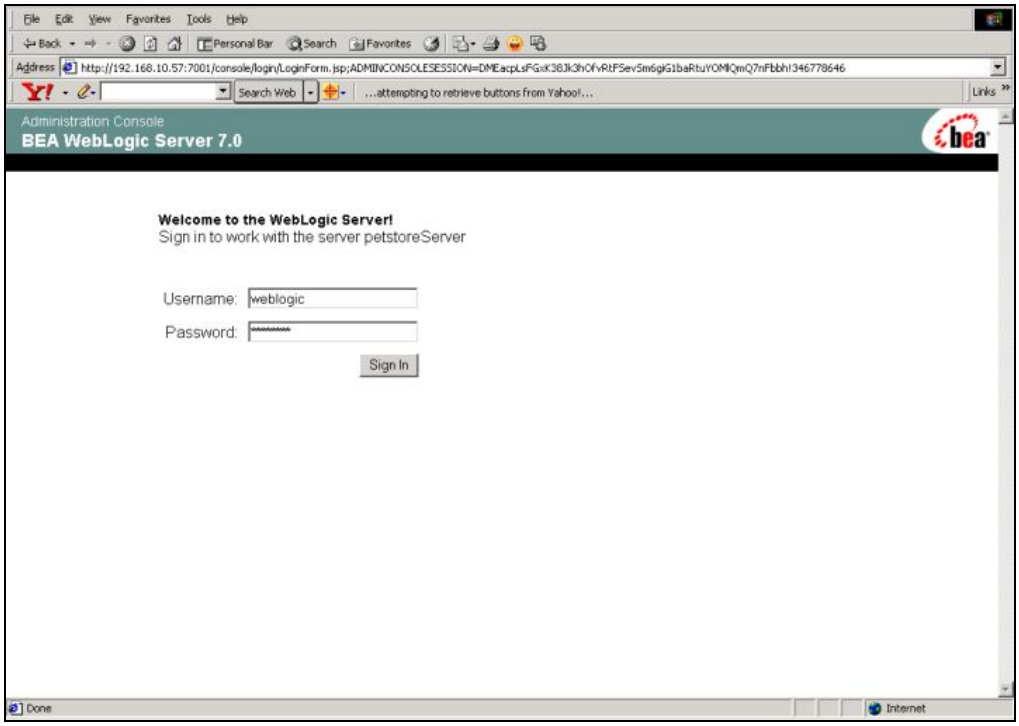


Figure 2.8: Specifying user name and password in the authentication dialog box

3. Upon successful authentication, the WebLogic 7.0 Administration console will appear (see Figure 2.10).

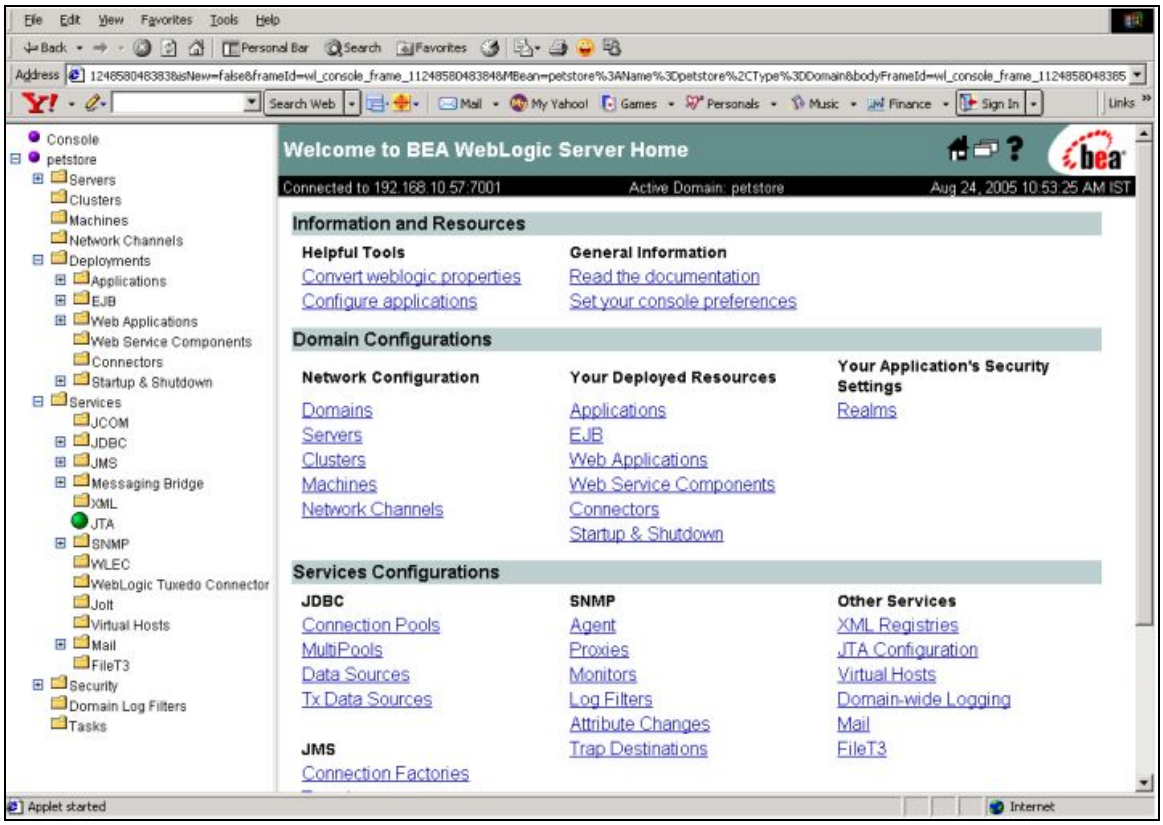


Figure 2.9: The Administration Console of the ORACLE WebLogic server 7.0

4. From the tree-structure in the left panel, choose the **Web Applications** node under the **Deployments** node. Then, click on the **Configure a new Web Application** link in the left pane to begin the deployment of the **egurkha.war** application (see Figure 2.10).

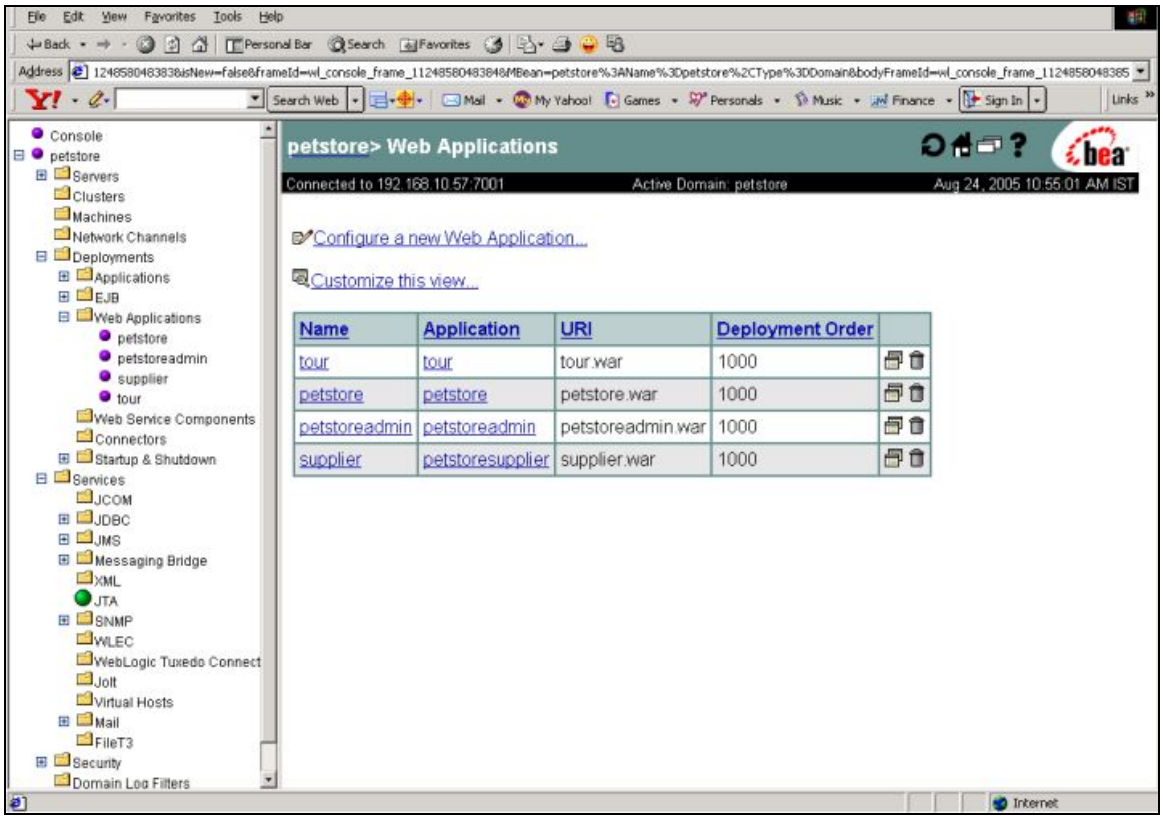


Figure 2.10: Screen where you can install or update an Application

5. Upon clicking the link, Figure 2.11 will appear, which will explain the procedure for application deployment. To proceed, click on the **upload it through the browser** link in the right panel of Figure 2.11.

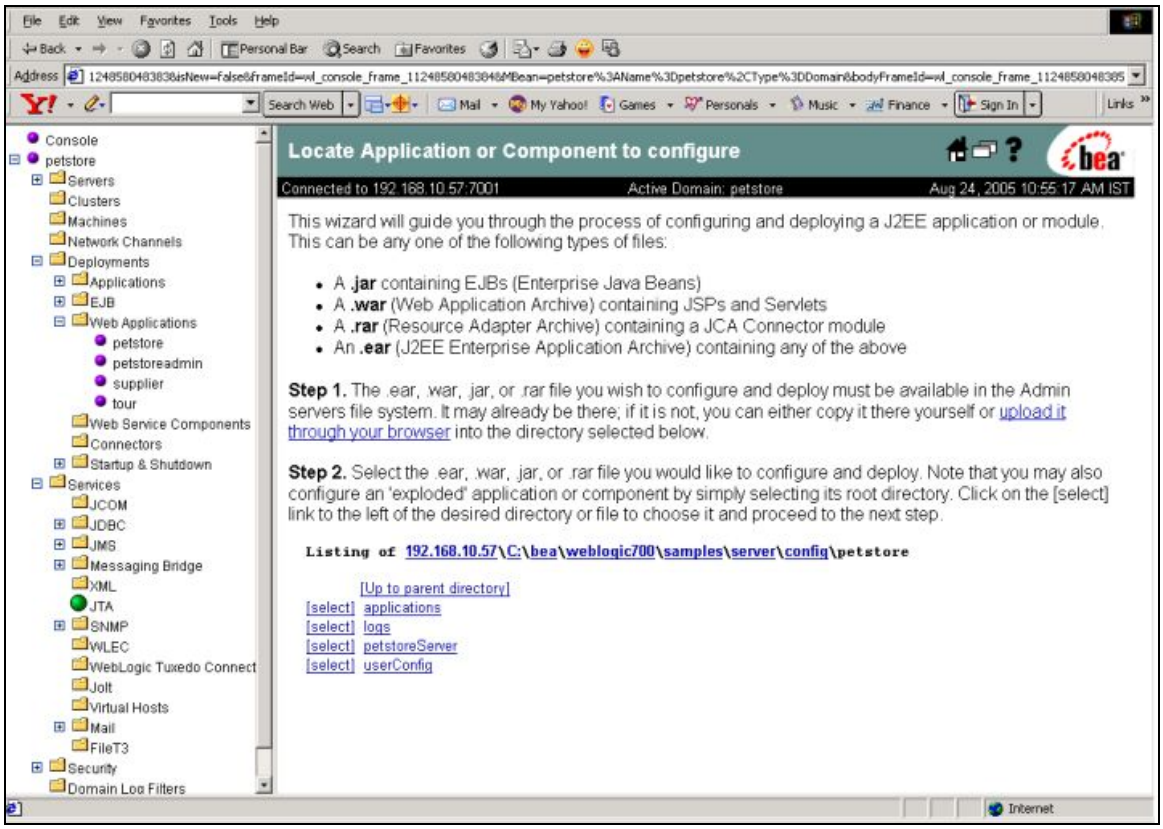


Figure 2.11: Attempting to upload the "egurkha.war" file

Note:

By default, the **egurkha.war** file will be searched for in the local disk of the system from which the browser is launched. Therefore, if you are trying to access the WebLogic console from a system that does not host an eG agent, then, before attempting to deploy the **egurkha.war** file, copy the war file from the <EG_HOME_DIR>\lib directory (in Windows environments. In Unix, this will be /opt/egurkha/lib) of an agent host, to your local disk.

6. In Figure 2.12 that appears, specify the full path to the **egurkha.war** file to be deployed on the WebLogic server. Alternatively, you can use the **Browse** button to locate the war file. Finally, click the **Upload** button in Figure 2.12 to upload the **war** file.

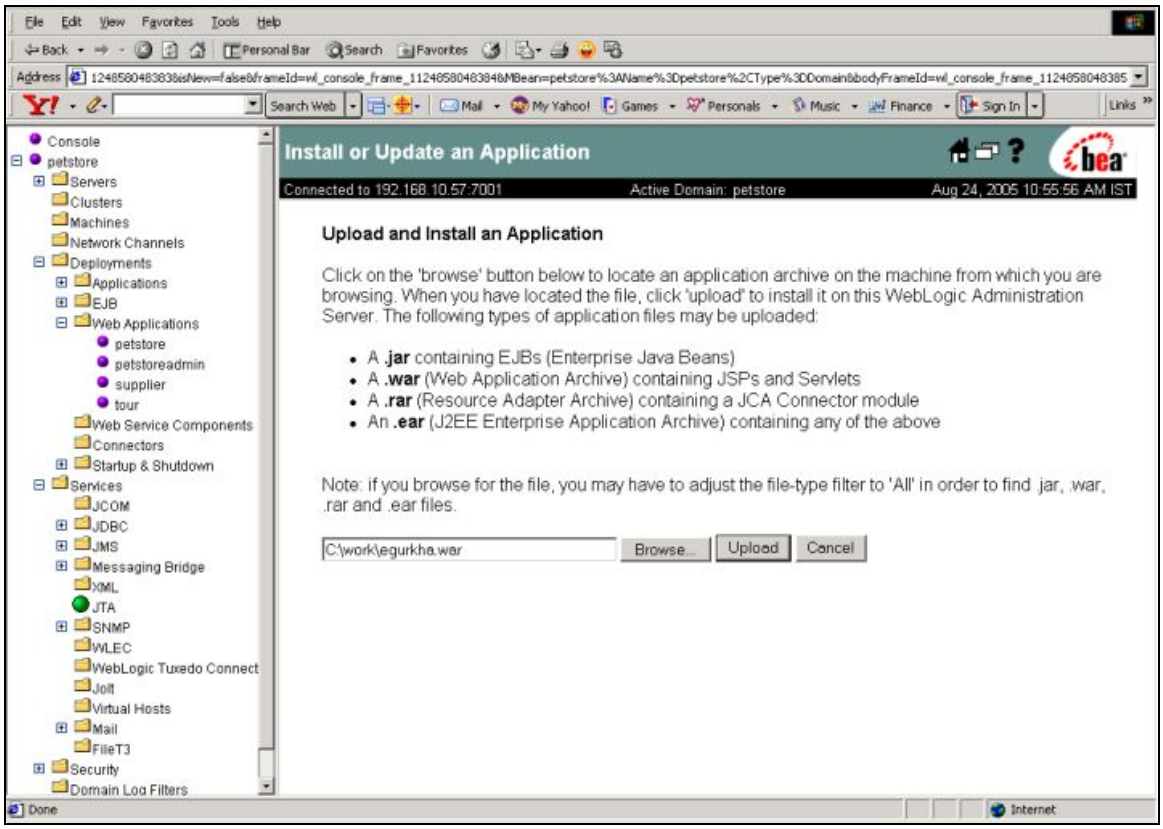


Figure 2.12: Screen informing you that egurkha.war file has been successfully uploaded

7. Upon a successful upload, Figure 2.13 will appear. At the bottom of the right panel of Figure 2.13, you will find displayed the name of the file that was just uploaded – i.e., **egurkha.war** – preceded by a **[select]** link. This is a clear indicator of the success of the upload process. To complete the deployment however, you will have to associate the deployed war file with one or more WebLogic server instances as you deem necessary. To achieve this, click on the **[select]** hyperlink that precedes the **egurkha.war** entry in the right panel of Figure 2.13.

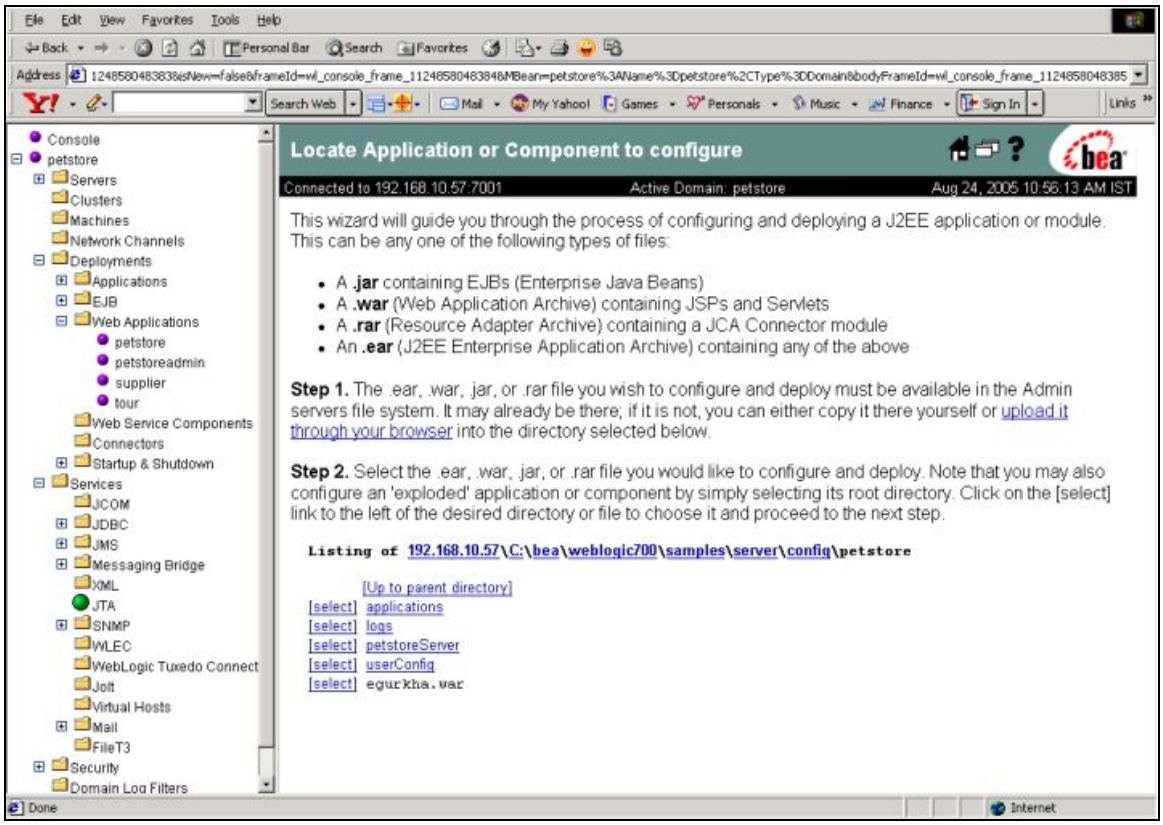


Figure 2.13: Indicating a successful upload

8. Doing so will open Figure 2.14. The **Available Servers** list in the right panel of Figure 2.14 will list all the available WebLogic server instances. From this list, choose the instances on which the **egurkha** application is to be deployed. Then, click on the **->** button to transfer the selection to the **Target Servers** list (see Figure 2.15). Finally, deploy the **egurkha** application on the **Target Servers** by clicking on the **Configure and Deploy** button in Figure 2.15.

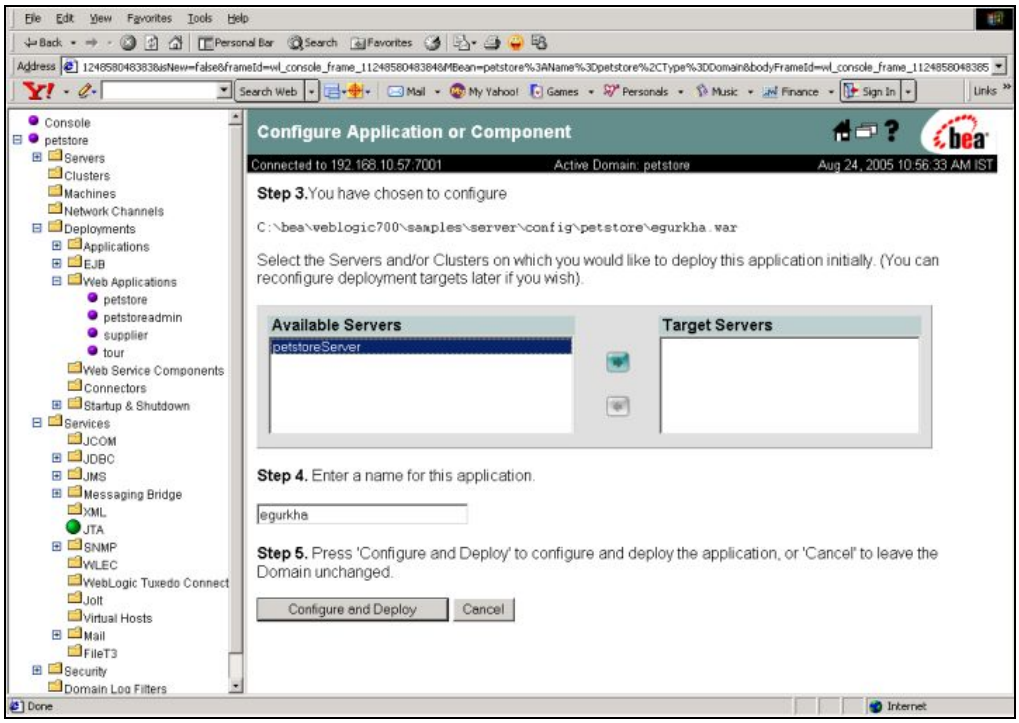


Figure 2.14: Selecting the WebLogic server instances on which the 'egurkha' application is to be deployed

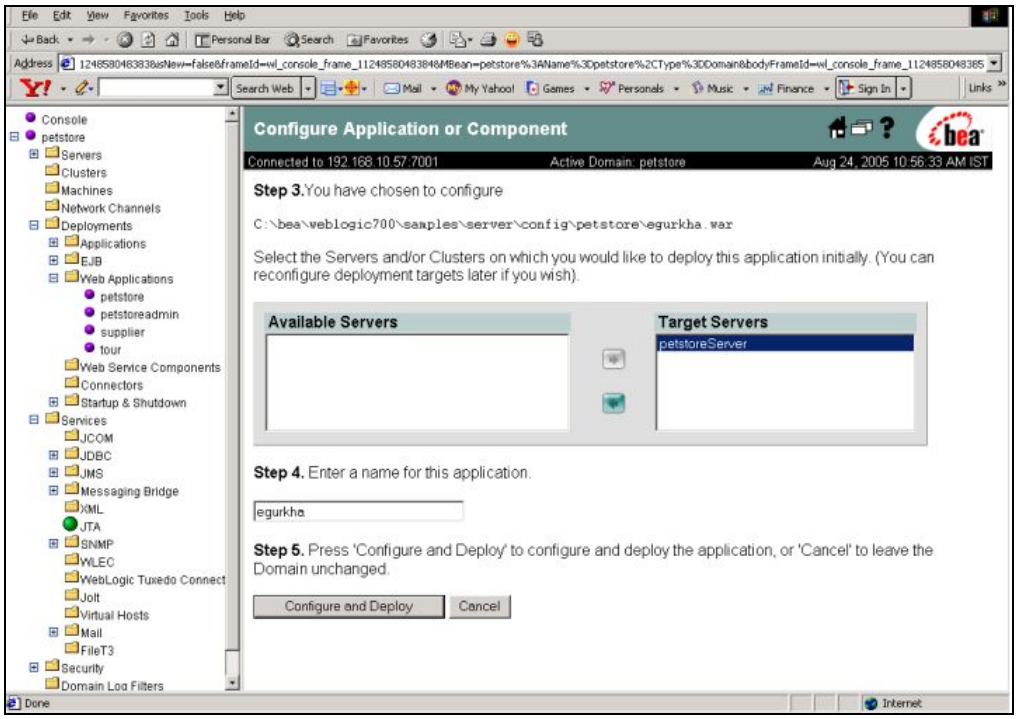


Figure 2.15: Deploying the 'egurkha' application on the Target Servers

9. Once the deployment completes and is successful, the **Deployment Status** section in the right panel

of Figure 2.16 will display **true** against each of the target servers. Similarly, the **Deployment Activity** section will display the **Completed** status for each of the target servers. On the contrary, if the **Deployment Status** is **false** and the **Deployment Activity** is "**Running…"**, it indicates that deployment is 'in progress'. In such a case, wait until the status changes. Upon completion of the deployment, the **egurkha** application will appear as a sub-node under the **Web Applications** tree in the left panel of Figure 2.16.

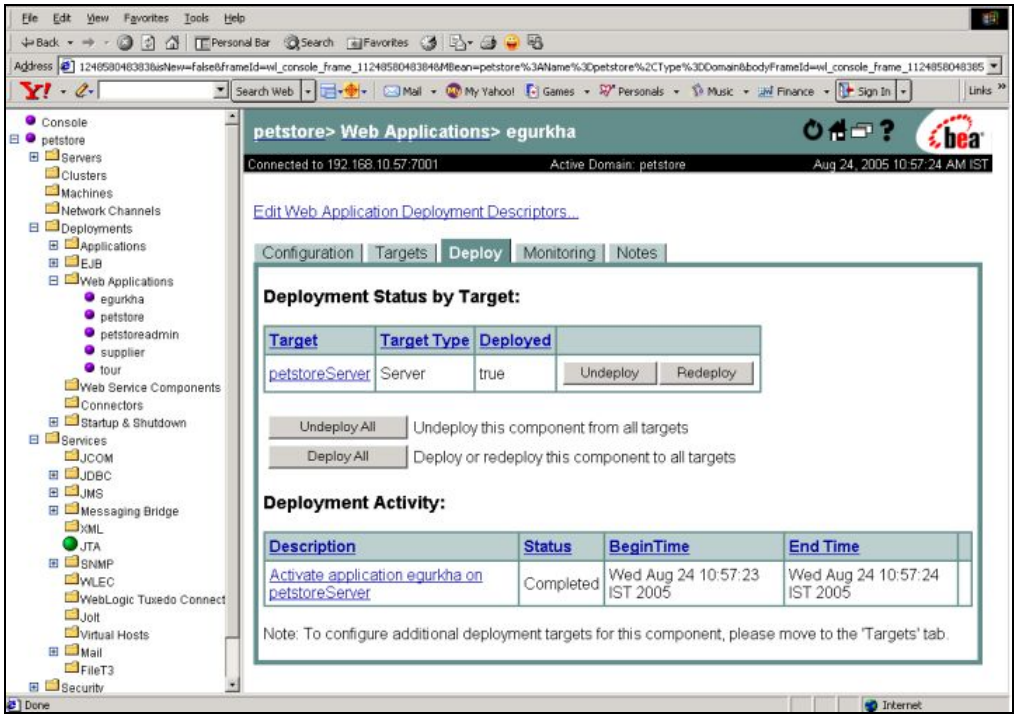


Figure 2.16: Successful deployment of the 'egurkha' application

2.1.3 Configuring an Oracle WebLogic Application Server 8.1

The eG agent package contains a web archive file called egurkha.war. On Unix environments, this file will be available in the /opt/egurkha/lib location. On Windows environment, this file will be available in the <EG_HOME_DIR>\lib directory. This file has to be installed on the WebLogic 8.0 (and above), which has to be monitored.

To deploy the **egurkha.war** file on a Oracle WebLogic application server 8.1 that requires monitoring, do the following:

1. Go to the following URL: *http:\\<WebLogic_server_IP>:<WebLogicPort>/console*

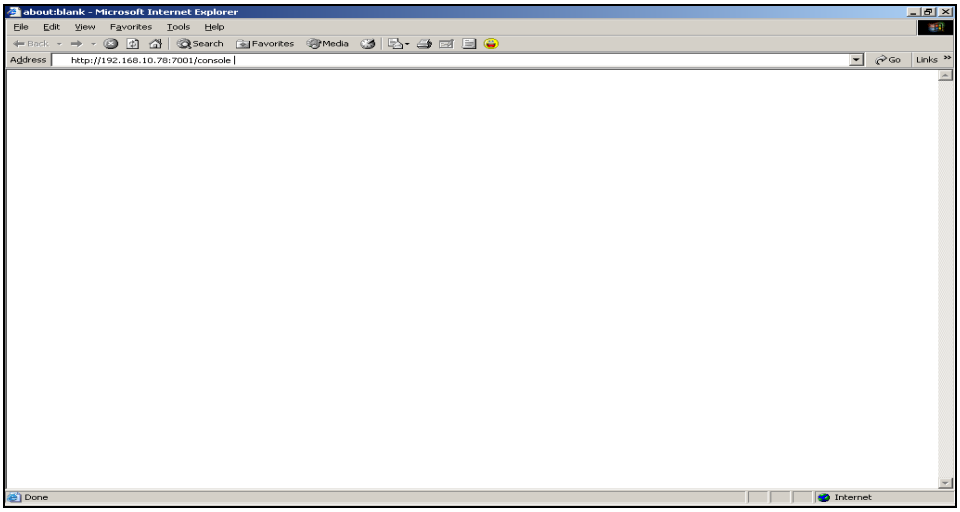


Figure 2.17: Connecting to the WebLogic console

2. Now, an authentication dialog (see Figure 2.18) that requires a username and password, will appear. Specify the same password you gave during the installation of the Weblogic server. If your environment is configured differently, please contact your WebLogic administrator to find out the username and password to use to login to the WebLogic console.

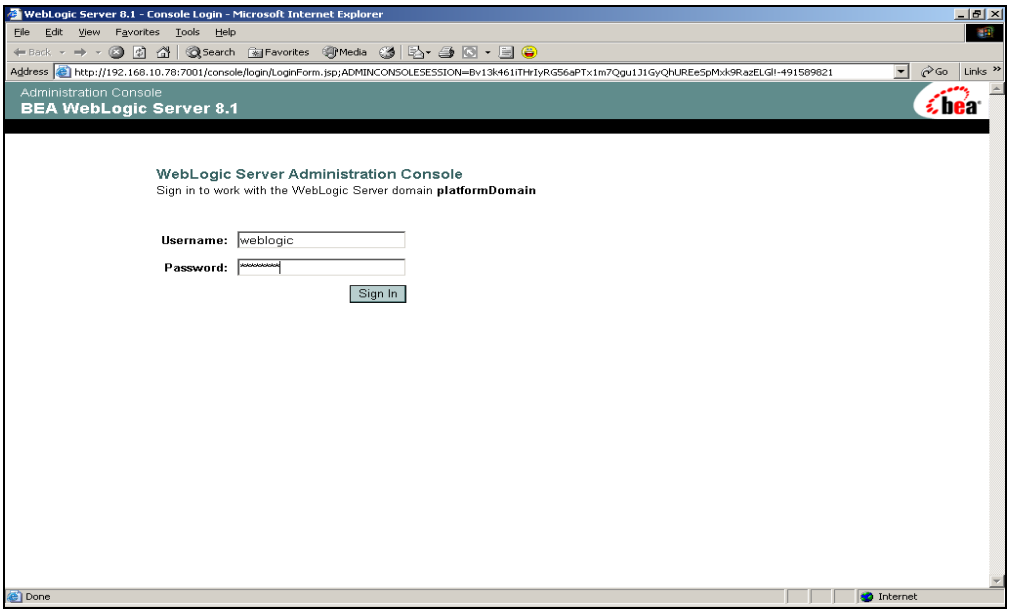


Figure 2.18: Specifying user name and password in the authentication dialog box

3. Upon successful authentication, the following WebLogic 8.1 Administration console will appear (see Figure 2.19).

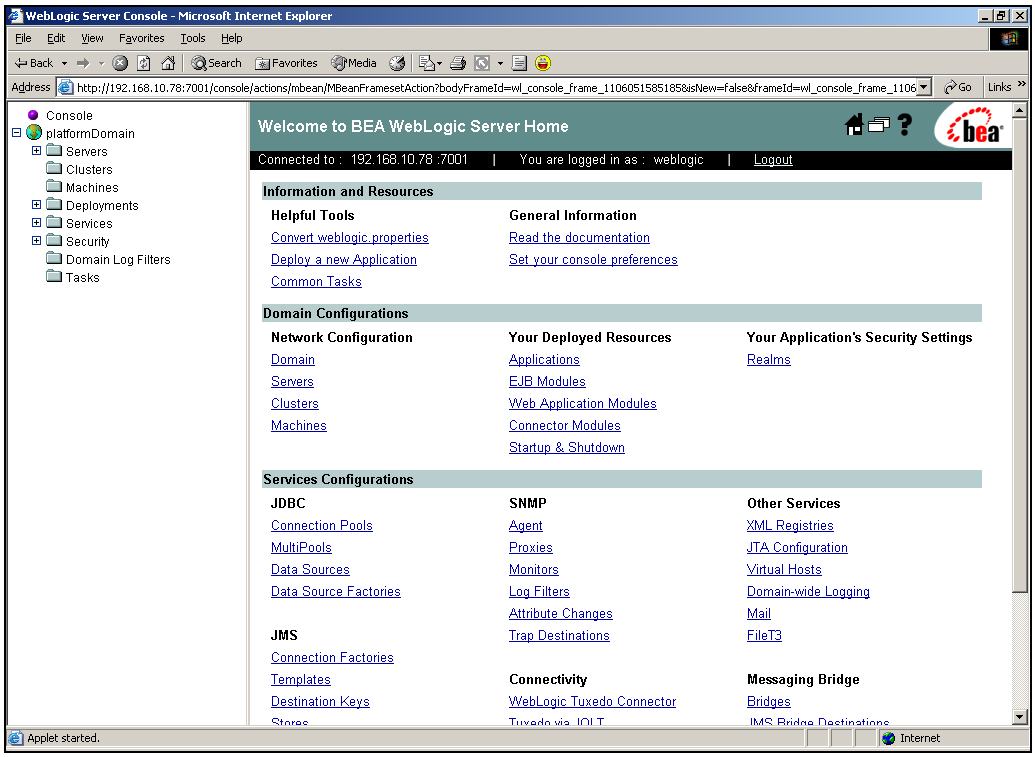


Figure 2.19: The Administration Console of the ORACLE WebLogic server 8.1

4. From the right panel, choose the **Web Application Modules** link under the **Domain Configurations** group. The following screen will then appear:

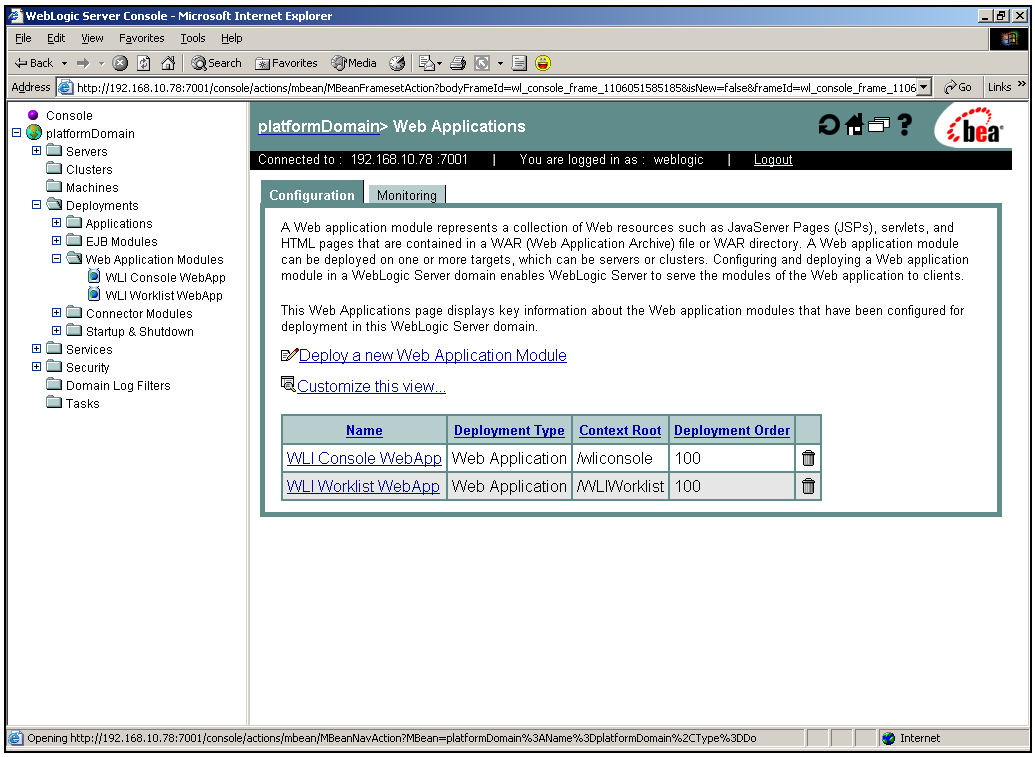


Figure 2.20: Screen where you can install or update an Application

5. From the right panel, choose the **Deploy a new Web Application Module** option, which will bring the screen represented by Figure 2.21 to light. Next, using the **Browse** button in the right panel, select the **egurkha.war** file (see Figure 2.21) from the <EG_HOME_DIR>\lib directory (in Unix, this will be /opt/egurkha/lib) and then, click on the **Upload** button.

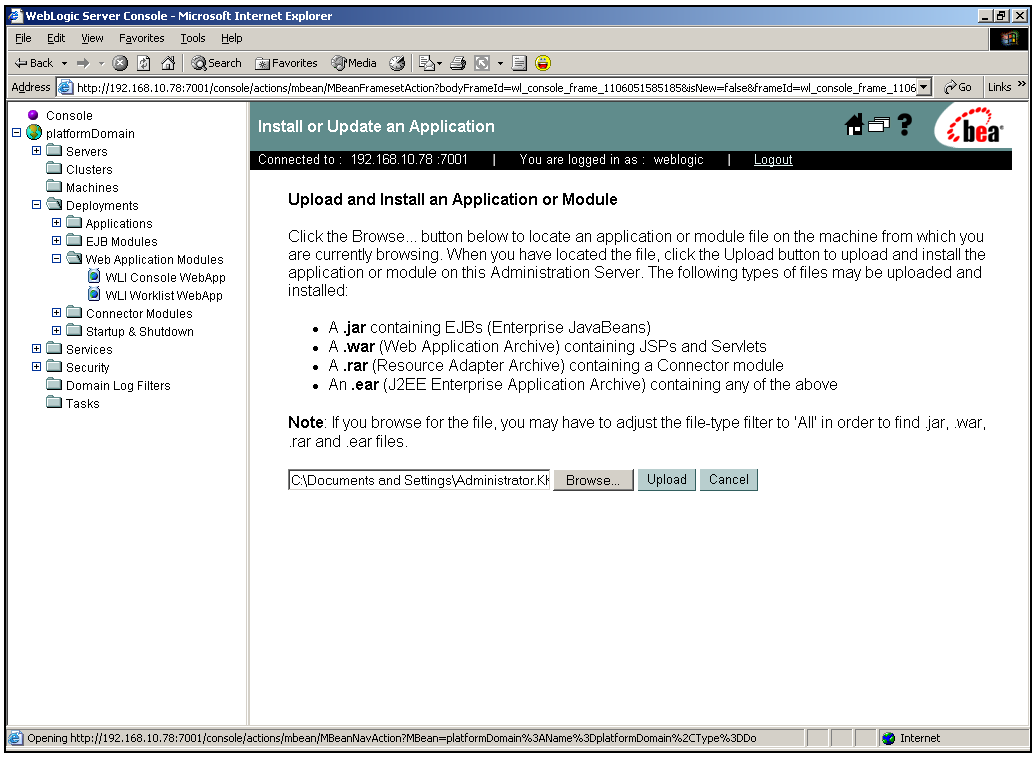


Figure 2.21: Specifying the location of the "egurkha.war" file

Note:

By default, the **egurkha.war** file will be searched for in the local disk of the system from which the browser is launched. Therefore, if you are trying to access the WebLogic console from a system that does not host an eG agent, then, before attempting to deploy the **egurkha.war** file, copy the war file from the <EG_HOME_DIR>\lib directory (in Windows environments. In Unix, this will be /opt/egurkha/lib) of an agent host, to your local disk. Then, proceed to perform the war deployment.

6. Next, select the archive path for the web application module being deployed by clicking on the radio button corresponding to the **egurkha.war** file in Figure 2.22. Then, click on the **Target Module** button therein.

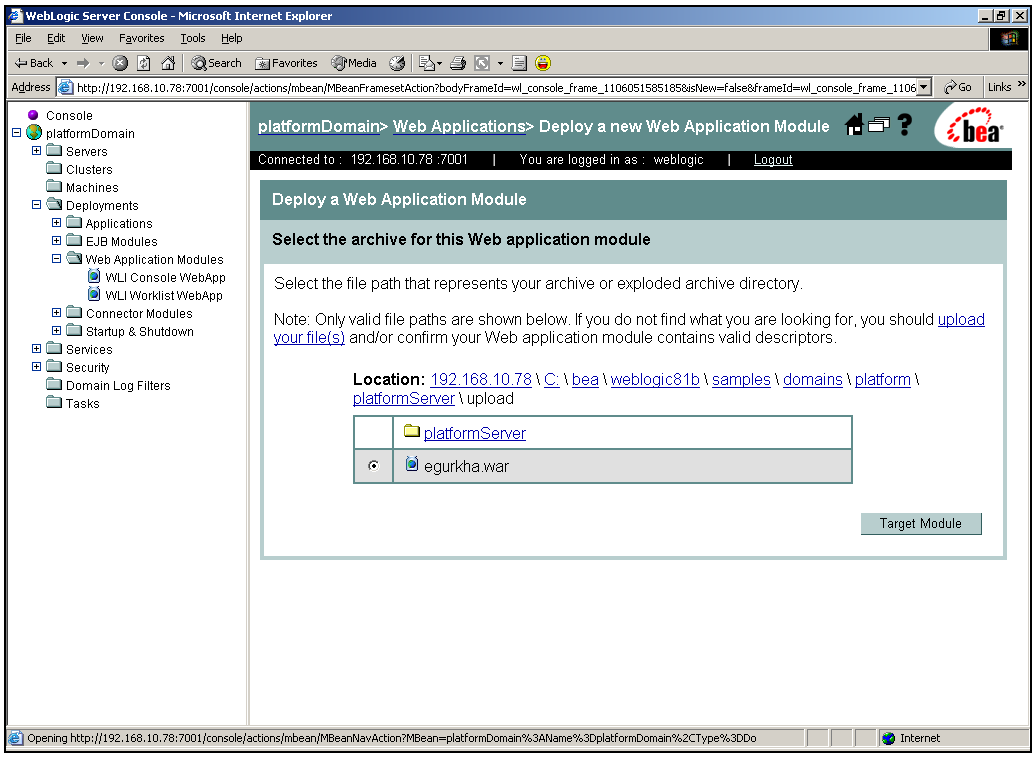


Figure 2.22: Screen informing you that egurkha.war file has been successfully uploaded

7. A summary of the details specified will then appear, to enable their review and reconfirmation (see Figure 2.23). Provide a name to identify the web application module being added in the **Name** text box of Figure 2.23.

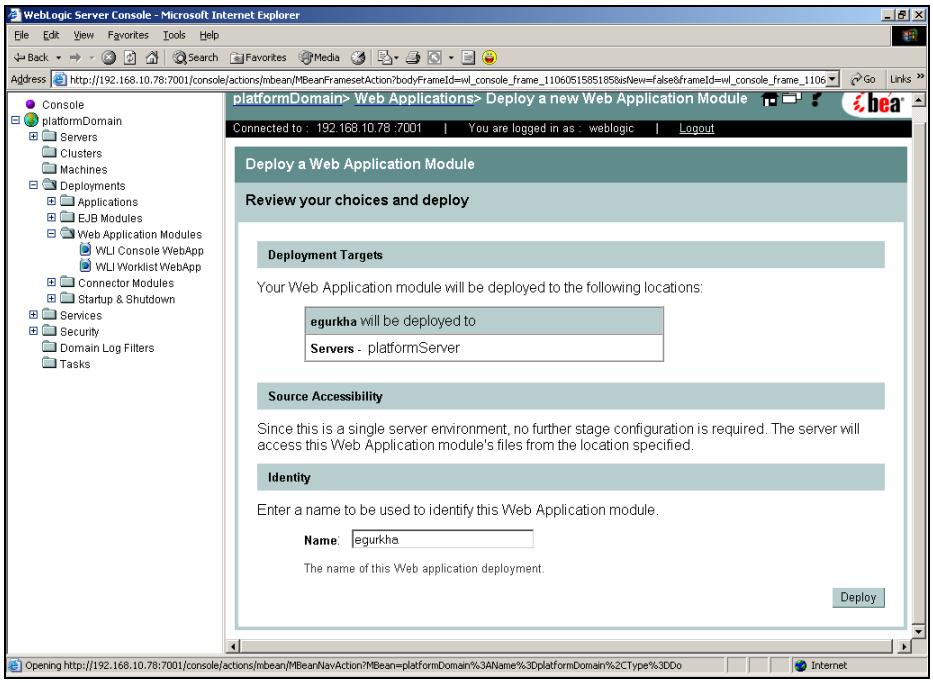


Figure 2.23: Summary of specifications

8. Finally, click on the **Deploy** button in Figure 2.23.

2.1.4 Deploying the 'egurkha.war' file on a WebLogic Server 7.x in a Clustered Environment

The eG agent package contains a web archive file **called egurkha7x.war**. On Unix environments, this file will be available in the **/opt/egurkha/lib** location. On Windows environment, this file will be available in the <EG_HOME_DIR>\lib directory. Prior to monitoring a WebLogic Server 7.x in a cluster, make sure that this file is deployed on the target WebLogic server to enable the eG agent to monitor it.

Before deploying the war file, make sure that you follow the steps below:

- Login to the system hosting the eG agent that is monitoring the WebLogic server 7.x.
- Copy the **egurkha7x.war** file from the /opt/egurkha/lib directory to any other directory on the eG agent host.
- Rename the war file as **egurkha.war**.

Then, to deploy the egurkha.war file on a ORACLE WebLogic application server 7.x in a clustered environment, do the following:

1. Connect to the **admin** server of the WebLogic cluster using the following URL: *http:\\<Admin_server_IP>:<AdminServerPort>/console*
2. Now, an authentication dialog (see Figure 2.24) that requires a username and password, will appear. Please contact your WebLogic administrator to find out the username and password to use to login to the WebLogic console.

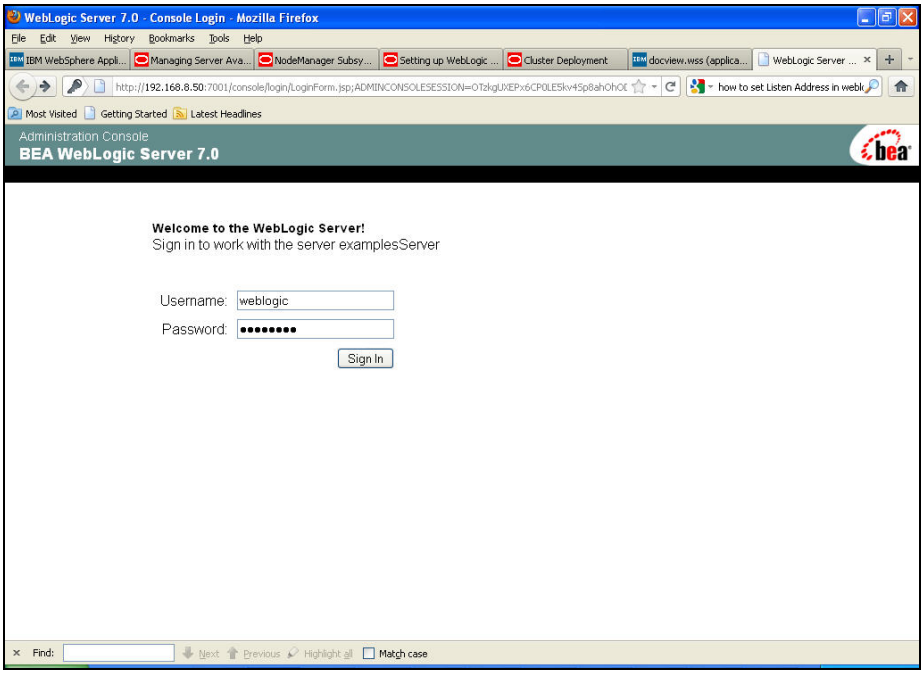


Figure 2.24: Specifying user name and password in the authentication dialog box

3. Upon successful authentication, the WebLogic Administration console will appear (see Figure 2.25).

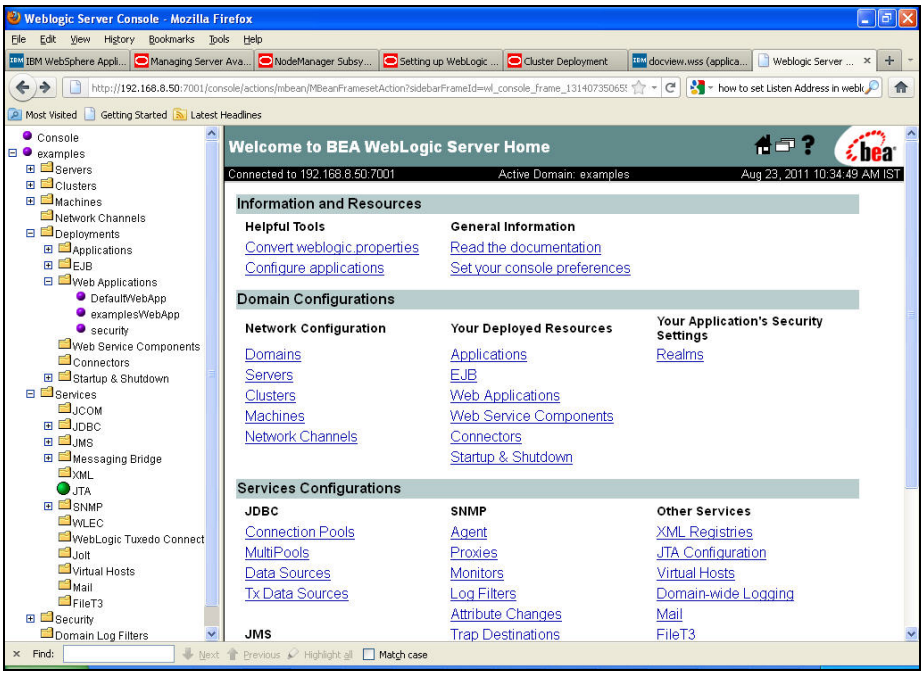


Figure 2.25: The Administration Console of the ORACLE WebLogic server 7.0 in a Cluster

4. From the tree-structure in the left panel, choose the **Web Applications** node under the **Deployments** node. Then, click on the **Configure a new Web Application** link in the left pane to begin the deployment of the **egurkha.war** application (see Figure 2.26).

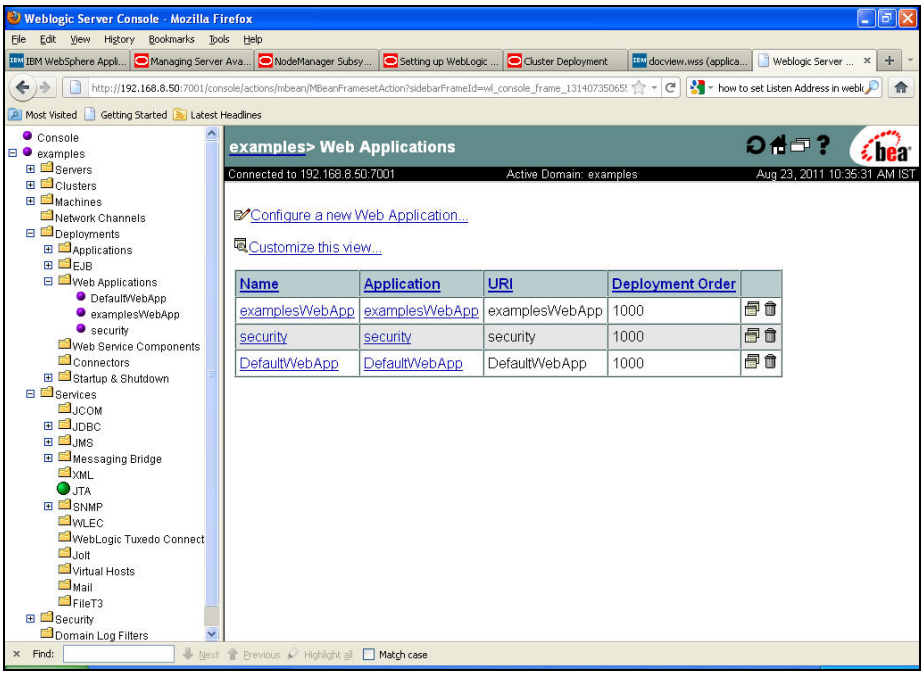


Figure 2.26: Screen where you can install or update an Application

5. Upon clicking the link, Figure 2.27 will appear, which explains procedure for application deployment. To proceed, click on the **upload it through the browser** link in the right panel of Figure 2.27.

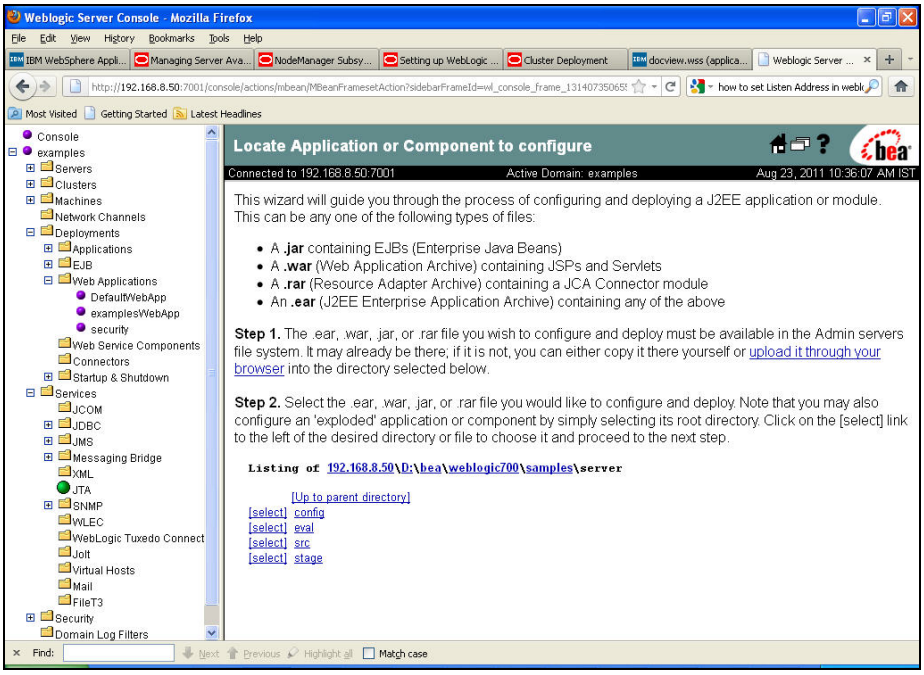


Figure 2.27: Attempting to upload the "egurkha.war" file

Note:

By default, the **egurkha.war** file will be searched for in the local disk of the system from which the browser is launched. Therefore, if you are trying to access the WebLogic console from a system that does not host an eG agent, then, before attempting to deploy the egurkha.war file, copy the war file from the <EG_HOME_DIR>\lib directory (in Windows environments. In Unix, this will be /opt/egurkha/lib) of an agent host, to your local disk. Then, proceed to perform the war deployment.

6. In Figure 2.28 that appears, specify the full path to the **egurkha.war** file to be deployed on the WebLogic server. Alternatively, you can use the **Browse** button to locate the war file. Finally, click the **Upload** button in Figure 2.28 to upload the **war** file.

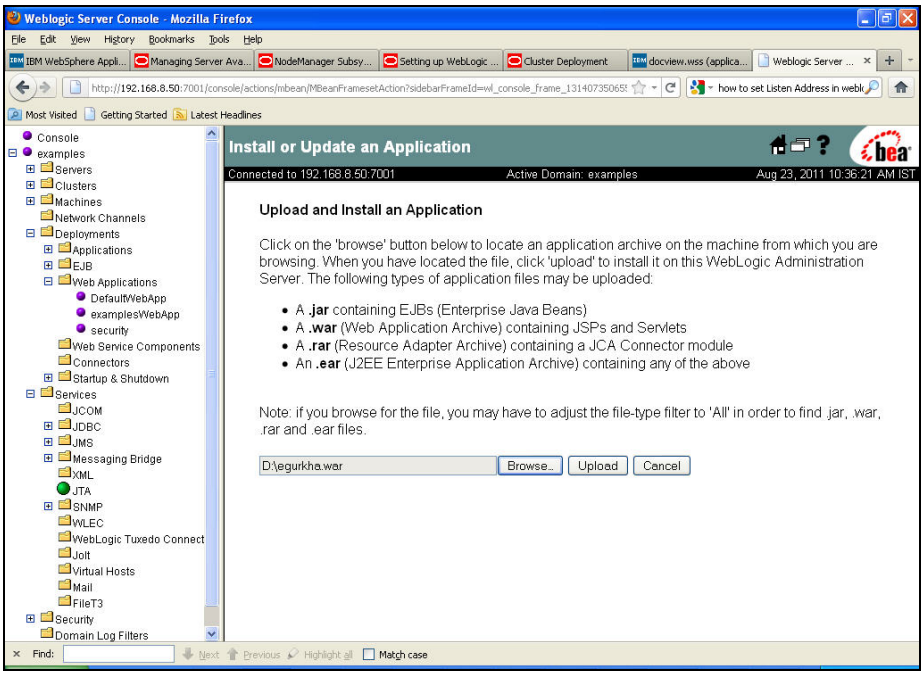


Figure 2.28: Screen informing you that egurkha.war file has been successfully uploaded

7. Upon a successful upload, Figure 2.29 will appear. At the bottom of the right panel of Figure 2.29, you will find displayed the name of the file that was just uploaded – i.e., **egurkha.war** – preceded by a **[select]** link. This is a clear indicator of the success of the upload process. To complete the deployment however, you will have to associate the deployed war file with one or more WebLogic server instances as you deem necessary. To achieve this, click on the **[select]** hyperlink that precedes the **egurkha.war** entry in the right panel of Figure 2.29.

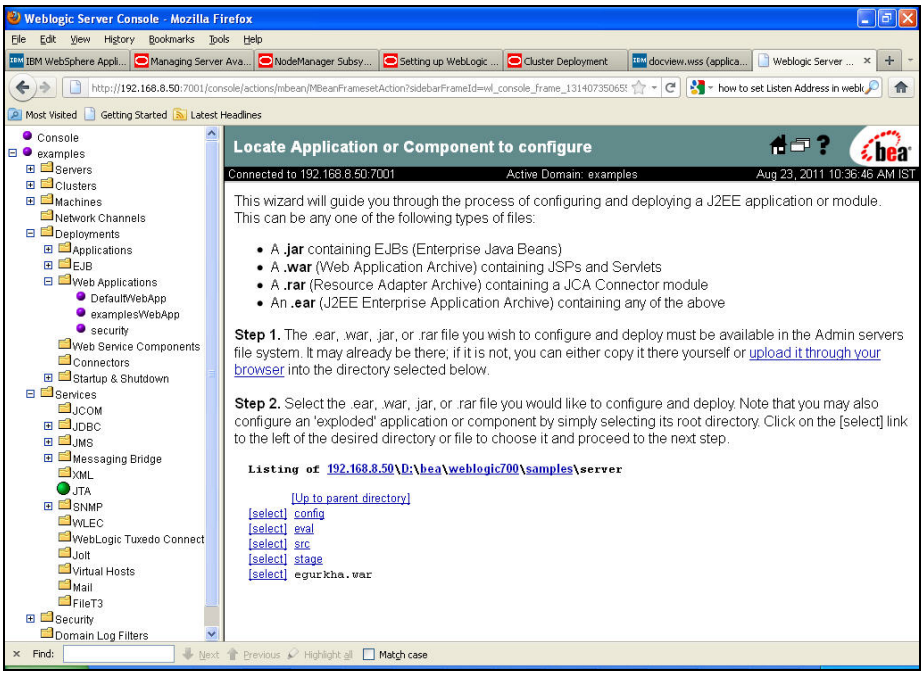


Figure 2.29: Indicating a successful upload

8. Doing so will open Figure 2.30. The **Available Servers** list in the right panel of Figure 2.30 will list all the available WebLogic server instances. From this list, choose the instances on which the **egurkha** application is to be deployed. Then, click on the –> button to transfer the selection to the **Target Servers** list (see Figure 2.30). Finally, deploy the **egurkha** application on the **Target Servers** by clicking on the **Configure and Deploy** button in Figure 2.31.

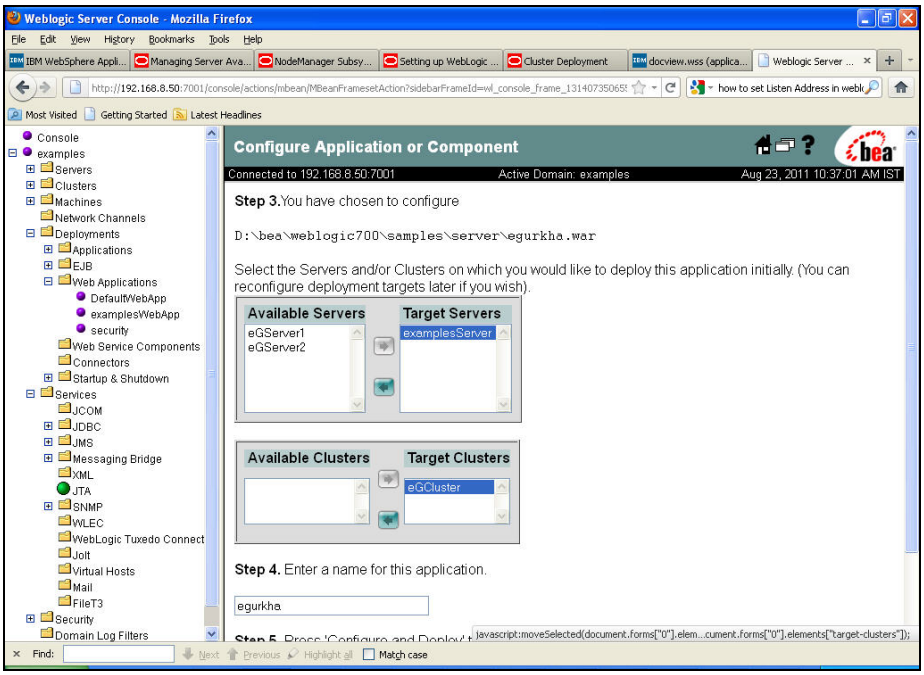


Figure 2.30: Selecting the WebLogic server instances on which the 'egurkha' application is to be deployed

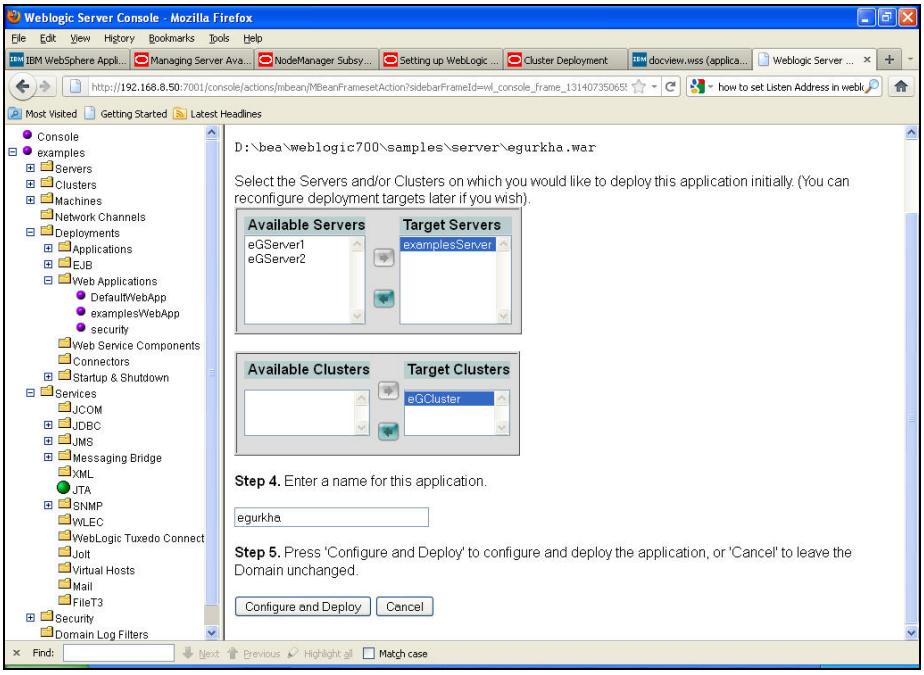


Figure 2.31: Deploying the 'egurkha' application on the Target Servers

9. Once the deployment completes and is successful, the **Deployment Status** section in the right panel of Figure 2.32 will display **true** against each of the target servers. Similarly, the **Deployment Activity** section will display the **Completed** status for each of the target servers. On the contrary, if the

Deployment Status is **false** and the **Deployment Activity** is **"Running…"**, it indicates that deployment is 'in progress'. In such a case, wait until the status changes.

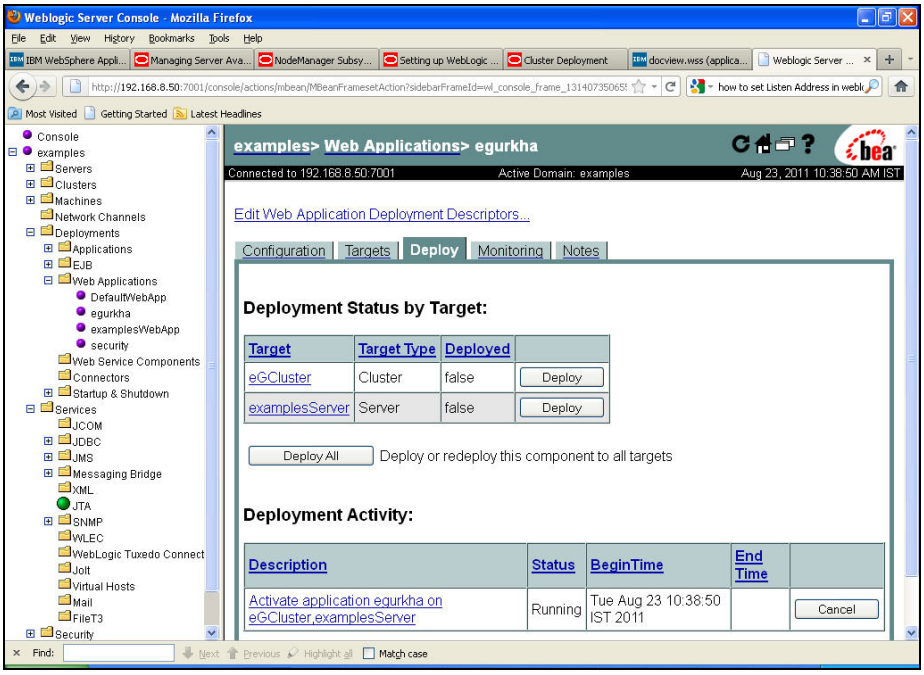


Figure 2.32: Successful deployment of the 'egurkha' application

2.1.5 Deploying the 'egurkha.war' file on a WebLogic Server 8.1 in a Clustered Environment

The eG agent package contains a web archive file called **egurkha.war**. On Unix environments, this file will be available in the **/opt/egurkha/lib** location. On Windows environment, this file will be available in the <EG_HOME_DIR>\lib directory. This file has to be installed on the WebLogic 8.0 (and above) server in a cluster, which has to be monitored.

To deploy the **egurkha.war** file on a Oracle WebLogic application server 8.1 that requires monitoring, do the following:

1. Connect to the **admin** server of the cluster using the URL: *http:\\<Admin_ server_ IP>:<AdminPort>/console*
2. Now, an authentication dialog (see Figure 2.33) that requires a username and password, will appear. Please contact your WebLogic administrator to find out the username and password to use to login to the WebLogic console.

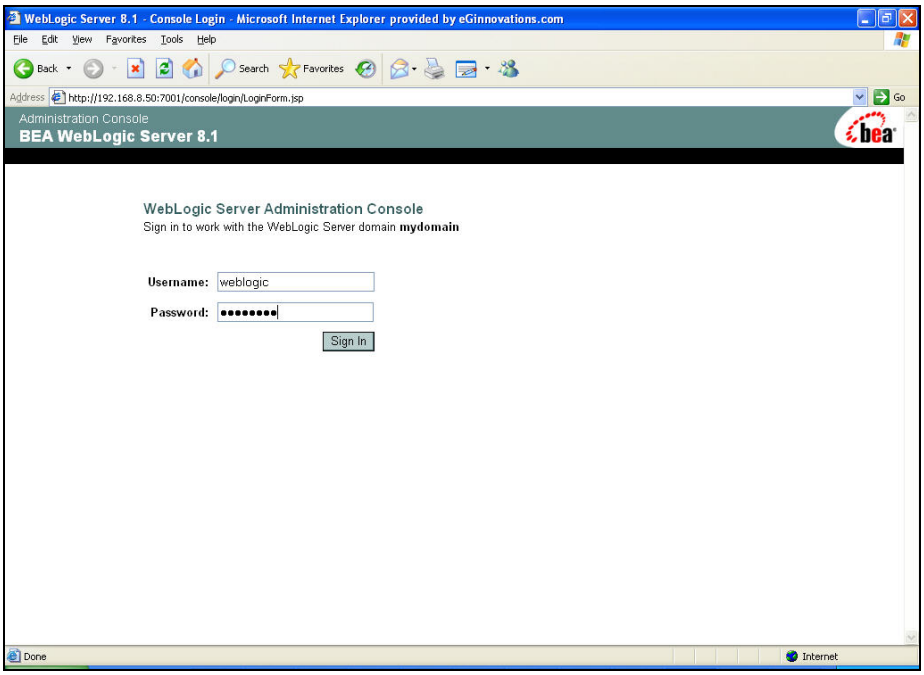


Figure 2.33: Specifying user name and password in the authentication dialog box

3. Upon successful authentication, the following WebLogic Administration console will appear (see Figure 2.34).

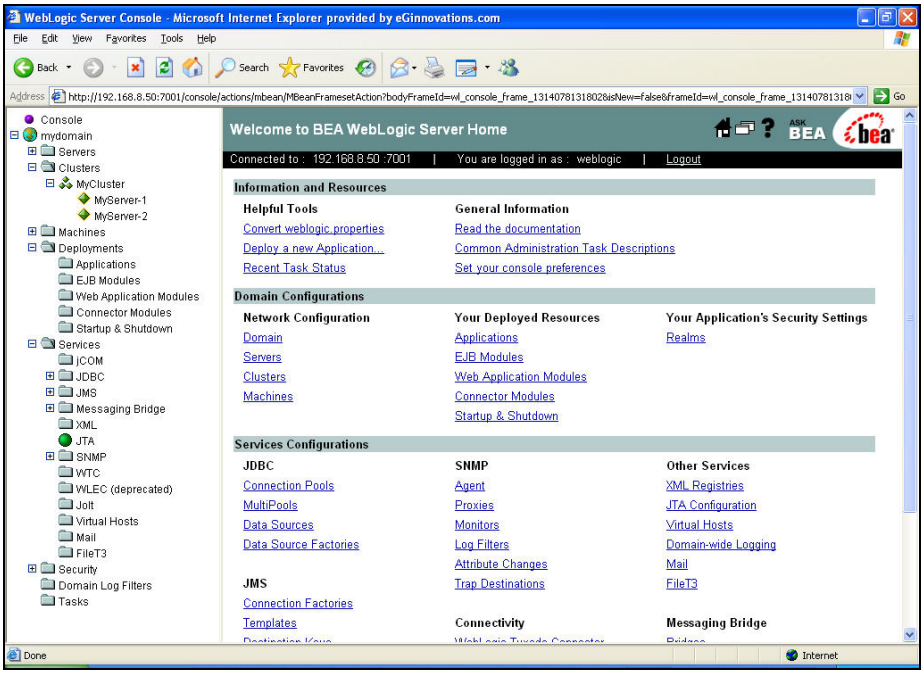


Figure 2.34: The Administration Console of the Oracle WebLogic server 8.1

4. From the right panel, choose the **Web Application Modules** link under the **Domain Configurations** group. The following screen will then appear:

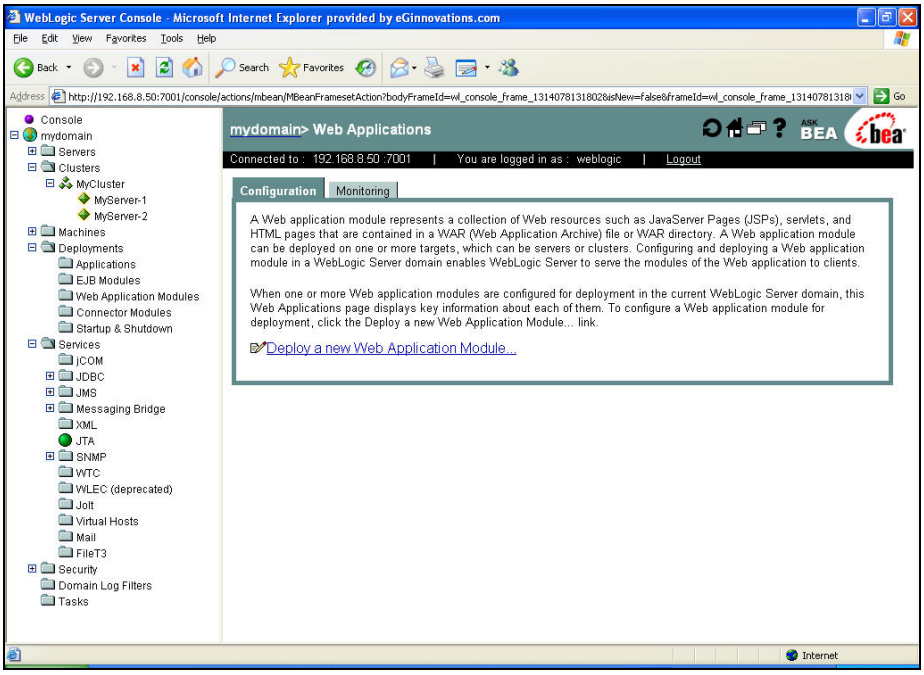


Figure 2.35: Screen where you can install or update an Application

5. From the right panel, choose the **Deploy a new Web Application Module** option, which will bring the screen represented by Figure 2.36 to light. Next, using the **Browse** button in the right panel, select the **egurkha.war** file (see Figure 2.36) from the <EG_HOME_DIR>\lib directory (in Unix, this will be /opt/egurkha/lib) and then, click on the **Upload** button.

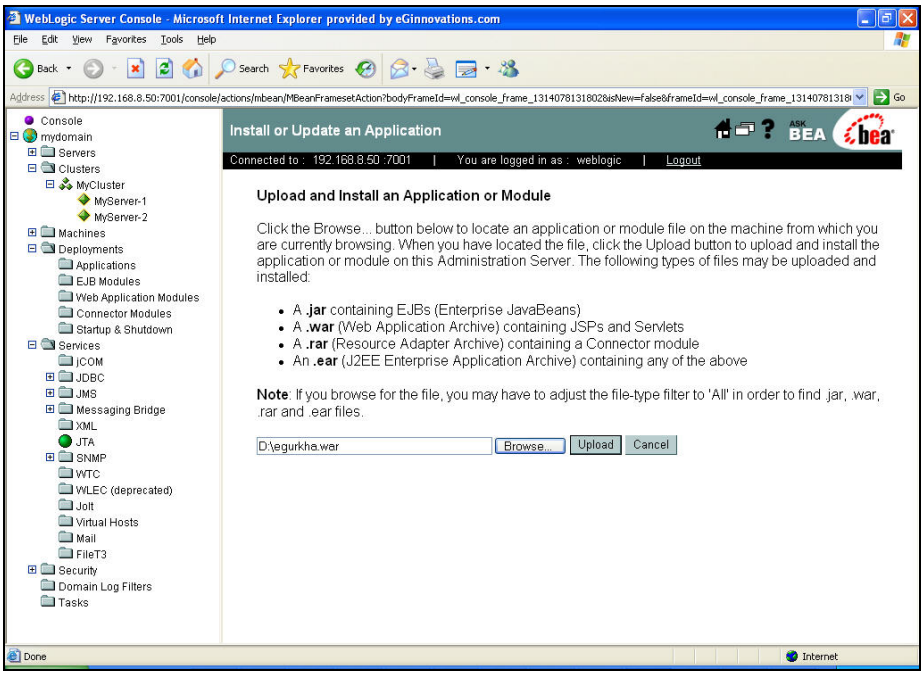


Figure 2.36: Specifying the location of the "egurkha.war" file

Note:

By default, the **egurkha.war** file will be searched for in the local disk of the system from which the browser is launched. Therefore, if you are trying to access the WebLogic console from a system that does not host an eG agent, then, before attempting to deploy the **egurkha.war** file, copy the war file from the <EG_HOME_DIR>\lib directory (in Windows environments. In Unix, this will be /opt/egurkha/lib) of an agent host, to your local disk. Then, proceed to perform the war deployment.

6. Next, select the archive path for the web application module being deployed. For this, use the **Location** links in Figure 2.37 to browse the location of the **egurkha.war** file, and then select the **egurkha.war** option as indicated by Figure 2.38. Then, click on the **Target Module** button therein.

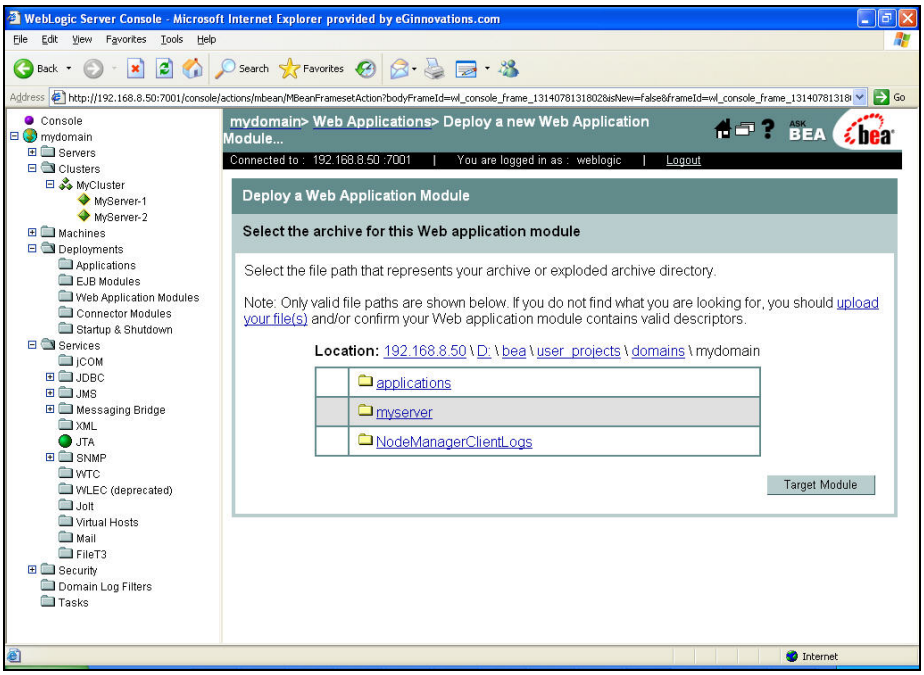


Figure 2.37: Browsing the location of the 'egurkha.war' file using the Location links

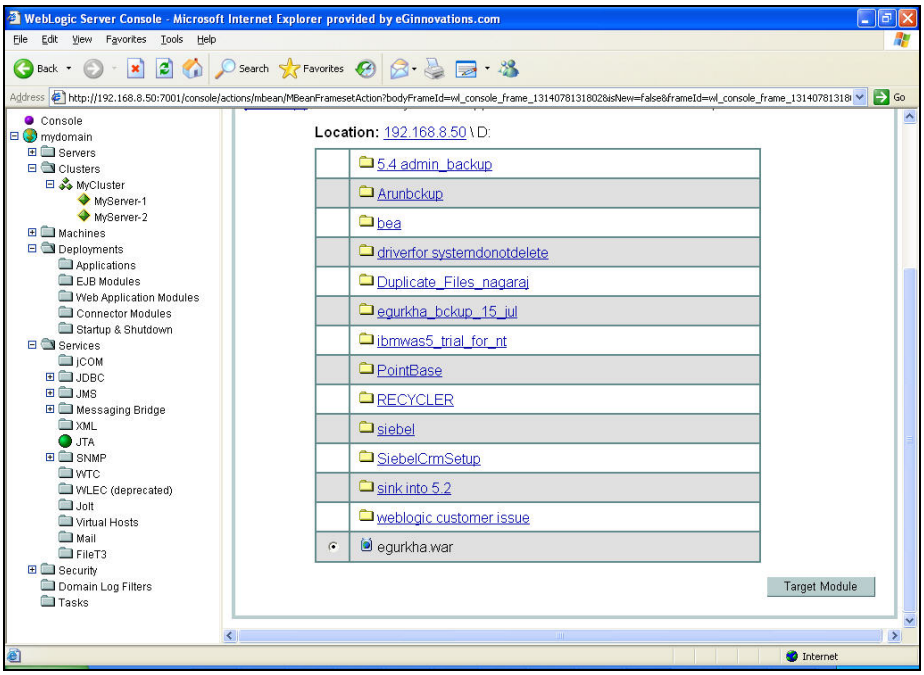


Figure 2.38: Selecting 'egurkha.war' as the archive path

7. Figure 2.39 will then appear listing the clustered and non-clustered (independent) WebLogic instances that are available. Select the instance(s) on which the **egurkha.war** file is to be deployed by selecting the corresponding check boxes or radio buttons. Then, click the **Continue** button.

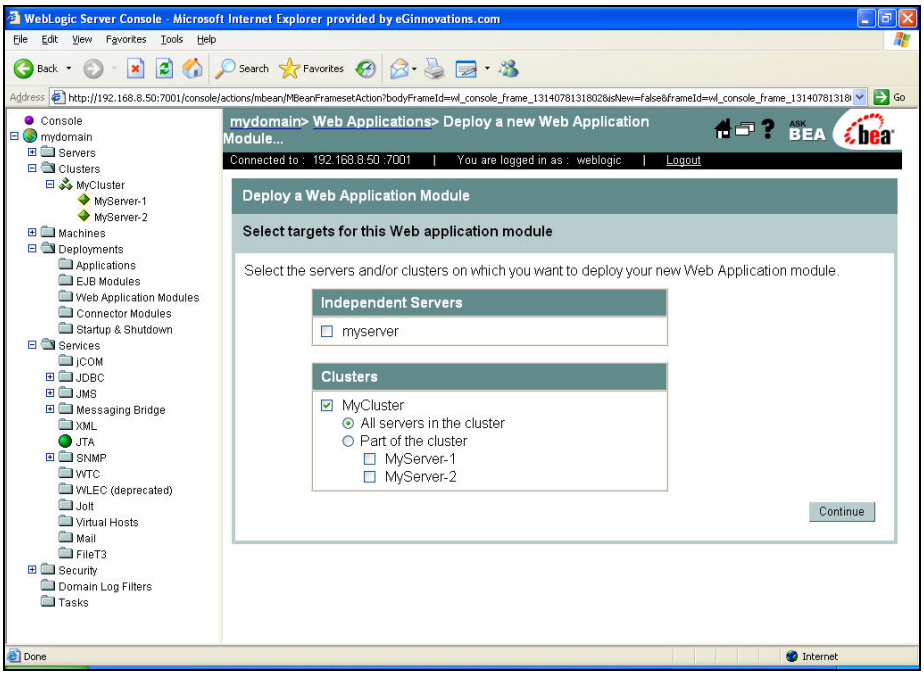


Figure 2.39: Selecting the WebLogic instance on which the WAR file is to be deployed

8. A summary of the details specified will then appear, to enable their review and reconfirmation (see Figure 2.40). Provide a name to identify the web application module being added in the **Name** text box of Figure 2.40.

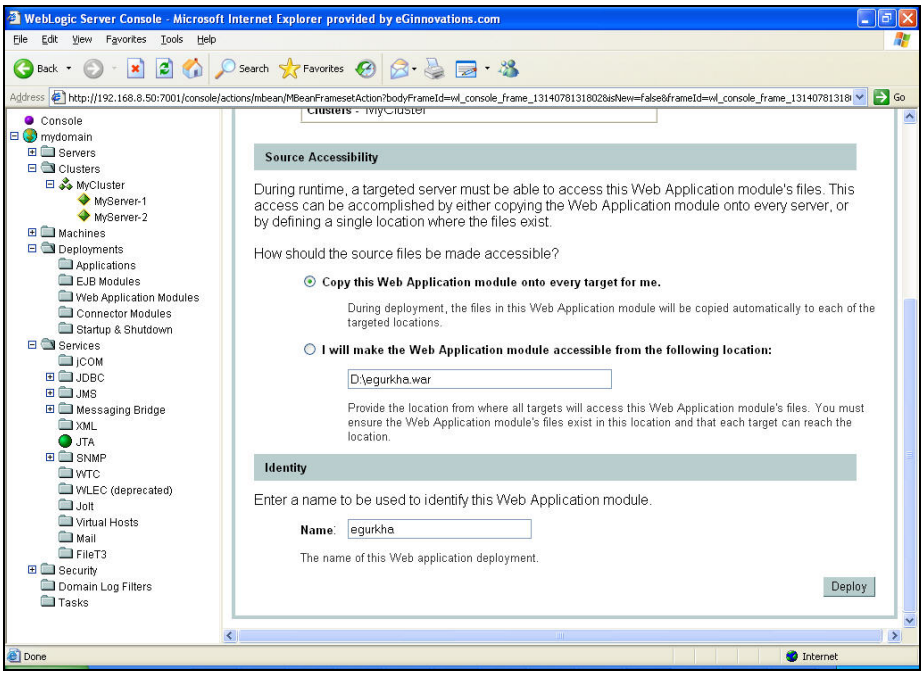


Figure 2.40: Viewing a summary of the specifications

9. Finally, click on the **Deploy** button in Figure 2.40. Upon successful deployment, Figure 2.41 will appear.

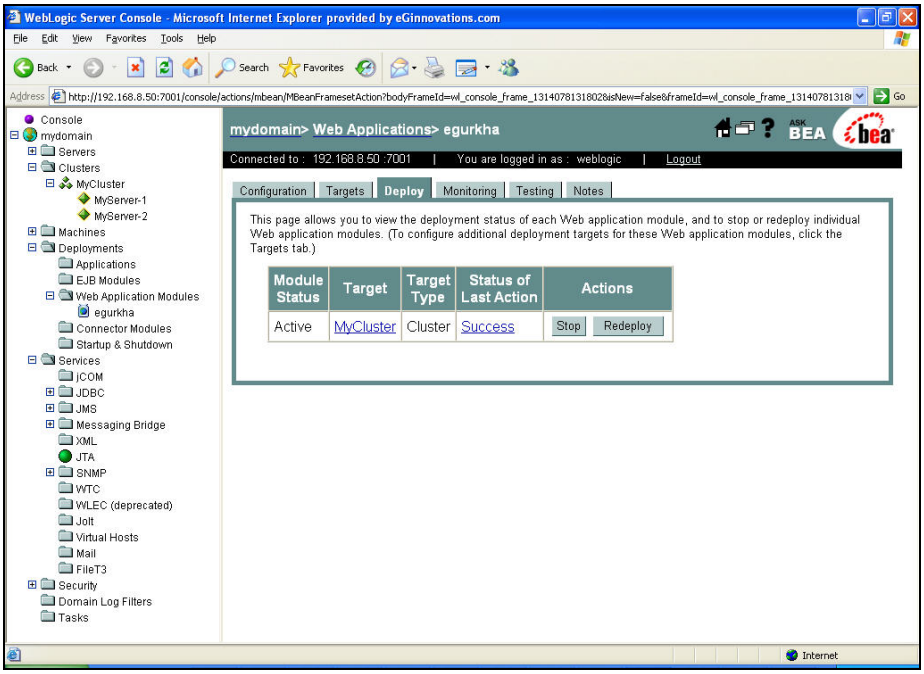


Figure 2.41: Successful deployment of the egurkha.war file on a WebLogic Server 8.x in a cluster

2.1.6 Enabling Garbage Collection Monitoring on the WebLogic Server 6/7/8

If the JVMGC test is to be executed for garbage collection (GC) monitoring, then the following additional steps need to be performed:

1. Edit the **startWebLogic.cmd** file (in Windows. In Unix, this will be **startWebLogic.sh**) in the <ORACLE_HOME>/<WEBLOGIC_HOME>/config/<DOMAIN>/ directory (in WebLogic 8.1, this will be <ORACLE_HOME>/<WEBLOGIC_HOME>/samples/domain/<DOMAIN>/ directory and in WebLogic 12c, this will be <ORACLE_HOME>/<WEBLOGIC_HOME>/user_project/domains/base_domain/bin directory).
2. Towards the end of the file, an entry to start the WebLogic server exists. Add a couple of lines to this entry as indicated by Figure 2.42.

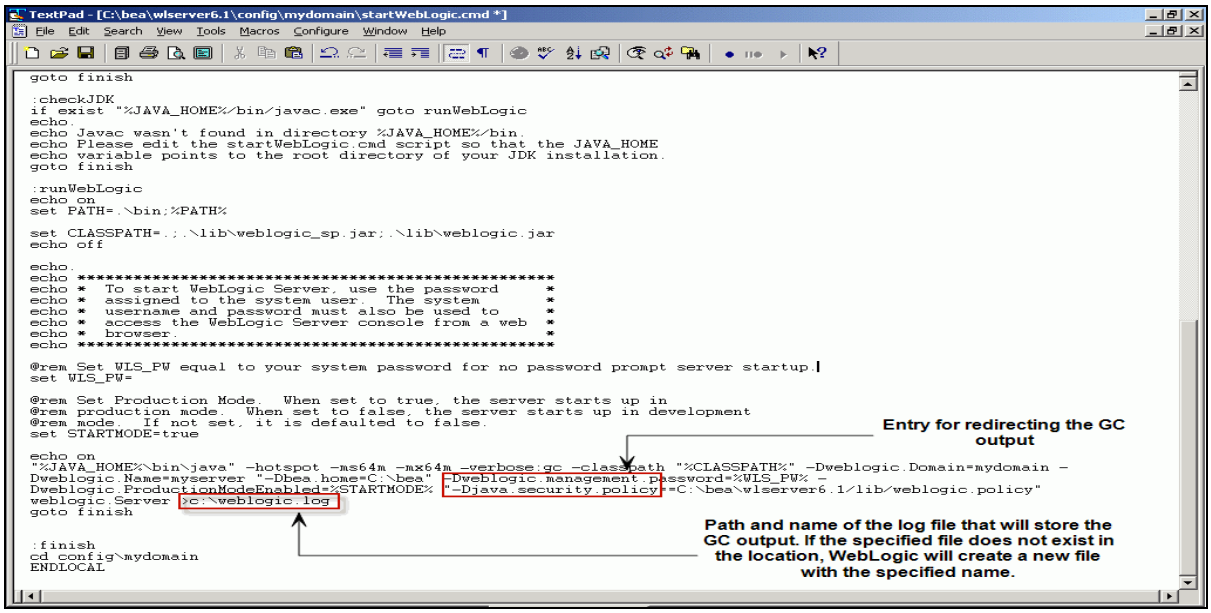


Figure 2.42: Editing the startWebLogic.cmd file

3. Start the WebLogic server.

2.1.7 Configuring an Execute Queue

To complete the deployment of the **egurkha** application, create a new execute queue in the WebLogic server. This execute queue has to be named eGQueue.

This section will discuss the steps involved in execute queue creation on all versions of WebLogic.

Configuring an Execute Queue on WebLogic 6.1

To create the special Execute queue for eG on WebLogic 6.1, do the following:

1. Connect to the WebLogic server console using the URL: *http://<WebLogicServerIP>:<WebLogicPort>/console*.
2. When prompted for a user name and password, provide the valid details and login to the console. Figure 2.43 then appears.

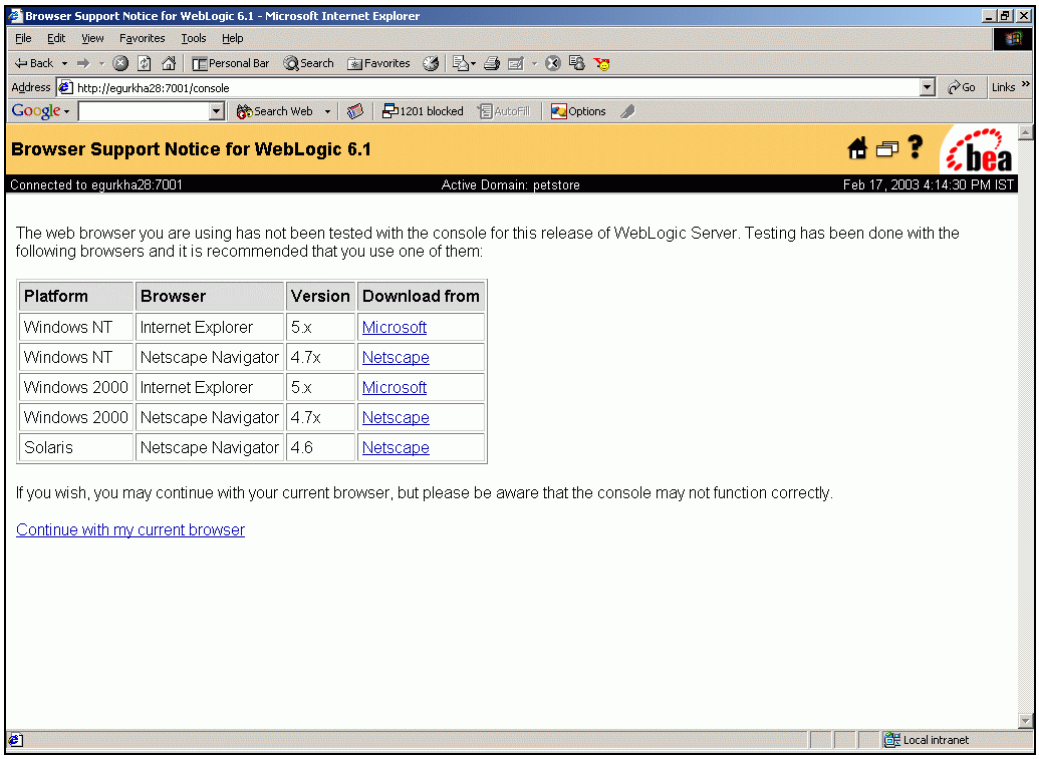


Figure 2.43: The WebLogic console

3. Click on the **Continue with my current browser** link in Figure 2.43 to move to the next page. Figure 2.44 will later appear.

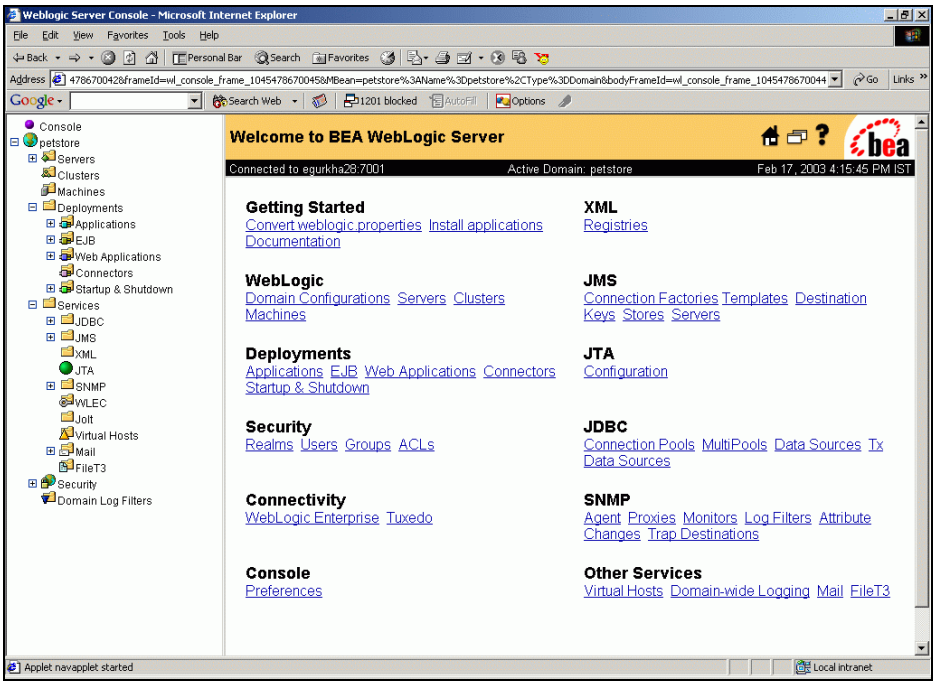


Figure 2.44: The Welcome screen

4. Expand the **Servers** node in the tree structure in the left pane of Figure 2.44, and click on the **petstoreServer** entry within. The properties of the selected server will then be displayed in the right pane (see Figure 2.45).

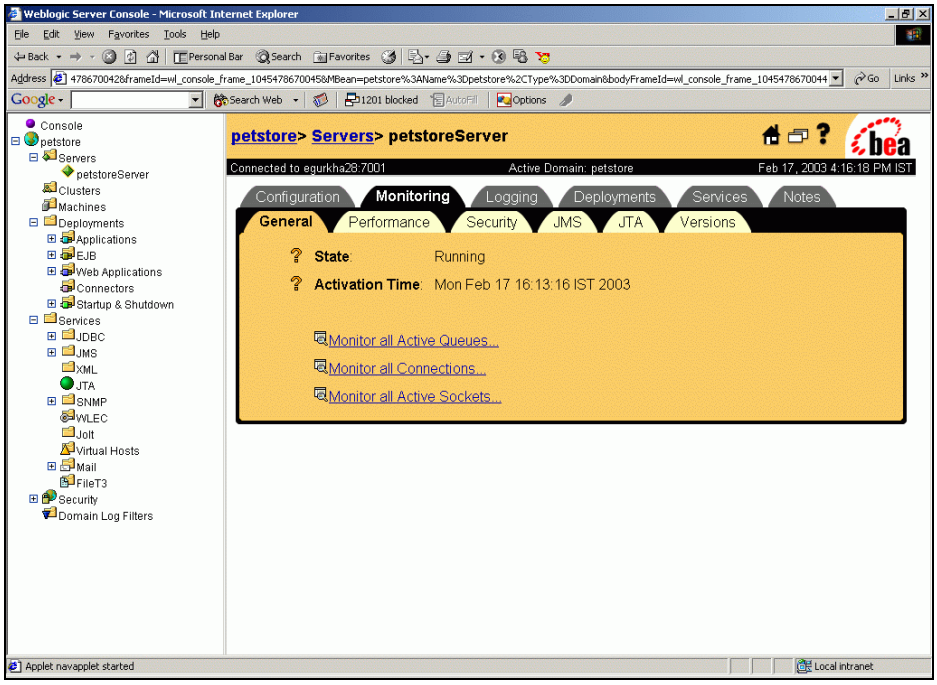


Figure 2.45: Selecting the petstoreServer

5. Click on the **Monitoring** tab in the right pane of Figure 2.45. Then, to configure a new execute queue, click on the **Configure Execute Queue** link the **Monitoring** tab of Figure 2.46.

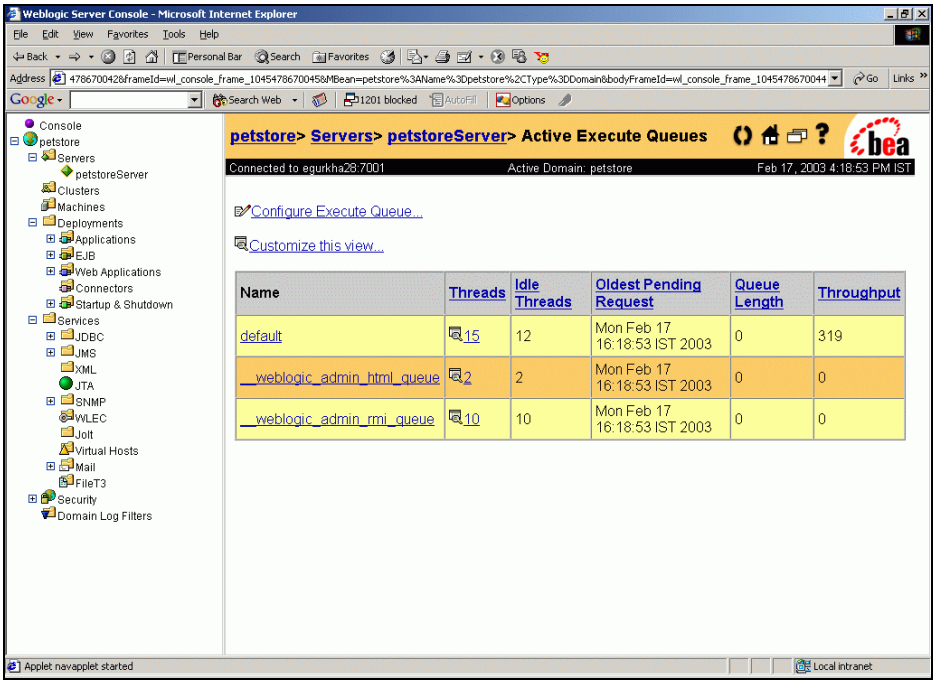


Figure 2.46: Clicking on the Configure Execute Queue link

6. Figure 2.47 that appears next will list the default execute queue on the WebLogic server. To create a new queue, click on the **Configure a new Execute Queue** link in Figure 2.47.

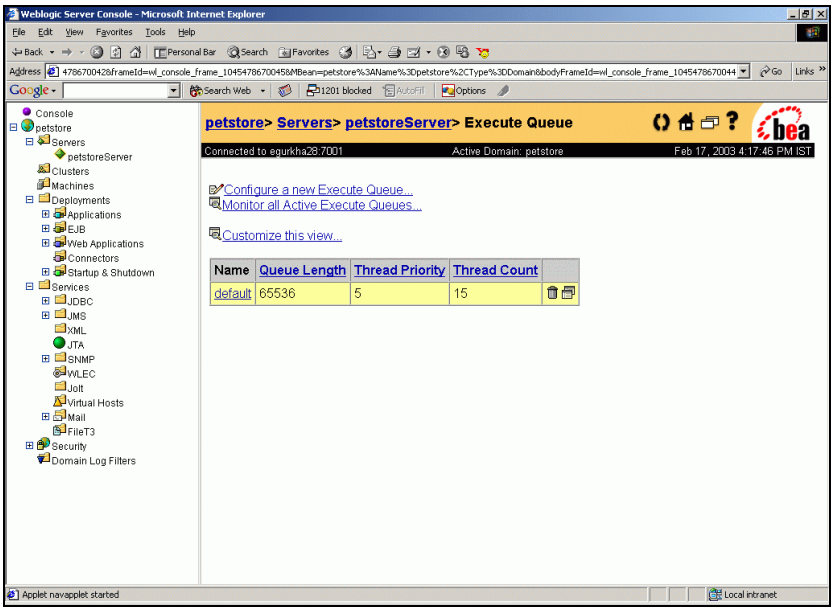


Figure 2.47: Clicking on the Configure a new Execute Queue link

7. When Figure 2.48 appears, specify the **Name** of the new execute queue as **eGQueue** and click the **Create** button to create the queue.

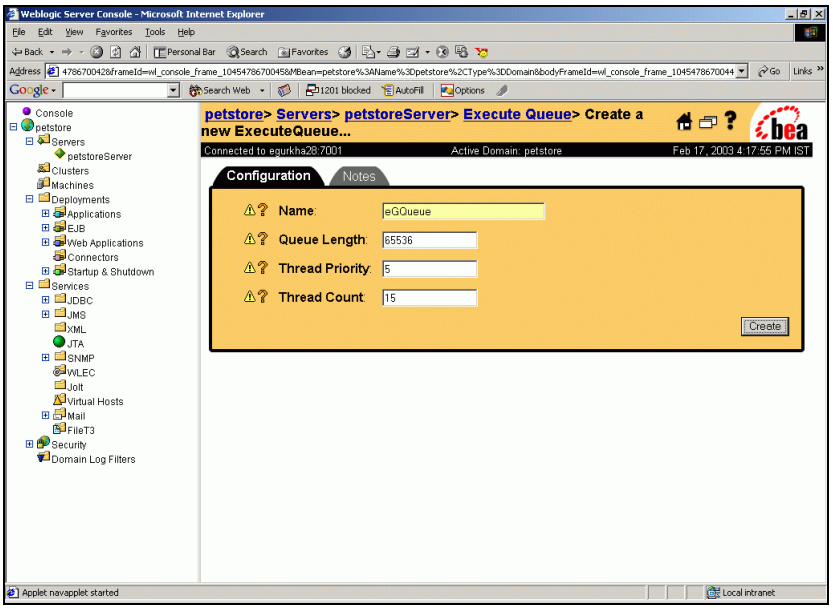


Figure 2.48: Specifying the name of the new execute queue

8. Finally, click on the **Apply** button in Figure 2.49 to apply the changes.

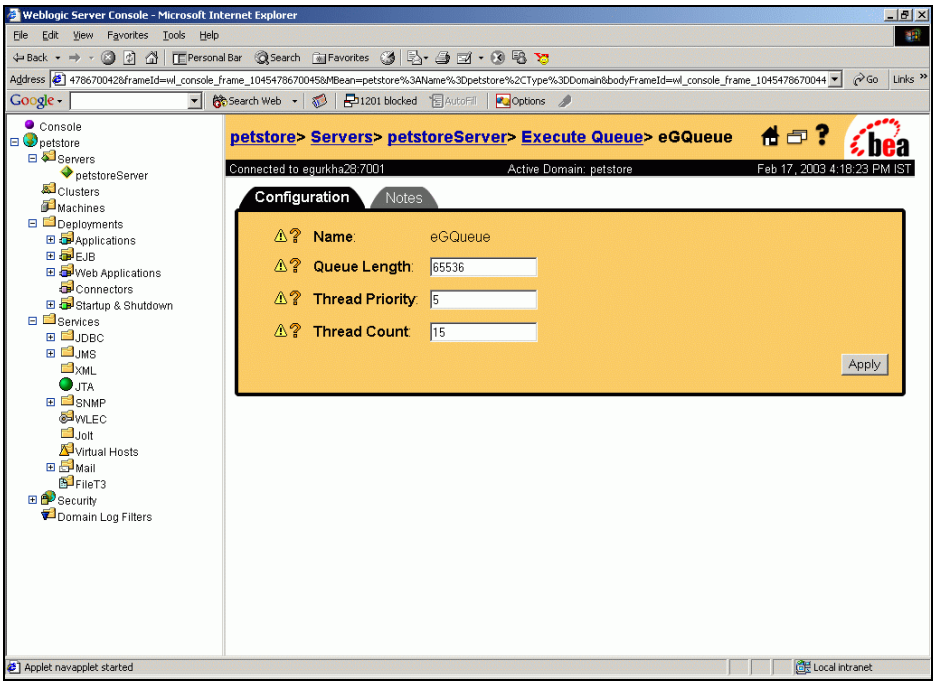


Figure 2.49: Applying the changes

9. Restart the WebLogic server.

Configuring an Execute Queue on WebLogic 7.0

To configure the special execute queue for eG on a WebLogic server 7.0, do the following:

1. Connect to the WebLogic server console using the URL: *http://<WebLogicServerIP>:<WebLogicServerPort>/console*.
2. When prompted for a user name and password, provide the valid details and login to the console. Figure 2.50 then appears.

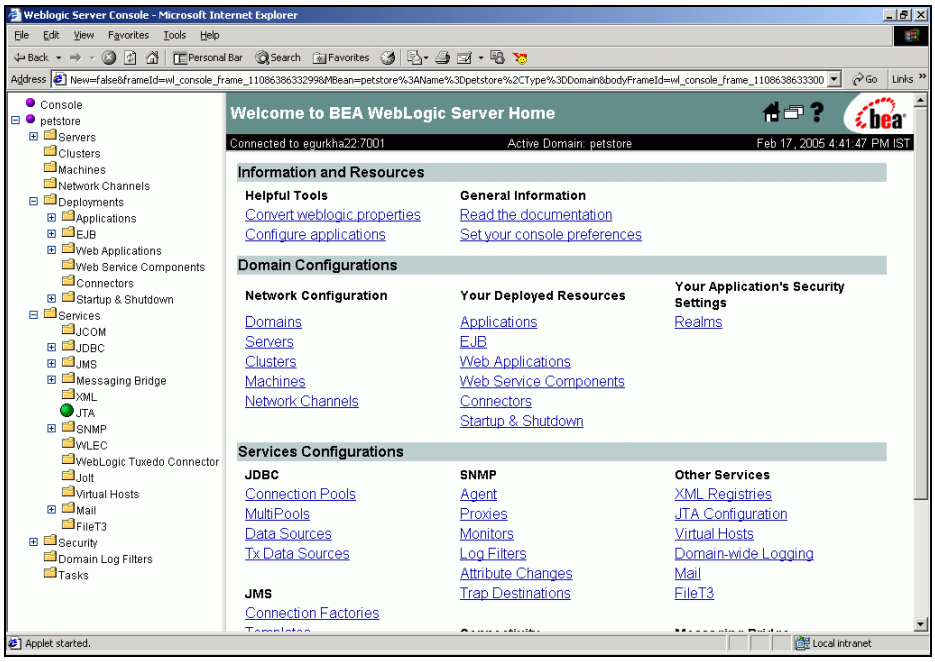


Figure 2.50: The WebLogic 7.0 console

3. Expand the **Servers** node in the tree structure in the left pane of Figure 2.50, and click on the **petstoreServer** entry within. The properties of the selected server will then be displayed in the right pane (see Figure 2.51).

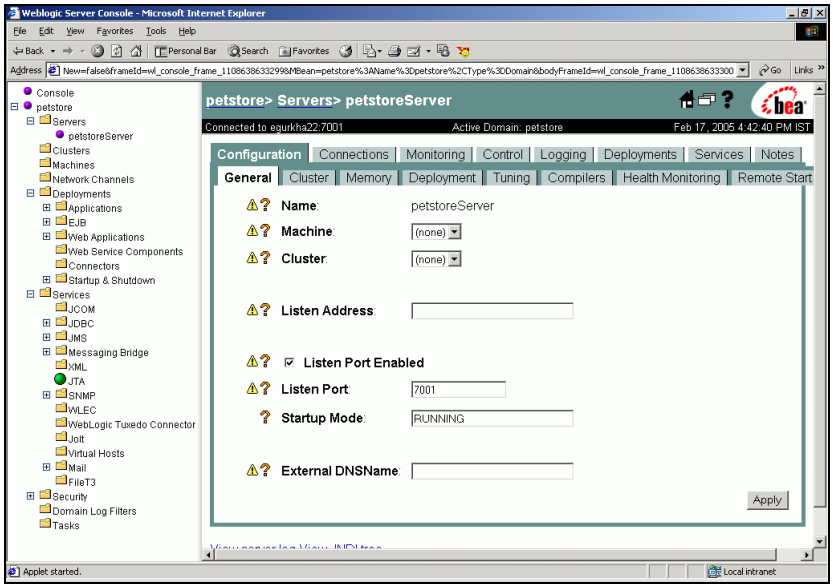


Figure 2.51: Selecting the petstoreServer

4. Click on the **Monitoring** tab in the right pane of Figure 2.51. Then, to configure a new execute queue, click on the **Monitor all Active Queues** link in the **Monitoring** tab of Figure 2.52.

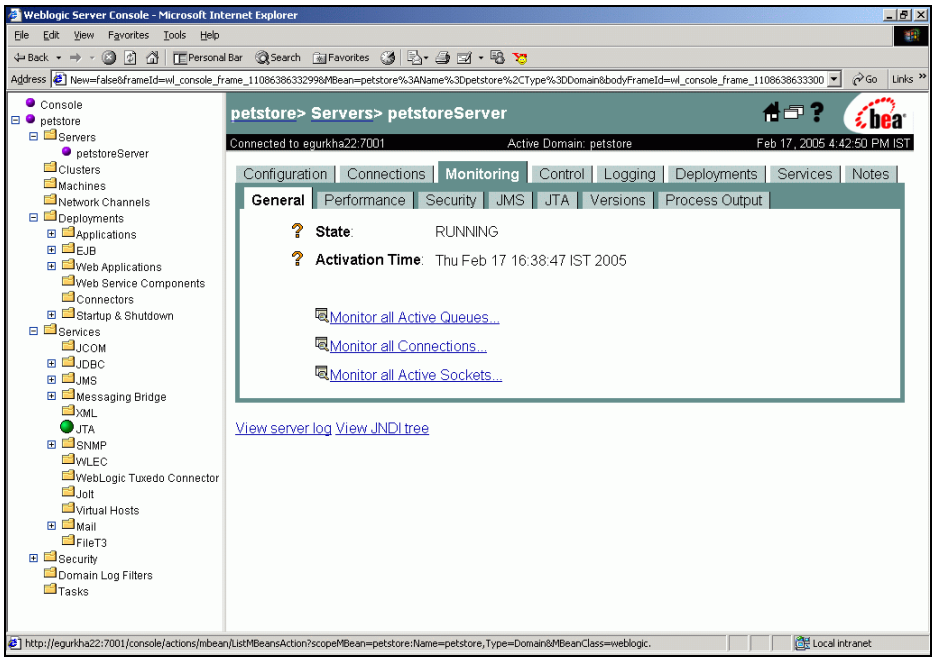


Figure 2.52: Clicking on the Monitor all Active Queues link

5. Figure 2.53 that appears next will list the details of the execute queues on the WebLogic server. To create a new queue, click on the **Configure Execute Queue** link in Figure 2.53.

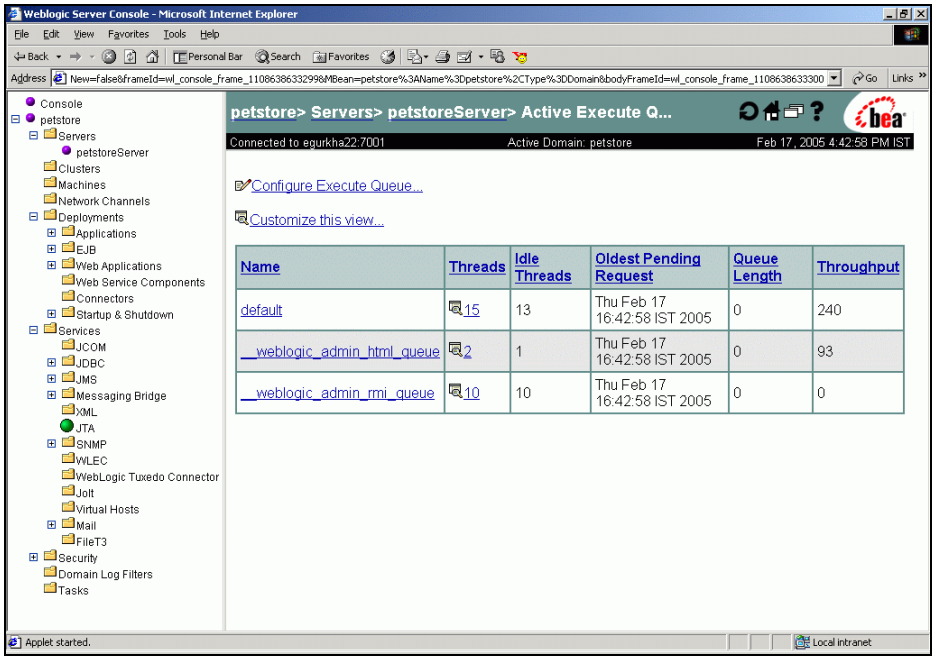


Figure 2.53: Clicking on the Configure Execute Queue link

6. When Figure 2.54 appears, click on the **Configure a new Execute Queue** link in it.

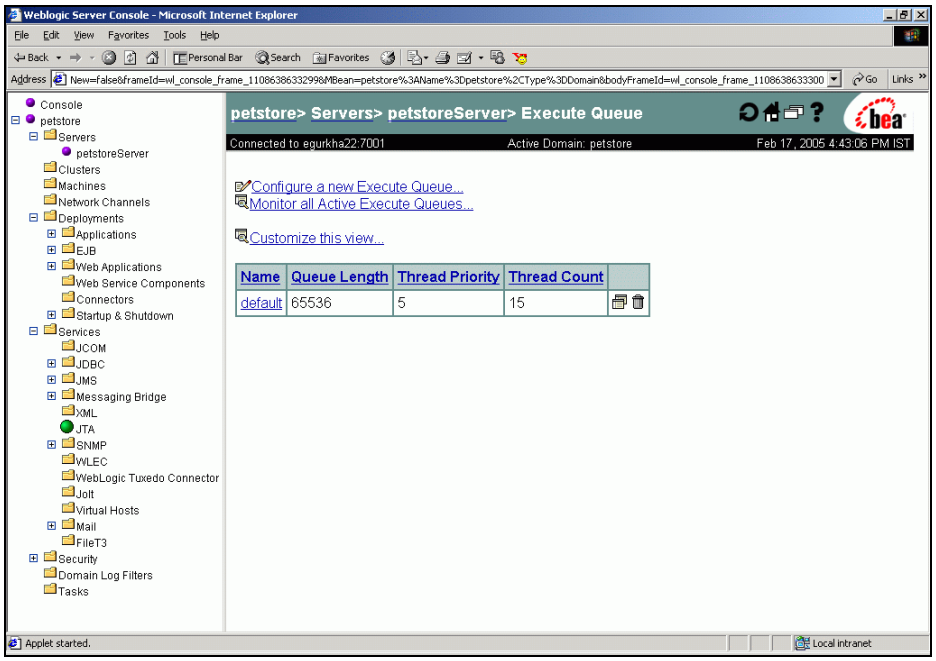


Figure 2.54: Clicking on the Configure a new Execute Queue link

7. In Figure 2.55 that appears, specify **eGQueue** as the **Name** of the new execute queue and click the **Create** button therein to create the new queue.

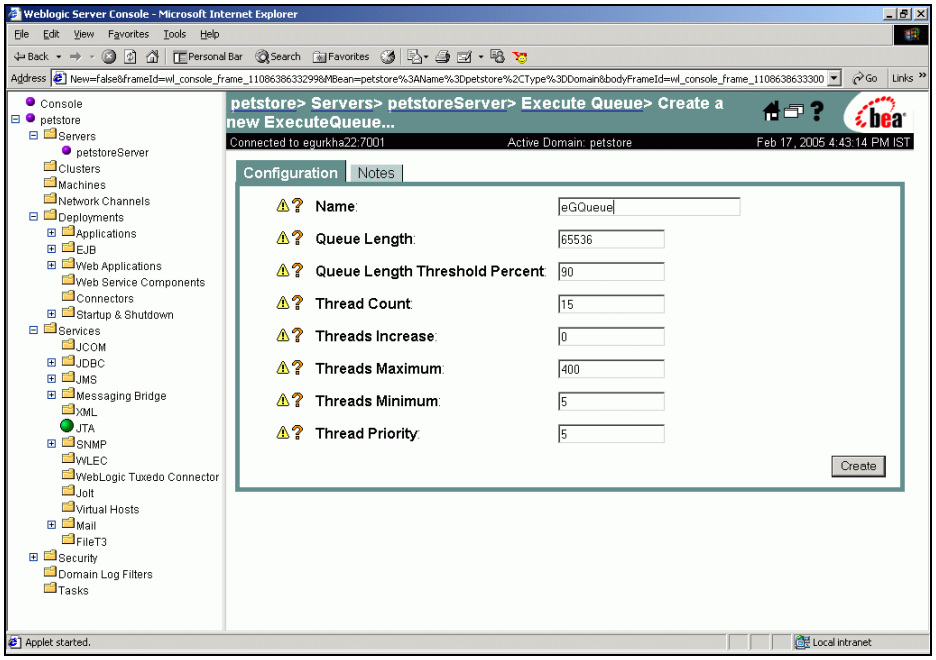


Figure 2.55: Specifying the name of the new execute queue

8. Next, click on the **Apply** button in Figure 2.56 to apply the changes.

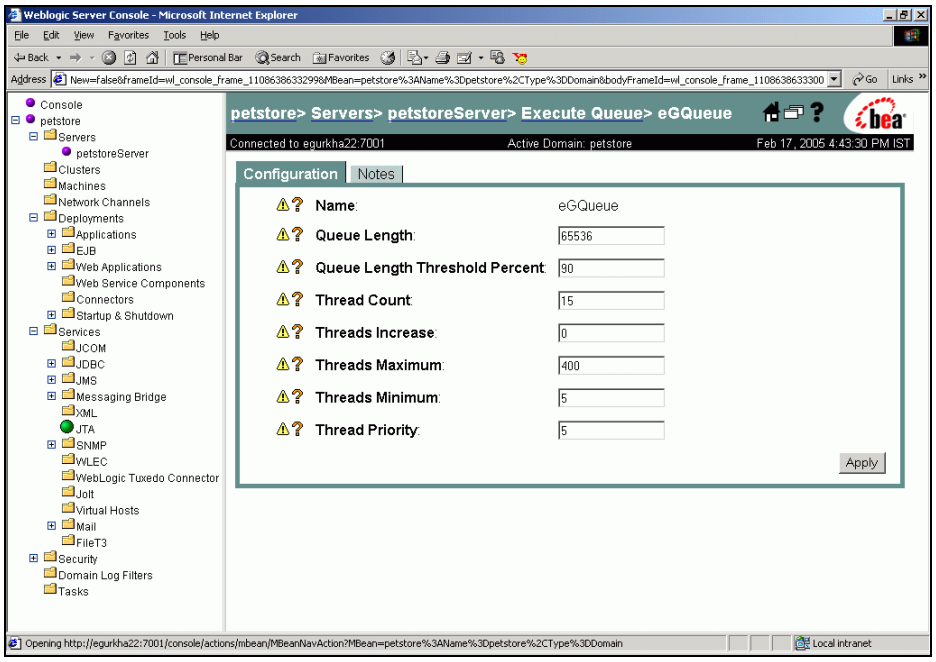


Figure 2.56: Applying the changes

9. Finally, restart the WebLogic server.

Configuring an Execute Queue on a Weblogic Server 8.1

To configure the special execute queue for eG on a WebLogic server 8.1, do the following:

1. Connect to the WebLogic server console using the URL: *http://<WeLogicServerIP>:<WebLogicPort>*.
2. When prompted for a user name and password, provide the valid details and login to the console.
3. Expand the **Servers** node in the tree structure in the left pane of the WebLogic console, and click on the **MedRecServer** entry within. The properties of the selected server will then be displayed in the right pane. Now, click on the **Monitoring** tab in the right pane and click on the **Monitor all Active Queues** link within (see Figure 2.57).

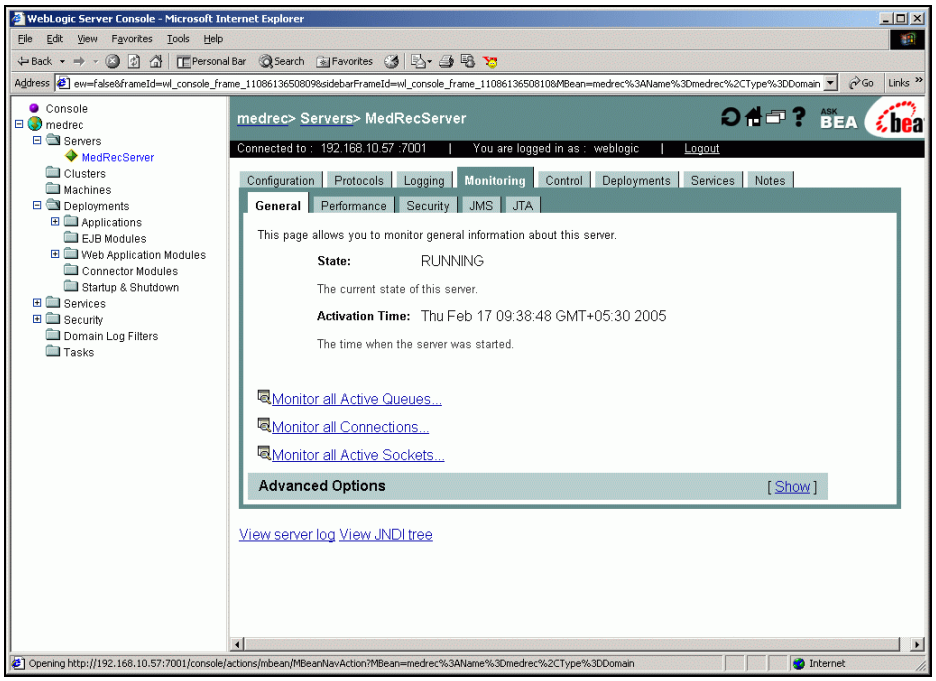


Figure 2.57: Selecting the MedRecServer and clicking on the Monitor all Active Queues link

4. Figure 2.58 that appears next will list the details of the execute queues on the WebLogic server. To create a new queue, first click on the **Configuration** tab in the right pane of Figure 2.58.

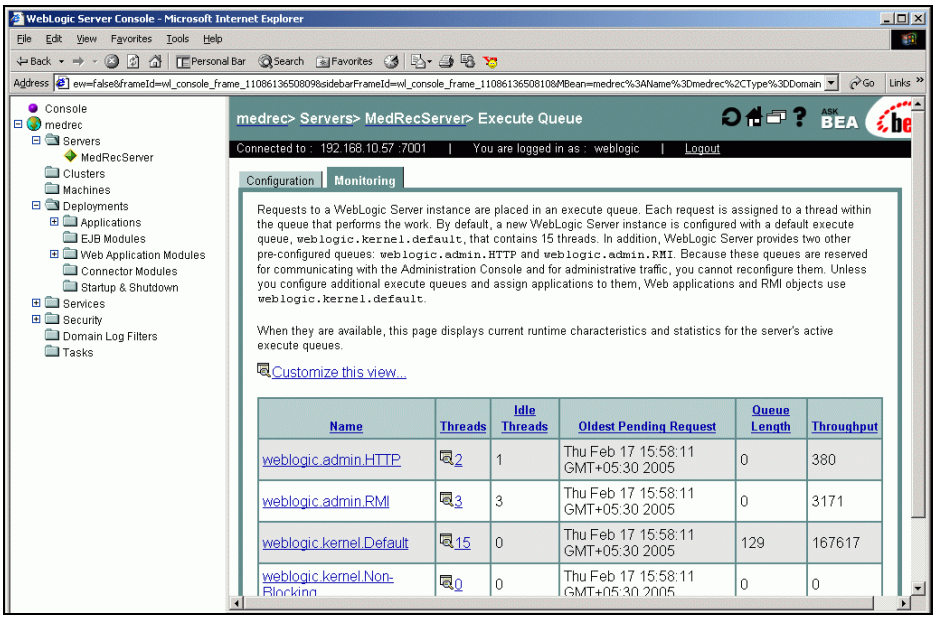


Figure 2.58: Clicking on the Configure Execute Queue link

5. When Figure 2.59 appears, click on the **Configure a new Execute Queue** link in it.

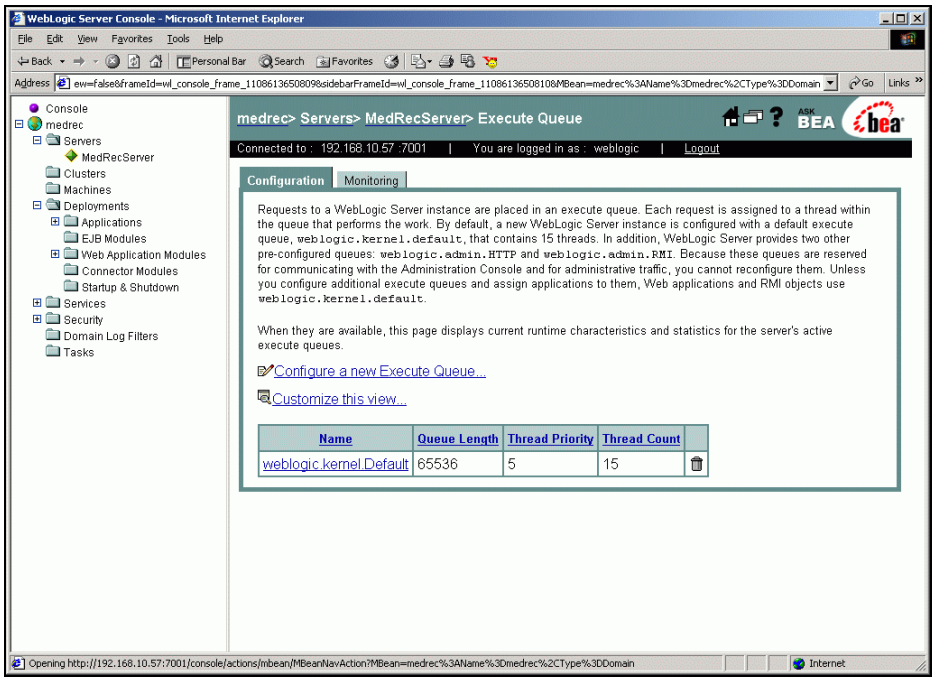


Figure 2.59: Clicking on the Configure a new Execute Queue link

6. In Figure 2.60 that appears, specify **eGQueue** as the **Name** of the new execute queue and click the **Create** button therein to create the new queue.

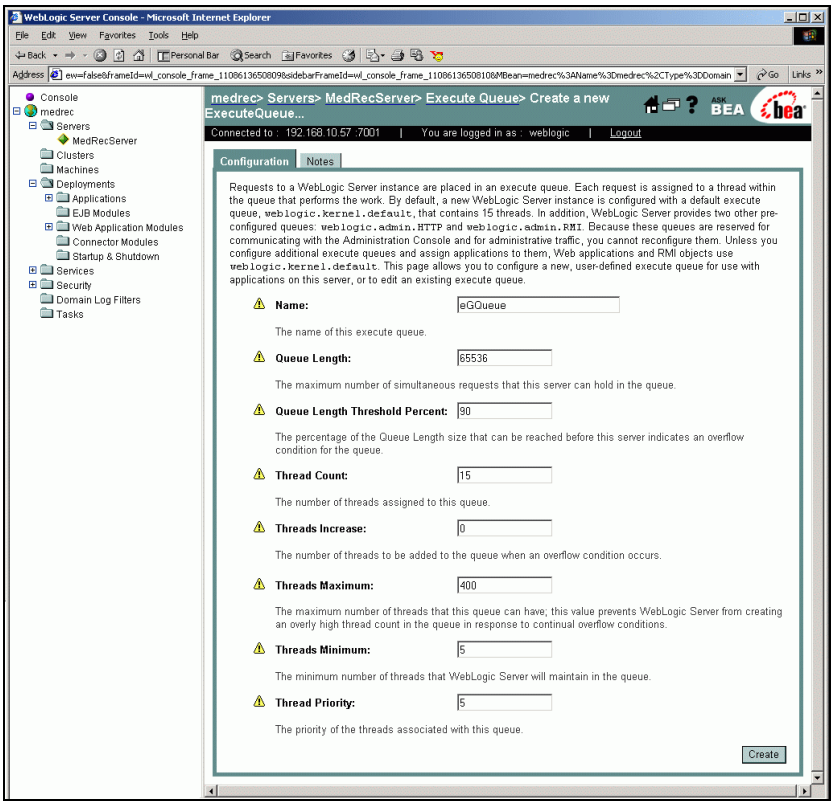


Figure 2.60: Specifying the name of the new execute queue

7. Next, click on the **Apply** button in Figure 2.61 to apply the changes.

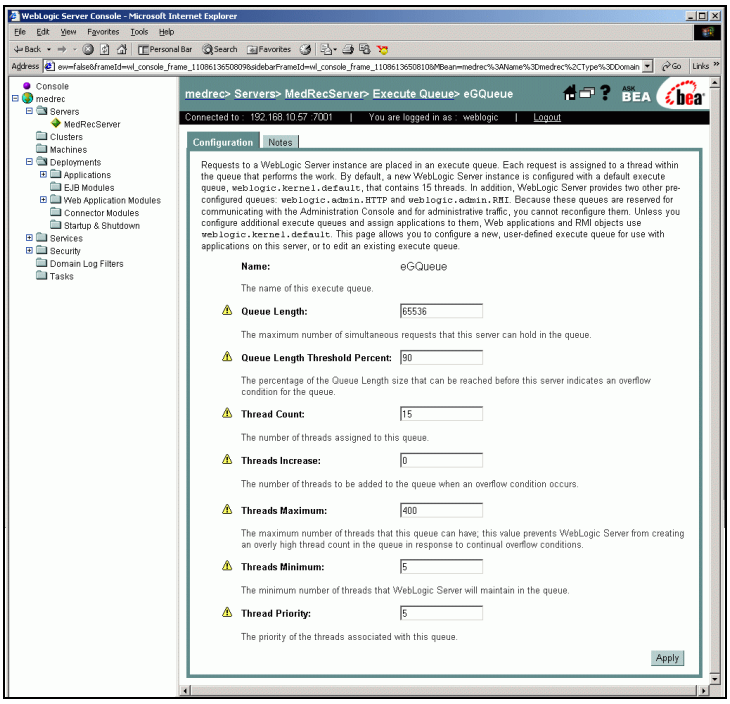


Figure 2.61: Applying the changes

8. Finally, restart the WebLogic server.

2.1.8 Deploying the 'egurkha.war' file on a WebLogic Portal server

To deploy the **egurkha.war** file on a WebLogic Portal server, do the following:

1. Connect to the WebLogic portal server console using the URL: *http://<WebLogicIP>:<WebLogicPort>.*
2. When prompted for a user name and password, provide the valid details and login to the console. Figure 2.62 then appears.

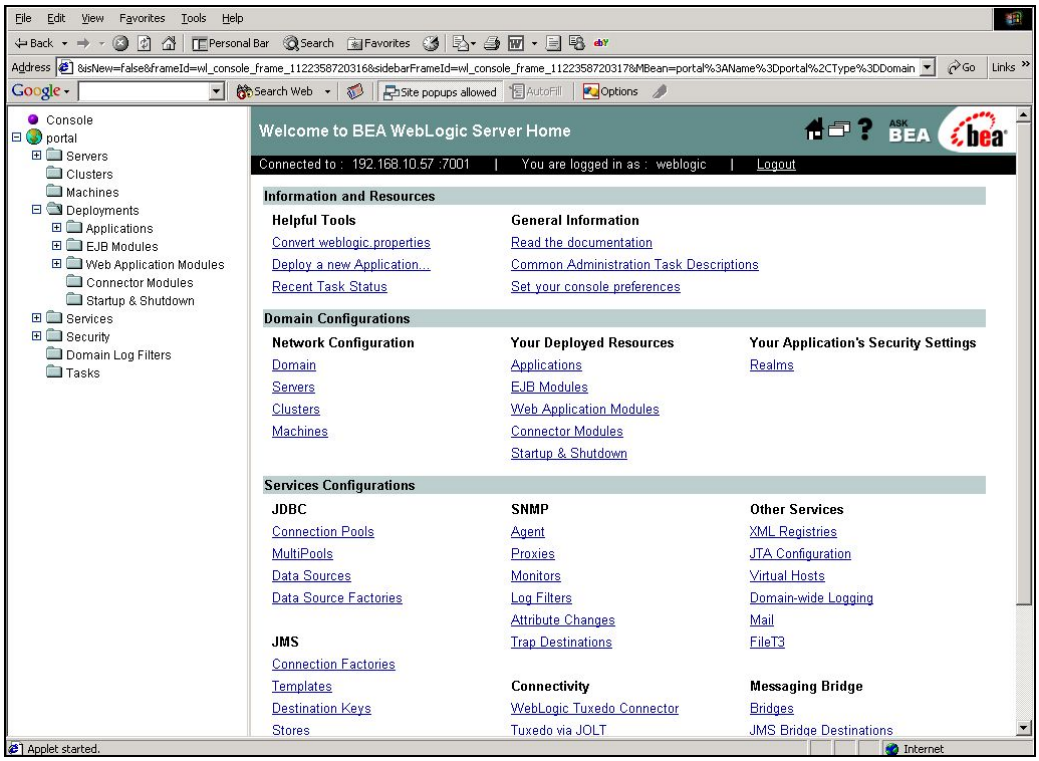


Figure 2.62: The WebLogic console

3. Expand the portal server node in the tree-structure in the left pane of Figure 2.62, and from within, click on the Deployments -> Web Application Modules node. Then, click on the **Deploy a new Web Application Module** hyperlink in the right pane of Figure 2.63.

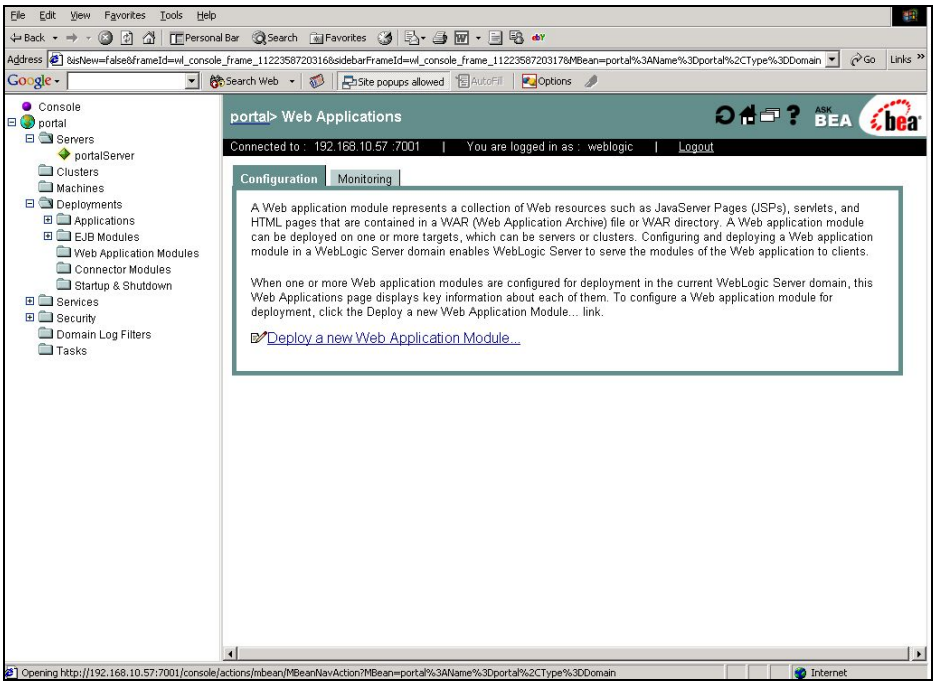


Figure 2.63: The Web Application Modules page

4. From the right pane of Figure 2.64 that appears, select the portal server on which **egurkha** application is to be deployed by clicking on it.

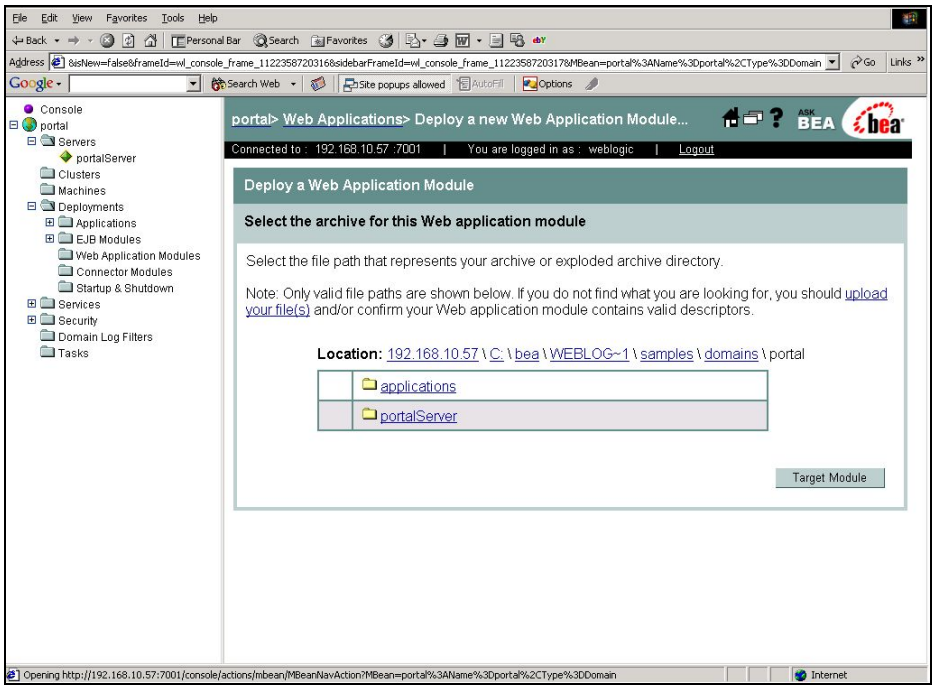


Figure 2.64: Selecting the portalServer link

5. Figure 2.65 then appears. To upload the **egurkha.war** file, click on the **upload your file(s)** link therein.

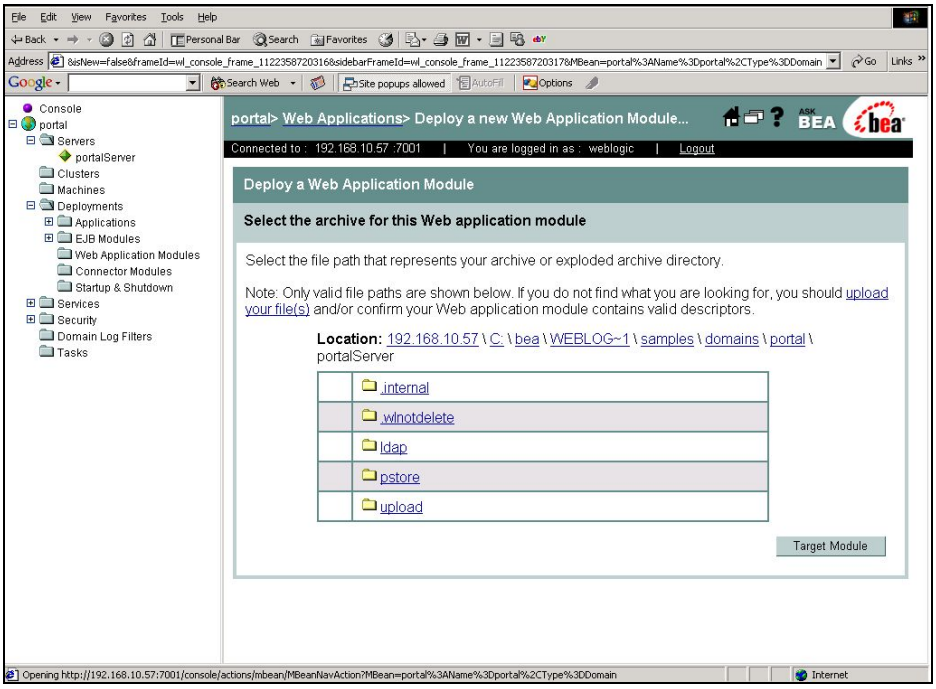


Figure 2.65: Clicking on the 'upload your file(s)' link

6. Use the **Browse** button in Figure 2.66 to locate the **egurkha.war** file to be uploaded to the portal server. Once the path is specified, click on the **Upload** button in Figure 2.66 to upload the file.

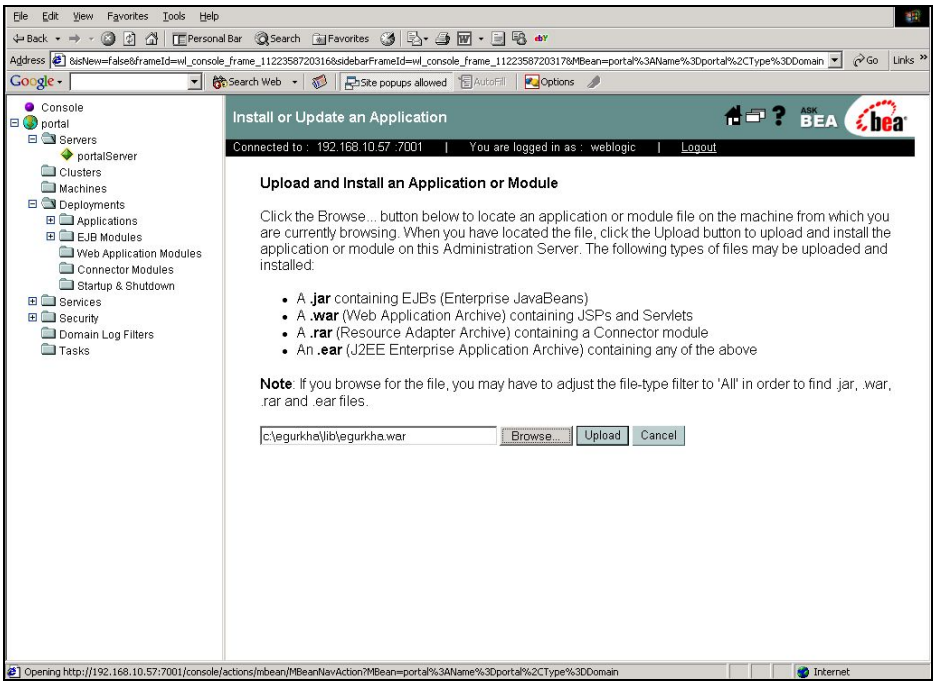


Figure 2.66: Specifying the full path the war file to be uploaded

7. The uploaded file will then be assigned to the chosen portal server as shown by Figure 2.67. Click on the radio button against **egurkha.war** and click on the **Target Module** button in Figure 2.67.

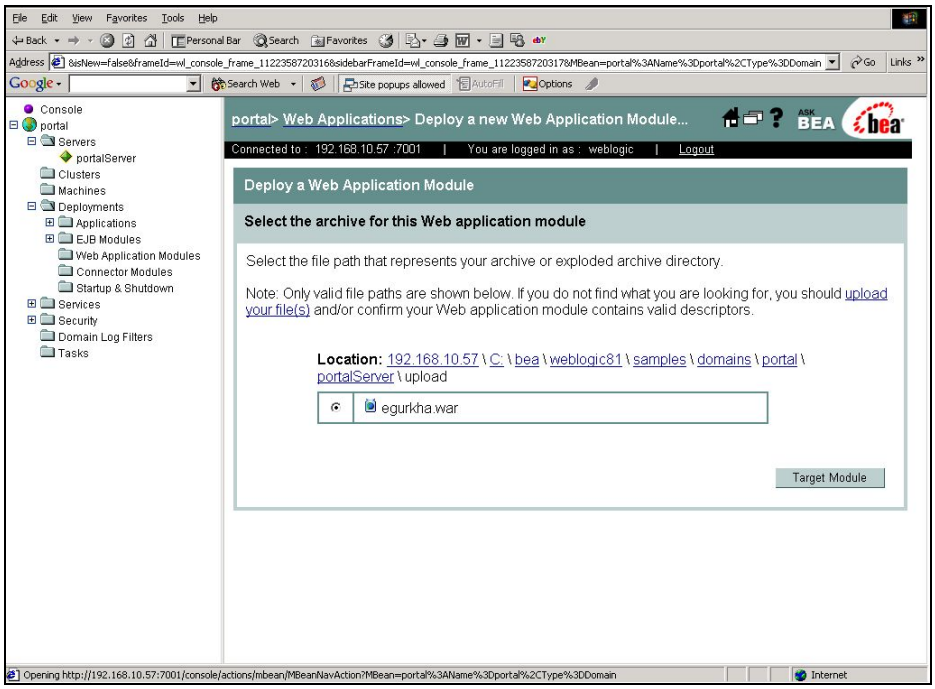


Figure 2.67: Setting the 'egurkha.war' application as the target module

8. In Figure 2.68, the **Name** of the uploaded application (**egurkha**) will be displayed. Now, click on the **Deploy** button to deploy the displayed application on the chosen portal server.

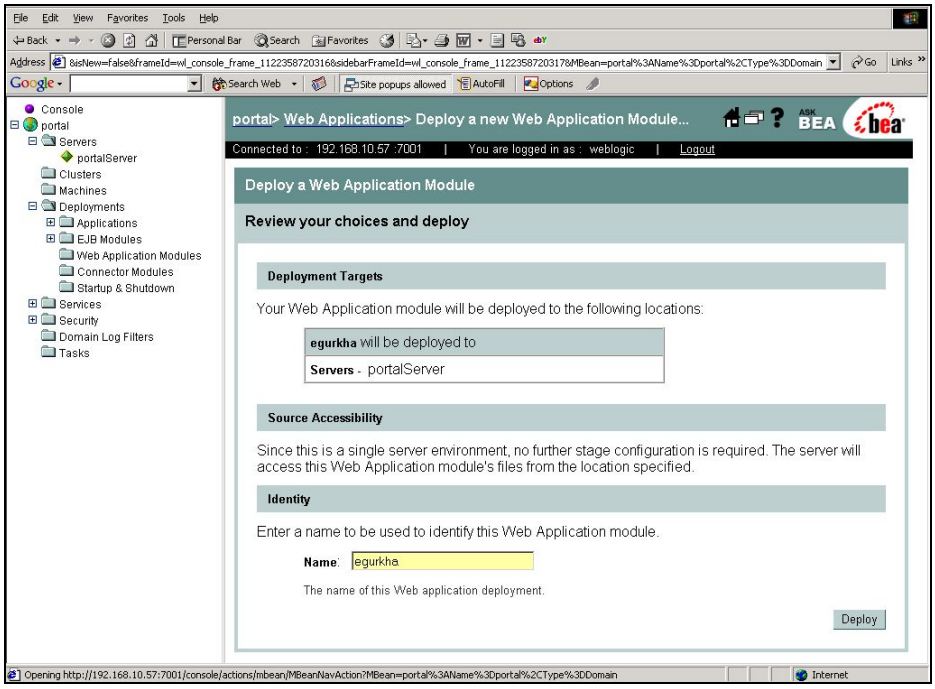


Figure 2.68: Providing a name for the web application to be deployed

9. Upon successful deployment of the **egurkha** application on the portal server, Figure 2.69 will appear indicating the same.

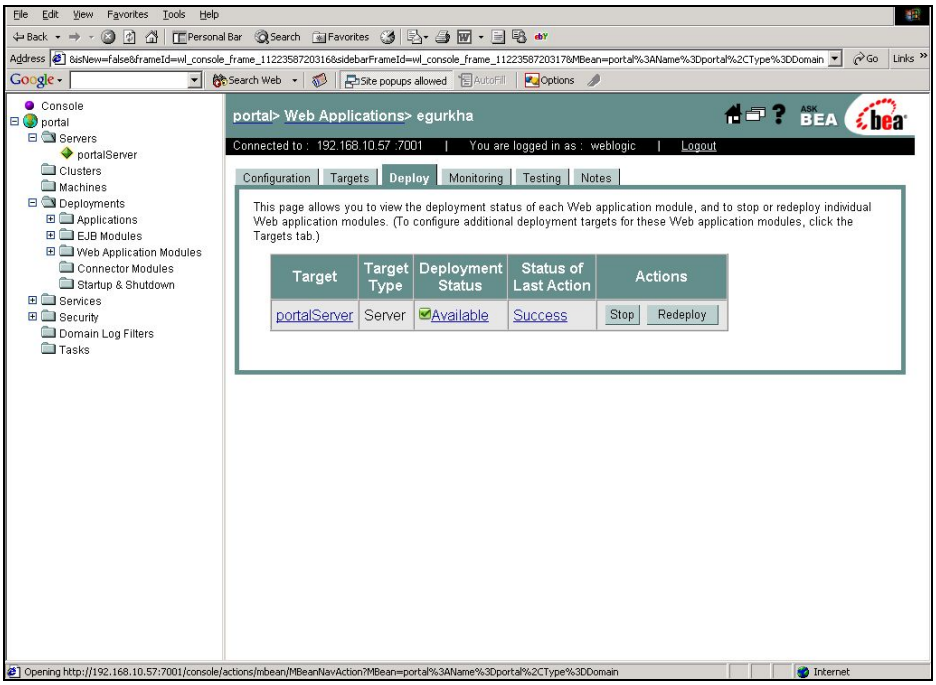


Figure 2.69: Successful deployment of the egurkha application

10. For further confirmation of the successful deployment, expand the **Web Application Modules** node in the tree-structure in the left pane of Figure 2.70. A sub-node named **egurkha** will be available within.

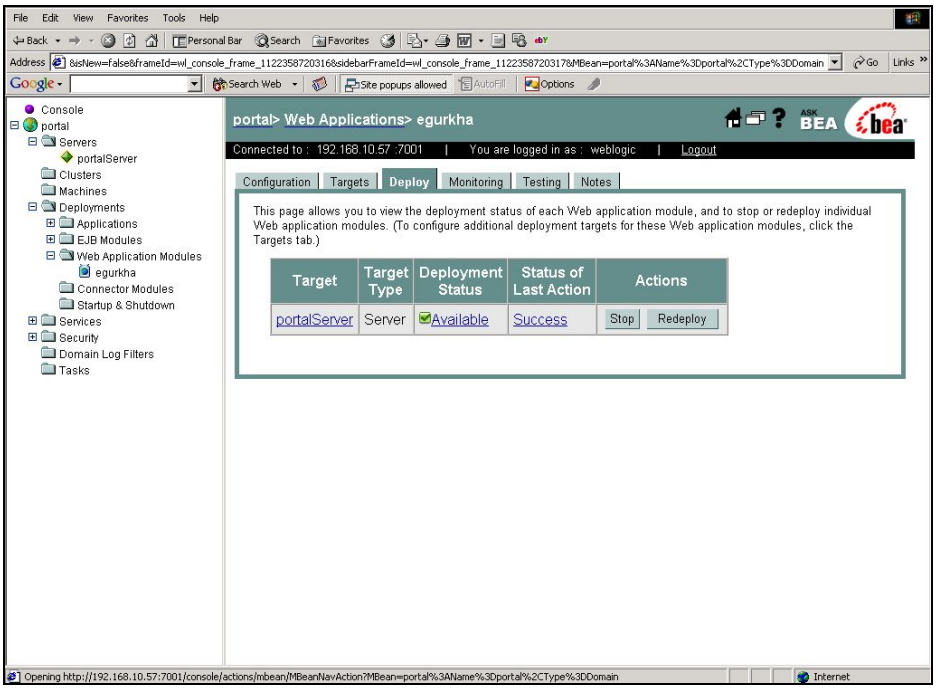


Figure 2.70: The 'egurkha' sub-node appearing under the Web Application Modules node

2.2 Managing the Oracle WebLogic Server Version 6/7/8

The eG Enterprise cannot automatically discover the Oracle WebLogic Server Version 6/7/8. This implies that you need to manually add the component for monitoring. Remember that the eG Enterprise automatically manages the components that are added manually. To manage a Oracle WebLogic 6/7/8 component, do the following:

1. Log into the eG administrative interface.
2. Follow the Components -> Add/Modify menu sequence in the **Infrastructure** tile of the **Admin** menu.
3. In the **COMPONENT** page that appears next, select Oracle WebLogic (6/7/8) as the **Component type**. Then, click the **Add New Component** button. This will invoke 2.2.

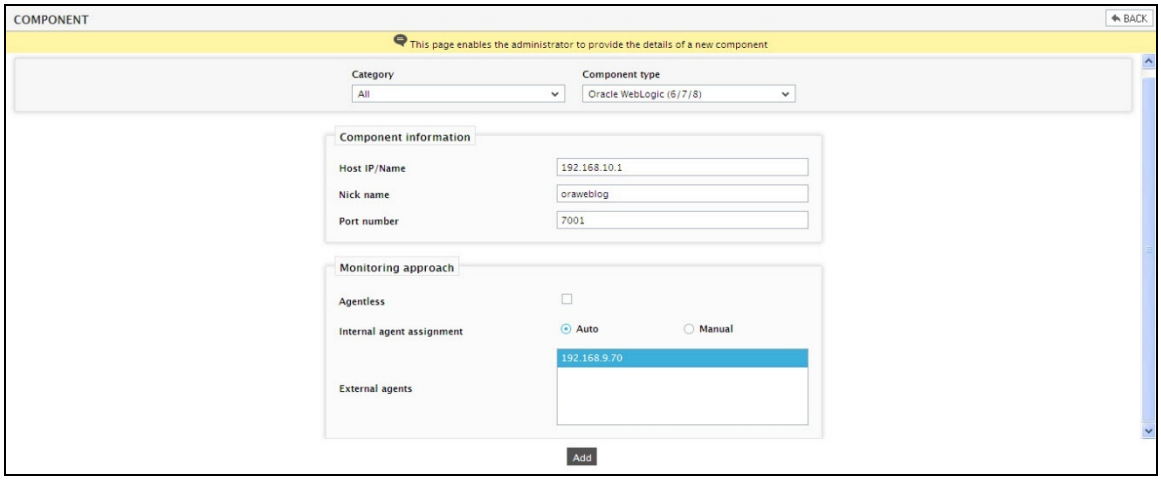


Figure 2.71: Adding the Oracle WebLogic (6/7/8) server

4. Specify the **Host IP/Name** and **Nick name** for the Oracle WebLogic (6/7/8) server (see 2.2). Then, click on the **Add** button to register the changes.
5. When you attempt to sign out, a list of unconfigured tests appears.

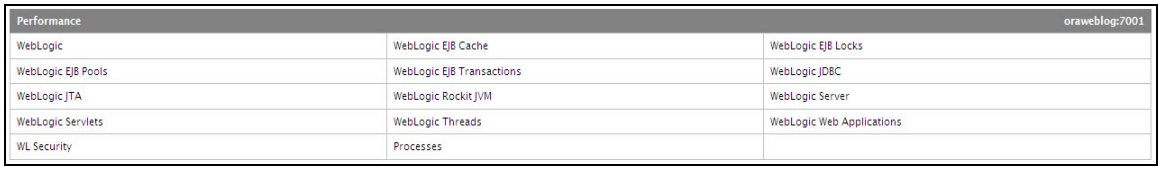


Figure 2.72: List of unconfigured tests for WebLogic (6/7/8) server

6. Click any test in the list of unconfigured tests. For instance, click on the **WebLogic EJB**

Transactions test to configure it. In the page that appears, specify the parameters as shown in Figure 2.73.

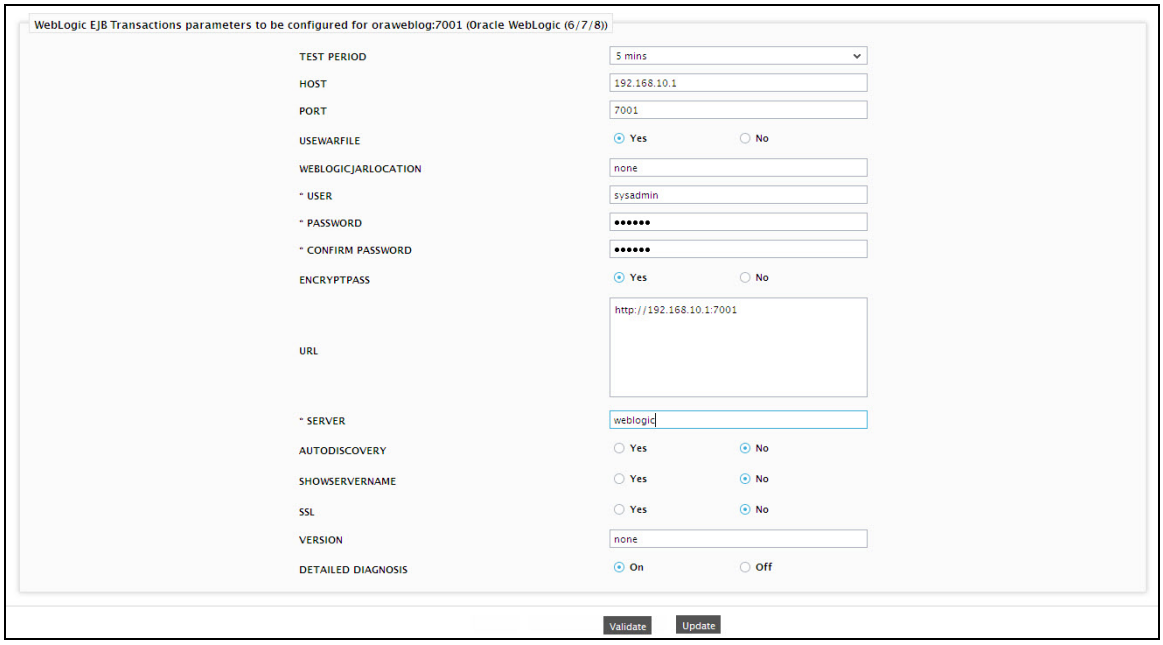


Figure 2.73: Configuring Weblogic EJB Transactions test

7. To know how to configure the parameters, refer to **Monitoring the WebLogic Server Ver. 6/7/8** chapter.
8. Next, try to sign out of the administrative interface. It will prompt you to configure the **Processes** test. Refer to *Monitoring Unix and Windows Servers* document for the details on configuring the **Processes** test.
9. Finally, sign out of the eG administrative interface.

2.3 Pre-requisites for Monitoring Oracle WebLogic Application Server Version 9 (and above)

To monitor the Oracle WebLogic Application Server Version 9 (and above), a set of pre-requisites should be fulfilled. These requirements are discussed in the following sections;

2.3.1 Configuring Oracle WebLogic Application Server 9.2

The eG agent package contains a web archive file called **egurkha9x.war**. On Unix environments, this file will be available in the /opt/egurkha/lib location. On Windows environment, this file will be available in the <EG_HOME_DIR>\lib directory. This file has to be installed on the WebLogic server

ver. 9 (and above), which has to be monitored. Prior to deploying this war file, make sure that you follow the steps below:

- Login to the system hosting the eG agent that is monitoring the WebLogic server.
- Copy the **egurkha9x.war** file from the /opt/egurkha/lib directory to any other directory on the eG agent host.
- Rename the war file as **egurkha.war**.

Next, follow the steps given below to deploy the war file:

1. Ensure that the WebLogic application server to be monitored has been started.
2. Next, go to the following URL: *http://<WebLogic_server_IP>:<WebLogicPort>/*.

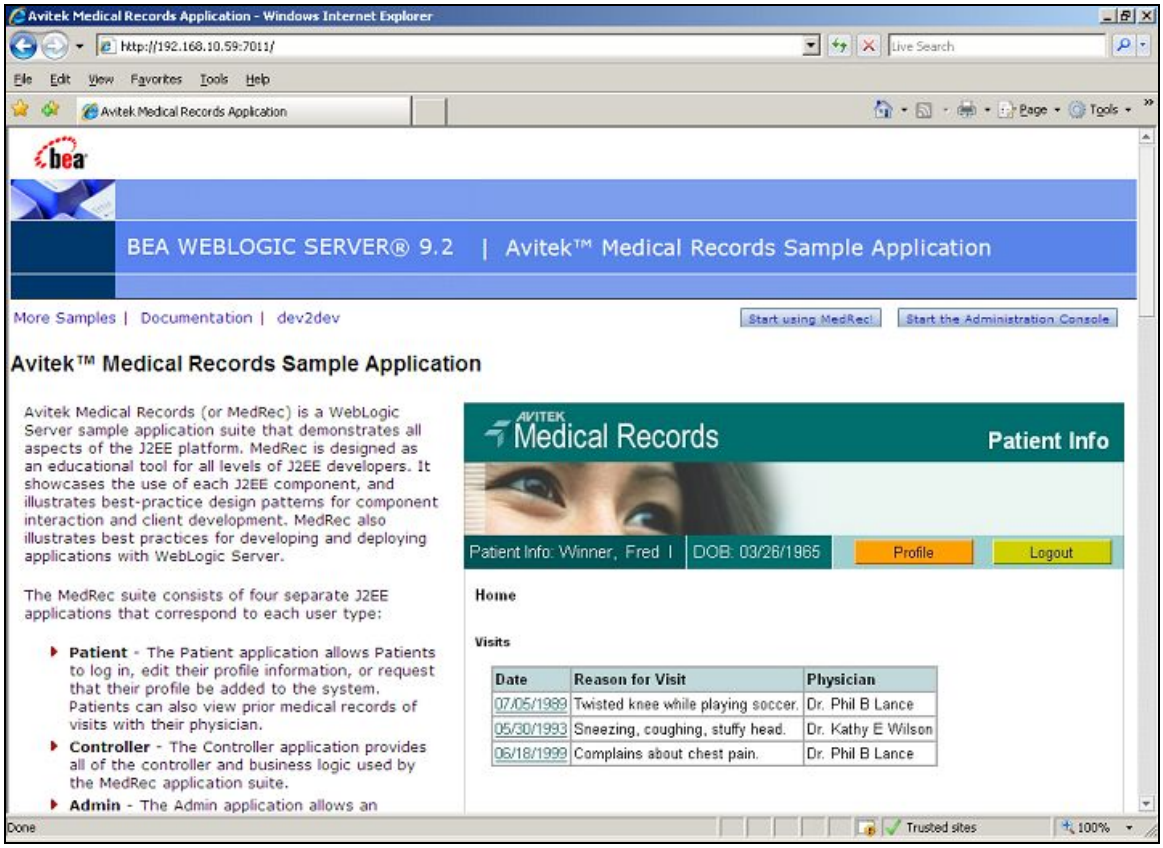


Figure 2.74: Connecting to the WebLogic console

3. To begin application deployment, click on the **Start the Administration Console** button in Figure 2.74 that appears next.
4. Now, an authentication dialog (see Figure 2.75) that requires a username and password, will appear. Specify the user name and password of the WebLogic administrator to login.

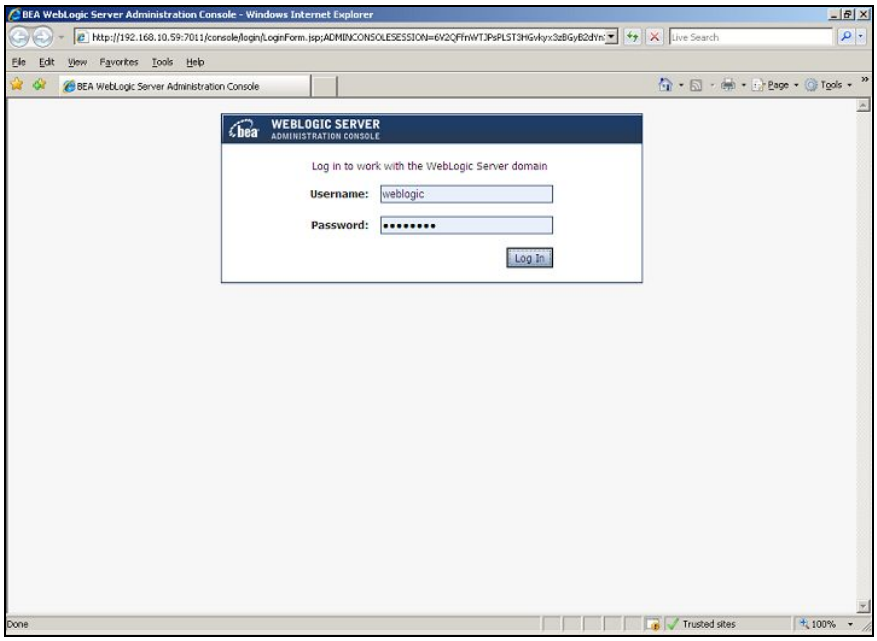


Figure 2.75: Specifying user name and password in the authentication dialog box

5. Upon successful authentication, the following WebLogic 9.2 Administration console will appear (see Figure 2.76).

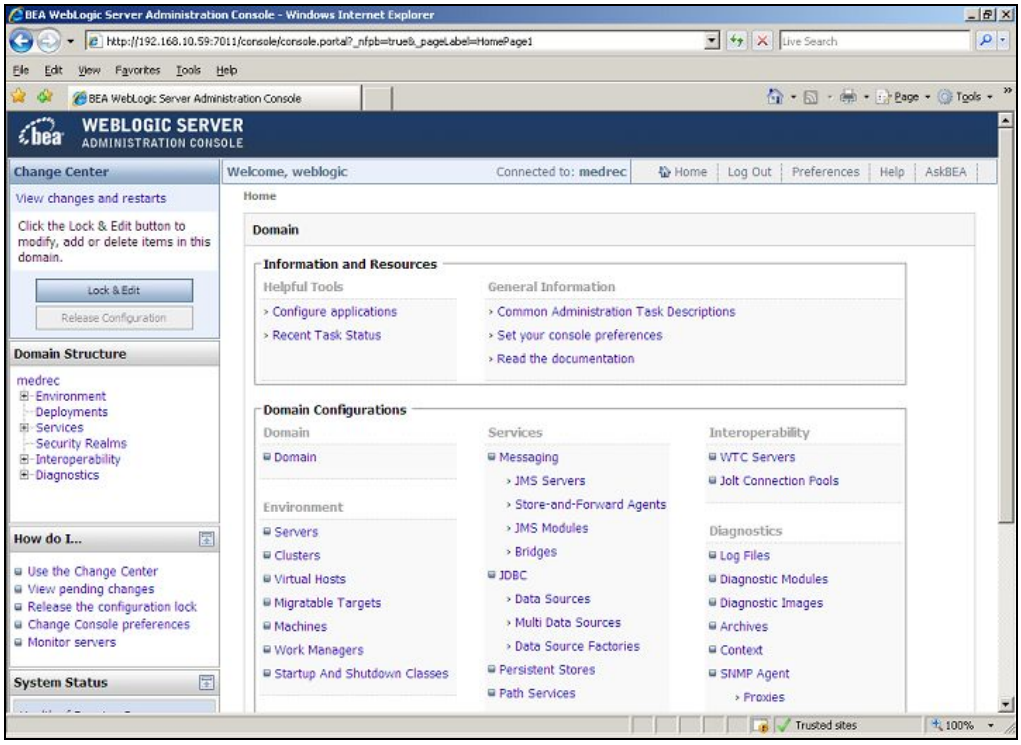


Figure 2.76: The Administration Console of the Oracle WebLogic server 9.2

6. From the **Domain Structure** section in the left panel of Figure 2.76, choose the **Deployments** node. A **Summary of Deployments** follows (see Figure 2.77), which lists all the applications that pre-exist on the WebLogic server (see the right panel).

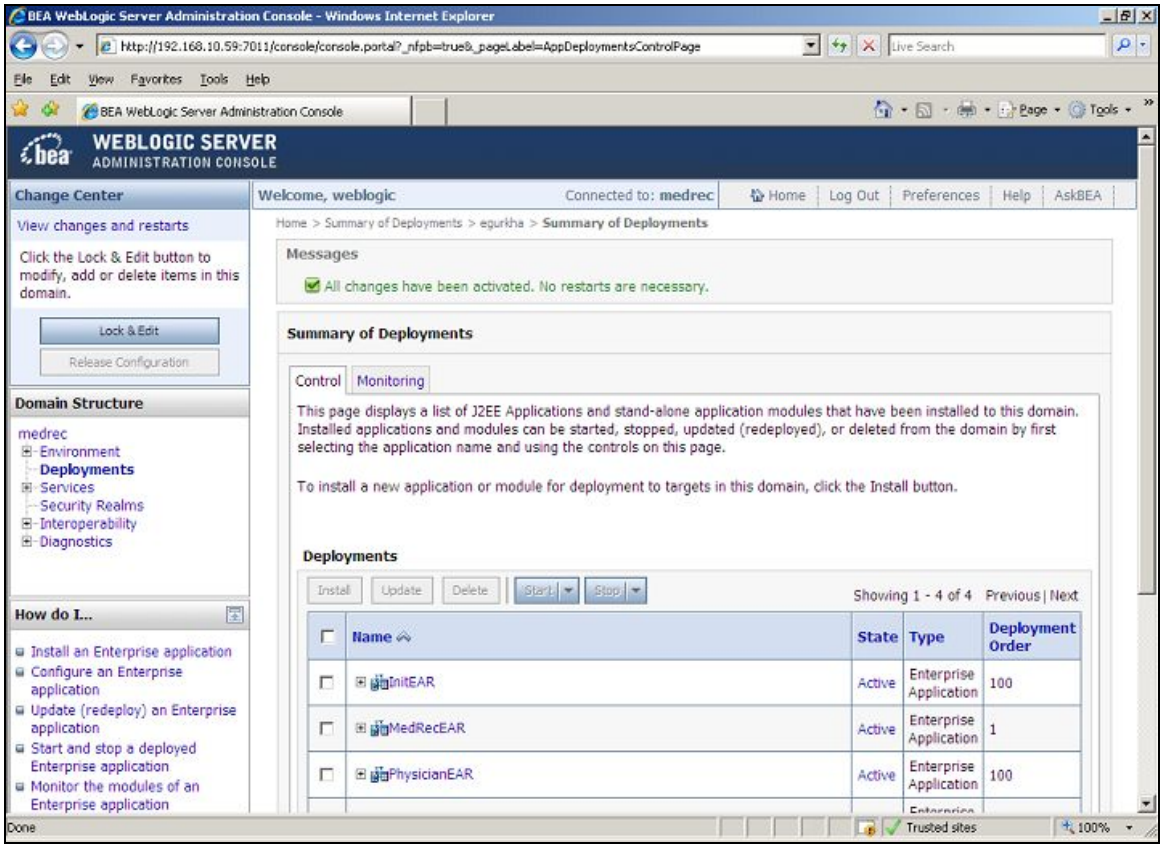


Figure 2.77: Screen displaying deployment summary

7. To add a new application, you first need to change to the *Edit* mode. To achieve this, click on the **Lock & Edit** button under the **Change Center** section in the left panel of Figure 2.77. Following which, the **Install** button in the **Deployments** section of the right panel, gets enabled (see Figure 2.78).

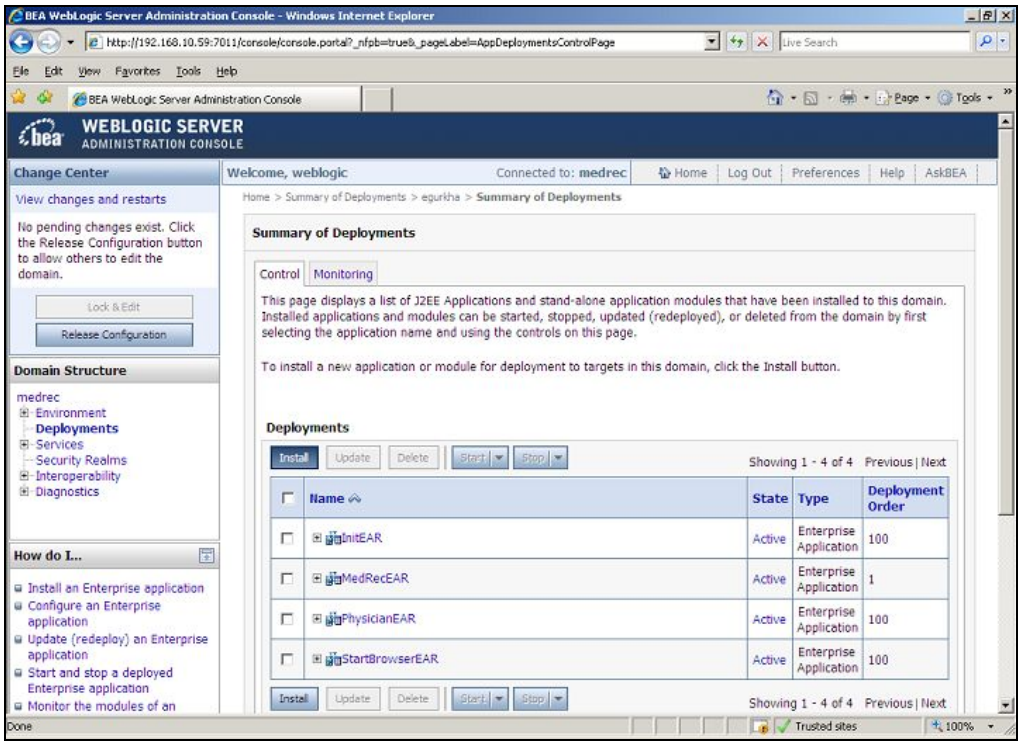


Figure 2.78: Switching to the Lock & Edit mode

8. To begin application deployment, click on the enabled **Install** button in the right panel of Figure 2.78 above. Next, using the slew of screens that appear one after another, browse for and specify the full path to the **egurkha.war** file (path: <EG_INSTALL_DIR>\agent\lib) to be deployed (see Figure 2.79, Figure 2.80, Figure 2.81, and Figure 2.82.

   Note:

   By default, the **egurkha.war** file will be searched for in the local disk of the system from which the browser is launched. Therefore, if you are trying to access the WebLogic console from a system that does not host an eG agent, then, before attempting to deploy the **egurkha.war** file, do the following:

   - Login to any agent host in your environment.
   - Copy the **egurkha9x.war** file in the <EG_AGENT_INSTALL_DIR>\libdirectory (on Windows; on Unix, this will be the /optegurkha/libdirectory) of the agent host to any other directory on that host.
   - Rename the **egurkha9x.war** file as **egurkha.war** file.
   - Copy the **egurkha.war** file to the local host from which the browser is launched.

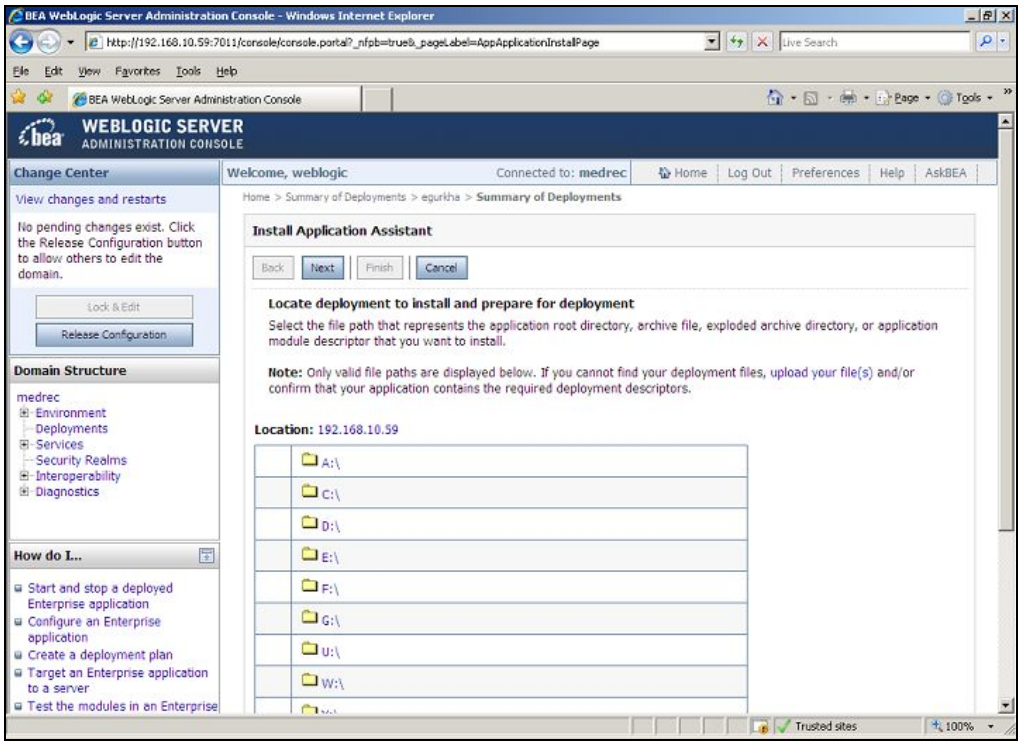


Figure 2.79: Specifying the full path to the egurkha.war file

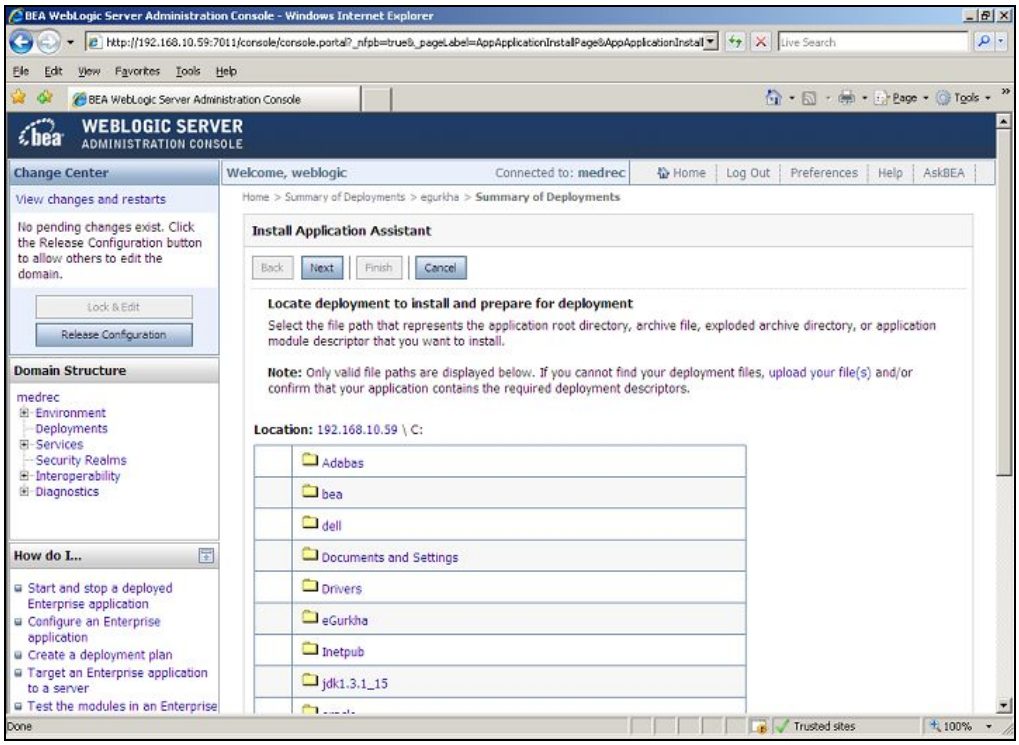


Figure 2.80: Specifying the full path to the egurkha.war file

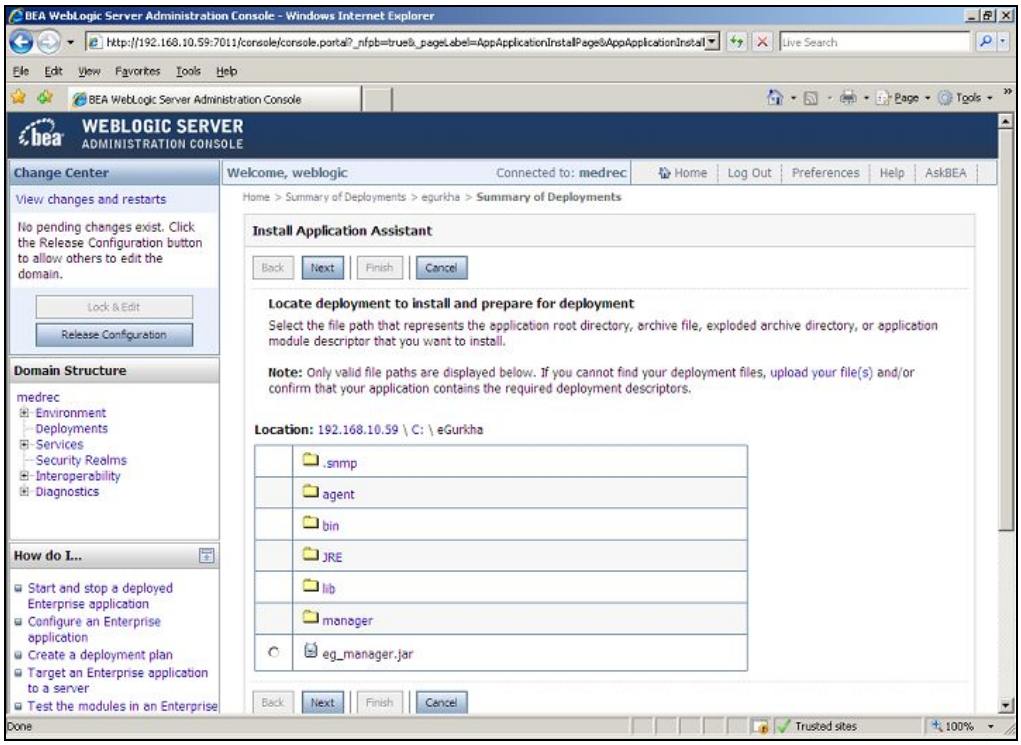


Figure 2.81: Specifying the full path to the egurkha.war file

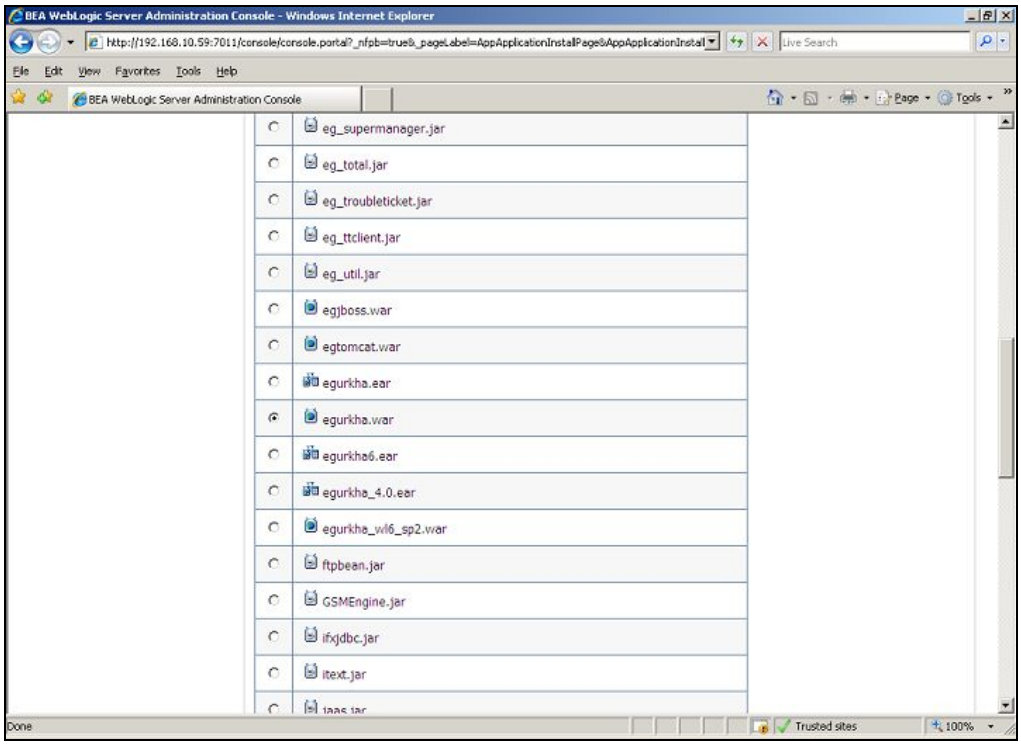


Figure 2.82: Specifying the full path to the egurkha.war file

9. Once the **egurkha.war** option is chosen from Figure 2.82, click the **Next** button (not visible in Figure 2.82) to complete the path specification.
10. Select the **Install this deployment as an application** option in Figure 2.83 that appears next, and then, click the **Next** button to move on.

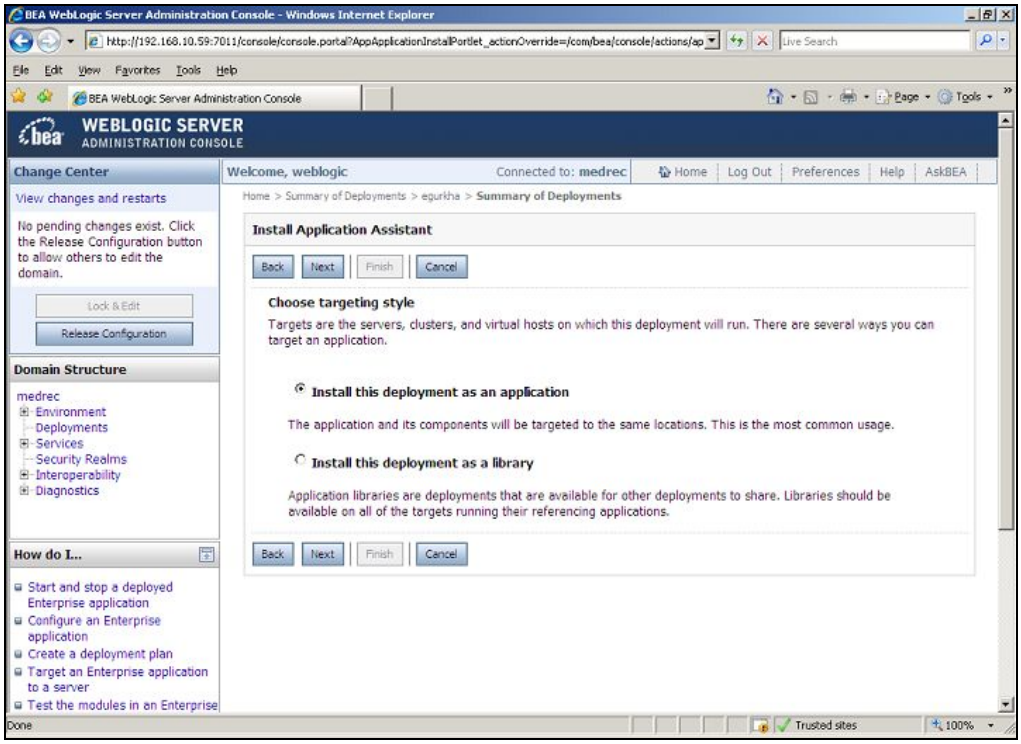


Figure 2.83: Choosing to install the deployment as an application

11. Provide a **Name** for the new application in Figure 2.84. By default, **egurkha** will be displayed therein. Then, click on the **Next** button to proceed.

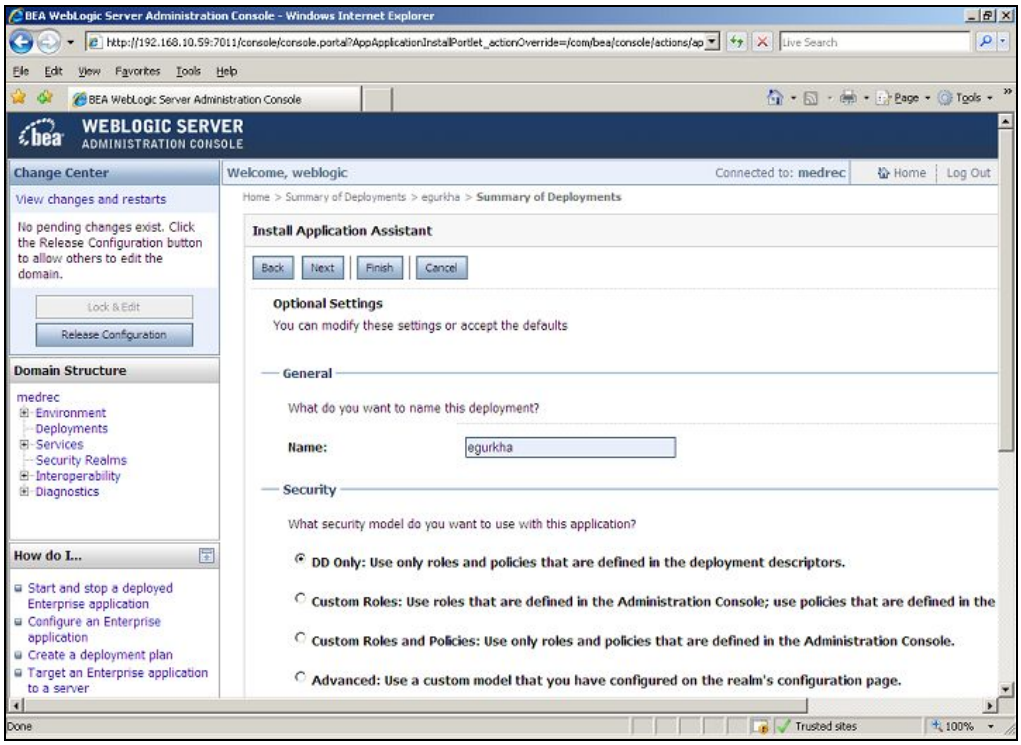


Figure 2.84: Providing a name for the application

12. To complete the application deployment, click on the **Finish** button in Figure 2.85.

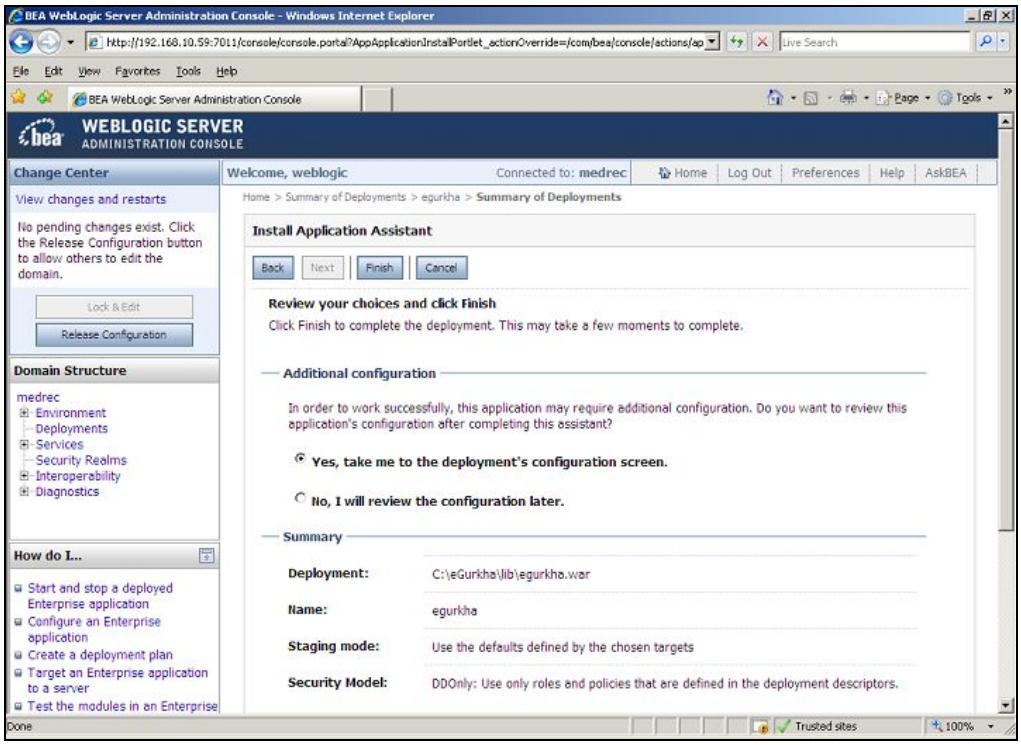


Figure 2.85: Completing the application deployment

13. The existing configuration settings for the new application is then made available for review. To save the settings, click on the **Save** button in Figure 2.86.

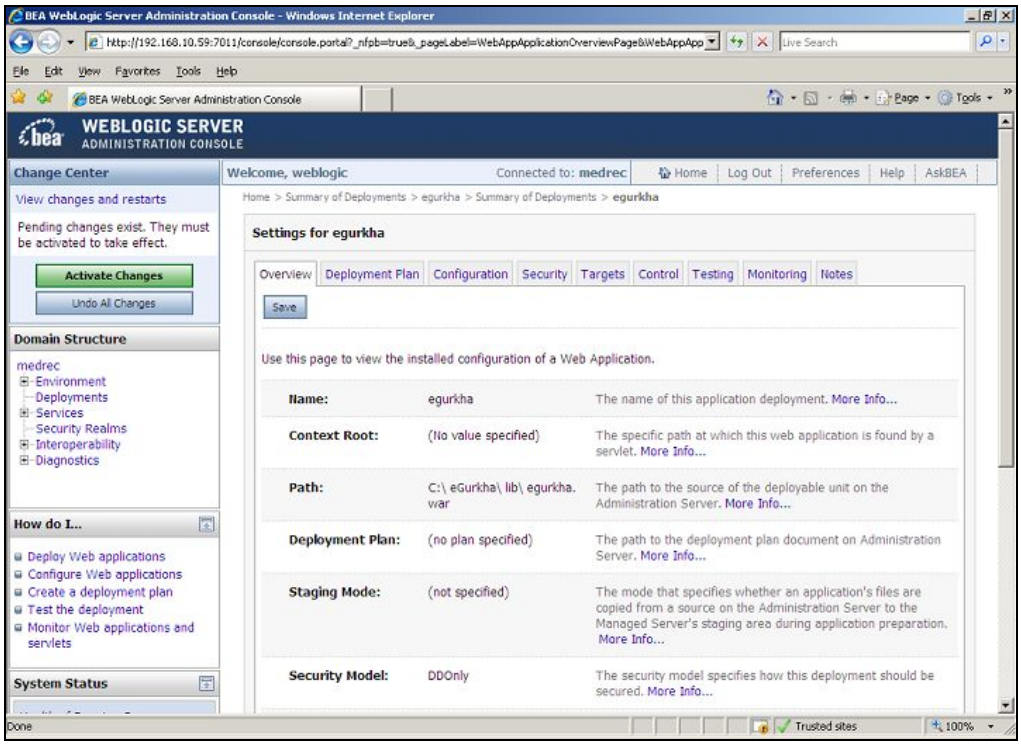


Figure 2.86: Saving the existing configuration

14. Figure 2.87 then appears, which lists the applications that have been deployed on the WebLogic server. If the **egurkha** application appears in the list, it is indicative of the successful installation of the **egurkha.war** file.

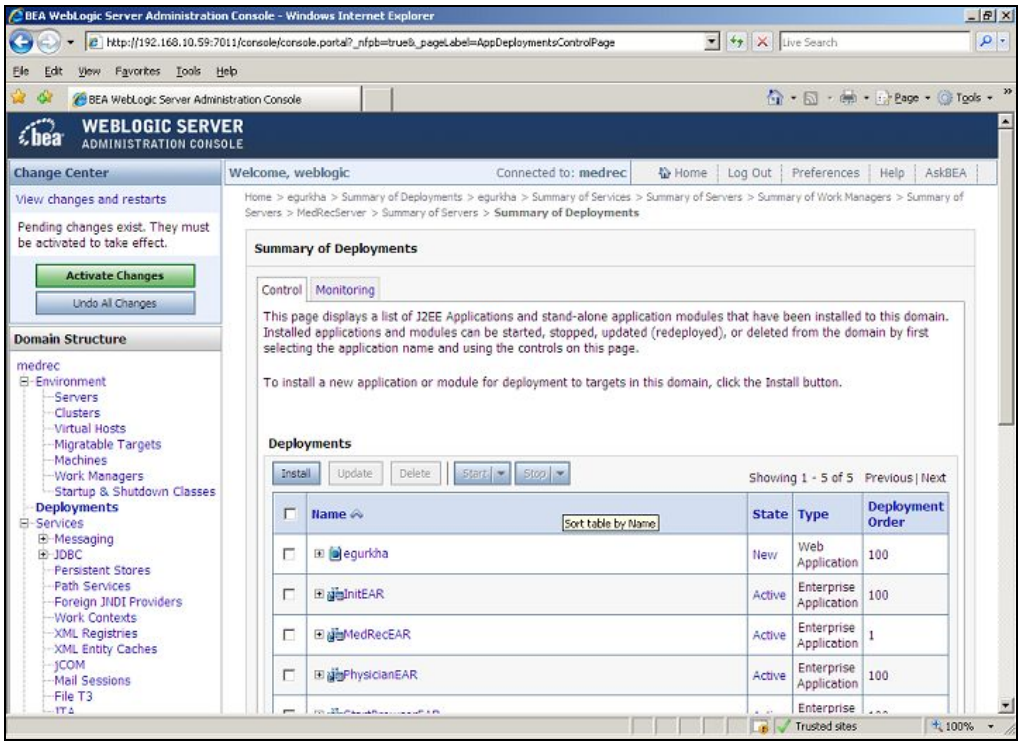


Figure 2.87: A proof of successful application deployment

15. To ensure that the changes are saved, click on the **Activate Changes** button under the **Change Center** section in the left panel of Figure 2.87. Upon successful change updation, a message to that effect appears at the top of the right panel, as depicted by Figure 2.88.

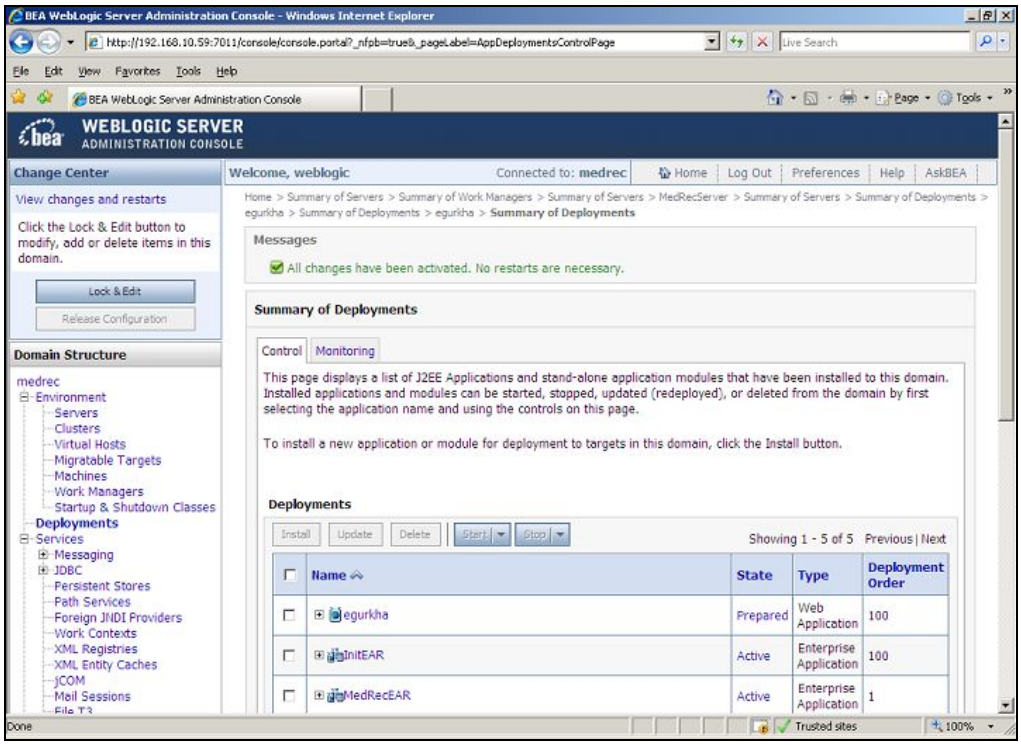


Figure 2.88: A message indicating that the changes have been activated

16. Now, proceed to start the application. To do so, first, click on the check box that corresponds to the **egurkha** entry in the **Deployments** list, click on the **Start** button above, and choose the **Servicing all requests** option from the drop-down menu (see Figure 2.89).

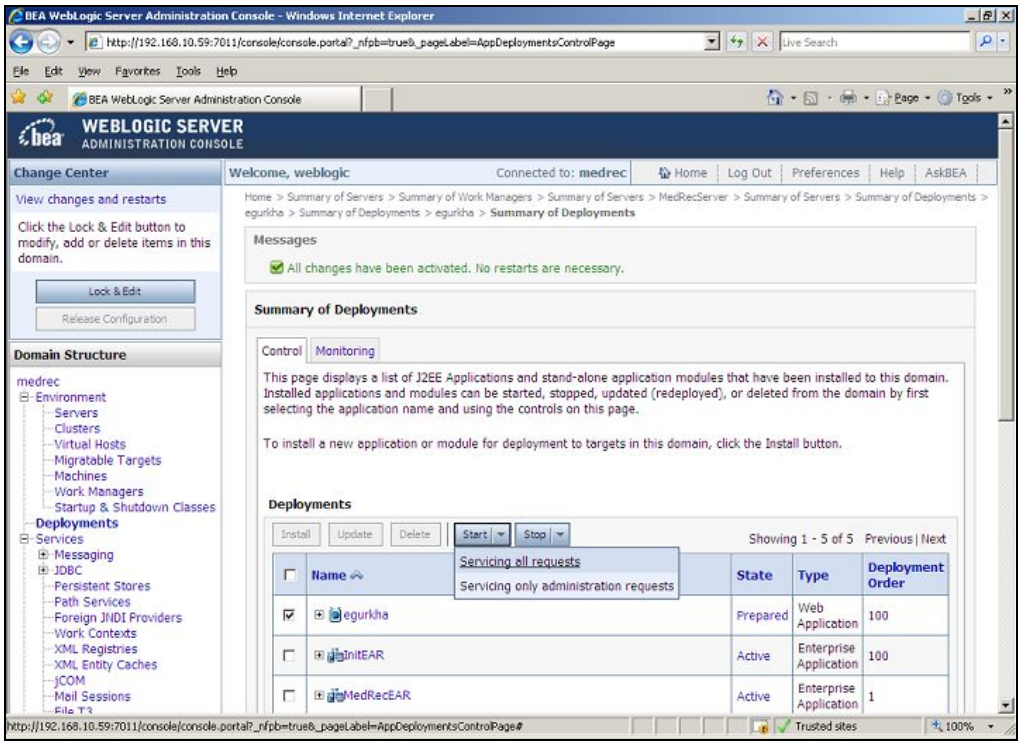


Figure 2.89: Starting the egurkha application

17. Click the **Yes** button in Figure 2.90 to confirm starting.

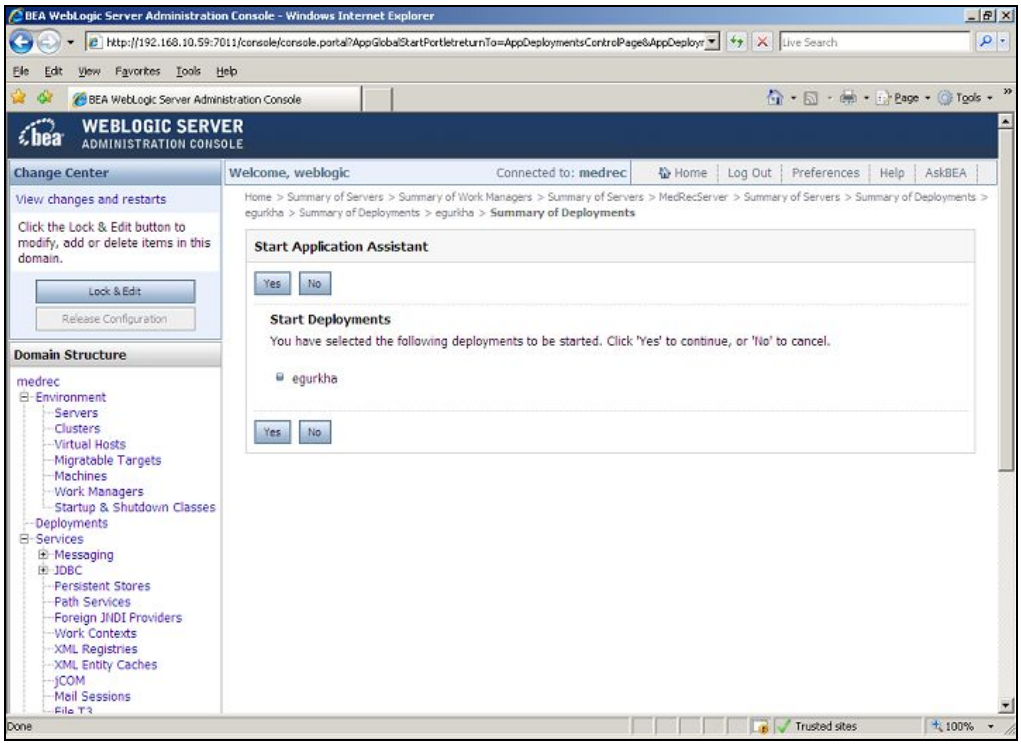


Figure 2.90: Providing confirmation for starting the egurkha application

18. If the application starts successfully, then a message to that effect appears, as depicted by the right panel of Figure 2.91.

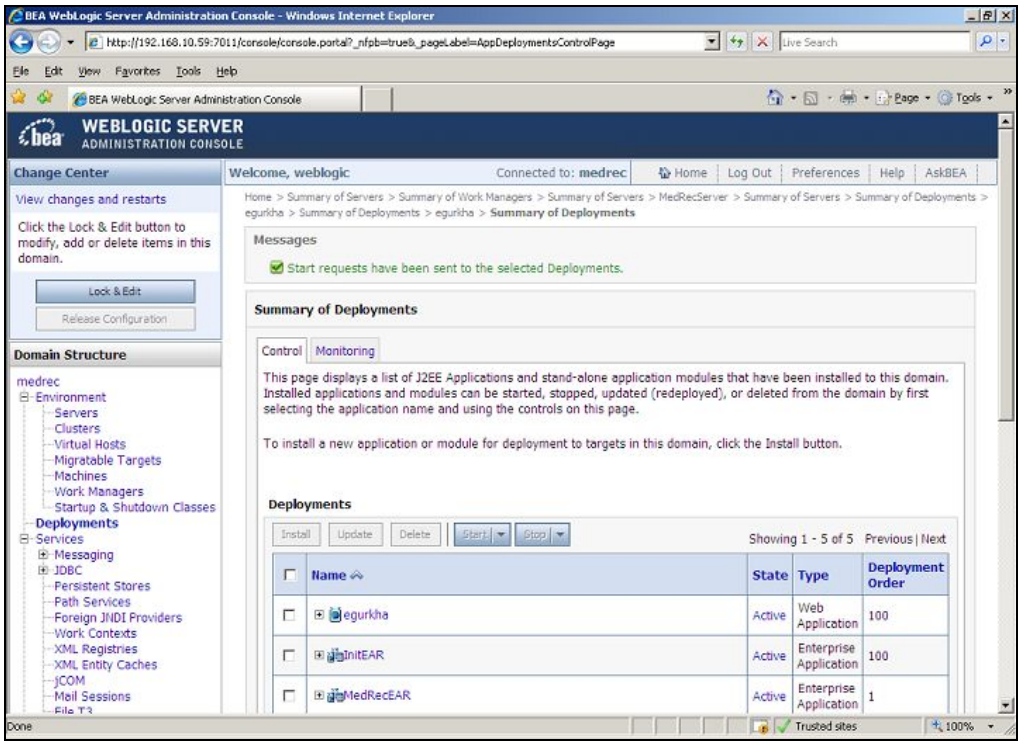


Figure 2.91: A message indicating that the application has started

2.3.2 Configuring a Oracle WebLogic Server 11G (and above)

The eG agent package contains a web archive file called **egurkha9x.war**. On Unix environments, this file will be available in the /opt/egurkha/lib location. On Windows environment, this file will be available in the <EG_HOME_DIR>\lib directory. This file has to be installed on the WebLogic server ver. 9 (and above), which has to be monitored. Prior to deploying this war file, make sure that you follow the steps below:

- Login to the system hosting the eG agent that is monitoring the WebLogic server.
- Copy the **egurkha9x.war** file from the **/opt/egurkha/lib** directory to any other directory on the eG agent host.
- Rename the war file as **egurkha.war**.

Next, follow the steps given below to deploy the war file:

1. Ensure that the WebLogic application server to be monitored has been started.
2. If the target WebLogic server is part of a cluster, then, connect to the **Admin** server of the cluster using the URL: *http://<Admin_server_IP>:<AdminServerPort>/console*. On the other hand, if

the target WebLogic server is a stand-alone server and not part of a cluster, then, connect to the targer using the URL: *http://<WebLogic_server_IP>:<WebLogicServerPort>/console*.

3. Figure 2.92 will then appear. Login to the WebLogic server/admin server by providing the required credentials in Figure 2.92.

Figure 2.92: Logging into the WebLogic server/admin server

4. Figure 2.93 will then appear.

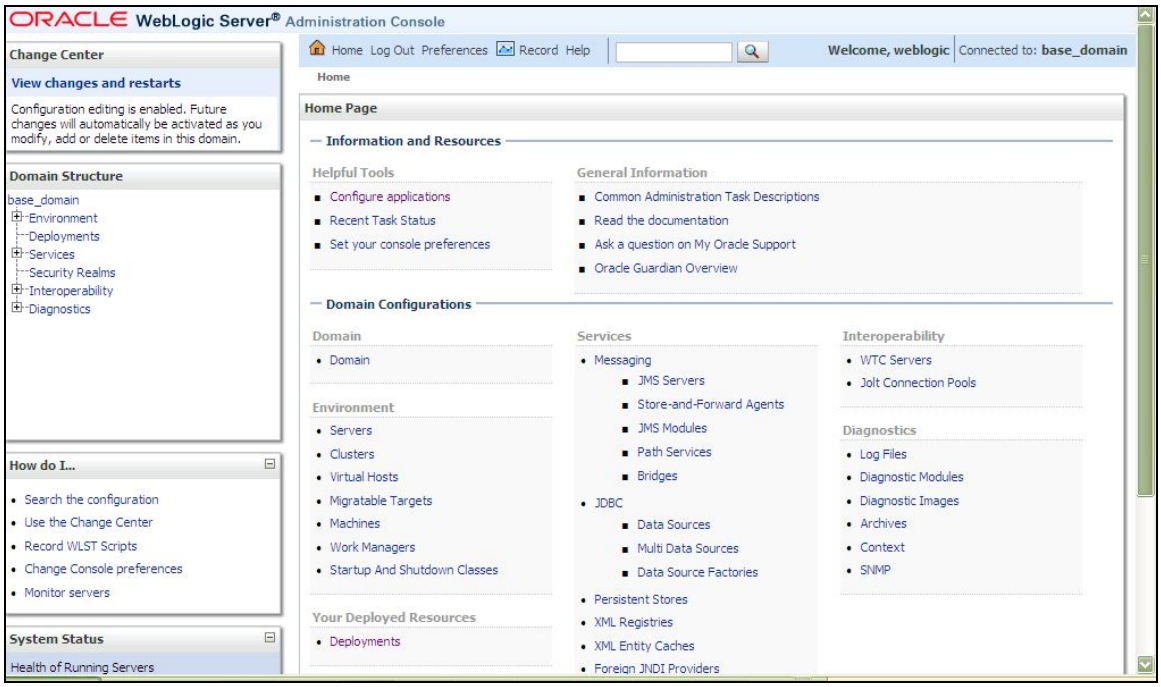


Figure 2.93: The WebLogic console

5. To begin application deployment, click on the **Deployments** node in the left panel of Figure 2.93. This will invoke Figure 2.94 where applications that pre-exist on the server will be displayed.

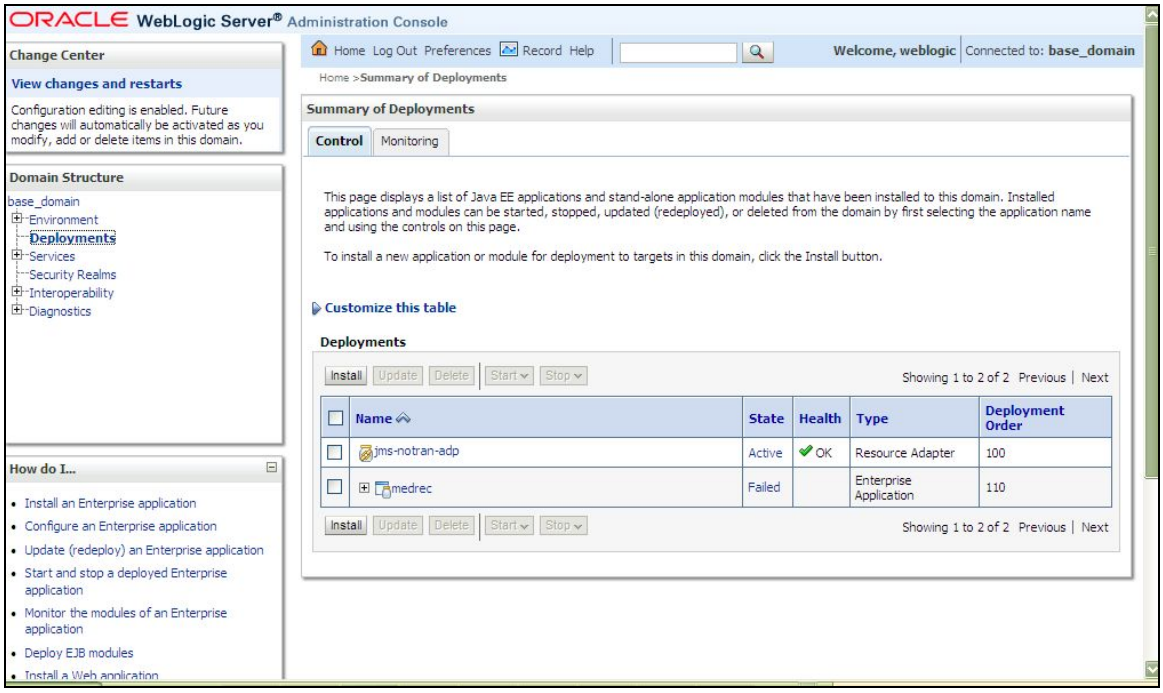


Figure 2.94: List of applications that pre-exist

6. To install a new application, click on the **Install** button in Figure 2.94. Figure 2.95 will then appear prompting you to specify the full **Path** to the directory on the target WebLogic server/admin server to which the **egurkha.war** file has been uploaded. To specify the path, click on the **upload your file(s)** link that is just above the **Path** text box in Figure 2.95.

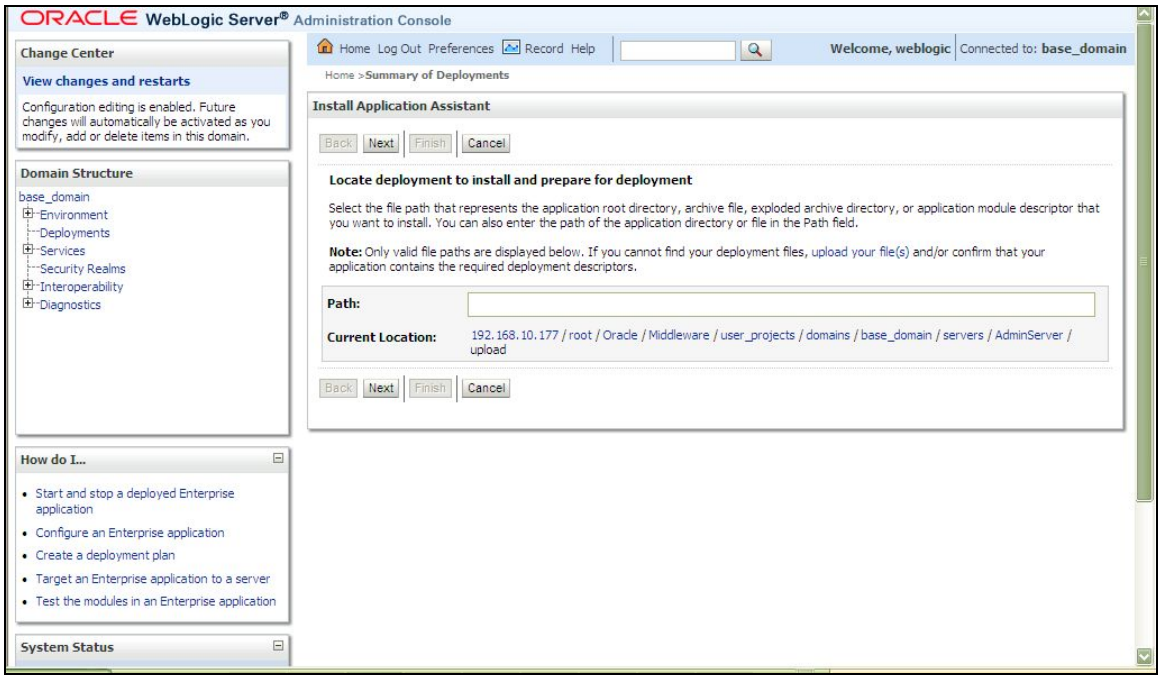


Figure 2.95: A page requesting you to enter the path to the directory to which the egurkha.war file has been uploaded

7. Figure 2.96 will then appear; in the **Deployment Archive** text box here, you will have to indicate the path from which the **egurkha.war** file has to be uploaded. To browse for the exact location of the **egurkha.war** file on the host, use the **Browse** button adjacent to the **Deployment Archive** text box. Typically, this file will be available in the <EG_INSTALL_DIR>\lib directory of the eG agent host.

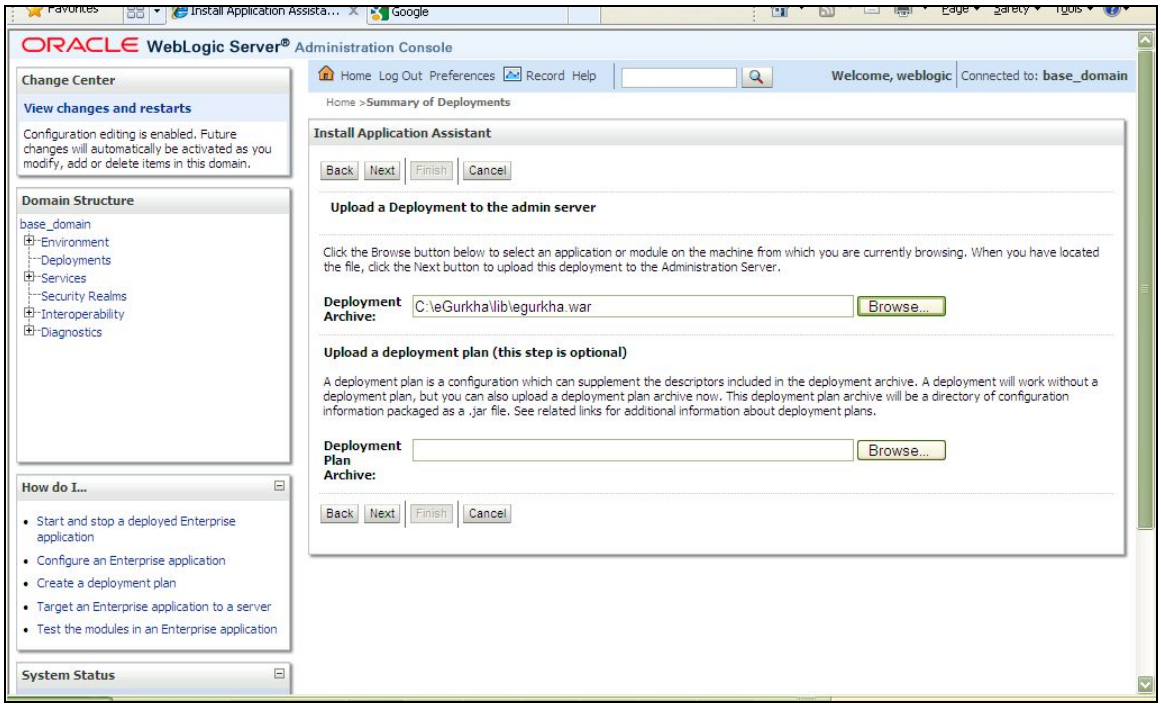


Figure 2.96: Specifying the path to the egurkha.war file

8. Once the **Deployment Archive** path is provided, click on the **Next** button in Figure 2.96 to proceed with the setup.

9. When Figure 2.97 appears, you will find that the **Path** text box reveals the full path to the directory on the target WebLogic server/admin server to which the **egurkha.war** file has been uploaded. Click on the **Next** button again to continue.

   Note:

   By default, the **egurkha.war** file will be searched for in the local disk of the system from which the browser is launched. Therefore, if you are trying to access the WebLogic console from a system that does not host an eG agent, then, before attempting to deploy the **egurkha.war** file, do the following:

   - Login to any agent host in your environment.
   - Copy the **egurkha9x.war** file in the <EG_AGENT_INSTALL_DIR>\lib directory (on Windows; on Unix, this will be the /optegurkha/lib directory) of the agent host to any other directory on that host.
   - Rename the **egurkha9x.war** file as **egurkha.war** file.
   - Copy the **egurkha.war** file to the local host from which the browser is launched.

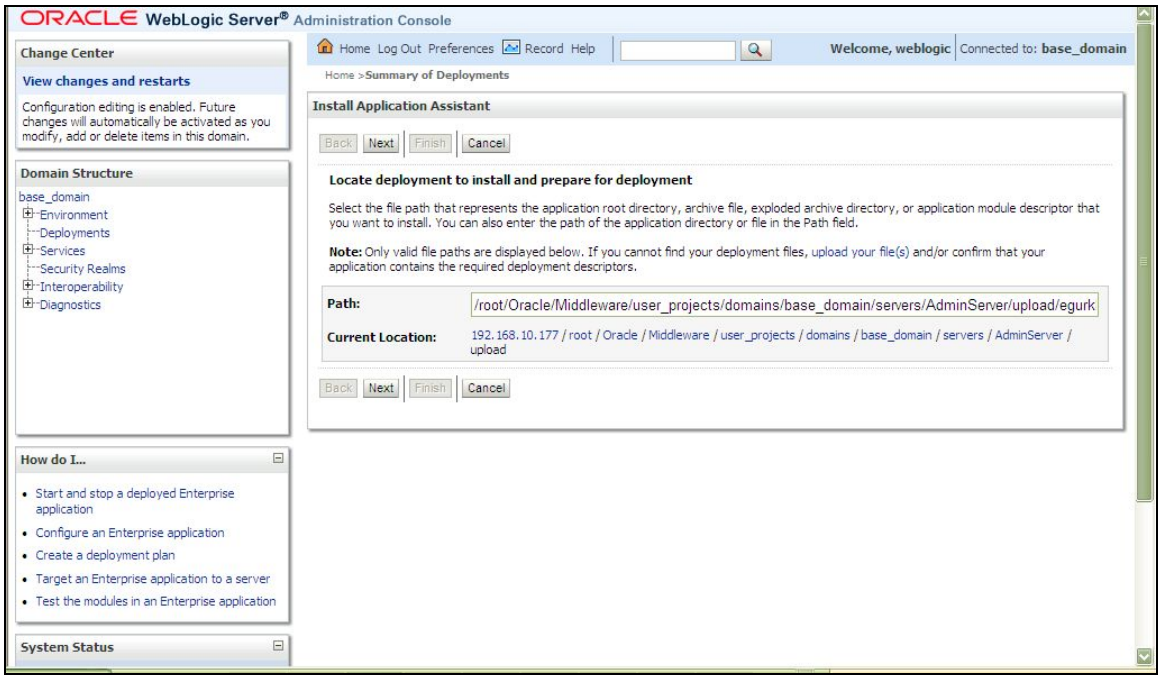


Figure 2.97: Path of the folder/directory to which the egurkha.war file has been uploaded

10. Then, select the **Install this deployment as an application** option from Figure 2.98, and then click **Next** to proceed.

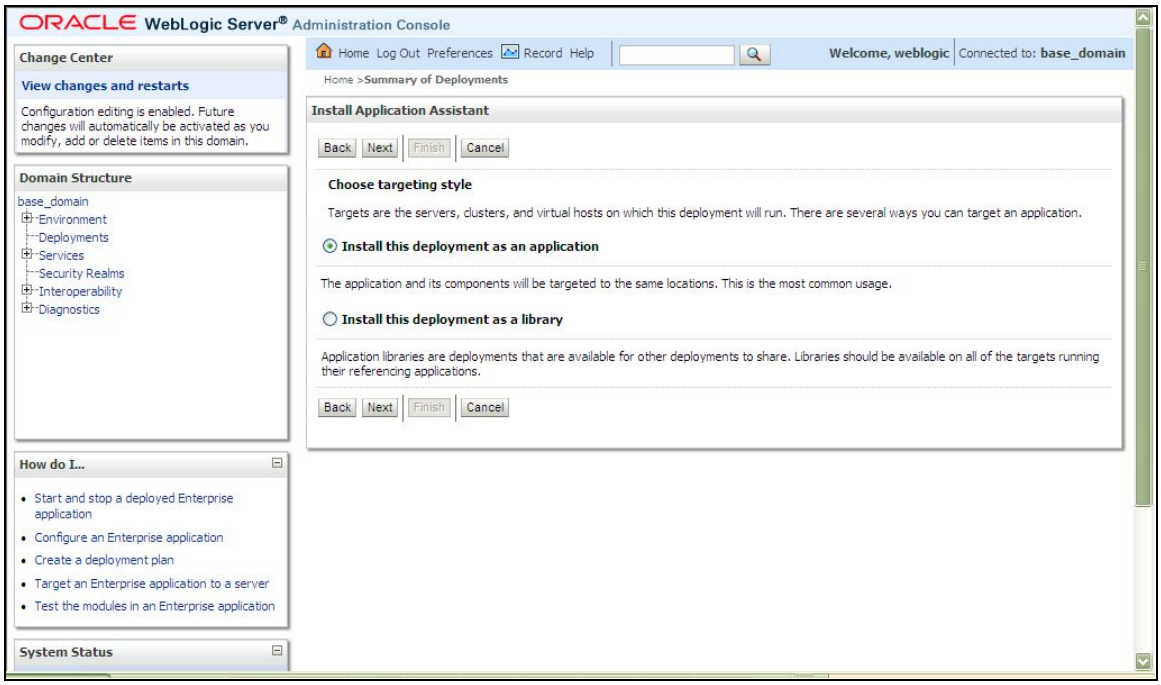


Figure 2.98: Choosing to install the deployment as an application

11. Next, pick the server on which the **egurkha** application is to be deployed, and then click the **Next** button.

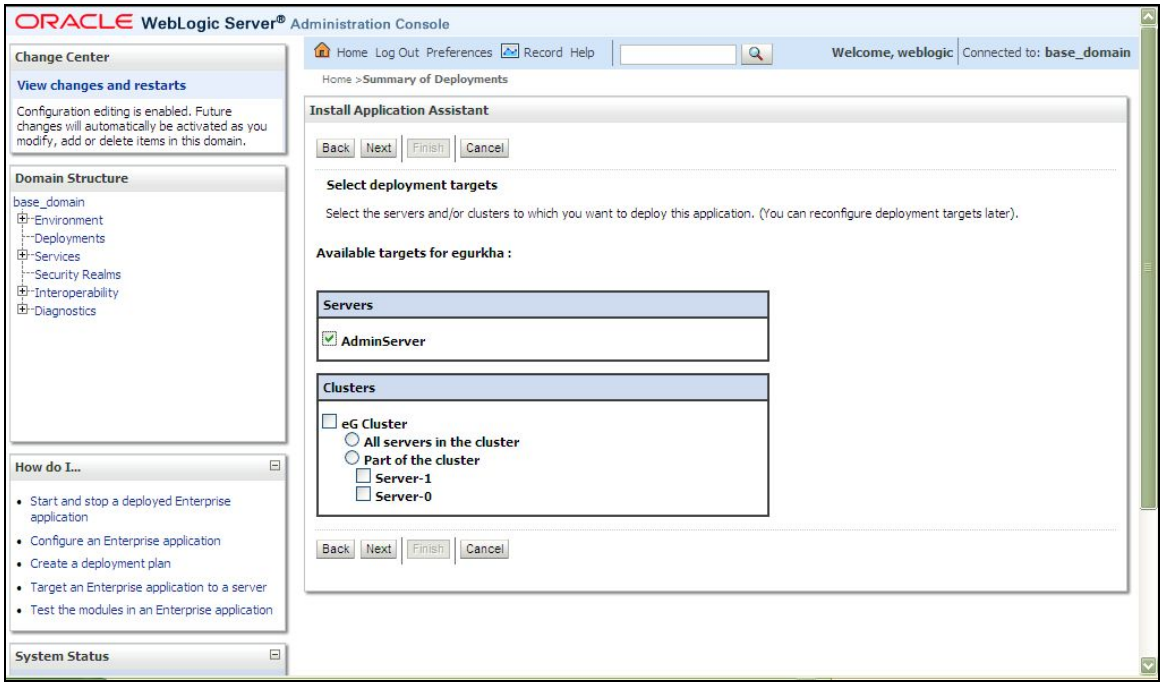


Figure 2.99: The Administration Console of the ORACLE WebLogic server 11G

12. In Figure 2.99 that then appears, assign a unique **Name** for the **egurkha** application, and click the **Next** button again to continue.

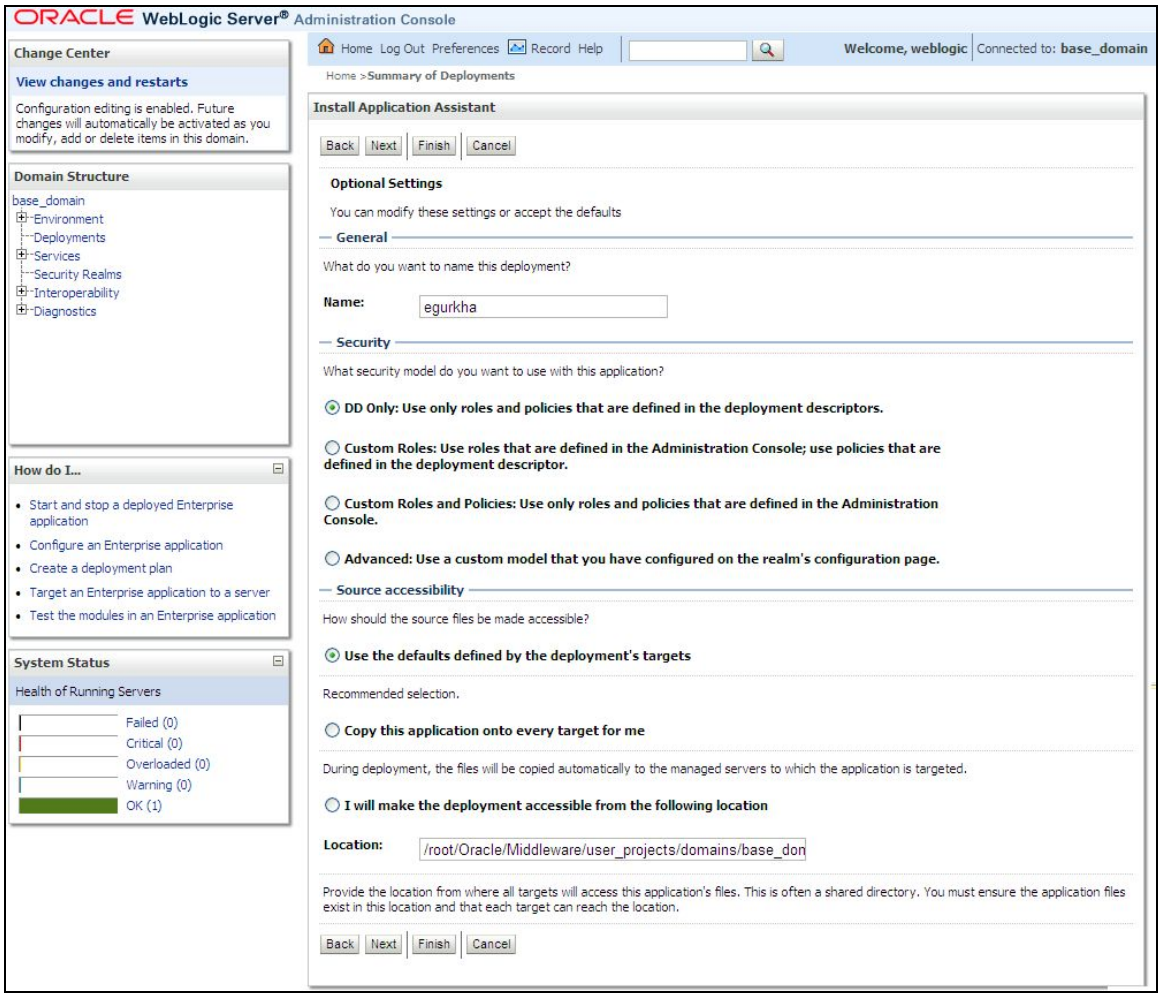


Figure 2.100: Naming the application

13. Quickly review your specifications using Figure 2.101, and then save the changes made to the configuration by clicking on the **Save** button in Figure 2.101. If updation is successful, then messages to that effect will appear in the console as depicted by Figure 2.102.

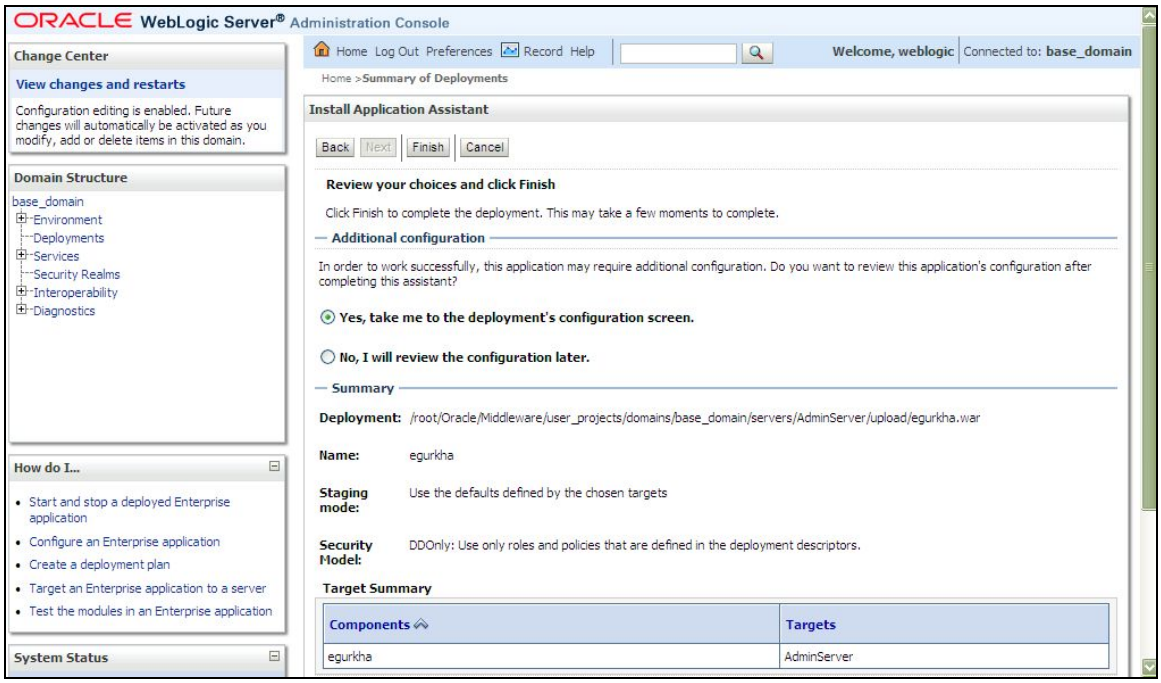


Figure 2.101: Reviewing the specifications for the egurkha.war application

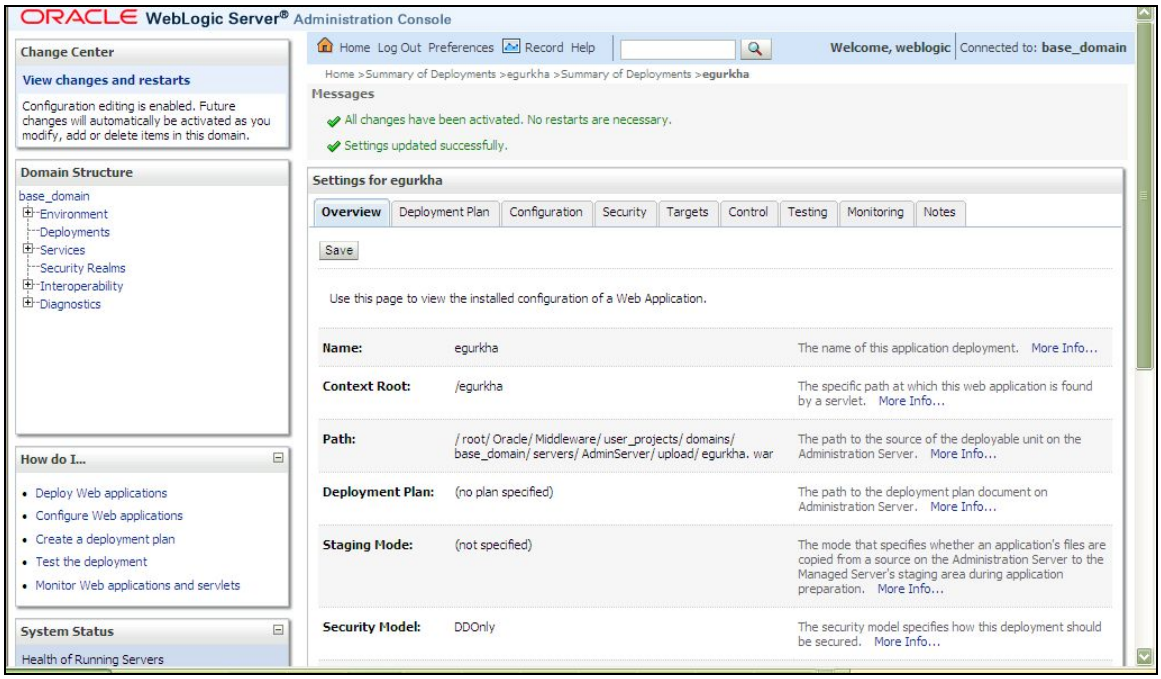


Figure 2.102: Messages that appear after saving changes to the WebLogic configuration

2.3.3 Enabling Garbage Collection Monitoring on the WebLogic Server 9 (and above)

If the JVMGC test is to be executed for garbage collection (GC) monitoring, then the following additional steps need to be performed:

1. Edit the **startWebLogic.cmd** file (in Windows. In Unix, this will be **startWebLogic.sh**) in the <ORACLE_HOME>\user_projects\domains\base_domain\<YOUR_DOMAIN> directory (on WebLogic Server ver. 10.3).

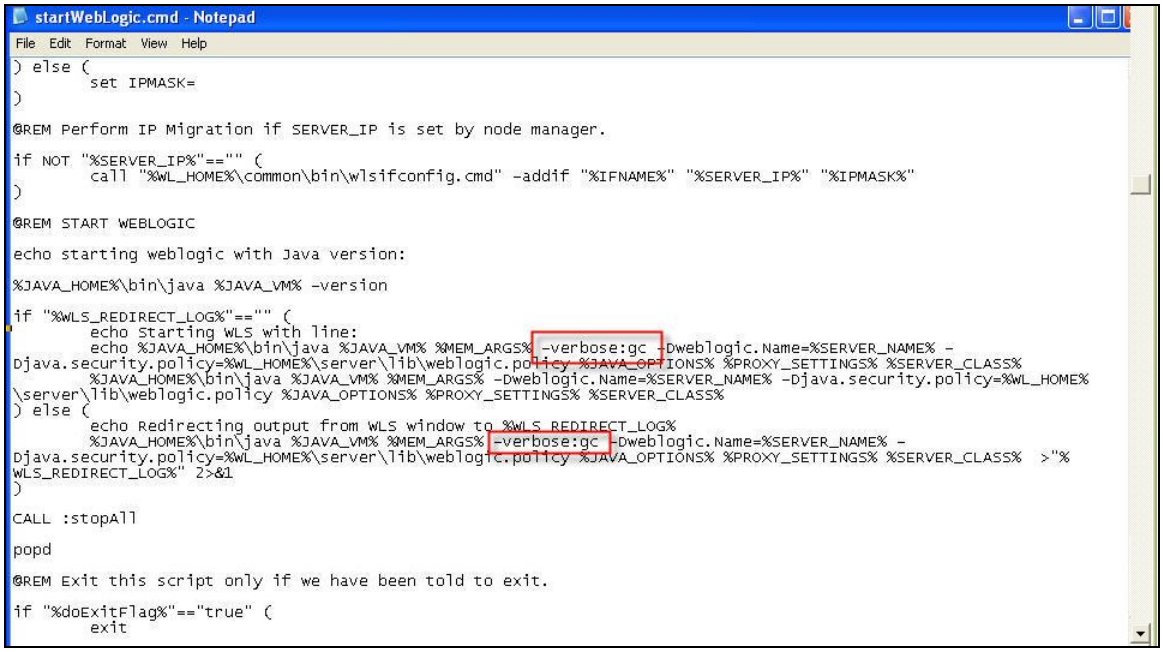


Figure 2.103: Adding entries for redireting GC output

2. In the file, the 'echo' command to start the WebLogic server exists. Add **-verbose:gc** to the line next to this command, as indicated by Figure 2.103.
3. Similarly, you will find an 'echo' command that will redirect the output from the WL window to a log file. Add the **-verbose:gc** entry to the line next to this command as well (as indicatd by Figure 2.103).
4. Then, save the file.

Figure 2.104: Specifying the path to the log file that will store the GC output

5. Next, edit the **startscript.xml** file in the <ORACLE_HOME>\user_projects\domains\base_domain\init-info directory (on WebLogic ver. 10.3) to configure the location of the log file which should store the GC output. For this purpose, look for the entry <operand value="%WLS_REDIRECT_LOG%"> in the file. Once you locate this entry, you will find that another **operand value** parameter appears below it, but this parameter is left blank by default. Fill up this blank by specifying the full path to the log file in which the GC output is to be stored, as indicated by Figure 2.104.
6. Finally, save the file.

2.3.4 Configuring the eG Agent to Report JVM-related Metrics for the WebLogic Server 9.0 (and above)

The eG agent can be optionally configured to execute a few additional tests on the JVM layer of the WebLogic monitoring model, so as to report critical statistics related to the WebLogic server's JVM. These statistics typically reveal the following:

- The count of classes loaded/unloaded (Java Classes test)
- JVM thread usage (JVM Threads test)
- CPU and memory usage of the JVM (JVM Cpu Usage test and JVM Memory Usage test)
- The effectiveness of the JVM's garbage collection activity (JVM Garbage Collections test)

- The uptime of the JVM (JVM Uptime test)
- Whether JMX is currently enabled/disabled on the target WebLogic server (JMX Connection to JVM test)
- The count and status of file descriptors (JVM File Descriptors test)

These additional tests are disabled by default for the WebLogic server.

These tests, if enabled, connect to the JRE used by the WebLogic application server to pull out the above- mentioned metrics. You can configure the eG agent to use either of the following methodologies for collecting the metrics of interest from the JRE:

- JMX (Java Management Extensions)
- SNMP (Simple Network Management Protocol)

This is why, while configuring each of these additional tests in the eG administrative interface, you are required to pick a monitoring **MODE** - i.e., either **JMX** or **SNMP**. The remaining test configuration depends upon the mode chosen.

Since both **JMX** and **SNMP** support are available for JRE 1.5 and above only, these additional tests **should be** enabled only if the WebLogic server being monitored runs JRE 1.5 and above.

Once the JRE version requirement is fulfilled, you must proceed to enable the JMX support for the JRE or SNMP support for the JRE, before enabling the optional tests.

If you choose to use JMX for pulling out the desired metrics from the JRE, then the following broad steps need to be followed:

- First, determine whether the JMX requires no authentication at all, or requires authentication (but no security)
- If JMX does not require authentication, follow the steps below:

1. Connect to the target WebLogic Application server and login to the WebLogic Administration Console. Once in the console, browse the **Domain Structure** tree in the left panel of the console for the **Environment** node. Expand the node and click the **Server** subnode within, as depicted by **Step-1** of Figure 2.105 below. The **Servers** list in the right panel will then display all the WebLogic server instances that are currently available on the monitored WebLogic server. Click on that instance for which JMX is to be enabled, as depicted by **Step-2** of Figure 2.105 below.

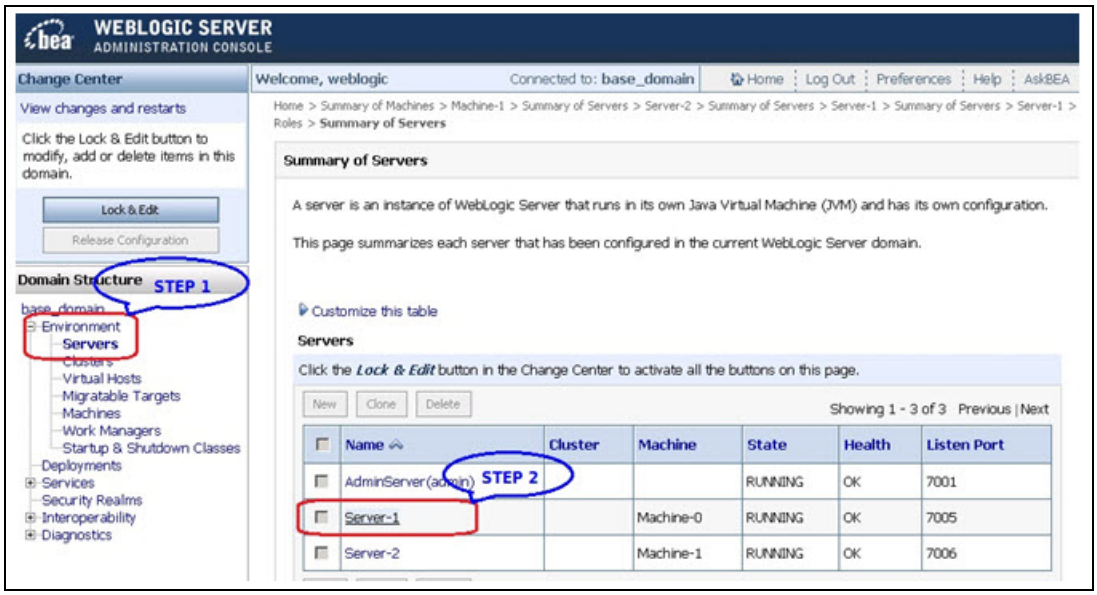


Figure 2.105: Selecting the WebLogic server instance to be monitored

2. When the Configuration-General tab page opens, click on the Server Start link, as shown by Step-3 of Figure 2.106, to start the chosen server instance.

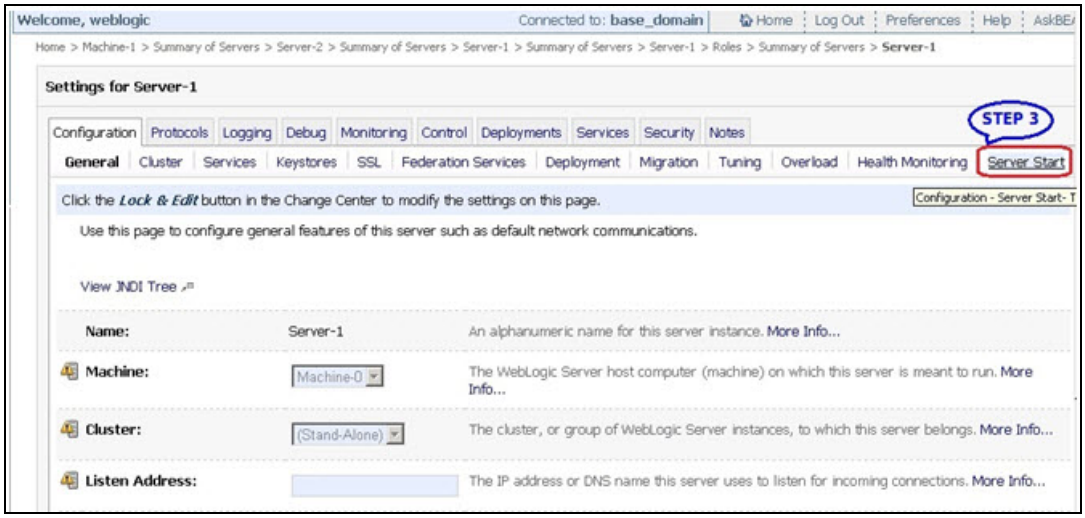


Figure 2.106: Starting the chosen WebLogic server instance

3. Next, click on the **Lock and Edit** button in the left panel, as indicated by Step-4 in Figure 2.107 below, to edit the configuration of the server instance.

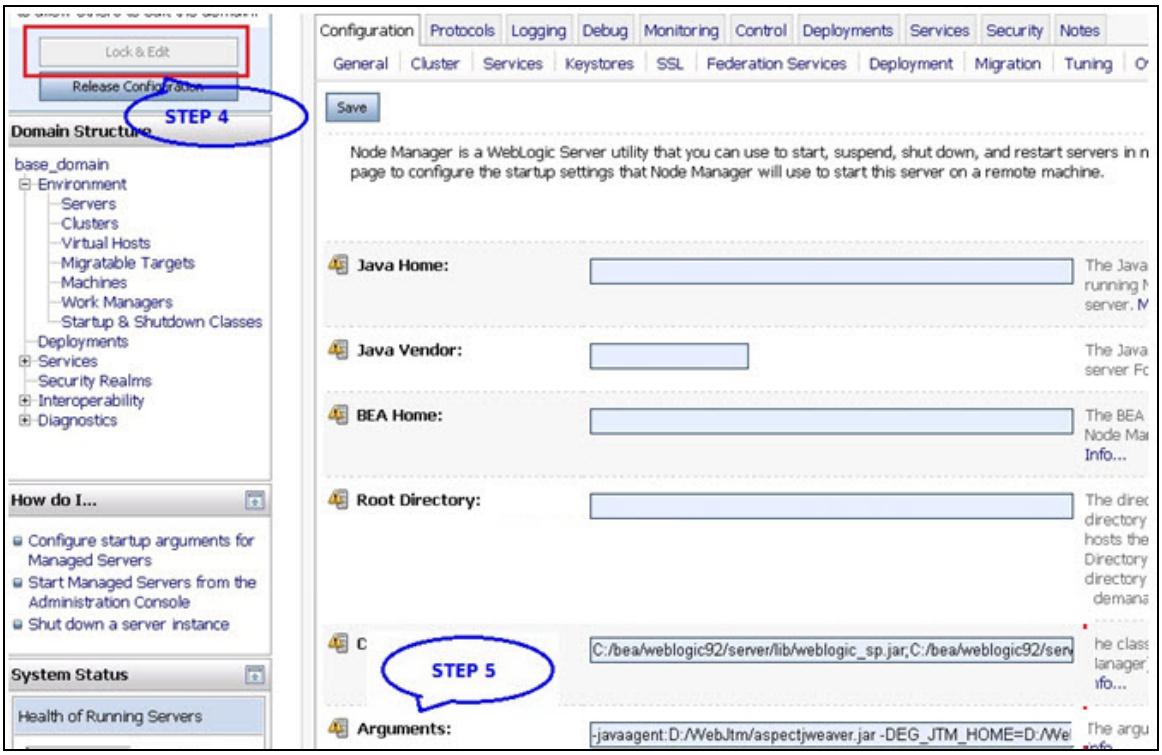


Figure 2.107: Editing the Arguments

4. Then, in the right panel, edit the **Arguments** parameter indicated by **Step-5** of Figure 2.107 above, as follows:

```
-javaagent:-Dcom.sun.management.jmxremote.port=11222 -
Dcom.sun.management.jmxremote.ssl=false -
Dcom.sun.management.jmxremote.authenticate=false
```

5. Then, save the configuration changes by clicking on the **Save** button indicated by Step-6 in Figure 2.108 below.

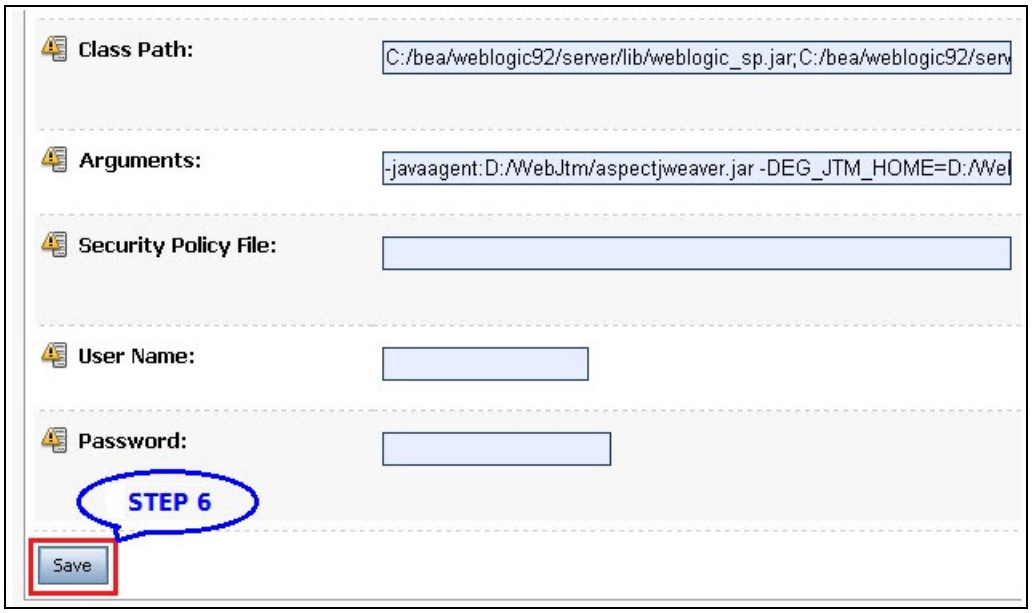


Figure 2.108: Saving the configuration changes

6. In the left panel, click the **Activate Changes** button, as depicted by Step-7 of Figure 2.109.

Figure 2.109: Activating the changes

7. Next, click on the **Control** tab page (see Step-8) in the right panel of Figure 2.110.

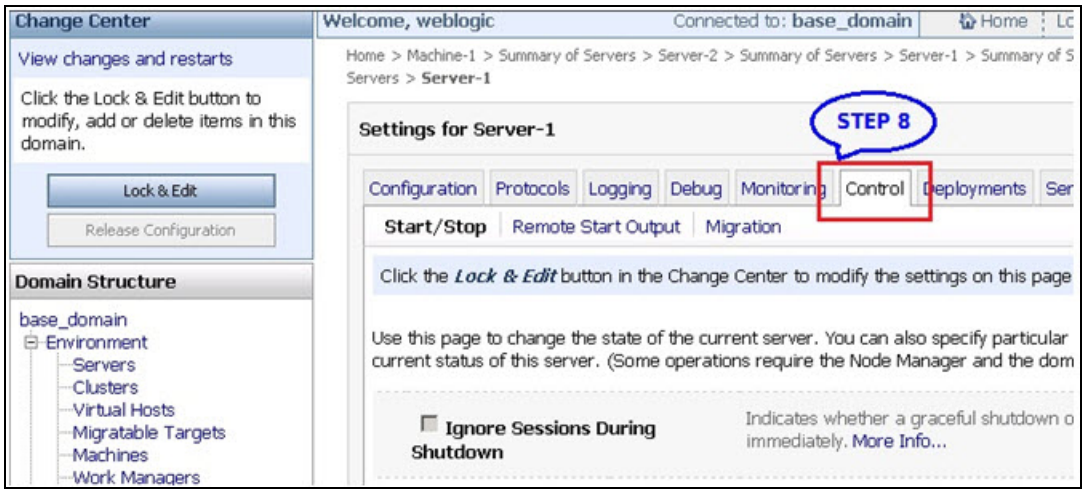


Figure 2.110: Clicking on the Control tab page

8. From the list of server instances displayed in the right panel, select the server instance for which JMX has been enabled. Then, proceed to shutdown the instance. You can shutdown an instance immediately or schedule the shutdown to occur automatically when all the pending processing on the instance is complete. To choose the shutdown mode, click on the down arrow next to the **Shutdown** button and select the **Force Shutdown Now** or **When Work Completes** option (as the case may be), as shown by Step-9 of Figure 2.111.

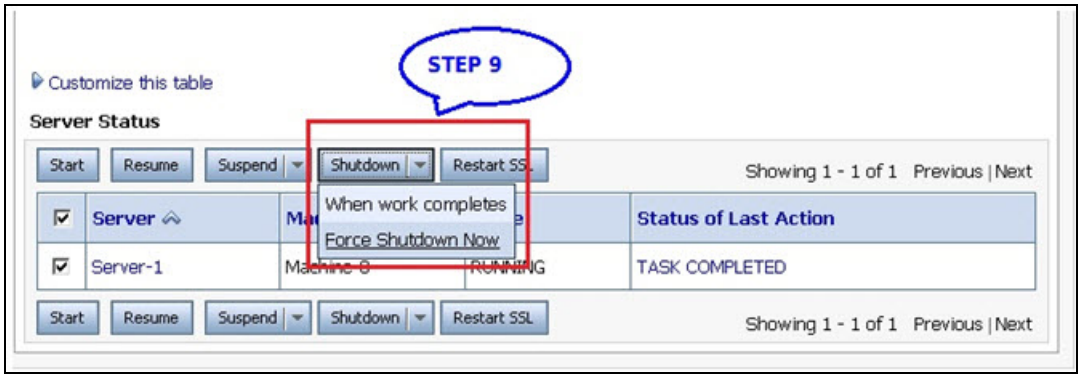


Figure 2.111: Shutting down the instance

9. Once the instance shuts down, click the **Start** button shown by Step-10 of Figure 2.112 to start the instance. If the **State** of the instance changes to **RUNNING** (as shown by Step-11), it clearly indicates that the instance has started.

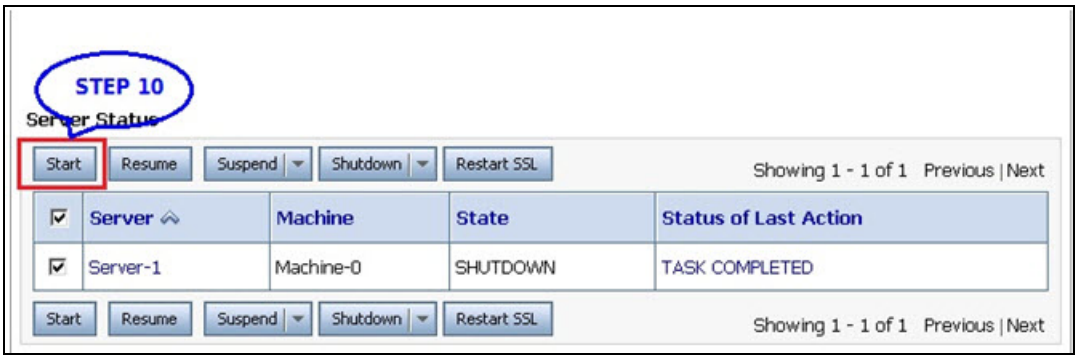


Figure 2.112: Starting the server instance

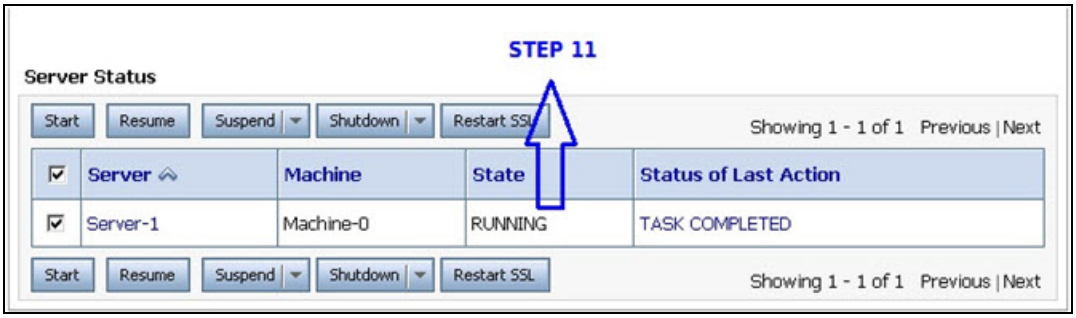


Figure 2.113: State change indicating that the server instance has started

10. If the JMX requires authentication (but no security), follow the steps below:

- Login to the target WebLogic server. If the server is executing on a Windows host, then, login to the host as a local/domain administrator.
- Next, copy the *jmxremote.password.template* file in the <JAVA_HOME>\jre\lib\management folder to any other location on the host, rename it as as *jmxremote.password*, and then, copy it back to the <JAVA_HOME>\jre\lib\management folder.
- Next, edit the *jmxremote.password* file and the *jmxremote.access* file to create a user with *read-write* access to the JMX. To know how to create such a user, refer to *Monitoring Java Applications* document.
- Then, proceed to make the *jmxremote.password* file secure by granting a single user “full access” to that file. To know how to achieve this, refer to the *Monitoring Java Applications* document.
- Edit the *management.properties* file that is used by the JRE of the target WebLogic server, and configure the following in it:

- The JMX remote port
- Whether JMX is SSL-enabled or not
- Whether JMX requires authentication or not
- The full path to the *jmxremote.access* file
- The full path to the *jmxremote.password* file

To know how to configure these, refer to the *Monitoring Java Applications* document.

- Then, save the file.
- Stop the WebLogic server.
- Then, edit the <WEBLOGIC_ HOME>\wlserver_ 10.0\samples\domains\wl_ server\bin\StartWebLogic.bat file.
- Add the following line in it:

```
set JAVA_OPTIONS=-Dcom.sun.management.config.file=<<WEBLOGIC_HOME>>\jrockit_150_
11\jre\lib\management\management.properties
```

- Finally, start the WebLogic server.
- Then, proceed to enable the JVM-related tests, and configure them using the instructions provided in the Monitoring Application Servers document. To enable one/more tests, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick WebLogic as the **Component type**, *Performance* as the **Test type**, choose the tests from the **DISABLED TESTS** list, and click on the **<<** button to move the tests to the **ENABLED TESTS** list. Finally, click the **Update** button.

Note:

To know how to enable SNMP support for the JRE, refer to the *Monitoring Java Applications* document.

2.4 Managing the Oracle WebLogic Application Server

The configured server should then be discovered and managed. To do so, adhere to the following steps:

1. Log into the administrative interface as an administrator.
2. The WebLogic server to be monitored may have been automatically discovered by the eG

manager. If it is, use the Infrastructure -> Components -> Manage / Unmanage menu to open the **COMPONENTS - MANAGE/UNMANAGE** page and manage the WebLogic server as depicted by Figure 2.114 and Figure 2.115 below.

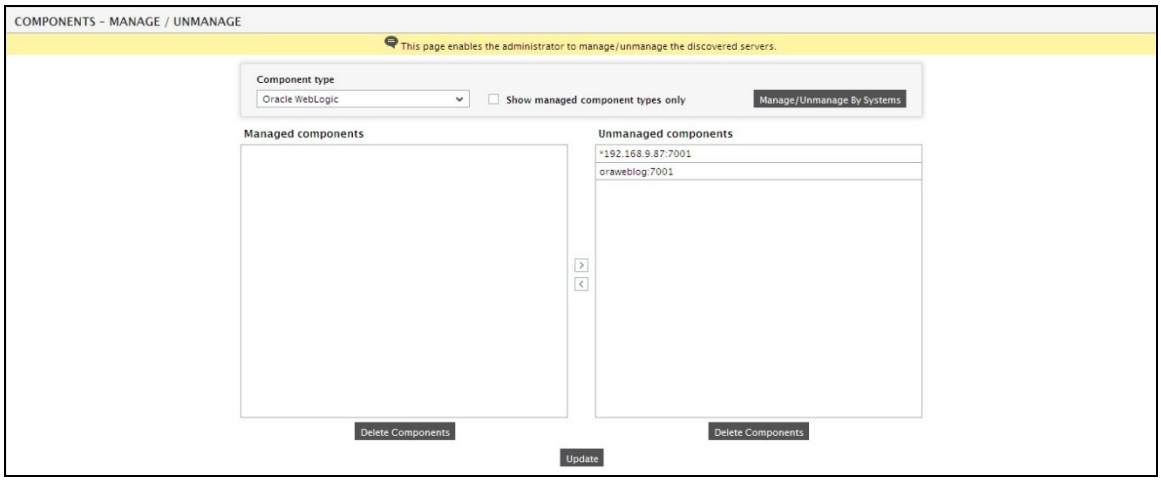


Figure 2.114: Viewing the list of unmanaged Oracle WebLogic servers in the COMPONENTS - MANAGE/UNMANAGE page

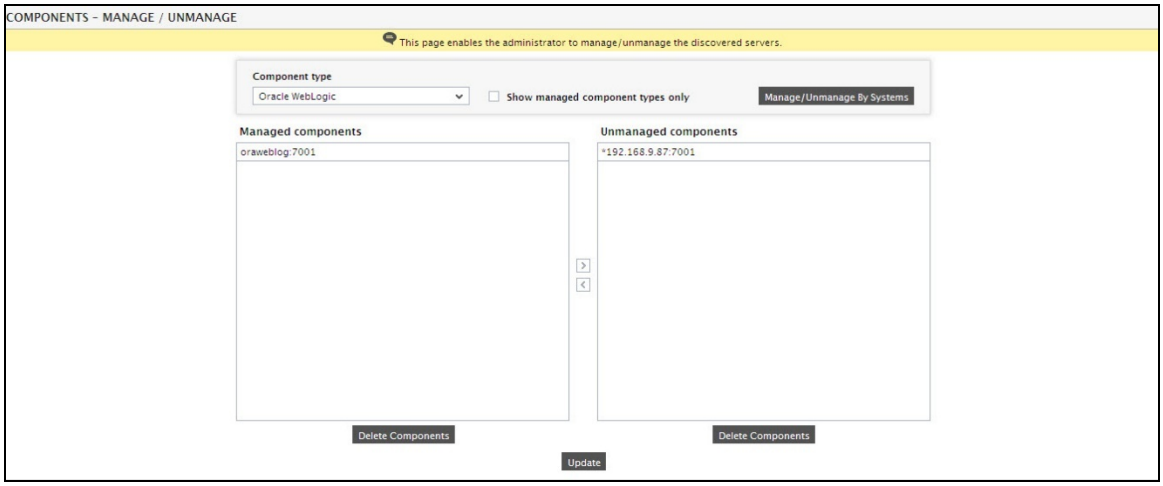


Figure 2.115: Managing a WebLogic server

3. Alternately, if the WebLogic Server is not discovered, it can be manually added using the **COMPONENTS** page that can be accessed through the menu sequence Infrastructure-> Add/Modify Servers . Such components will be automatically managed by the eG Enterprise system.
4. Now, if you proceed to log out of the administrative interface, a list of unconfigured tests will appear prompting you to configure tests for the WebLogic Server.

5. Click on **WebLogic EJB Transactions** test to move to the page that facilitates its configuration (see Figure 2.116).

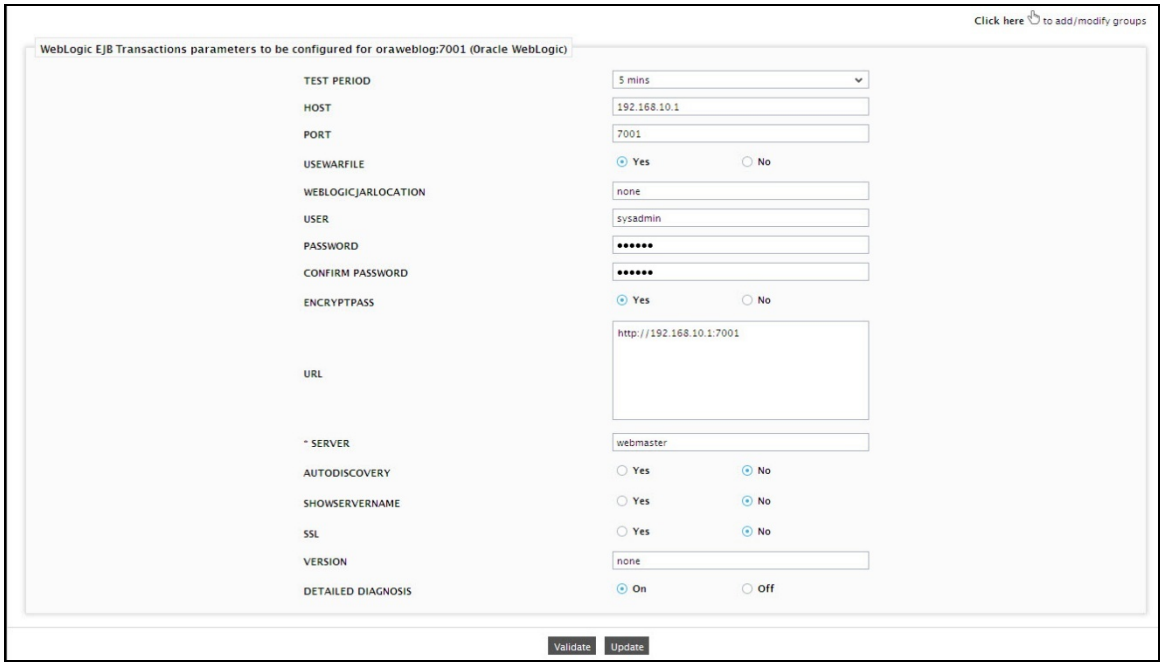


Figure 2.116: Configuring the Weblogic EJB Transactions test for a WebLogic server

6. The test parameter section that appears (as shown in Figure 2.116), specify the following:
7. Wait for the test to run once, and then proceed to **RECONFIGURE** the same for creating the EJB groups. eG monitors only those components that belong to an EJB group.
8. To begin configuring, first click on the **RECONFIGURE** button in the test configuration page. This will result in the display of Figure 2.116 once again.
9. Clicking on the **Click here** hyperlink in Figure 2.116 above will take you to a page depicted by Figure 2.117 below. This page lists the EJB groups that pre-exist (if any), the EJB components that have been added to the group, and allows the administrator to create and configure new groups, or delete/modify existing ones.

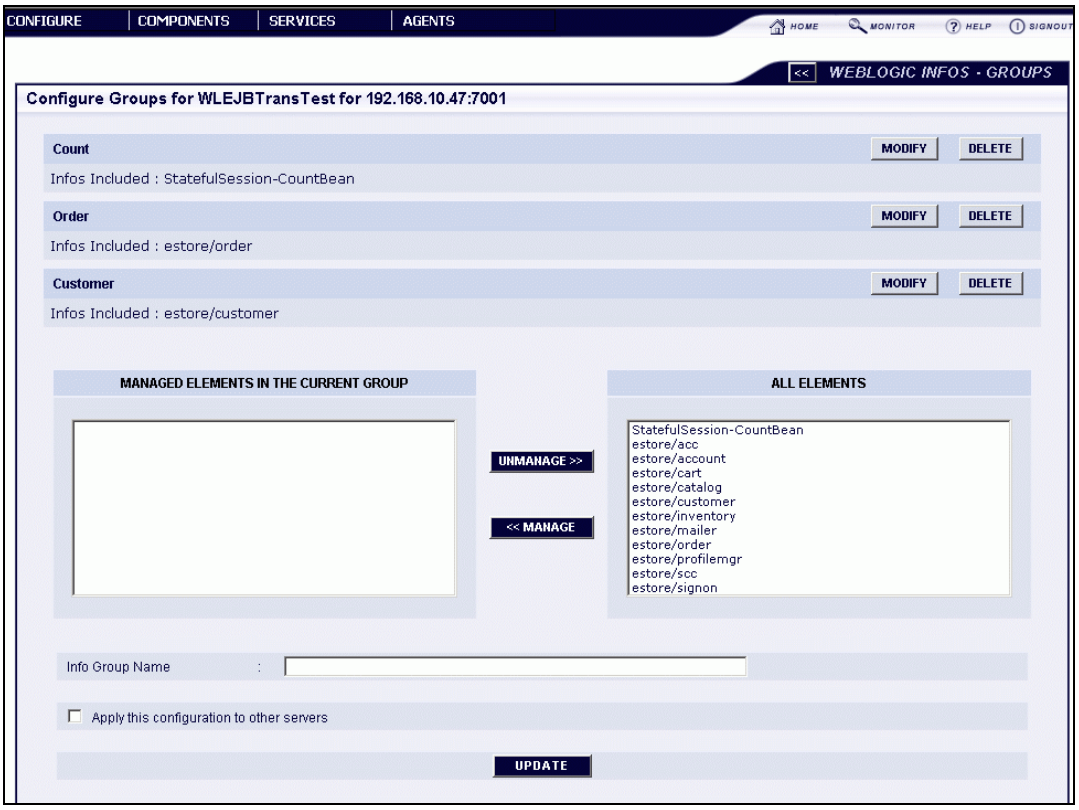


Figure 2.117: Figure 1.121: A page listing existing EJB groups (if any)

10. To configure a new EJB group, first select the EJBs to be added to the group from the **ALL ELEMENTS** list in Figure 2.118. All the EJBs hosted on a WebLogic server are discovered when the agent starts executing on the WebLogic Server's host. These discovered EJBs will be listed in the **ALL ELEMENTS** list. This list will get updated every time the agent is restarted / when the rediscovery scheduled for every four hours occurs.

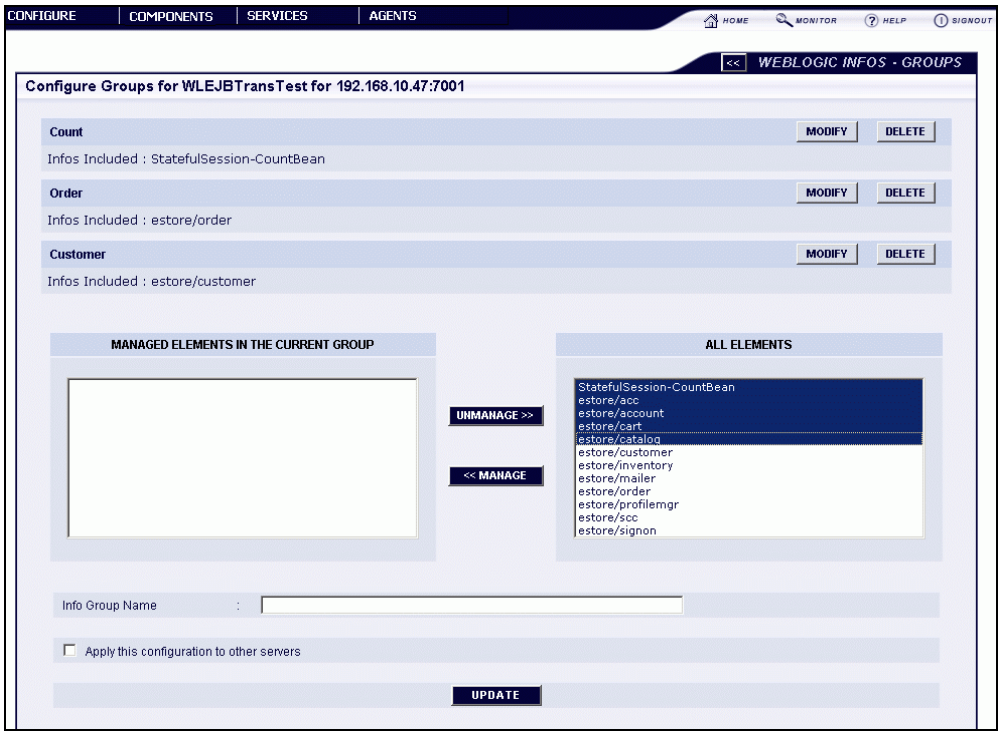


Figure 2.118: Selecting the EJBs to be added to the new group

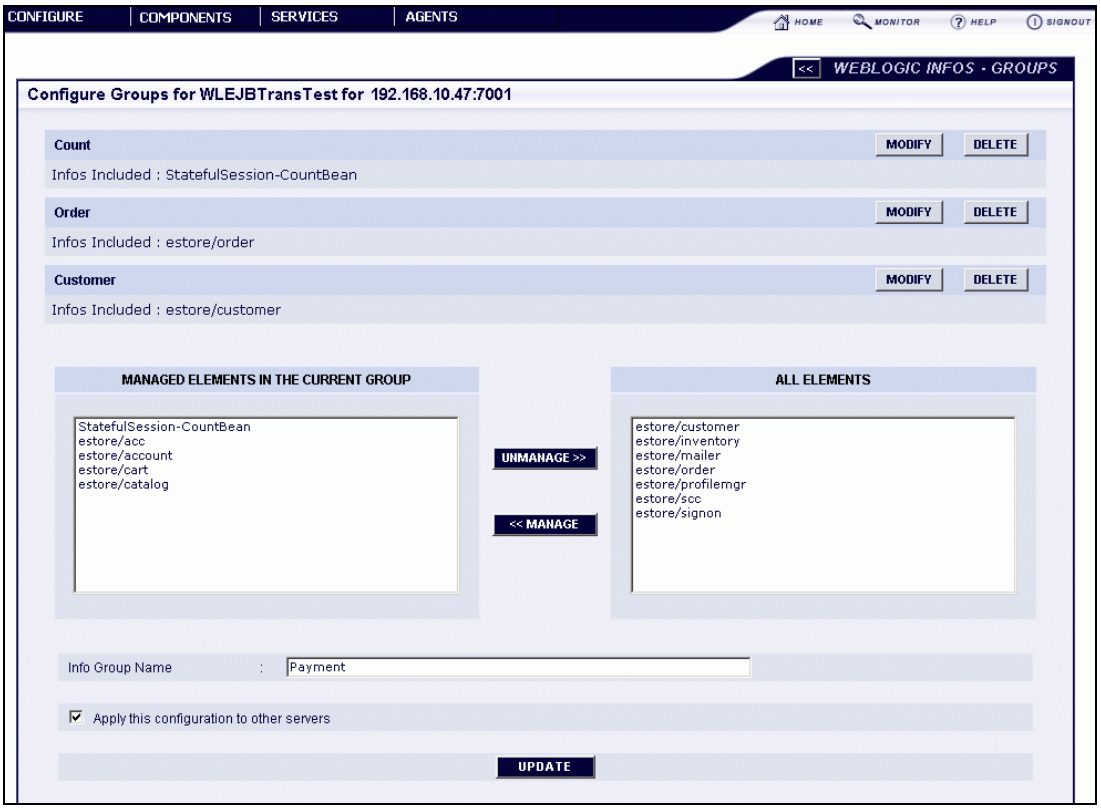


Figure 2.119: Managing the selected EJB components

11. Next, click on the **MANAGE** button and add the selection to the **MANAGED ELEMENTS IN THE CURRENT GROUP** list (see Figure 2.118 and Figure 2.119). Similarly, multiple EJBs can be added to the **MANAGED ELEMENTS IN THE CURRENT GROUP** list. The EJB components so managed will thus form a part of the new EJB group. On the contrary, to unmanage a managed EJB component, select the component from the **MANAGED ELEMENTS IN THE CURRENT GROUP** list, and click the **UNMANAGE** button. This will transfer the selected EJBs back to the **ALL ELEMENTS** list.
12. Then, assign a name to the newly created EJB group in the **Info Group Name** text box (see Figure 2.119). If the EJB group so created is to be associated with other WebLogic servers in the environment, then, select the **Apply this configuration to other servers** check box of Figure 2.119.
13. Finally, click the **UPDATE** button to register the changes.
14. If the **Apply this configuration to other servers** check box is selected, then upon clicking the **UPDATE** button, Figure 2.120 will appear.

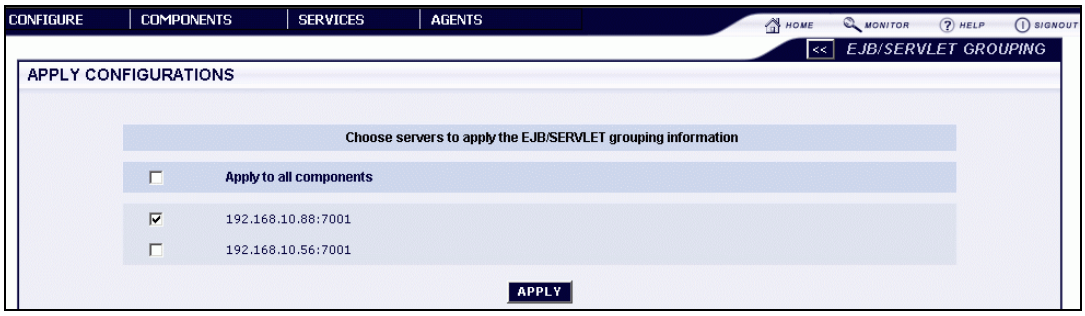


Figure 2.120: Associating the EJB group with other WebLogic servers

15. Using Figure 2.120, select the WebLogic servers to which the EJB group configuration is to be applied, by clicking on the corresponding check boxes. To select all the displayed servers, click on the **Apply to all components** check box. Finally, click the **APPLY** button in Figure 2.120.
16. A summary of the selection will then be displayed (see Figure 2.121). To delete the displayed group, click on the **DELETE** button alongside it (see Figure 2.121). If you had earlier associated the group with other WebLogic servers, then all or a few of these associations can be removed by selecting the **Apply configuration to other servers** check box before clicking on the corresponding **DELETE** button. This will lead you back to Figure 2.120, where you can select the WebLogic servers for which the EJB group configuration has to be removed. Clicking on the **APPLY** button after selecting the servers will ensure that the corresponding EJB group configuration does not apply to the chosen servers any longer. To modify a displayed group, click on the **MODIFY** button corresponding to the group name (see Figure 2.121). To return to the WLEJBTrans test configuration page, click the **<<** button in Figure 2.121.

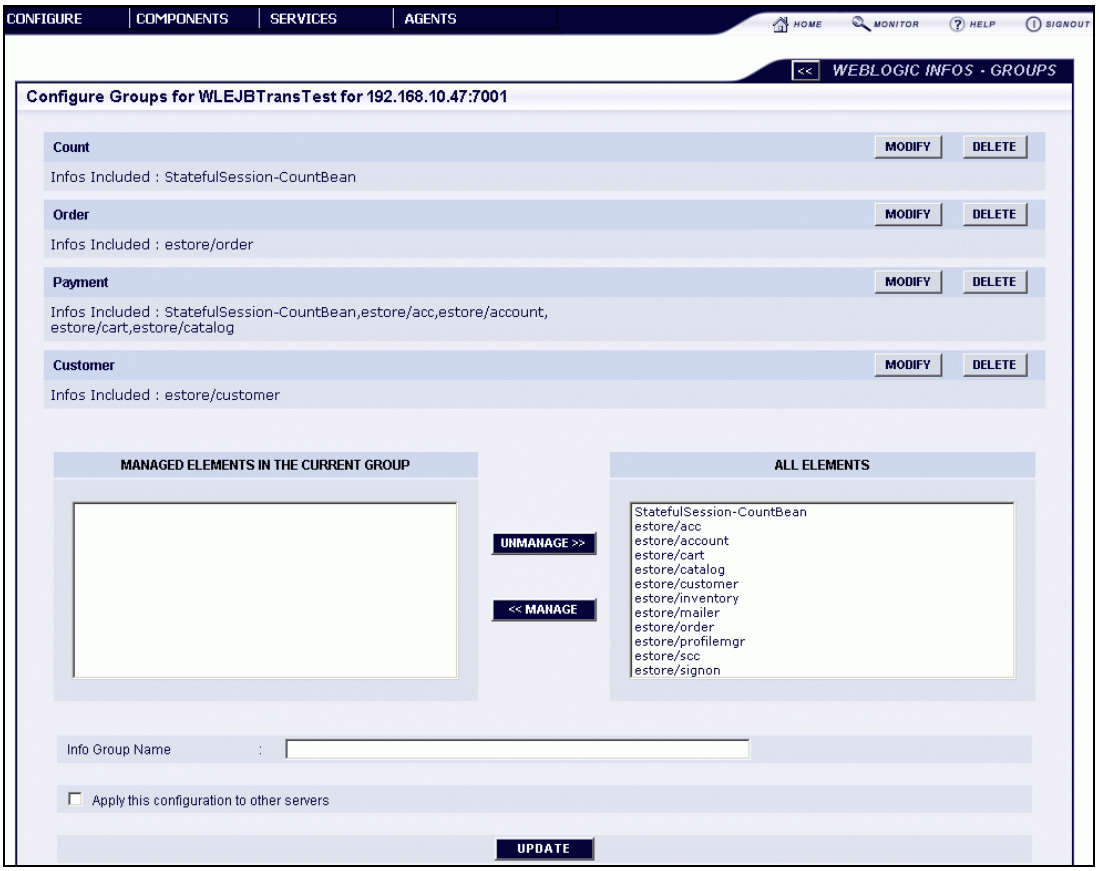


Figure 2.121: Summary of the selection

17. Now, attempt to sign out of the admin interface yet again. This time around, you will be prompted to configure the **Processes** test for the WebLogic server. Configure the patterns of processes that are critical for WebLogic operations to be monitored by this test.

18. The WebLogicServlets test, by default, auto-discovers all the servlets on the monitored WebLogic server, and reports key statistics pertaining to every servlet. Alternatively, like EJBs, you can configure groups of servlets and monitor every group so created. To create one/more servlet groups, first, open the **AGENTS –SPECIFIC TEST CONFIGURATION** page following the menu sequence: Agents -> Tests -> Configure -> Specific. In the **AGENTS –SPECIFIC TEST CONFIGURATION** page, pick **WebLogic** as the **Component type**, and a specific component from the Component name list boxes. Next, select Performance from the Test type list box. This will display complete list of unconfigured tests available for the WebLogic server in the **UNCONFIGURED TESTS** list box. Locate the WebLogicServlets test in the **UNCONFIGURED TESTS** list and click the **Configure** button corresponding to that test (see Figure 2.122).

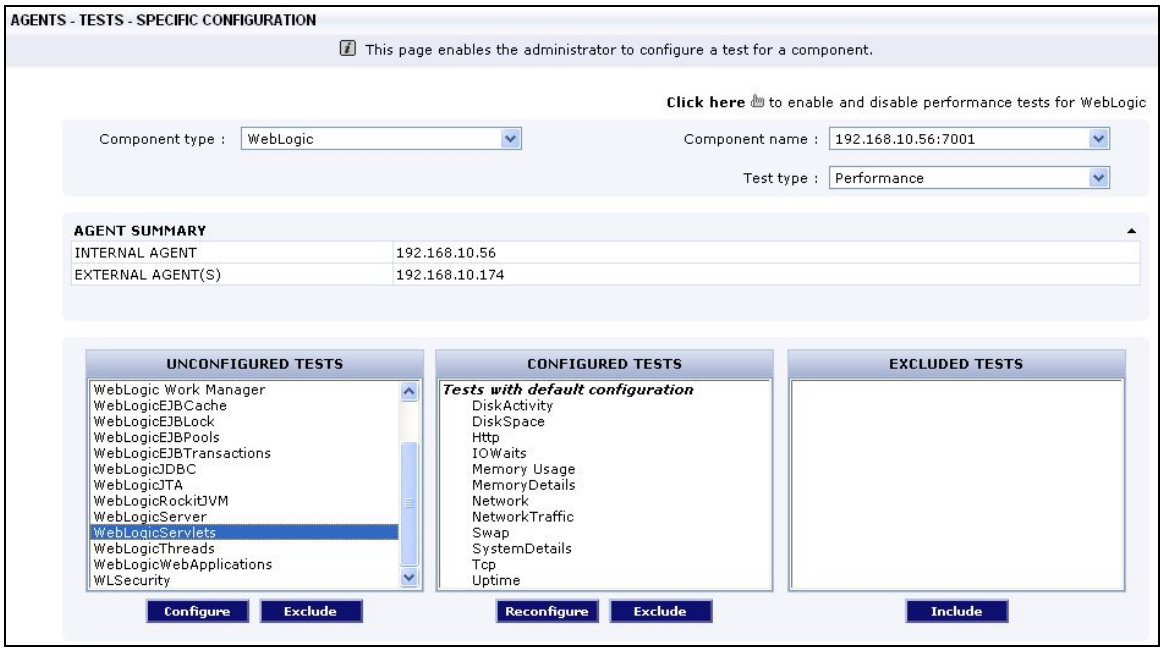


Figure 2.122: List of tests available for a WebLogic server

19. Figure 2.123 then appears revealing the parameters for the **WebLogicServlets** test.

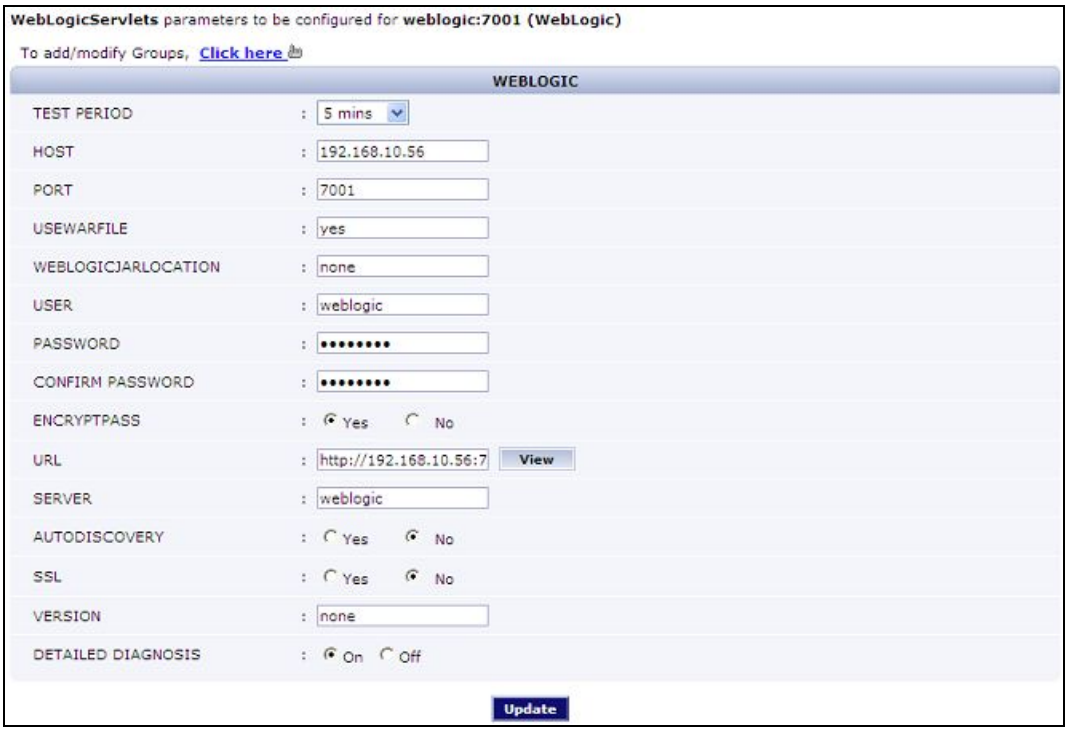


Figure 2.123: Configuring the WeblogicServlet test

20. Configure the WeblogicServlets test (see Figure 2.123) in the same manner as the **WeblogicEJBTransactions** test, and click the **Update** button in Figure 2.123 to register the changes.
21. Allow the test to run once and reconfigure it by clicking the **RECONFIGURE** button in the test configuration page. Figure 2.123 will reappear. This time, click on the **Click here** hyperlink in Figure 2.123 to configure the servlet groups. The eG Enterprise suite will monitor only those components that belong to a servlet group.
22. Upon clicking the hyperlink, Figure 2.124 will appear, which will list the servlet groups that pre-exist (if any), the servlets that have been added to the group, and allows the administrator to create and configure new ones, and delete/modify existing ones.
23. To configure a new servlet group, first select the servlets to be added to the group from the **ALL ELEMENTS** list in Figure 2.124. All the servlets hosted on a WebLogic server are discovered when the agent starts executing on the WebLogic Server's host. These discovered servlets will be listed in the **ALL ELEMENTS** list. This list will get updated every time the agent is restarted / when the rediscovery scheduled for every four hours occurs.

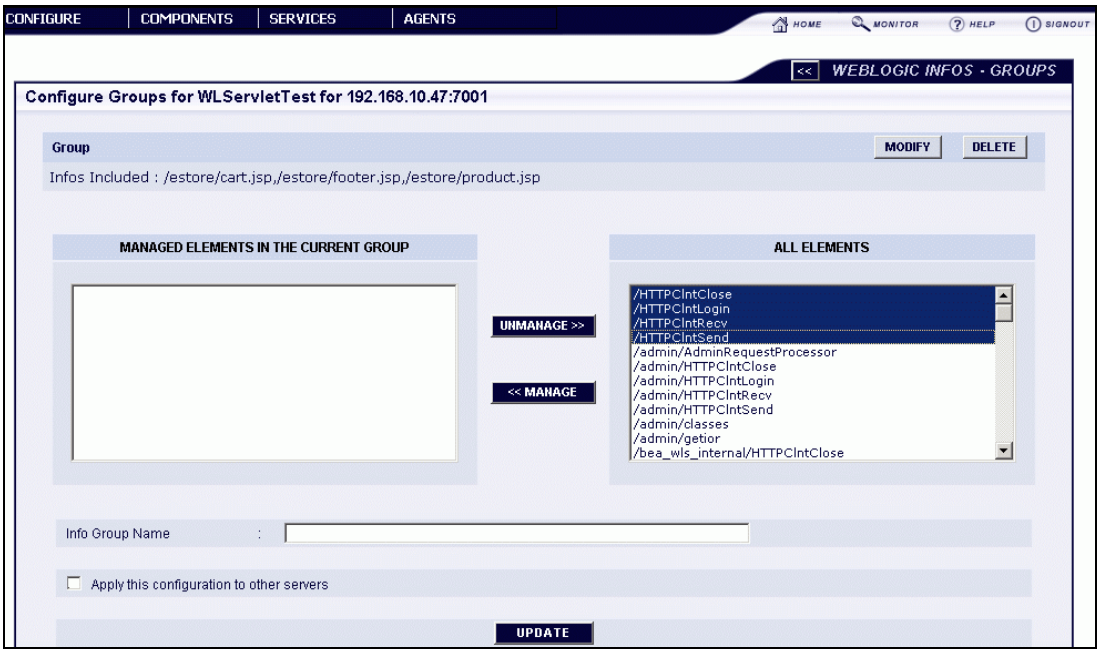


Figure 2.124: Selecting the servlets to be added to the new group

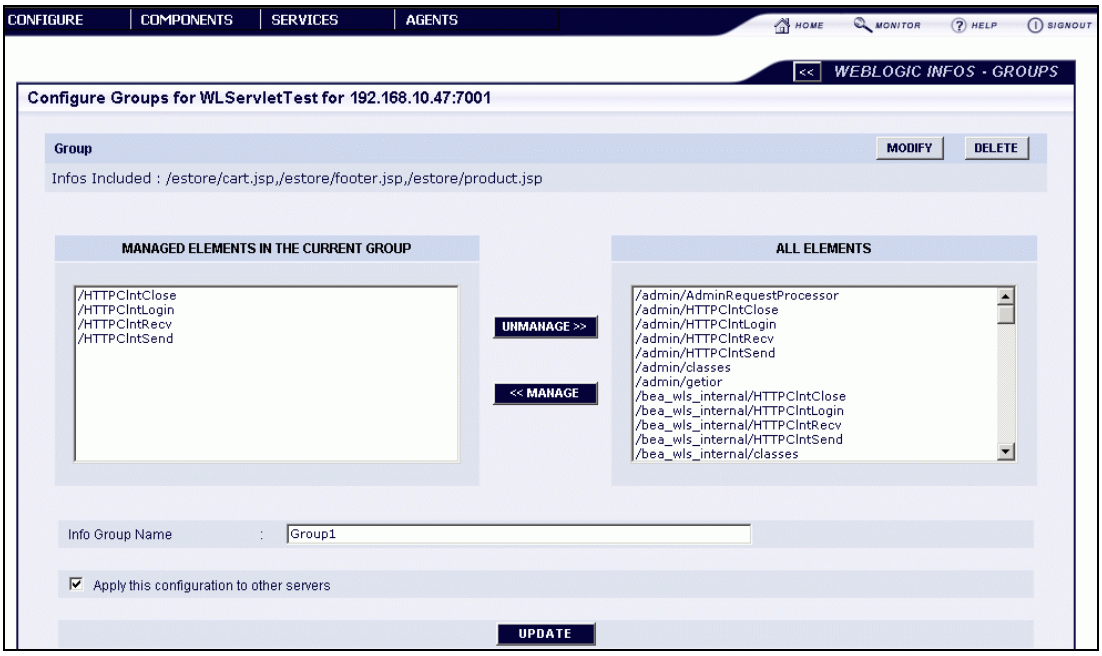


Figure 2.125: Managing the selected servlets

24. Next, click on the **MANAGE** button and add the selection to the **MANAGED ELEMENTS IN THE CURRENT GROUP** list (see Figure 2.124 and Figure 2.125). Similarly, multiple servlets can be added to the **MANAGED ELEMENTS IN THE CURRENT GROUP** list. The servlets so managed will thus form a part of the new servlet group. To unmanage a managed servlet, select the component from the **MANAGED ELEMENTS IN THE CURRENT GROUP** list, and click the **UNMANAGE** button. This will transfer the selected servlets back to the **ALL ELEMENTS** list.
25. Then, assign a name to the newly created servlet group in the **Info Group Name** text box (see Figure 2.125).
26. If the servlet group so created is to be associated with other WebLogic servers in the environment, then, select the **Apply this configuration to other servers** check box of Figure 2.123.
27. Finally, click the **UPDATE** button to register the changes.
28. If the **Apply this configuration to other servers** check box is selected, then upon clicking the **UPDATE** button, Figure 2.126 will appear.

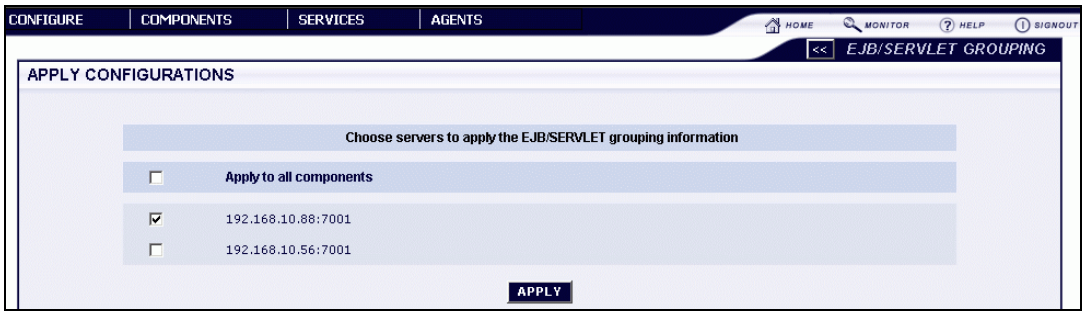


Figure 2.126: Associating the servlet group with other WebLogic servers

29. A summary of the selection will then be displayed (see Figure 2.127). To delete the displayed group, click on the **DELETE** button alongside it (see Figure 2.127). If you had earlier associated a group with other WebLogic servers, then all or a few of these associations can be removed by selecting the **Apply configuration to other servers** check box before clicking on the corresponding **DELETE** button. This will lead you back to Figure 2.126, where you can select the WebLogic servers for which the servlet group configuration has to be removed. Clicking on the **APPLY** button after selecting the servers will ensure that the corresponding servlet group configuration does not apply to the chosen servers any longer. To modify a displayed group, click on the **MODIFY** button corresponding to the group name (see Figure 2.127). To return to the WLServlet test configuration page, click the **<<** button in Figure 2.127.

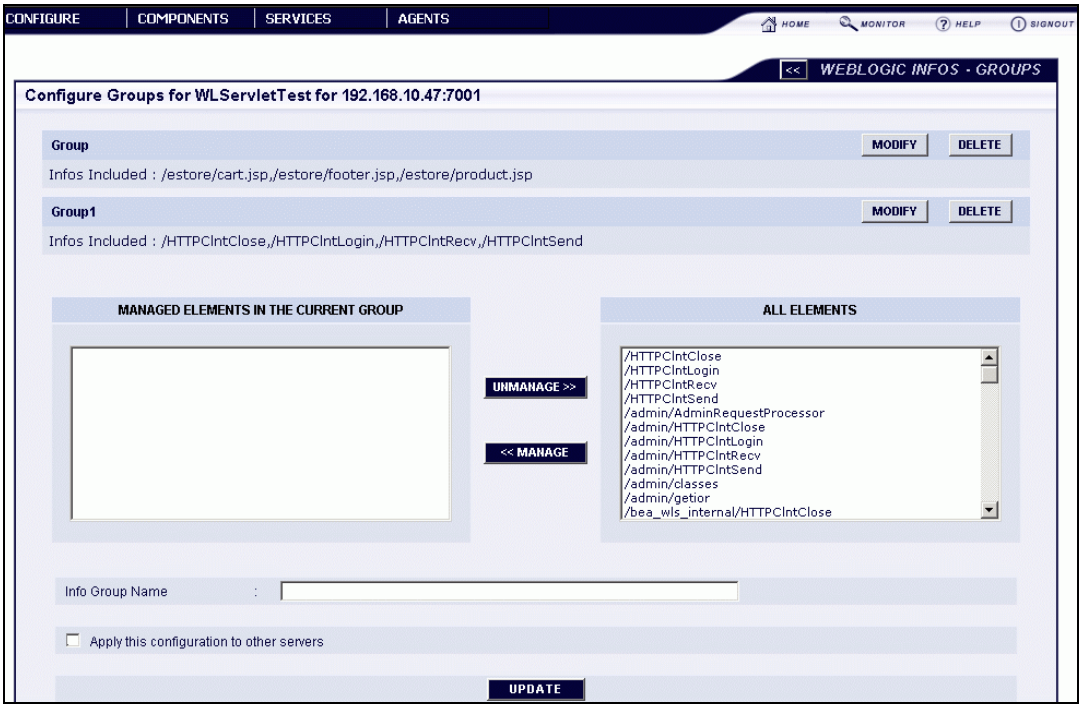


Figure 2.127: Summary of the selection

30. Now that the tests are configured, proceed to view the measurements gathered by these tests on monitoring the WebLogic Server.

Chapter 3: Monitoring the WebLogic Server Ver. 9.0 (and above)

The special WebLogic monitoring model (see Figure 3.1) that eG Enterprise offers provides uses JMX (Java Management extension), the new standard for managing java components, for monitoring the WebLogic server 9.0 (and above). JMX allows users to instrument their applications and control or monitor them using a management console. Using this mechanism, over a hundred critical metrics relating to a WebLogic server instance can be monitored in real-time and alerts can be generated based on user-defined thresholds or auto-computed baselines.

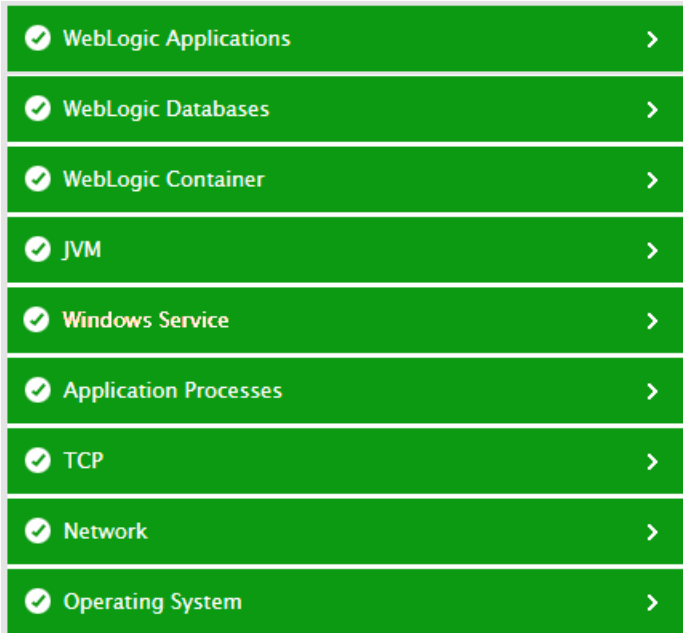


Figure 3.1: Layer model of the WebLogic Application server

The sections that will follow discuss the top 4 layers of Figure 3.1, and the metrics they report. In addition, the **Application Processes** layer will also be touched upon, as it includes an additional test for WebLogic servers called the **Windows Service Resources** test. The remaining layers have been extensively dealt with in the *Monitoring Unix and Window Servers* document.

3.1 The Application Processes Layer

The default **Processes** test mapped to this layer reports the availability and resource usage of the processes that are critical to the functioning of the WebLogic server. For more details about the **Processes** test, refer to the Monitoring Unix and Windows Servers document.

If the WebLogic server is operating on a Windows host, then, optionally, you can configure a **Windows Service Resources** test for the server. This test reports the availability and resource usage of a configured service.

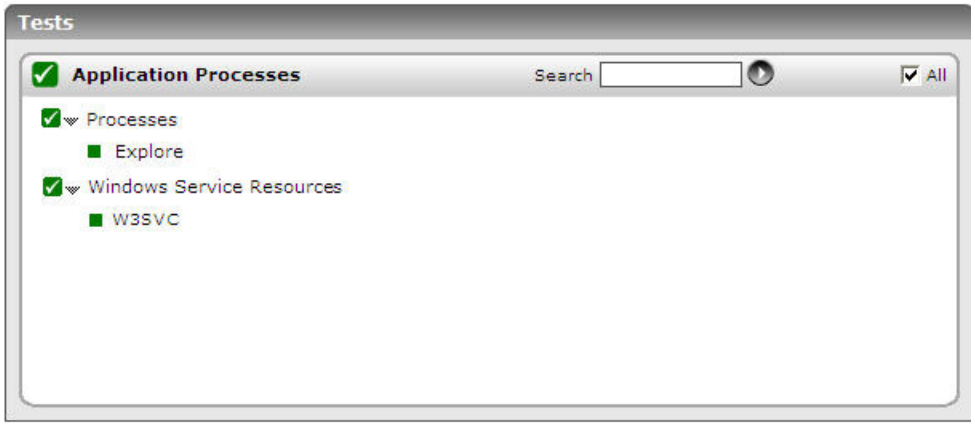


Figure 3.2: The tests mapped to the Application Processes layer of the WebLogic server

3.1.1 Windows Service Resources Test

For a configured service, this test reports whether that service is up and running or not. In addition, the test automatically determines the ID and name of the process that corresponds to the configured service, and measures the CPU and memory usage of that process and the I/O load imposed by the process.

Note:

This test executes only on Windows hosts.

This test is disabled by default. **Enable this test only if the WebLogic server is operating on a Windows host**. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *WebLogic* as the **Component type**, *Performance* as the **Test type**, choose this 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 WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the ServiceName configured.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the WebLogic server for which this test is to be configured.
Port	The port number of the WebLogic server.
ServiceName	Specify the exact name of the service to be monitored. For eg., to monitor the World Wide Web Publishing service, the ServiceName should be: *W3SVC*. If your service name embeds white spaces, then specify the service name within "double-quotes".

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Service availability	Indicates whether the configured service is available or not.	Percent	If the service exists on the target host and is currently running, then this measure will report the value 100. On the other hand, if the service exists but is not running, then this measure will report the value 0. If the service does not exist, then the test will report the value Unknown.
CPU utilization	Indicates the percentage of CPU utilized by the process that corresponds to the configured ServiceName.	Percent	A very high value could indicate that the service is consuming excessive CPU resources.
Memory utilization	For the process corresponding to the specified ServiceName, this value represents the ratio of the resident set size of the process to the physical memory of the host system, expressed as a percentage.	Percent	A sudden increase in memory utilization for a process may be indicative of memory leaks in the application.
Handle count	Indicates the number of handles opened by the	Number	An increasing trend in this measure is indicative of a memory leak in the

Measurement	Description	Measurement Unit	Interpretation
	process mapped to the configured ServiceName.		service.
Number of threads	Indicates the number of threads that are used by the process that corresponds to the configured ServiceName.	Number	
Virtual memory used	Indicates the amount of virtual memory that is being used by the process that corresponds to the configured ServiceName.	MB	
I/O data rate	Indicates the rate at which the process mapped to the configured servicename is reading and writing bytes in I/O operations.	Kbytes/Sec	This value counts all I/O activity generated by a process and includes file, network and device I/Os.
I/O data operations	Indicates the rate at which the process corresponding to the specified ServiceName is issuing read and write data to file, network and device I/O operations.	Operations/Sec	
I/O read data rate	Indicates the rate at which the process that corresponds to the configured service name is reading data from file, network and device I/O operations.	Kbytes/Sec	
I/O write data rate	Indicates the rate at which the process (that corresponds to the configured ServiceName) is writing data to file, network and device I/O	Kbytes/Sec	

Measurement	Description	Measurement Unit	Interpretation
	operations.		
Page fault rate	Indicates the total rate at which page faults are occurring for the threads of the process that maps to the configured ServiceName.	Faults/Sec	A page fault occurs when a thread refers to a virtual memory page that is not in its working set in main memory. This may not cause the page to be fetched from disk if it is on the standby list and hence already in main memory, or if it is in use by another process with whom the page is shared.
Memory working set	Indicates the current size of the working set of the process that maps to the configured ServiceName.	MB	The Working Set is the set of memory pages touched recently by the threads in the process. If free memory in the computer is above a threshold, pages are left in the Working Set of a process even if they are not in use. When free memory falls below a threshold, pages are trimmed from Working Sets. If they are needed they will then be soft-faulted back into the Working Set before leaving main memory. By tracking the working set of a process over time, you can determine if the application has a memory leak or not.

3.2 The JVM Layer

A Java virtual machine (JVM), an implementation of the Java Virtual Machine Specification, interprets compiled java binary code for a computer's processor (or "hardware platform") so that it can perform a Java program's instructions. The Java Virtual Machine Specification defines an abstract -- rather than a real -- machine or processor. The Specification specifies an instruction set, a set of registers, a stack, a garbage heap, and a method area.

The tests associated with the **JVM** layer of WebLogic enables administrators to perform the following functions:

- Assess the effectiveness of the garbage collection activity performed on the JVM heap
- Monitor WebLogic thread usage
- Evaluate the performance of the BEA JRockit JVM

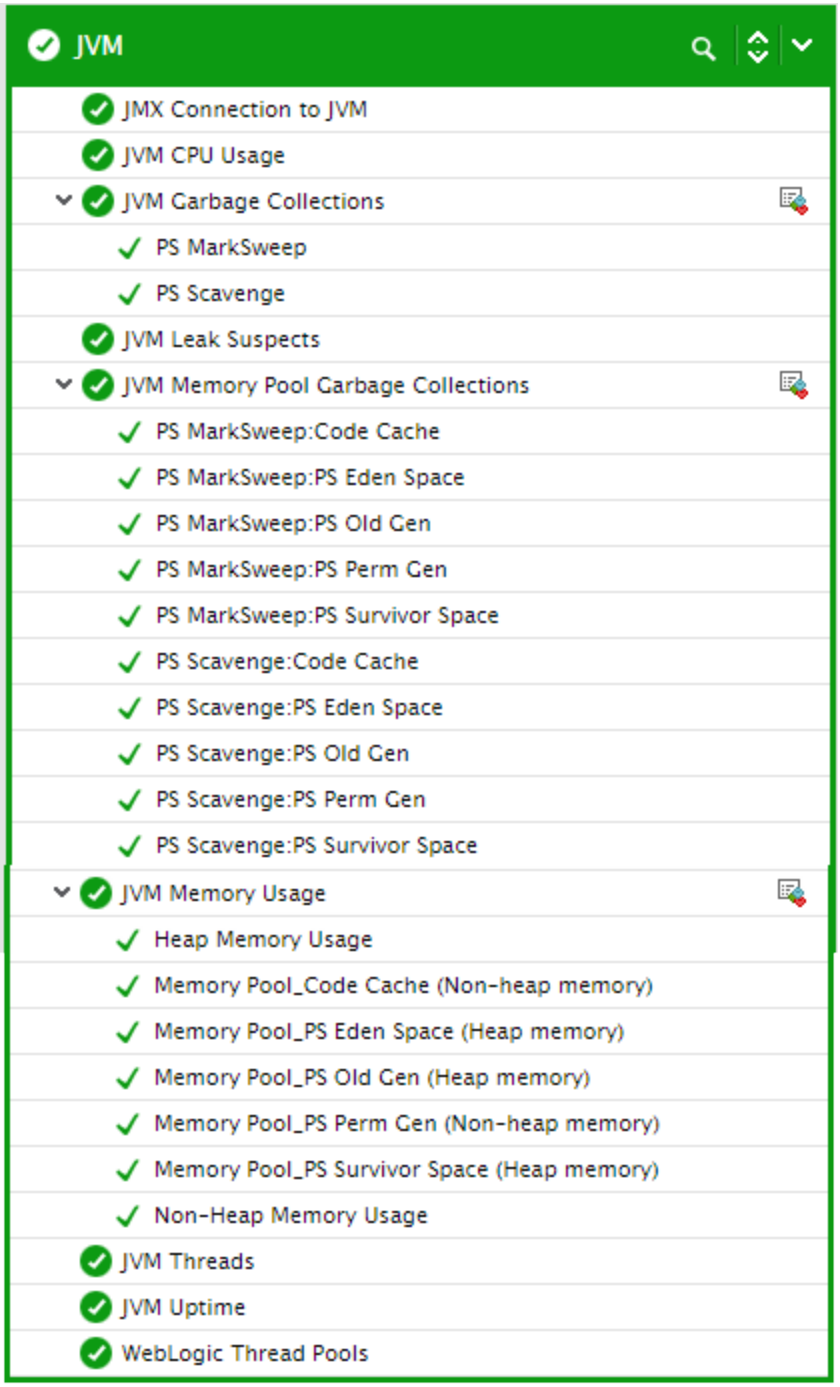


Figure 3.3: The tests associated with the JVM layer

3.2.1 WebLogic Rockit JVM Test

This test exposes runtime data about the JRockit Virtual Machine (VM) that is running the current WebLogic Server instance.

This test will work only if the following conditions are fulfilled:

- The Weblogic Server must be launched on the JRockit JVM
- The *managementapi.jar* should be on the Weblogic server's startup classpath

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every WebLogic server being monitored.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The IP address of the WebLogic server.
Port	The port number of the WebLogic server.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the

Parameter	Description
	metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the HOSTand PORT). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the egurkha war file is deployed to the admin server, and it is up and running.
Version	The Version textbox indicates the version of the Weblogic server to be managed. The default value is "none", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "none", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UserWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UserWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UserWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WebLogicJarLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the *weblogic.jar* file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total heap	Indicates the amount of memory currently allocated to the Virtual Machine's Java heap.	MB	
Used heap	Indicates the amount of Java heap memory that is currently being used by the Virtual Machine.	MB	If the value of this measure increases consistently, it is indicative of heavy load on the Virtual Machine.
Free heap	Indicates the amount of Java heap memory that is currently free in the Virtual Machine.	MB	A very low value of this measure is a cause of concern, as it indicates a heavy utilization of the JVM heap. Consider increasing the JVM heap size under such circumstances.
Total nursery	Indicates the amount of memory that is currently allocated to the nursery.	MB	
GC count	Indicates the number of garbage collection runs that have occurred since the Virtual Machine was started.	Number	If GC has run too many times during a short interval, it indicates that the JVM is in dire need of free heap for normal functioning. Moreover, frequent GC executions could cause application performance to deteriorate. In order to avoid this, it is recommended that you increase the heap size or alter the GC frequency.
GC time	Indicates the time that the Virtual Machine has spent on all garbage collection runs since the VM was started.	Secs	
Total load	Indicates the percentage of load that the Virtual Machine is placing on all processors in the host computer.	Percent	
Percent heap used	Indicates the percentage of the total Java heap memory that is currently being used by the Virtual Machine.	Percent	

3.2.2 WebLogic Thread Pools Test

Starting from WebLogic server release 9.0, every server instance uses a self-tuned thread-pool. All requests, whether related to system administration or application activity—are processed by this single thread pool. The self-tuning thread pool would also adjust its pool size automatically based on the throughput history that WLS gathers every 2 seconds and based on queue size.

This test monitors how the self-tuning thread pool is being used, and in the process reports whether there are adequate idle threads in the pool to handle additional workload that may be imposed on the WebLogic server. The test also turns the spot light on the request (if any) that is hogging threads, and enables you to quickly capture a sudden/consistent increase in queue size, which in turn might impact the pool size.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the self-tuning thread pool on the WebLogic server being monitored.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The IP address of the WebLogic server.
Port	The port number of the WebLogic server.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.

Parameter	Description
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the HOSTand PORT). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the egurkha war file is deployed to the admin server, and it is up and running.
Version	The Version textbox indicates the version of the Weblogic server to be managed. The default value is "none", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "none", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UserWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UserWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UserWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WebLogicJarLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the *weblogic.jar* file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Active threads	Indicates the total number of active threads in this pool.	Number	A high value for this measure is indicative of a high load on the applications deployed on the WebLogic server. This measure is also useful for determining usage trends. For example, it can show the time of day and the day of the week in which you usually reach peak thread count. In addition, the creation of too many threads can result in out of memory errors or thrashing. By watching this metric, you can reduce excessive memory consumption before it's too late.
Hogging threads	Indicates the number of threads that are currently hogged by a request.	Number	Ideally, the value of this measure should be low. A very high value indicates that a request is using up too many threads. Hogging threads will either be declared as stuck after the configured timeout or will return to the pool before that. The self-tuning mechanism will backfill if necessary. WebLogic Server automatically detects when a thread in a pool becomes "stuck." Because a stuck thread cannot complete its current work or accept new work, the server logs a message each time it diagnoses a stuck thread. WebLogic Server diagnoses a thread as stuck if it is continually working (not idle) for a set period of time. You can tune a server's thread detection behavior by changing the length of time before a thread is diagnosed as stuck, and by changing

Measurement	Description	Measurement Unit	Interpretation
			the frequency with which the server checks for stuck threads. Although you can change the criteria WebLogic Server uses to determine whether a thread is stuck, you cannot change the default behavior of setting the "warning" and "critical" health states when all threads in a particular execute queue become stuck.
Idle threads	Indicates the number of idle threads (i.e., the threads that are ready to process a new job as and when it arrives) in the pool.	Number	A high value is desired for this measure.
Queue length	Indicates the number of pending requests in the priority queue.	Number	This measure comprises of both the internal system requests and requests made by the user. A low value is desired for this measure. A high value or a sudden increase in this value may indicate a sudden slowdown in responsiveness or a performance bottleneck.
Standby threads	Indicates the number of threads that are currently in the standby pool.	Number	Threads that are not needed to handle the present work load are designated as standby and are added to the standby pool. These threads are activated when more threads are needed.
Throughput	Indicates the number of requests in the priority queue that are completed.	Number	The queue monitors throughput over time and based on history, determines whether to adjust the thread count or not. For example, if historical throughput statistics indicate that a higher thread count increased throughput, the server increases it. Similarly, if statistics indicate that fewer threads did not reduce

Measurement	Description	Measurement Unit	Interpretation
			throughput, the count will be reduced.
Total threads	Indicates the total number of threads in this pool.	Number	

3.3 Tests Disabled by Default for the JVM Layer

The tests discussed above are enabled by default for a WebLogic server. Besides these tests, the eG agent can be optionally configured to execute a few other tests on the WebLogic server's JVM so as to report critical statistics related to the Java transactions, classes loaded/unloaded, threads used, CPU and memory resources used, garbage collection activity, uptime of the JVM, etc. These additional tests are disabled by default for the WebLogic server. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, choose *WebLogic* as 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.

These JVM tests have been discussed below.

When a user initiates a transaction to a Java-based web application, the transaction typically travels via many Java components before completing execution and sending out a response to the user.

Figure 3.4 reveals some of the Java components that a web transaction/web request visits during its journey.

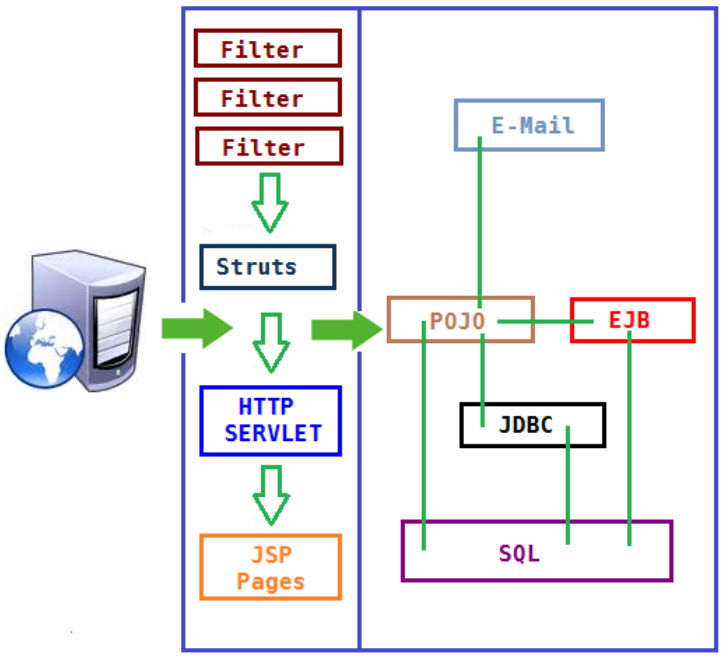


Figure 3.4: The layers through which a Java transaction passes

The key Java components depicted by Figure 3.4have been briefly described below:

- **Filter**: A filter is a program that runs on the server before the servlet or JSP page with which it is associated. All filters must implement *javax.servlet.Filter*. This interface comprises three methods: init, doFilter, and destroy.
- **Servlet**: A servlet acts as an intermediary between the client and the server. As servlet modules run on the server, they can receive and respond to requests made by the client. If a servlet is designed to handle HTTP requests, it is called an HTTP Servlet.
- **JSP**: Java Server Pages are an extension to the Java servlet technology. A JSP is translated into Java servlet before being run, and it processes HTTP requests and generates responses like any servlet. Translation occurs the first time the application is run.
- **Struts** : The Struts Framework is a standard for developing well- architected Web applications. Based on the Model- View- Controller (MVC) design paradigm, it distinctly separates all three levels (Model, View, and Control).

A delay experienced by any of the aforesaid Java components can adversely impact the total response time of the transaction, thereby scarring the user experience with the web application. In addition, delays in JDBC connectivity and slowdowns in SQL query executions (if the application interacts with a database), bottlenecks in delivery of mails via the Java Mail API (if used), and any

slow method calls, can also cause insufferable damage to the 'user-perceived' health of a web application.

The challenge here for administrators is to not just isolate the slow transactions, but to also accurately identify where the transaction slowed down and why - is it owing to inefficient JSPs? poorly written servlets or struts? poor or the lack of any JDBC connectivity to the database? long running queries? inefficient API calls? or delays in accessing the POJO methods? The eG JTM Monitor provides administrators with answers to these questions!

With the help of the **Java Transactions** test, the eG JTM Monitor traces the route a configured web transaction takes, and captures live the total responsiveness of the transaction and the response time of each Java component it visits en route. This way, the solution proactively detects transaction slowdowns, and also precisely points you to the Java components causing it - is it the Filters? JSPs? Servlets? Struts? JDBC? SQL query? Java Mail API? or the POJO? In addition to revealing where (i.e., at which Java component) a transaction slowed down, the solution also provides the following intelligent insights, on demand, making root-cause identification and resolution easier:

- A look at the methods that took too long to execute, thus leading you to those methods that may have contributed to the slowdown;
- Single-click access to each invocation of a chosen method, which provides pointers to when and where a method spent longer than desired;
- A quick glance at SQL queries and Java errors that may have impacted the responsiveness of the transaction;

Using these interesting pointers provided by the eG JTM Monitor, administrators can diagnose the root-cause of transaction slowdowns within minutes, rapidly plug the holes, and thus ensure that their critical web applications perform at peak capacity at all times!

Before attempting to monitor Java transactions using the eG JTM Monitor, the following configurations will have to be performed:

1. In the <EG_INSTALL_DIR>\lib directory (on Windows; on Unix, this will be /opt/egurkha/lib) of the eG agent, you will find the following files:
    - eg_jtm.jar
    - aspectjrt.jar
    - aspectjweaver.jar
    - jtmConn.props

- jtmLogging.props
- jtmOther.props

2. Login to the system hosting the Java application to be monitored.
3. If the eG agent will be 'remotely monitoring' the target Java application (i.e., if the Java application is to be monitored in an 'agentless manner'), then, copy all the files mentioned above from the <EG_INSTALL_DIR>\lib directory (on Windows; on Unix, this will be /opt/egurkha/lib) of the eG agent to any location on the Java application host.
4. Then, proceed to edit the start-up script of the Java application being monitored, and append the following lines to it:

```
set JTM_HOME=<<PATH OF THE LOCAL FOLDER CONTAINING THE JAR FILES AND PROPERTY FILES
LISTED ABOVE>>
"-javaagent:%JTM_HOME%\aspectjweaver.jar"
"-DEG_JTM_HOME=%JTM_HOME%"
```

   Note that the above lines will change based on the operating system and the web/web application server being monitored.

   Then, add the **eg_jtm.jar**, **aspectjrt.jar**, and **aspectjweaver.jar** files to the classpath of the Java application being monitored.

   Finally, save the file. Once this is done, then, the next time the Java application starts, the eG JTM Monitor scans the web requests to the application for configured URL patterns. When a match is found, the eG JTM Monitor collects the desired metrics and stores them in memory.

   Then, every time the eG agent runs the **Java Transactions** test, the agent will poll the eG JTM Monitor (on the target application) for the required metrics, extract the same from the application's memory, and report them to the eG manager.

5. Next, edit the **jtmConn.props** file. You will find the following lines in the file:

```
#Contains the connection properties of eGurkha Java Transaction Monitor
JTM_Port=13631
Designated_Agent=
```

   By default, the JTM Port parameter is set to 13631. If the Java application being monitored listens on a different JTM port, then specify the same here. In this case, when managing a Java Application using the eG administrative interface, specify the JTM Port that you set in the **jtmConn.props** file as the Port of the Java application.

Also, against the Designated_Agent parameter, specify the IP address of the eG agent which will poll the eG JTM Monitor for metrics. If no IP address is provided here, then the eG JTM Monitor will treat the host from which the very first 'measure request' comes in as the Designated_Agent.

Note:

In case a specific Designated_Agent is not provided, and the eG JTM Monitor treats the host from which the very first 'measure request' comes in as the Designated_Agent, then if such a Designated_Agent is stopped or uninstalled for any reason, the eG JTM Monitor will wait for a maximum of 10 measure periods for that 'deemed' Designated_Agent to request for metrics. If no requests come in for 10 consecutive measure periods, then the eG JTM Monitor will begin responding to 'measure requests' coming in from any other eG agent.

6. Finally, save the **jtmConn.props** file.

Then, proceed to configure the **Java Transactions** test as discussed below.

3.3.1 Java Transactions Test

Target of the test : A Java application/web application server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for each configured URL pattern.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The IP address of the host for which this test is to be configured.
Port	The port number at which the specified host listens; if Java Transaction Monitoring is enabled for the target Java application, then the JTM Port has to be specified here.
JTM Port	Specify the port number configured as the JTM Port in the **jtmConn.props** file described in the procedure outlined above.
URL Patterns	Provide a comma-separated list of the URL patterns of web requests/transactions to be monitored. The format of your specification should be as follows: *<DisplayName_of_Pattern>:<Transaction_Pattern>*. For instance, your specification can be: *login:*log*,ALL:*,pay:*pay**
Filtered URL Patterns	Provide a comma-separated list of the URL patterns of transactions/web requests to be excluded from the monitoring scope of this test. For example, **blog*,*paycheque**

Parameter	Description
Slow URL Threshold	The *Slow transactions* measure of this test will report the number of transactions (of the configured patterns) for which the response time is higher than the value (in seconds) specified here.
Method Exec Cutoff	The detailed diagnosis of the *Slow transactions* measure allows you to drill down to a **URL tree**, where the methods invoked by a chosen transaction are listed in the descending order of their execution time. By configuring an execution duration (in seconds) here, you can have the **URL Tree** list only those methods that have been executing for a duration greater the specified value. For instance, if you specify *5* here, the URL tree for a transaction will list only those methods that have been executing for over 5 seconds, thus shedding light on the slow method calls alone.
Max Slow URLs per Test Period	Specify the number of top-n transactions (of a configured pattern) that should be listed in the detailed diagnosis of the *Slow transactions* measure, every time the test runs. By default, this is set to *10*, indicating that the detailed diagnosis of the *Slow transactions* measure will by default list the top-10 transactions, arranged in the descending order of their response times.
Max Error URLs per Test Period	Specify the number of top-n transactions (of a configured pattern) that should be listed in the detailed diagnosis of the *Error transactions* measure, every time the test runs. By default, this is set to *10*, indicating that the detailed diagnosis of the *Error transactions* measure will by default list the top-10 transactions, in terms of the number of errors they encountered.
DD Frequency	Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD Frequency.
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: • The eG manager license should allow the detailed diagnosis capability • Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total transactions	Indicates the total number of transactions of this pattern that the target application handled during the last measurement period.	Number	
Avg. response time	Indicates the average time taken by the transactions of this pattern to complete execution.	Secs	Compare the value of this measure across patterns to isolate the type of transactions that were taking too long to execute. You can then take a look at the values of the other measures to figure out where the transaction is spending too much time.
Slow transactions	Indicates the number of transactions of this pattern that were slow during the last measurement period.	Number	This measure will report the number of transactions with a response time higher than the configured slow url threshold. A high value is a cause for concern, as too many slow transactions to an application can significantly damage the user experience with that application. Use the detailed diagnosis of this measure to know which transactions are slow.
Slow transactions response time	Indicates the average time taken by the slow transactions of this pattern to execute.	Secs	
Error transactions	Indicates the number of transactions of this pattern that experienced errors during the last measurement period.	Number	A high value is a cause for concern, as too many error-prone transactions to an application can significantly damage the user experience with that application. Use the detailed diagnosis of this

Measurement	Description	Measurement Unit	Interpretation
			measure to isolate the error transactions.
Error transactions response time	Indicates the average duration for which the transactions of this pattern were processed before an error condition was detected.	Secs	
Filters	Indicates the number of filters that were accessed by the transactions of this pattern during the last measurement period.	Number	A filter is a program that runs on the server before the servlet or JSP page with which it is associated.
Filters response time	Indicates the average time spent by the transactions of this pattern at the Filters layer.	Secs	Typically, the init, doFilter, and destroy methods are called at the Filters layer. Issues in these method invocations can increase the time spent by a transaction in the Filters Java component. Compare the value of this measure across patterns to identify the transaction pattern that spent the maximum time with the Filters component. If one/more transactions of a pattern are found to be slow, then, you can compare the value of this measure with the other response time values reported by this test to determine where the slowdown actually occurred - in the filters, in JSPs, in servlets, in struts, in exception handling, when executing JDBC/SQL queries, when sending Java mails, or when accessing POJOs.
JSPs accessed	Indicates the number of JSPs accessed by the transactions of this pattern during the last	Number	

Measurement	Description	Measurement Unit	Interpretation
	measurement period.		
JSPs response time	Indicates the average time spent by the transactions of this pattern at the JSP layer.	Secs	Compare the value of this measure across patterns to identify the transaction pattern that spent the maximum time in JSPs. If one/more transactions of a pattern are found to be slow, then, you can compare the value of this measure with the other response time values reported by this test to determine where the slowdown actually occurred - in the filters, in JSPs, in servlets, in struts, in exception handling, when executing JDBC/SQL queries, when sending Java mails, or when accessing POJOs..
HTTP Servlets Accessed	Indicates the number of HTTP servlets that were accessed by the transactions of this pattern during the last measurement period.	Number	
HTTP servlets response time	Indicates the average time taken by the HTTP servlets for processing the HTTP requests of this pattern.	Secs	Badly written servlets can take too long to execute, and can hence obstruct the smooth execution of the dependent transactions. By comparing the value of this measure across patterns, you can figure out which transaction pattern is spending the maximum time in Servlets. If one/more transactions of a pattern are found to be slow, then, you can compare the value of this measure with the other response time values reported by this test to determine where the slowdown actually occurred - in the filters, in JSPs, in servlets, in struts, in exception handling, when executing

Measurement	Description	Measurement Unit	Interpretation
			JDBC/SQL queries, when sending Java mails, or when accessing POJOs.
Generic servlets accessed	Indicates the number of generic (non-HTTP) servlets that were accessed by the transactions of this pattern during the last measurement period.	Number	
Generic servlets response time	Indicates the average time taken by the generic (non-HTTP) servlets for processing transactions of this pattern.	Secs	Badly written servlets can take too long to execute, and can hence obstruct the smooth execution of the dependent transactions. By comparing the value of this measure across patterns, you can figure out which transaction pattern is spending the maximum time in Servlets. If one/more transactions of a pattern are found to be slow, then, you can compare the value of this measure with the other response time values reported by this test to determine where the slowdown actually occurred - in the filters, in JSPs, in servlets, in struts, in exception handling, when executing JDBC/SQL queries, when sending Java mails, or when accessing POJOs.
JDBC queries	Indicates the number of JDBC statements that were executed by the transactions of this pattern during the last measurement period.	Number	The methods captured by the eG JTM Monitor from the Java class for the JDBC sub-component include: Commit (), rollback(..), close(),GetResultSet(), executeBatch(), cancel(), connect (String, Properties), getConnection (..),getPooledConnection(..)
JDBC response time	Indicates the average time taken by the transactions of this	Secs	By comparing the value of this measure across patterns, you can figure out which transaction pattern is taking the

Measurement	Description	Measurement Unit	Interpretation
	pattern to execute JDBC statements.		most time to execute JDBC queries. If one/more transactions of a pattern are found to be slow, then, you can compare the value of this measure with the other response time values reported by this test to determine where the slowdown actually occurred - in the filters, in JSPs, in servlets, in struts, in exception handling, when executing JDBC/SQL queries, when sending Java mails, or when accessing POJOs.
SQL statements executed	Indicates the number of SQL queries executed by the transactions of this pattern during the last measurement period.	Number	
SQL statement time avg.	Indicates the average time taken by the transactions of this pattern to execute SQL queries.	Secs	Inefficient queries can take too long to execute on the database, thereby significantly delaying the responsiveness of the dependent transactions. To know which transactions have been most impacted by such queries, compare the value of this measure across the transaction patterns. If one/more transactions of a pattern are found to be slow, then, you can compare the value of this measure with the other response time values reported by this test to determine where the slowdown actually occurred - at the filters layer, JSPs layer, servlets layer, struts layer, in exception handling, when executing JDBC/SQL queries, when sending Java mails, or when accessing POJOs.
Exceptions seen	Indicates the number of	Number	Ideally, the value of this measure should

Measurement	Description	Measurement Unit	Interpretation
	exceptions encountered by the transactions of this pattern during the last measurement period.		be 0.
Exceptions response time	Indicates the average time which the transactions of this pattern spent in handling exceptions.	Secs	If one/more transactions of a pattern are found to be slow, then, you can compare the value of this measure with the other response time values reported by this test to determine where the slowdown actually occurred - at the filters layer, JSPs layer, servlets layer, struts layer, in exception handling, when executing JDBC/SQL queries, when sending Java mails, or when accessing POJOs.
Struts accessed	Indicates the number of struts accessed by the transactions of this pattern during the last meaurement period.	Number	The Struts framework is a standard for developing well-architected Web applications.
Struts response time	Indicates the average time spent by the transactions of this pattern at the Struts layer.	Secs	If you compare the value of this measure across patterns, you can figure out which transaction pattern spent the maximum time in Struts. If one/more transactions of a pattern are found to be slow, then, you can compare the value of this measure with the other response time values reported by this test to determine where the slowdown actually occurred - in the filters, in JSPs, in servlets, in struts, in exception handling, when executing JDBC/SQL queries, when sending Java mails, or when accessing POJOs.
Java mails	Indicates the number of mails sent by the transactions of this	Number	The eG JTM Monitor captures any mail that has been sent from the monitored application using Java Mail API. Mails

Measurement	Description	Measurement Unit	Interpretation
	pattern during the last measurement period, using the Java mail API.		sent using other APIs are ignored by the eG JTM Monitor.
Java mail API time	Indicates the average time taken by the transactions of this pattern to send mails using the Java mail API.	Secs	If one/more transactions of a pattern are found to be slow, then, you can compare the value of this measure with the other response time values reported by this test to determine where the slowdown actually occurred - in the filters, in JSPs, in servlets, in struts, in exception handling, when executing JDBC/SQL queries, when sending Java mails, or when accessing POJOs.
POJOs	Indicates the number of transactions of this pattern that accessed POJOs during the last measurement period.	Number	Plain Old Java Object (POJO) refers to a 'generic' method in JAVA Language. All methods that are not covered by any of the Java components (eg., JSPs, Struts, Servlets, Filters, Exceptions, Queries, etc.) discussed above will be automatically included under POJO. When reporting the number of POJO methods, the eG agent will consider only those methods with a response time value that is higher than the threshold limit configured against the method exec cutoff parameter.
POJO avg. access time	Indicates the average time taken by the transactions of this pattern to access POJOs.	Secs	If one/more transactions of a pattern are found to be slow, then, you can compare the value of this measure with the other response time values reported by this test to determine where the slowdown actually occurred - in the filters, in JSPs, in servlets, in struts, in exception handling, when executing JDBC/SQL queries, when sending Java mails, or when accessing POJOs.

The detailed diagnosis of the *Slow transactions* measure lists the top-10 (by default) transactions of a configured pattern that have violated the response time threshold set using the slow url threshold parameter of this test. Against each transaction, the date/time at which the transaction was initiated/requested will be displayed. Besides the request date/time, the remote host from which the transaction request was received and the total response time of the transaction will also be reported. This response time is the sum total of the response times of each of the top methods (in terms of time taken for execution) invoked by that transaction. To compute this sum total, the test considers only those methods with a response time value that is higher than the threshold limit configured against the method exec cutoff parameter.

In the detailed diagnosis, the transactions will typically be arranged in the descending order of the total response time; this way, you would be able to easily spot the slowest transaction. To know what caused the transaction to be slow, you can take a look at the subcomponent details column of the detailed diagnosis. Here, the time spent by the transaction in each of the Java components (FILTER, STRUTS, servlets, jsps, pojos, sql, jdbc, etc.) will be listed, thus leading you to the exact Java component where the slowdown occurred.

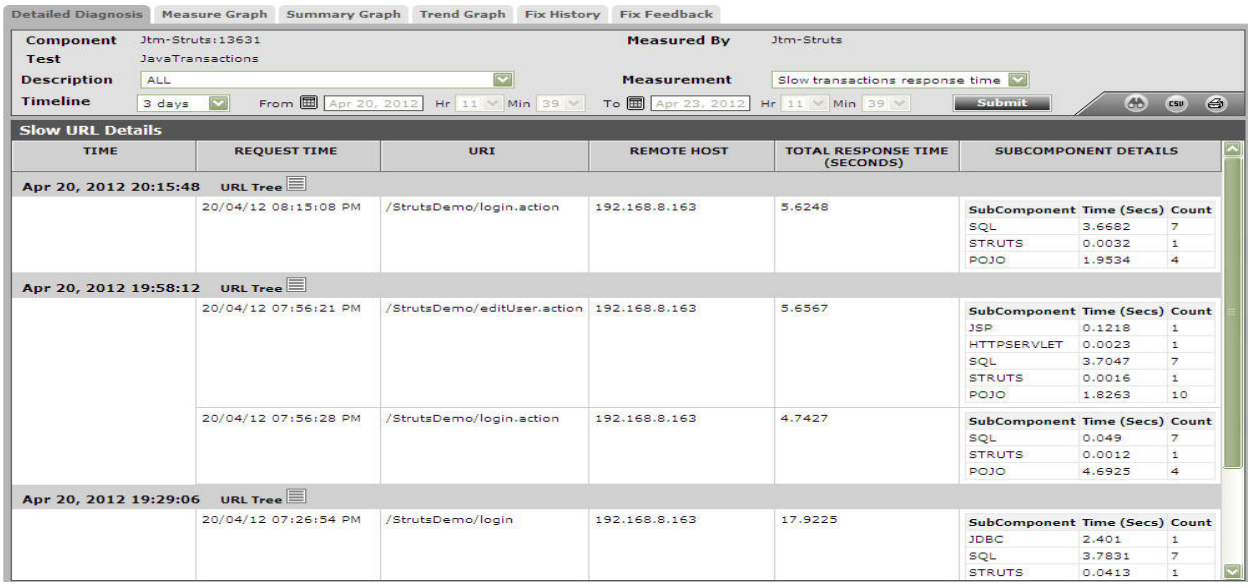


Figure 3.5: The detailed diagnosis of the Slow transactions measure

You can even perform detailed method-level analysis to isolate the methods taking too long to execute. For this, click on the **URL Tree** link. Figure 3.6 will then appear. In the left panel of Figure 3.6, you will find the list of transactions that match a configured pattern; these transactions will be sorted in the descending order of their *Total Response Time* (by default). This is indicated by the **Total Response Time** option chosen by default from the **Sort by** list in Figure 3.6. If you select a transaction from the left panel, an **At-A-Glance** tab page will open by default in the right panel, providing quick, yet deep insights into the performance of the chosen transaction and the reasons

for its slowness. This tab page begins by displaying the **URL** of the chosen transaction, the total **Response time** of the transaction, the time at which the transaction was last requested, and the **Remote Host** from which the request was received.

If the **Response time** appears to be very high, then you can take a look at the **Method Level Breakup** section to figure out which method called by which Java component (such as **FILTER, STRUTS, SERVLETS, JSPS, POJOS, SQL, JDBC,** etc.) could have caused the slowdown. This section provides a horizontal bar graph, which reveals the percentage of time for which the chosen transaction spent executing each of the top methods (in terms of execution time) invoked by it. The legend below clearly indicates the top methods and the layer/sub-component that invoked each method. Against every method, the number of times that method was invoked in the **Measurement Time**, the **Duration** (in Secs) for which the method executed, and the percentage of the total execution time of the transaction for which the method was in execution will be displayed, thus quickly pointing you to those methods that may have contributed to the slowdown. The methods displayed here and featured in the bar graph depend upon the **METHOD EXEC CUTOFF** configuration of this test - in other words, only those methods with an execution duration that exceeds the threshold limit configured against **METHOD EXEC CUTOFF** will be displayed in the **Method Level Breakup** section.

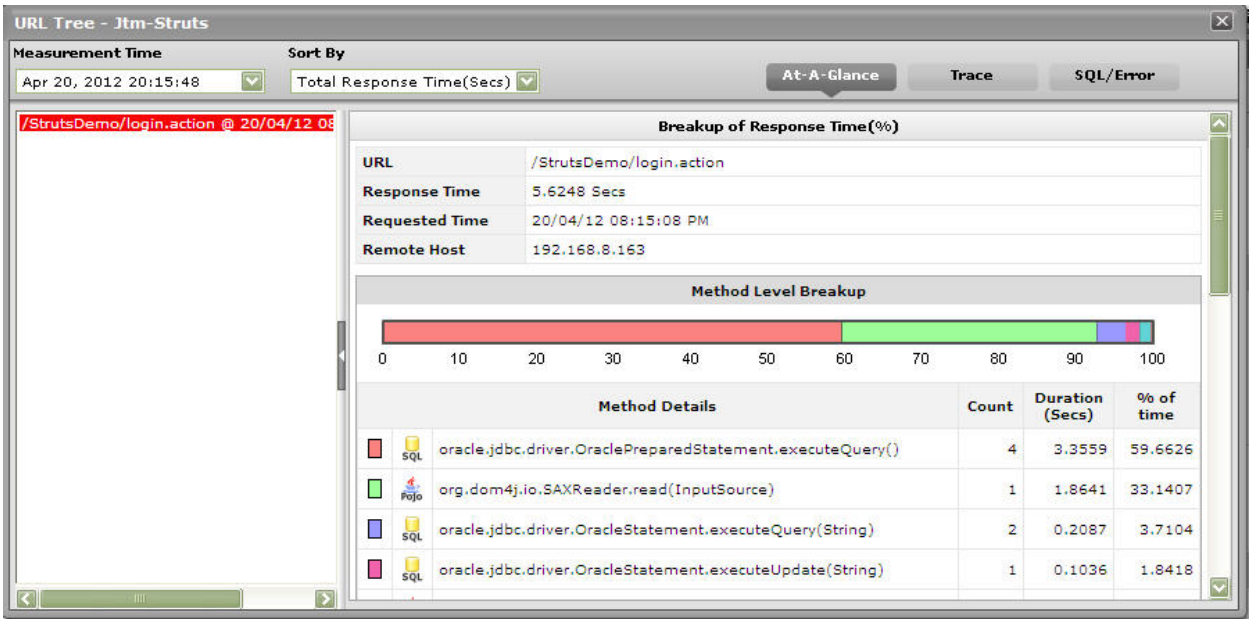


Figure 3.6: The Method Level Breakup section in the At-A-Glance tab page

While the **Method Level Breakup** section provides method-level insights into responsiveness, for a sub-component or layer level breakup of responsiveness scroll down the **At-A-Glance** tab to view the **Component Level Breakup** section (seeFigure 3.7). Using this horizontal bar graph, you can quickly tell where - i.e., in which Java component - the transaction spent the maximum time. A quick

glance at the graph's legend will reveal the Java components the transaction visited, the number of methods invoked by Java component, the **Duration (Secs)** for which the transaction was processed by the Java component, and what **Percentage** of the total transaction response time was spent in the Java component.

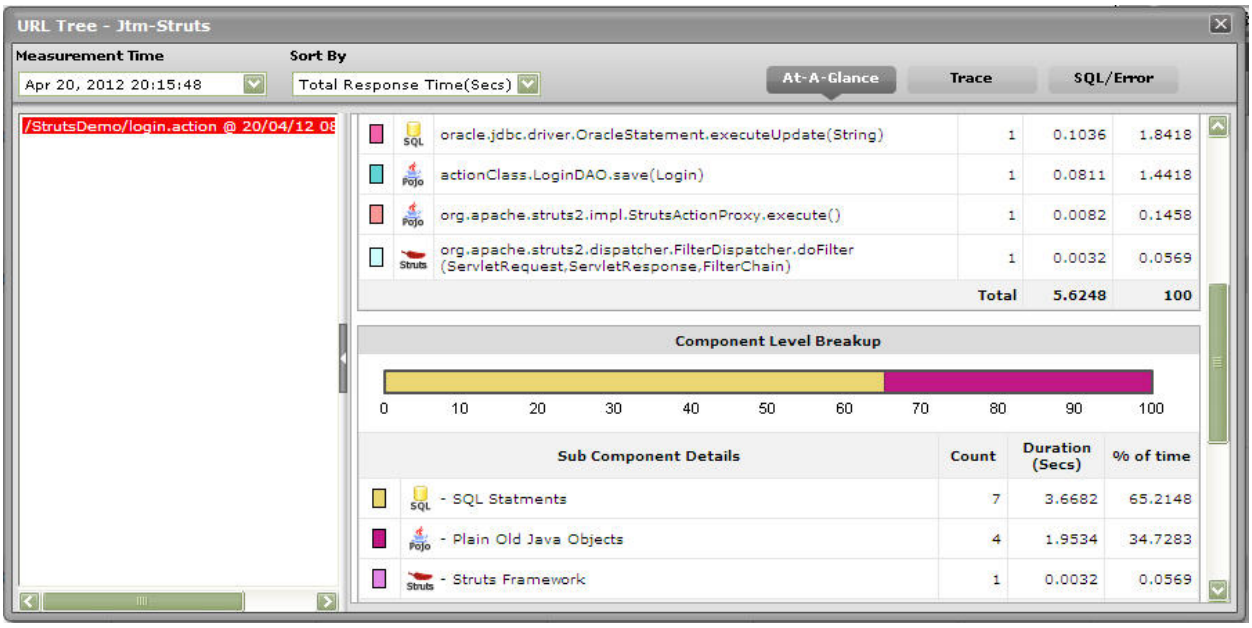


Figure 3.7: The Component Level Breakup section in the At-A-Glance tab page

Besides Java methods, where the target Java application interacts with the database, long-running SQL queries can also contribute to the poor responsiveness of a transaction. You can use the **At-A-Glance** tab page to determine whether the transaction interacts with the database or not, and if so, how healthy that interaction is. For this, scroll down the **At-A-Glance** tab page.

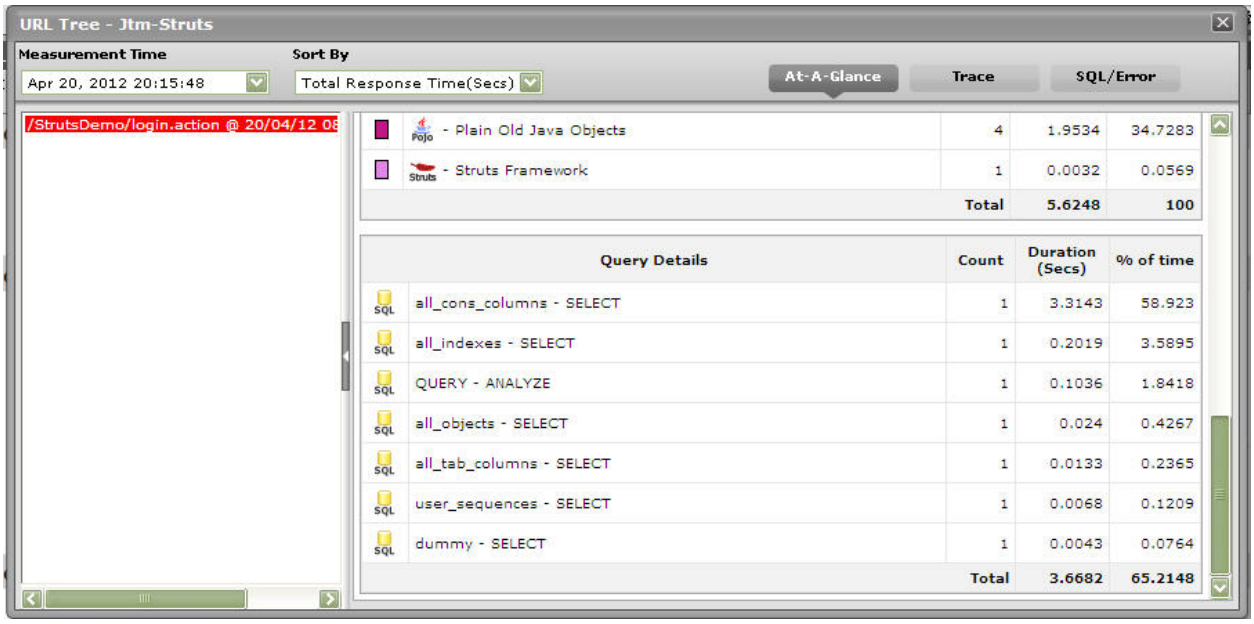


Figure 3.8: Query Details in the At-A-Glance tab page

Upon scrolling, you will find query details below the **Component Level Breakup** section. All the SQL queries that the chosen transaction executes on the backend database will be listed here in the descending order of their **Duration**. Corresponding to each query, you will be able to view the number of times that query was executed, the **Duration** for which it executed, and what percentage of the total transaction response time was spent in executing that query. A quick look at this tabulation would suffice to identify the query which executed for an abnormally long time on the database, causing the transaction's responsiveness to suffer. For a detailed query description, click on the query. Figure 3.9 will then pop up displaying the complete query and its execution duration.

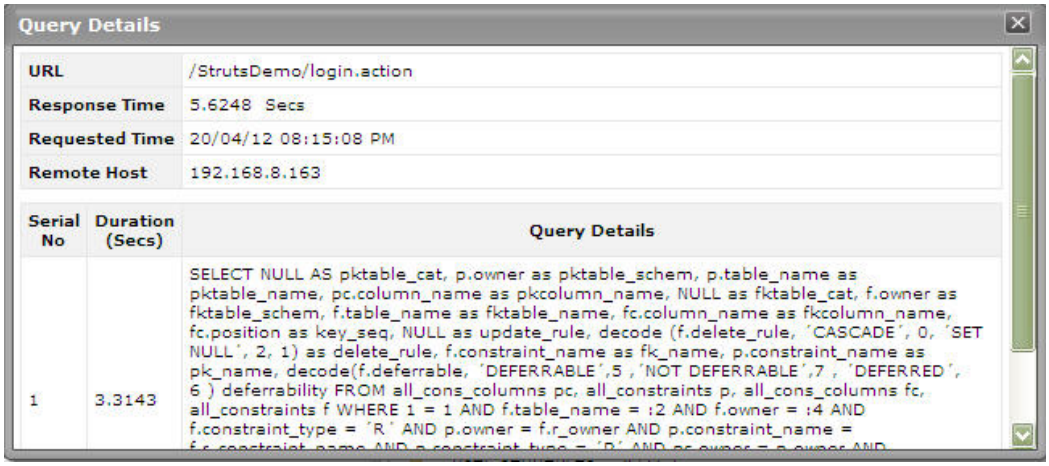


Figure 3.9: Detailed description of the query clicked on

This way, the **At-A-Glance** tab page allows you to analyze, at-a-glance, all the factors that can influence transaction response time - be it Java methods, Java components, and SQL queries - and enables you to quickly diagnose the source of a transaction slowdown. If, for instance, you figure out that a particular Java method is responsible for the slowdown, you can zoom into the performance of the 'suspect method' by clicking on that method in the **Method Level Breakup** section of the **At-A-Glance** tab page. This will automatically lead you to the **Trace** tab page, where all invocations of the chosen method will be highlighted (see Figure 3.10).

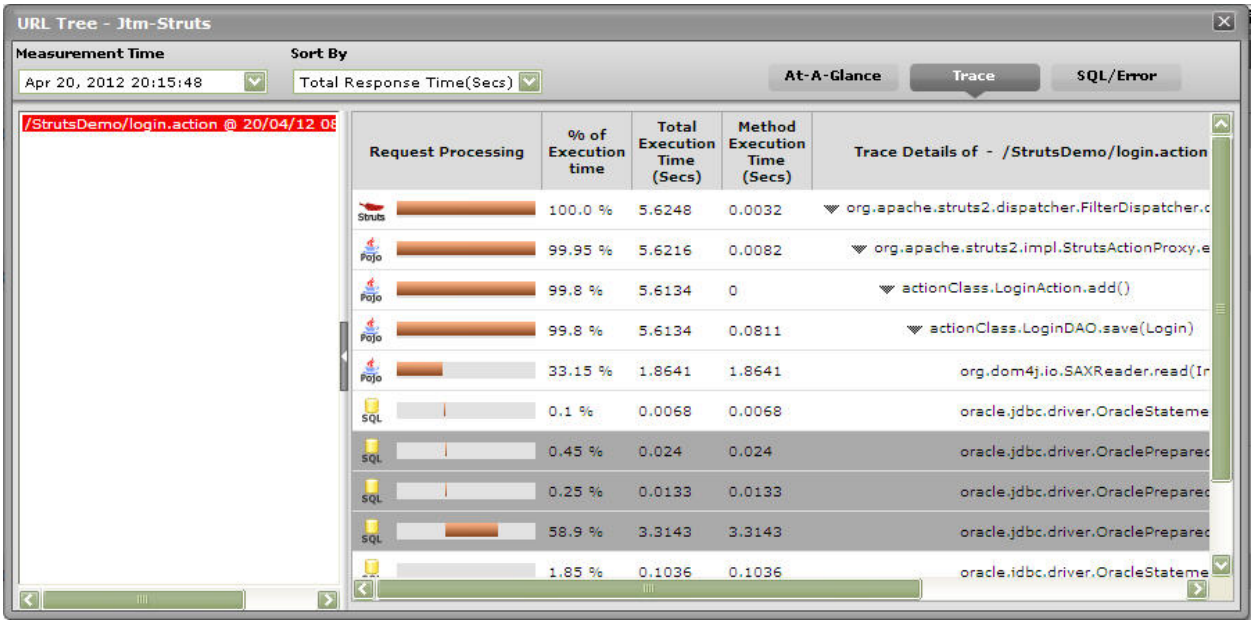


Figure 3.10: The Trace tab page displaying all invocations of the method chosen from the Method Level Breakup section

Typically, clicking on the **Trace** tab page will list all the methods invoked by the chosen transaction, starting with the very first method. Methods and sub-methods (a method invoked within a method) are arranged in a tree-structure, which can be expanded or collapsed at will. To view the sub-methods within a method, click on the **arrow icon** that precedes that method in the **Trace** tab page. Likewise, to collapse a tree, click once again on the **arrow icon**. Using the tree-structure, you can easily trace the sequence in which methods are invoked by a transaction.

If a method is chosen for analysis from the **Method Level Breakup** section of the **At-A-Glance** tab page, the **Trace** tab page will automatically bring your attention to all invocations of that method by highlighting them (as shown by Figure 3.10). Likewise, if a Java component is clicked in the **Component Level Breakup** section of the **At-A-Glance** section, the **Trace** tab page will automatically appear, displaying all the methods invoked from the chosen Java component (as shown by Figure 3.11).

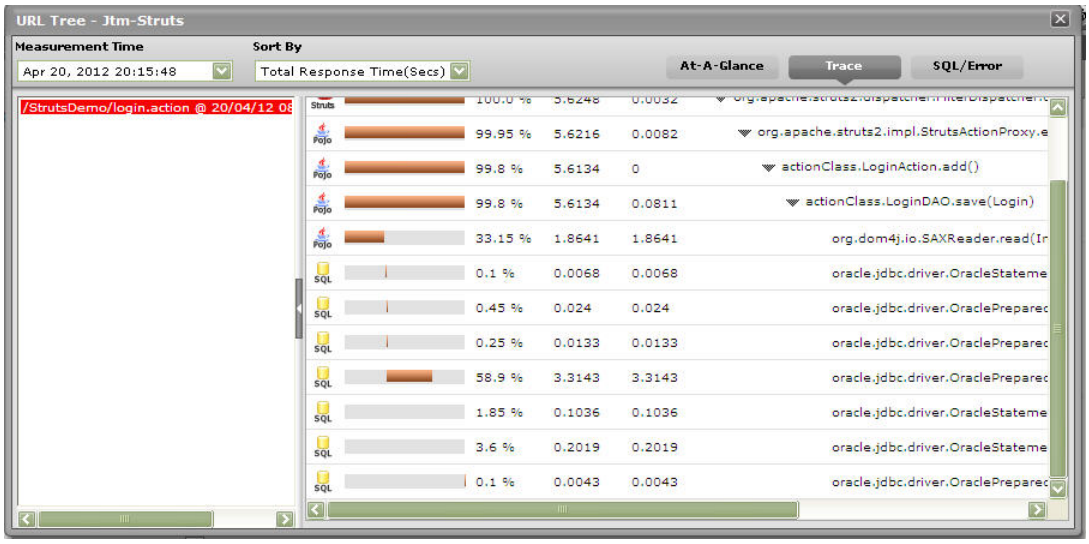


Figure 3.11: The Trace tab page displaying all methods invoked at the Java layer/sub-component chosen from the Component Level Breakup section

For every method, the **Trace** tab page displays a **Request Processing** bar, which will accurately indicate when, in the sequence of method invocations, the said method began execution and when it ended; with the help of this progress bar, you will be able to fairly judge the duration of the method, and also quickly tell whether any methods were called prior to the method in question. In addition, the **Trace** tab page will also display the time taken for a method to execute (**Method Execution Time**) and the percentage of the time the transaction spent in executing that method. The most time-consuming methods can thus be instantly isolated.

The **Trace** tab page also displays the **Total Execution Time** for each method - this value will be the same as the **Method Execution Time** for 'stand-alone' methods - i.e., methods without any sub-methods. In the case of methods with sub-methods however, the **Total Execution Time** will be the sum total of the **Method Execution Time** of each sub-method invoked within. This is because, a 'parent' method completes execution only when all its child/sub-methods finish executing.

With the help of the **Trace** tab page therefore, you can accurately trace the method that takes the longest to execute, when that method began execution, and which 'parent method' (if any) invoked the method.

Next, click on the **SQL/Errors** tab page. This tab page lists all the SQL queries the transaction executes on its backend database, and/or all the errors detected in the transaction's Java code. The query list (see Figure 3.12) is typically arranged in the descending order of the query execution **Duration**, and thus leads you to the long-running queries right away! You can even scrutinize the time-consuming query on-the-fly, and suggest improvements to your administrator instantly.

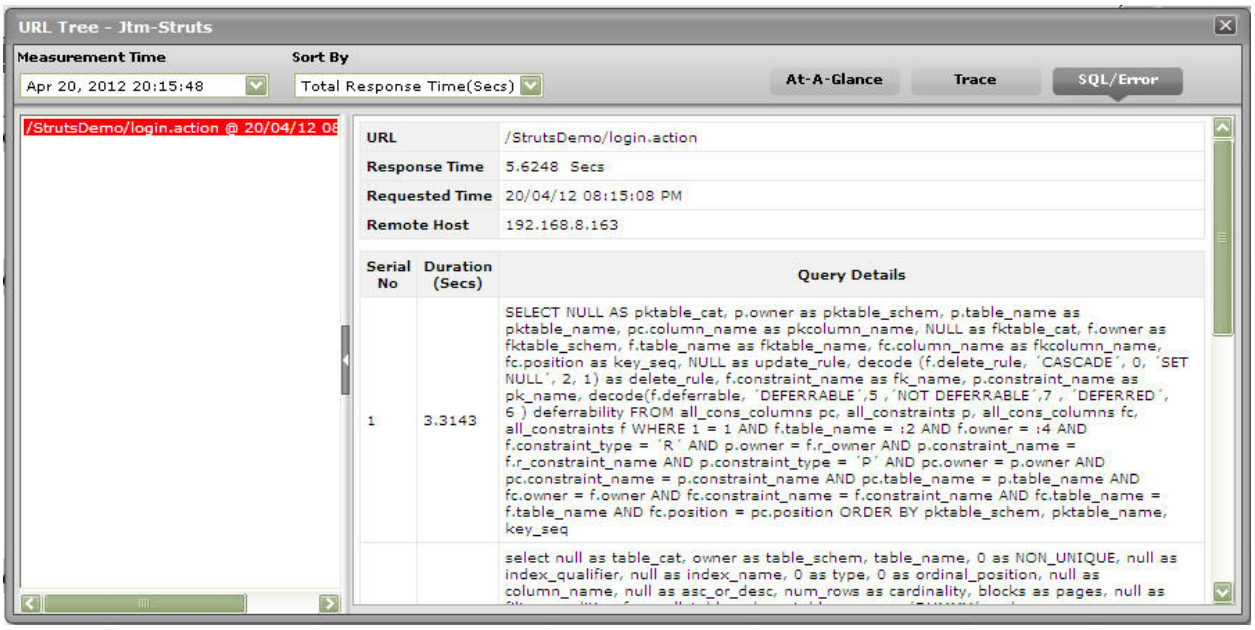


Figure 3.12: Queries displayed in the SQL/Error tab page

When displaying errors, the **SQL/Error** tab page does not display the error message alone, but displays the complete code block that could have caused the error to occur. By carefully scrutinizing the block, you can easily zero-in on the 'exact line of code' that could have forced the error - this means that besides pointing you to bugs in your code, the **SQL/Error** tab page also helps you initiate measures to fix the same.

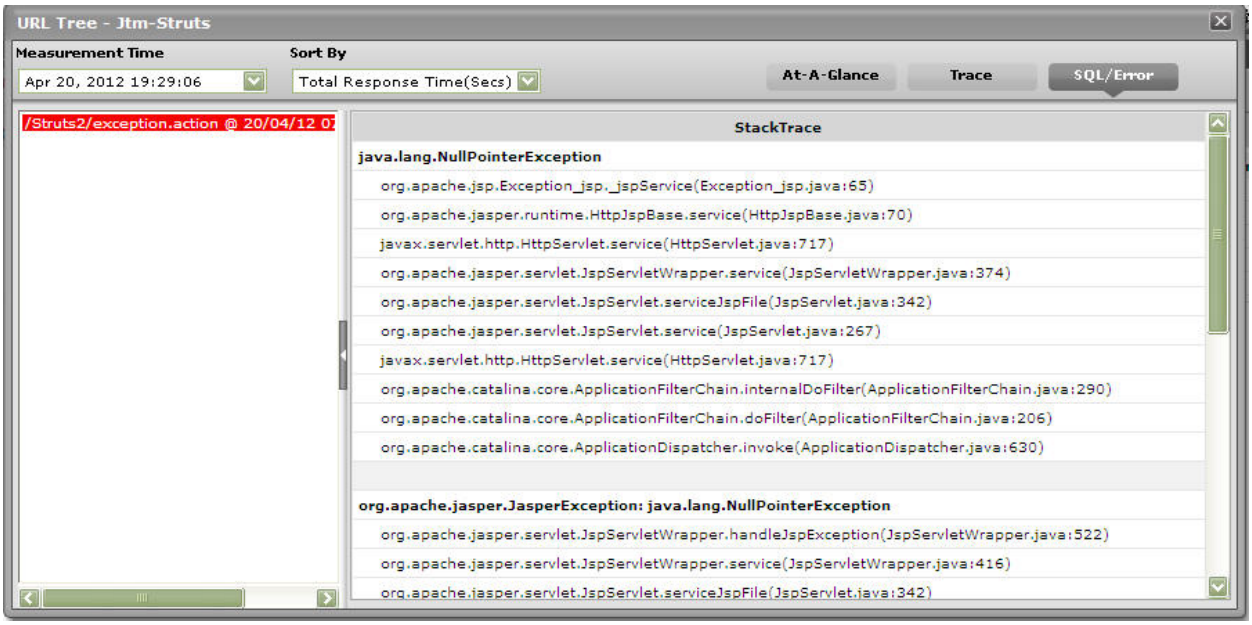


Figure 3.13: Errors displayed in the SQL/Error tab page

This way, with the help of the three tab pages - **At-A-Glance, Trace**, and **SQL/Error** - you can effectively analyze and accurately diagnose the root-cause of slowdowns in transactions to your Java applications.

The detailed diagnosis of the *Error transactions* measure reveals the top-10 (by default) transactions, in terms of **TOTAL RESPONSE TIME**, that have encountered errors. To know the nature of the errors that occurred, click on the **URL Tree** icon in Figure 3.14. This will lead you to the **URL Tree** window, which has already been elaborately discussed.

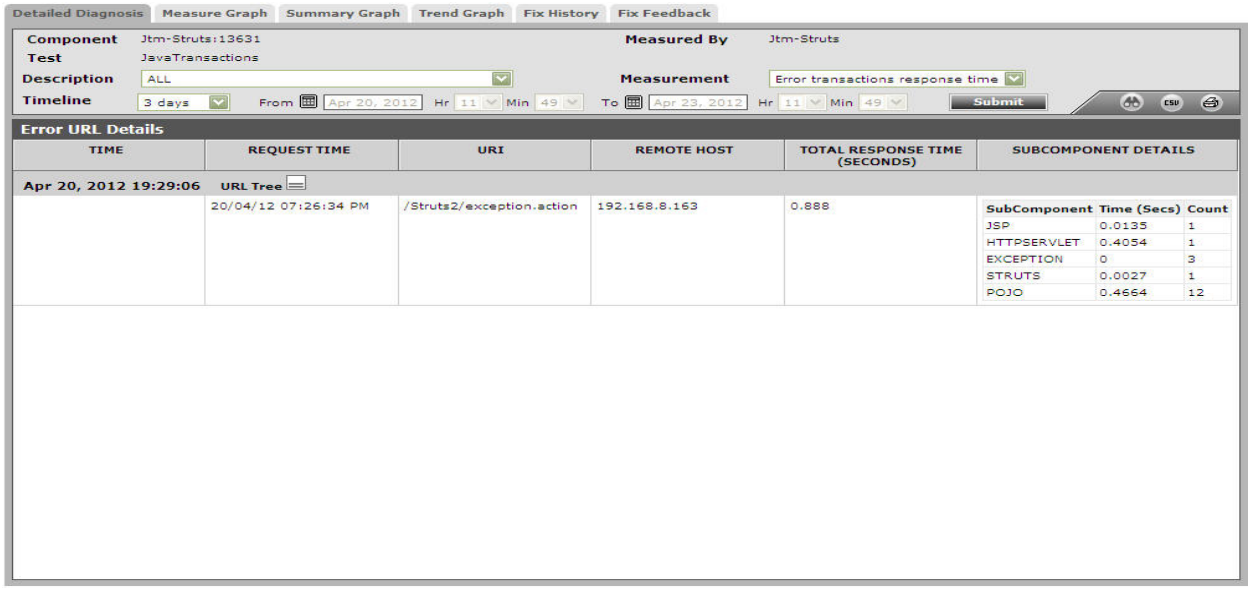


Figure 3.14: The detailed diagnosis of the Error transactions measure

3.3.2 JVM GC Test

Manual memory management is time consuming, and error prone. Most programs still contain leaks. This is all doubly true with programs using exception-handling and/or threads. Garbage collection (GC) is a part of WebLogic's JVM that automatically determines what memory a program is no longer using, and recycles it for other use. It is also known as "automatic storage (or memory) reclamation". The JVMGCTest reports the performance statistics pertaining to the JVM's garbage collection.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every GC.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The IP address of the WebLogic server.
Port	The port number of the WebLogic server.
JRE Home	The path to the directory in which the JVM to be monitored exists.
LogFileName	The full path to the log file which stores the GC output.
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
Number of GC	The number of times garbage collection has happened.	Number	If adequate memory is not allotted to the JVM, then the value of this measure would be very high. A high value of this measure is indicative of a high frequency of GC. This is not a good sign, as GC, during its execution, has the tendency of suspending an application, and a high frequency of GC would only adversely impact the application's performance. To avoid this, it is recommended that you allot sufficient memory to the JVM.

Measurement	Description	Measurement Unit	Interpretation
			The detailed diagnosis of the Number of GC measure, if enabled, provides details such as the heap size before GC was performed, the heap size after GC was performed, and the total time spent by the JVM in garbage collection. The difference between the heap sizes will help administrators figure out how effective GC is. The total GC time will help administrators gauge how severely GC has impacted application performance
Total GC time	The sum of the time taken by all garbage collections.	Secs	If adequate memory is not allotted to the JVM, then the value of this measure would be very high. This is not a good sign, as GC, during its execution, has the tendency of suspending an application, and a high value of this measure would only adversely impact the application's performance. To avoid this, it is recommended that you allot sufficient memory to the JVM.
Avg GC frequency	The frequency with which the JVM performed GC.	Sec	If adequate memory is not allotted to the JVM, then the value of this measure would be very low. A low value of this measure is indicative of a high frequency of GC. This is not a good sign, as GC, during its execution, has the tendency of suspending an application, and a high frequency of GC would only adversely impact the application's performance. To avoid this, it is recommended that you allot sufficient memory to the JVM.
Avg GC pause	The average time the application is suspended	Secs	If the garbage collection takes more time to complete, then it indicates a

Measurement	Description	Measurement Unit	Interpretation
	while garbage collection is in progress.		very high memory allocation to the JVM. This again hinders application performance.
Avg GC overhead	The percentage of time utilized by the JVM for garbage collection	Percent	By carefully examining the application behavior in terms of memory utilization, you should arrive at an optimal ratio of the number of times the GC needs to run and how long it should take to complete. Accordingly, memory allocation to the JVM can be performed.
Max GC pause	The maximum time spent by the JVM on garbage collection, during the last measurement period	Secs	
Avg heap before GC	The average heap size prior to garbage collection	KB	
Avg heap after GC	The average heap size after garbage collection	KB	The difference between the value of this measure and the Avg heap before GC measure provides the amount of memory that has been released by GC. This value is a good indicator of the effectiveness of GC.

3.3.3 Java Classes Test

This test reports the number of classes loaded/unloaded from the memory.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal/remote agent

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

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
Mode	This test can extract metrics from the Java application using either of the following mechanisms: • Using SNMP-based access to the Java runtime MIB statistics; • By contacting the Java runtime (JRE) of the application via JMX To configure the test to use SNMP, select the **SNMP** option. On the other hand, choose the JMX option to configure the test to use JMX instead. By default, the **JMX** option is chosen here.
JMX Port	This parameter appears only if the mode is set to **JMX**. Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
JNDIName	This parameter appears only if the Mode is set to JMX. The JNDIName is a lookup name for connecting to the JMX connector. By default, this is JMXRMI. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
User, Password, and Confirm Password	These parameters appear only if the Mode is set to **JMX**. If JMX requires **authentication only** (but no security), then ensure that the User and Password parameters are configured with the credentials of a user with *read-write* access to JMX. To know how to create this user, refer to the *Monitoring Java Applications* document. Confirm the password by retyping it in the Confirm Password text box.
SNMPPort	This parameter appears only if the Mode is set to **SNMP**. Here specify the port number through which the server exposes its SNMP MIB. Ensure that you specify the same port you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
SNMPVersion	This parameter appears only if the Mode is set to **SNMP**. By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the SNMPversion list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP

Parameter	Description
	v2 or **v3**, then select the corresponding option from this list.
SNMPCommunity	This parameter appears only if the Mode is set to **SNMP**. The SNMP community name that the test uses to communicate with the firewall. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the SNMPVersion chosen is **v3**, then this parameter will not appear.
UserName	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only when **v3** is selected as the SNMPVersion. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against this parameter.
Context	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only when v3 is selected as the SNMPVersion. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the SNMPEngineID value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a contextName). If the Username provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the Username in the Context text box. By default, this parameter is set to *none*.
AuthPass	Specify the password that corresponds to the above-mentioned Username. This parameter once again appears only if the SNMPversion selected is **v3**.
Confirm Password	Confirm the AuthPass by retyping it here.
AuthType	This parameter too appears only if **v3** is selected as the SNMPVersion. From the Authtype list box, choose the authentication algorithm using which SNMP v3 converts the specified username and password into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm
EncryptFlag	This flag appears only when **v3** is selected as the SNMPVersion. By default, the eG

Parameter	Description
	agent does not encrypt SNMP requests. Accordingly, the this flag is set to **No** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **Yes** option.
EncryptType	If this EncryptFlag is set to **Yes**, then you will have to mention the encryption type by selecting an option from the EncryptType list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard
EncryptPassword	Specify the encryption password here.
Confirm Password	Confirm the encryption password by retyping it here.
Timeout	This parameter appears only if the Mode is set to **SNMP**. Specify the duration (in seconds) within which the SNMP query executed by this test should time out in this text box. The default is 10 seconds.
Data Over TCP	This parameter appears only if the Mode is set to **SNMP**. By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic – for instance, certain types of data traffic or traffic pertaining to specific components – to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the monitored target over TCP (and not UDP). For this, set this flag to **Yes**. By default, this flag is set to **No**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Classes loaded	Indicates the number of classes currently loaded into memory.	Number	Classes are fundamental to the design of Java programming language. Typically, Java applications install a variety of class loaders (that is, classes that implement java.lang.ClassLoader) to allow different portions of the container, and the applications running on the container, to have access to different repositories of available classes and resources. A consistent

Measurement	Description	Measurement Unit	Interpretation
			decrease in the number of classes loaded and unloaded could indicate a road-block in the loading/unloading of classes by the class loader. If left unchecked, critical resources/classes could be rendered inaccessible to the application, thereby severely affecting its performance.
Classes unloaded	Indicates the number of classes currently unloaded from memory.	Number	
Total classes loaded	Indicates the total number of classes loaded into memory since the JVM started, including those subsequently unloaded.	Number	

3.3.4 JVM Threads Test

This test reports the status of threads on the JVM, and also reveals resource-hungry threads, so that threads that are unnecessarily consuming CPU resources can be killed.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal/remote agent

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

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
Mode	This test can extract metrics from the Java application using either of the following mechanisms: • Using SNMP-based access to the Java runtime MIB statistics; • By contacting the Java runtime (JRE) of the application via JMX

Parameter	Description
	To configure the test to use SNMP, select the **SNMP** option. On the other hand, choose the JMX option to configure the test to use JMX instead. By default, the **JMX** option is chosen here.
JMX Port	This parameter appears only if the mode is set to **JMX**. Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
JNDIName	This parameter appears only if the Mode is set to JMX. The JNDIName is a lookup name for connecting to the JMX connector. By default, this is JMXRMI. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
User, Password, and Confirm Password	These parameters appear only if the Mode is set to **JMX**. If JMX requires **authentication only** (but no security), then ensure that the User and Password parameters are configured with the credentials of a user with *read-write* access to JMX. To know how to create this user, refer to the *Monitoring Java Applications* document. Confirm the password by retyping it in the Confirm Password text box.
SNMPPort	This parameter appears only if the Mode is set to **SNMP**. Here specify the port number through which the server exposes its SNMP MIB. Ensure that you specify the same port you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
SNMPVersion	This parameter appears only if the Mode is set to **SNMP**. By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the SNMPversion list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list.
SNMPCommunity	This parameter appears only if the Mode is set to **SNMP**. The SNMP community name that the test uses to communicate with the firewall. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the SNMPVersion chosen is **v3**, then this parameter will not appear.
UserName	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only when **v3** is selected as the SNMPVersion. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access

Parameter	Description
	privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against this parameter.
Context	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only when v3 is selected as the SNMPVersion. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the SNMPEngineID value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a contextName). If the Username provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the Username in the Context text box. By default, this parameter is set to *none*.
AuthPass	Specify the password that corresponds to the above-mentioned Username. This parameter once again appears only if the SNMPversion selected is **v3**.
Confirm Password	Confirm the AuthPass by retyping it here.
AuthType	This parameter too appears only if **v3** is selected as the SNMPVersion. From the Authtype list box, choose the authentication algorithm using which SNMP v3 converts the specified username and password into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm
EncryptFlag	This flag appears only when **v3** is selected as the SNMPVersion. By default, the eG agent does not encrypt SNMP requests. Accordingly, the this flag is set to **No** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **Yes** option.
EncryptType	If this EncryptFlag is set to **Yes**, then you will have to mention the encryption type by selecting an option from the EncryptType list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard
EncryptPassword	Specify the encryption password here.

Parameter	Description
Confirm Password	Confirm the encryption password by retyping it here.
Timeout	This parameter appears only if the Mode is set to **SNMP**. Specify the duration (in seconds) within which the SNMP query executed by this test should time out in this text box. The default is 10 seconds.
Data Over TCP	This parameter appears only if the Mode is set to **SNMP**. By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic – for instance, certain types of data traffic or traffic pertaining to specific components – to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the monitored target over TCP (and not UDP). For this, set this flag to **Yes**. By default, this flag is set to **No**.
Pct Low CPU Util Threads	This test reports the number of threads in the JVM that are consuming low CPU. This thread count will include only those threads for which the CPU usage is equal to or lesser than the value specified in the Pct Low CPU Util Threads text box. The default value displayed here is 30.
Pct Medium CPU Util Threads	This test reports the number of threads in the JVM that are consuming CPU to a medium extent. This thread count will include only those threads for which the CPU usage is higher than the Pct Low CPU Util Threads configuration and is lower than or equal to the value specified in the Pct Medium CPU Util Threads text box. The default value displayed here is 50.
Pct High CPU Util Threads	This test reports the number of threads in the JVM that are consuming high CPU. This thread count will include only those threads for which the CPU usage is either greater than the Pct Medium CPU Util Threads configuration, or is equal to or greater than the value specified in the Pct High CPU Util Threads text box. The default value displayed here is 70.
DD Frequency	Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD Frequency.
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

Parameter	Description
	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
Total threads	Indicates the total number of threads (including daemon and non-daemon threads).	Number	
Runnable threads	Indicates the current number of threads in a runnable state.	Number	The detailed diagnosis of this measure, if enabled, provides the name of the threads, the CPU usage by the threads, the time for which the thread was in a blocked state, waiting state, etc.
Blocked threads	Indicates the number of threads that are currently in a blocked state.	Number	If a thread is trying to take a lock (to enter a synchronized block), but the lock is already held by another thread, then such a thread is called a blocked thread. The detailed diagnosis of this measure, if enabled, provides in-depth information related to the blocked threads.
Waiting threads	Indicates the number of threads that are currently in a waiting state.	Number	A thread is said to be in a Waiting state if the thread enters a synchronized block, tries to take a lock that is already held by another thread, and hence, waits till the other thread notifies that it has released the lock. Ideally, the value of this measure

Measurement	Description	Measurement Unit	Interpretation
			should be low. A very high value could be indicative of excessive waiting activity on the JVM. You can use the detailed diagnosis of this measure, if enabled, to figure out which threads are currently in the waiting state. While waiting, the Java application program does no productive work and its ability to complete the task-at-hand is degraded. A certain amount of waiting may be acceptable for Java application programs. However, when the amount of time spent waiting becomes excessive or if the number of times that waits occur exceeds a reasonable amount, the Java application program may not be programmed correctly to take advantage of the available resources. When this happens, the delay caused by the waiting Java application programs elongates the response time experienced by an end user. An enterprise may use Java application programs to perform various functions. Delays based on abnormal degradation consume employee time and may be costly to corporations.
Timed waiting threads	Indicates the number of threads in a TIMED_ WAITING state.	Number	When a thread is in the TIMED_ WAITING state, it implies that the thread is waiting for another thread to do something, but will give up after a specified time out period. To view the details of threads in the TIMED_WAITING state, use the detailed diagnosis of this measure, if enabled.
Low CPU threads	Indicates the number of	Number	

Measurement	Description	Measurement Unit	Interpretation
	threads that are currently consuming CPU lower than the value configured in the Pct Low CPU Util Threads text box.		
Medium CPU threads	Indicates the number of threads that are currently consuming CPU that is higher than the value configured in the Pct Low CPU Util Threads text box and is lower than or equal to the value specified in the Pct Medium CPU Util Threads text box.	Number	
High CPU threads	Indicates the number of threads that are currently consuming CPU that is either greater than the percentage configured in the Pct Medium CPU Util Threads or lesser than or equal to the value configured in the Pct High CPU Util Threads text box.	Number	Ideally, the value of this measure should be very low. A high value is indicative of a resource contention at the JVM. Under such circumstances, you might want to identify the resource-hungry threads and kill them, so that application performance is not hampered. To know which threads are consuming excessive CPU, use the detailed diagnosis of this measure.
Peak threads	Indicates the highest number of live threads since JVM started.	Number	
Started threads	Indicates the the total number of threads started (including daemon, non-daemon, and terminated) since JVM started.	Number	
Daemon threads	Indicates the current number of live daemon threads.	Number	

Measurement	Description	Measurement Unit	Interpretation
Deadlock threads	Indicates the current number of deadlocked threads.	Number	Ideally, this value should be 0. A high value is a cause for concern, as it indicates that many threads are blocking one another causing the application performance to suffer. The detailed diagnosis of this measure, if enabled, lists the deadlocked threads and their resource usage.

Note:

If the Mode for the **JVM Threads** test is set to **SNMP**, then the detailed diagnosis of this test will not display the **Blocked Time** and **Waited Time** for the threads. To make sure that detailed diagnosis reports these details also, do the following:

a. Login to the server host.

b. Go to the <JAVA_HOME>\jre\lib\management folder used by the WebLogic server, and edit the *management.properties* file in that folder.

c. Append the following line to the file:

```
com.sun.management.enableThreadContentionMonitoring
```

d. Finally, save the file.

Note:

While viewing the measures reported by the **JVM Thread** test, you can also view the resource usage details and the stack trace information for all the threads, by clicking on the **STACK TRACE** link in the **Measurements** panel.

A stack trace (also called stack backtrace or stack traceback) is a report of the active stack frames instantiated by the execution of a program. It is commonly used to determine what threads are currently active in the JVM, and which threads are in each of the different states – i.e., alive, blocked, waiting, timed waiting, etc.

Typically, when the JVM begins exhibiting erratic resource usage patterns, it often takes administrators hours, even days to figure out what is causing this anomaly – could it be owing to one/more resource-intensive threads being executed by the WebLogic server? If so, what is causing the thread to erode resources? Is it an inefficient piece of code? In which case, which line of code could be the most likely cause for the spike in resource usage? To be able to answer these questions accurately, administrators need to know the complete list of threads that are executing on the JVM,

view the stack trace of each thread, analyze each stack trace in a top-down manner, and trace where the problem originated.

3.3.5 JVM CPU Usage Test

This test measures the CPU utilization of the JVM. If the JVM experiences abnormal CPU usage levels, you can use this test to instantly drill down to the classes and the methods within the classes that contributing to the resource contention at the JVM.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal/remote agent

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

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
Mode	This test can extract metrics from the Java application using either of the following mechanisms: • Using SNMP-based access to the Java runtime MIB statistics; • By contacting the Java runtime (JRE) of the application via JMX To configure the test to use SNMP, select the **SNMP** option. On the other hand, choose the JMX option to configure the test to use JMX instead. By default, the **JMX** option is chosen here.
JMX Port	This parameter appears only if the mode is set to **JMX**. Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
JNDIName	This parameter appears only if the Mode is set to JMX. The JNDIName is a lookup name for connecting to the JMX connector. By default, this is JMXRMI. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.

Parameter	Description
User, Password, and Confirm Password	These parameters appear only if the Mode is set to **JMX**. If JMX requires **authentication only** (but no security), then ensure that the User and Password parameters are configured with the credentials of a user with *read-write* access to JMX. To know how to create this user, refer to the *Monitoring Java Applications* document. Confirm the password by retyping it in the Confirm Password text box.
SNMPPort	This parameter appears only if the Mode is set to **SNMP**. Here specify the port number through which the server exposes its SNMP MIB. Ensure that you specify the same port you configured in the *management.properties* file in the <JAVA_ HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
SNMPVersion	This parameter appears only if the Mode is set to **SNMP**. By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the SNMPversion list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list.
SNMPCommunity	This parameter appears only if the Mode is set to **SNMP**. The SNMP community name that the test uses to communicate with the firewall. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the SNMPVersion chosen is **v3**, then this parameter will not appear.
UserName	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only when **v3** is selected as the SNMPVersion. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against this parameter.
Context	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only when v3 is selected as the SNMPVersion. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the SNMPEngineID value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a contextName). If the Username provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the Username in the Context text

Parameter	Description
	box. By default, this parameter is set to *none*.
AuthPass	Specify the password that corresponds to the above-mentioned Username. This parameter once again appears only if the SNMPVersion selected is **v3**.
Confirm Password	Confirm the AuthPass by retyping it here.
AuthType	This parameter too appears only if **v3** is selected as the SNMPVersion. From the Authtype list box, choose the authentication algorithm using which SNMP v3 converts the specified username and password into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm
EncryptFlag	This flag appears only when **v3** is selected as the SNMPVersion. By default, the eG agent does not encrypt SNMP requests. Accordingly, the this flag is set to **No** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **Yes** option.
EncryptType	If this EncryptFlag is set to **Yes**, then you will have to mention the encryption type by selecting an option from the EncryptType list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard
EncryptPassword	Specify the encryption password here.
Confirm Password	Confirm the encryption password by retyping it here.
Timeout	This parameter appears only if the Mode is set to **SNMP**. Specify the duration (in seconds) within which the SNMP query executed by this test should time out in this text box. The default is 10 seconds.
Data Over TCP	This parameter appears only if the Mode is set to **SNMP**. By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic – for instance, certain types of data traffic or traffic pertaining to specific components – to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the monitored target over TCP (and not UDP). For this, set this flag to **Yes**. By default, this flag is set to **No**.

Parameter	Description
Profiler Home	JIP (Java Interactive Profiler) is a high performance, low overhead profiler that is written entirely in Java and used extensively to gather performance data pertaining to a JVM. The eG agent comes bundled with JIP, and takes the help of JIP to provide detailed diagnosis information related to the CPU usage of the JVM. To make sure that this test contacts JIP for detailed resource usage metrics, you need to indicate the location of the JIP in the Profiler Home text box. Typically, the files related to this profiler are available in <EG_INSTALL_DIR>\lib\jip directory (in Windows; in Unix, this will be: /opt/egurkha/lib/jip). If only a single Java application on a host is being monitored, then the *jip* folder can remain in the same location. The profiler home parameter should then be configured with /opt/egurkha/lib/jip or <EG_INSTALL_DIR>\lib\jip, as the case may be. However, if more than one Java application on a single host is to be monitored, then first ensure that the *jip* folder is copied to two different locations on the same host. The profiler home parameter should in this case be configured with the location of the *jip* folder that corresponds to the Java application for which this test is being currently configured. For instance, say japp1 and japp2 are 2 Java applications that are being managed by the eG Enterprise system. Assume that the *jip* folder has been copied to the *C:\japp1* and *D:\japp2* folders. Now, while configuring this test for the japp1 application, specify *C:\japp\jip* in profiler homE. Similarly, when configuring this test for the japp2 application, enter *D:\japp2\jip* in profiler home.
Profiler	The JIP can be turned on or off while the JVM is still running. For the eG agent to be able to report detailed diagnosis of the CPU usage metric, the Profiler should be turned on. Accordingly, the Profiler flag is set to **On** by default. If you do not want detailed diagnosis, then you can set the Profiler flag to **Off**.
DD Frequency	Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD Frequency.
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

Parameter	Description
	• 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
CPU Utilization of JVM	Indicates the percentage of CPU currently utilized by the JVM.	Percent	Ideally, this value should be low. An unusually high value or a consistent increase in this value is indicative of abnormal CPU usage, and could warrant further investigation. In such a situation, you can use the detailed diagnosis of this measure, if enabled, to determine which methods invoked by which classes are causing the steady/sporadic spikes in CPU usage.

3.3.6 JVM Memory Usage Test

This test monitors every memory type on the JVM and reports how efficiently the JVM utilizes the memory resources of each type.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for every memory type on the JVM being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
Mode	This test can extract metrics from the Java application using either of the following

Parameter	Description
	mechanisms: • Using SNMP-based access to the Java runtime MIB statistics; • By contacting the Java runtime (JRE) of the application via JMX To configure the test to use SNMP, select the **SNMP** option. On the other hand, choose the JMX option to configure the test to use JMX instead. By default, the **JMX** option is chosen here.
JMX Port	This parameter appears only if the mode is set to **JMX**. Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
JNDIName	This parameter appears only if the Mode is set to JMX. The JNDIName is a lookup name for connecting to the JMX connector. By default, this is JMXRMI. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
User, Password, and Confirm Password	These parameters appear only if the Mode is set to **JMX**. If JMX requires **authentication only** (but no security), then ensure that the User and Password parameters are configured with the credentials of a user with *read-write* access to JMX. To know how to create this user, refer to the *Monitoring Java Applications* document. Confirm the password by retyping it in the Confirm Password text box.
SNMPPort	This parameter appears only if the Mode is set to **SNMP**. Here specify the port number through which the server exposes its SNMP MIB. Ensure that you specify the same port you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
SNMPVersion	This parameter appears only if the Mode is set to **SNMP**. By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the SNMPversion list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list.
SNMPCommunity	This parameter appears only if the Mode is set to **SNMP**. The SNMP community name that the test uses to communicate with the firewall. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the SNMPVersion chosen is **v3**, then this parameter will not appear.
UserName	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only

Parameter	Description
	when **v3** is selected as the SNMPVersion. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against this parameter.
Context	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only when v3 is selected as the SNMPVersion. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the SNMPEngineID value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a contextName). If the Username provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the Username in the Context text box. By default, this parameter is set to *none*.
AuthPass	Specify the password that corresponds to the above-mentioned Username. This parameter once again appears only if the SNMPversion selected is **v3**.
Confirm Password	Confirm the AuthPass by retyping it here.
AuthType	This parameter too appears only if **v3** is selected as the SNMPVersion. From the Authtype list box, choose the authentication algorithm using which SNMP v3 converts the specified username and password into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm
EncryptFlag	This flag appears only when **v3** is selected as the SNMPVersion. By default, the eG agent does not encrypt SNMP requests. Accordingly, the this flag is set to **No** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **Yes** option.
EncryptType	If this EncryptFlag is set to **Yes**, then you will have to mention the encryption type by selecting an option from the EncryptType list. SNMP v3 supports the following encryption types:

Parameter	Description
	• **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard
EncryptPassword	Specify the encryption password here.
Confirm Password	Confirm the encryption password by retyping it here.
Timeout	This parameter appears only if the Mode is set to **SNMP**. Specify the duration (in seconds) within which the SNMP query executed by this test should time out in this text box. The default is 10 seconds.
Data Over TCP	This parameter appears only if the Mode is set to **SNMP**. By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic – for instance, certain types of data traffic or traffic pertaining to specific components – to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the monitored target over TCP (and not UDP). For this, set this flag to **Yes**. By default, this flag is set to **No**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Initial memory	Indicates the amount of memory initially allocated at startup.	MB	The detailed diagnosis of this measure, when enabled, reveals the memory pools on the JVM, the memory type of every pool, and the memory manager managing the pool.
Used memory	Indicates the amount of memory currently used.	MB	It includes the memory occupied by all objects including both reachable and unreachable objects.
Available memory	Indicates the amount of memory guaranteed to be available for use by the JVM.	MB	The amount of *Available memory* may change over time. The Java virtual machine may release memory to the system and committed memory could be less than the amount of memory initially allocated at startup. Committed will always be greater than or equal to used memory.

Measurement	Description	Measurement Unit	Interpretation
Free memory	Indicates the amount of memory currently available for use by the JVM.	MB	This is the difference between Available memory and Used memory. Ideally, the value of this measure should be high.
Max free memory	Indicates the maximum amount of memory allocated for the JVM.	MB	
Used percentage	Indicates the percentage of used memory.	Percent	Ideally, the value of this measure should be low. A very high value of this measure could indicate excessive memory consumption by the JVM, which in turn, could warrant further investigation.

3.3.7 JVM Uptime Test

This test helps track whether a scheduled reboot of the JVM has occurred or not.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal/remote agent

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

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
Mode	This test can extract metrics from the Java application using either of the following mechanisms: • Using SNMP-based access to the Java runtime MIB statistics; • By contacting the Java runtime (JRE) of the application via JMX

Parameter	Description
	To configure the test to use SNMP, select the **SNMP** option. On the other hand, choose the JMX option to configure the test to use JMX instead. By default, the **JMX** option is chosen here.
JMX Port	This parameter appears only if the mode is set to **JMX**. Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
JNDIName	This parameter appears only if the Mode is set to **JMX**. The JNDIName is a lookup name for connecting to the JMX connector. By default, this is JMXRMI. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
User, Password, and Confirm Password	These parameters appear only if the Mode is set to **JMX**. If JMX requires **authentication only** (but no security), then ensure that the User and Password parameters are configured with the credentials of a user with *read-write* access to JMX. To know how to create this user, refer to the *Monitoring Java Applications* document. Confirm the password by retyping it in the Confirm Password text box.
SNMPPort	This parameter appears only if the Mode is set to **SNMP**. Here specify the port number through which the server exposes its SNMP MIB. Ensure that you specify the same port you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
SNMPVersion	This parameter appears only if the Mode is set to **SNMP**. By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the SNMPversion list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list.
SNMPCommunity	This parameter appears only if the Mode is set to **SNMP**. The SNMP community name that the test uses to communicate with the firewall. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the SNMPVersion chosen is **v3**, then this parameter will not appear.
UserName	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only when **v3** is selected as the SNMPVersion. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access

Parameter	Description
	privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against this parameter.
Context	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only when v3 is selected as the SNMPVersion. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the SNMPEngineID value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a contextName). If the Username provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the Username in the Context text box. By default, this parameter is set to *none*.
AuthPass	Specify the password that corresponds to the above-mentioned Username. This parameter once again appears only if the SNMPVersion selected is **v3**.
Confirm Password	Confirm the AuthPass by retyping it here.
AuthType	This parameter too appears only if **v3** is selected as the SNMPVersion. From the Authtype list box, choose the authentication algorithm using which SNMP v3 converts the specified username and password into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm
EncryptFlag	This flag appears only when **v3** is selected as the SNMPVersion. By default, the eG agent does not encrypt SNMP requests. Accordingly, the this flag is set to **No** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **Yes** option.
EncryptType	If this EncryptFlag is set to **Yes**, then you will have to mention the encryption type by selecting an option from the EncryptType list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard
EncryptPassword	Specify the encryption password here.

Parameter	Description
Confirm Password	Confirm the encryption password by retyping it here.
Timeout	This parameter appears only if the Mode is set to **SNMP**. Specify the duration (in seconds) within which the SNMP query executed by this test should time out in this text box. The default is 10 seconds.
Data Over TCP	This parameter appears only if the Mode is set to **SNMP**. By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic – for instance, certain types of data traffic or traffic pertaining to specific components – to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the monitored target over TCP (and not UDP). For this, set this flag to **Yes**. By default, this flag is set to **No**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Has JVM been restarted?	Indicates whether or not the JVM has restarted during the last measurement period.		If the value of this measure is *No*, it indicates that the JVM has not restarted. The value *Yes* on the other hand implies that the JVM has indeed restarted. The numeric values that correspond to the reboot states discussed above are listed in the table below: **State** / **Value**: Yes – 1; No – 0 **Note:** By default, this measure reports the value Yes or No to indicate whether a JVM has restarted. The graph of this measure however, represents the same using the numeric equivalents – 0 or 1.

Measurement	Description	Measurement Unit	Interpretation
Uptime during the last measure period	Indicates the time period that the JVM has been up since the last time this test ran.	Secs	If the JVM has not been restarted during the last measurement period and the agent has been running continuously, this value will be equal to the measurement period. If the JVM was restarted during the last measurement period, this value will be less than the measurement period of the test. For example, if the measurement period is 300 secs, and if the JVM was restarted 120 secs back, this metric will report a value of 120 seconds. The accuracy of this metric is dependent on the measurement period – the smaller the measurement period, greater the accuracy.
Total uptime of the JVM	Indicates the total time that the JVM has been up since its last reboot.	Secs	Administrators may wish to be alerted if a JVM has been running without a reboot for a very long period. Setting a threshold for this metric allows administrators to determine such conditions.

3.3.8 JVM Garbage Collections Test

Manual memory management is time consuming, and error prone. Most programs still contain leaks. This is all doubly true with programs using exception-handling and/or threads. Garbage collection (GC) is a part of a Java application's JVM that automatically determines what memory a program is no longer using, and recycles it for other use. It is also known as "automatic storage (or memory) reclamation". The JvmGarbageCollection test reports the performance statistics pertaining to the JVM's garbage collection.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for every garbage collector configured on the server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
Mode	This test can extract metrics from the Java application using either of the following mechanisms: • Using SNMP-based access to the Java runtime MIB statistics; • By contacting the Java runtime (JRE) of the application via JMX To configure the test to use SNMP, select the **SNMP** option. On the other hand, choose the JMX option to configure the test to use JMX instead. By default, the **JMX** option is chosen here.
JMX Port	This parameter appears only if the mode is set to **JMX**. Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
JNDIName	This parameter appears only if the Mode is set to JMX. The JNDIName is a lookup name for connecting to the JMX connector. By default, this is JMXRMI. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
User, Password, and Confirm Password	These parameters appear only if the Mode is set to **JMX**. If JMX requires **authentication only** (but no security), then ensure that the User and Password parameters are configured with the credentials of a user with *read-write* access to JMX. To know how to create this user, refer to the *Monitoring Java Applications* document. Confirm the password by retyping it in the Confirm Password text box.
SNMPPort	This parameter appears only if the Mode is set to **SNMP**. Here specify the port number through which the server exposes its SNMP MIB. Ensure that you specify the same port you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).

Parameter	Description
SNMPVersion	This parameter appears only if the Mode is set to **SNMP**. By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the SNMPversion list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list.
SNMPCommunity	This parameter appears only if the Mode is set to **SNMP**. The SNMP community name that the test uses to communicate with the firewall. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the SNMPVersion chosen is **v3**, then this parameter will not appear.
UserName	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only when **v3** is selected as the SNMPVersion. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against this parameter.
Context	This parameter appears only if the Mode is set to **SNMP**. This parameter appears only when v3 is selected as the SNMPVersion. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the SNMPEngineID value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a contextName). If the Username provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the Username in the Context text box. By default, this parameter is set to *none*.
AuthPass	Specify the password that corresponds to the above-mentioned Username. This parameter once again appears only if the SNMPVersion selected is **v3**.
Confirm Password	Confirm the AuthPass by retyping it here.
AuthType	This parameter too appears only if **v3** is selected as the SNMPVersion. From the Authtype list box, choose the authentication algorithm using which SNMP v3 converts the specified username and password into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm

Parameter	Description
	• **SHA** – Secure Hash Algorithm
EncryptFlag	This flag appears only when **v3** is selected as the SNMPVersion. By default, the eG agent does not encrypt SNMP requests. Accordingly, the this flag is set to **No** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **Yes** option.
EncryptType	If this EncryptFlag is set to **Yes**, then you will have to mention the encryption type by selecting an option from the EncryptType list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard
EncryptPassword	Specify the encryption password here.
Confirm Password	Confirm the encryption password by retyping it here.
Timeout	This parameter appears only if the Mode is set to **SNMP**. Specify the duration (in seconds) within which the SNMP query executed by this test should time out in this text box. The default is 10 seconds.
Data Over TCP	This parameter appears only if the Mode is set to **SNMP**. By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic – for instance, certain types of data traffic or traffic pertaining to specific components – to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the monitored target over TCP (and not UDP). For this, set this flag to **Yes**. By default, this flag is set to **No**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
No of garbage collections started	Indicates the number of times garbage collection started to release dead objects form memory.	Number	
Time taken for	Indicates the time taken to	Secs	Ideally, the value of both these

Measurement	Description	Measurement Unit	Interpretation
garbage collection	perform the current garbage collection operation.		measures should be low. This is because, the garbage collection (GC) activity tends to suspend the operations of the application until such time that GC ends. Longer the GC time, longer it would take for the application to resume its functions. To minimize the impact of GC on application performance, it is best to ensure that GC activity does not take too long to complete.
Percent of time spent by JVM for garbage collection	Indicates the percentage of time spent by JVM in garbage collection.	Percent	

3.3.9 JVM Memory Pool Garbage Collections Test

While the **JVM Garbage Collections** test reports statistics indicating how well each collector on the JVM performs garbage collection, the measures reported by the **JVM Memory Pool Garbage Collections** test help assess the impact of the garbage collection activity on the availability and usage of memory in each memory pool of the JVM. Besides revealing the count of garbage collections per collector and the time taken by each collector to perform garbage collection on the individual memory pools, the test also compares the amount of memory used and available for use pre and post garbage collection in each of the memory pools. This way, the test enables administrators to guage the effectiveness of the garbage collection activity on the memory pools, and helps them accurately identify those memory pools where enough memory could not reclaimed or where the garbage collectors spent too much time.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for every *GarbageCollector:MemoryPool* pair on the JVM of the server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.

Parameter	Description
Port	The port at which the specified host listens. By default, this is *NULL*.
Measure Mode	This test allows you the option to collect the desired metrics using one of the following methodologies: • By contacting the Java runtime (JRE) of the application via JMX • Using GC logs To use JMX for metrics collections, set the Measure Mode to **JMX**. On the other hand, if you intend to use the GC log files for collecting the required metrics, set the measure mode to **Log File**. In this case, you would be required to enable GC logging. The procedure for this has been detailed in the Monitoring Java Applications document.
JMX Port	This parameter appears only if the Measure Mode is set to **JMX**. Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
JNDIName	This parameter appears only if the Measure Mode is set to **JMX**. The JNDIName is a lookup name for connecting to the JMX connector. By default, this is JMXRMI. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
User, Password, and Confirm Password	These parameters appear only if the Measure Mode is set to **JMX**. If JMX requires **authentication only** (but no security), then ensure that the User and Password parameters are configured with the credentials of a user with *read-write* access to JMX. To know how to create this user, refer to the *Monitoring Java Applications* document. Confirm the password by retyping it in the Confirm Password text box.
JRE Home	This parameter will be available only if the Measure Mode is set to **Log File**. Specify the full path to the Java Runtime Environment (JRE) used by the target application.
LogFileName	This parameter will be available only if the Measure Mode is set to **Log File**. Specify the full Mpath to the GC log file to be used for metrics collection.

Measurements made by the test

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td>Has garbage collection happened?</td><td>Indicates whether garbage collection occurred on this memory pool in the last measurement period.</td><td></td><td>This measure reports the value Yes if garbage collection took place or No if it did not take place on the memory pool.

The numeric values that correspond to the measure values of Yes and No are listed below:

<table><tr><th>State</th><th>Value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>0</td></tr></table>
Note:

By default, this measure reports the value Yes or No to indicate whether a GC occurred on a memory pool or not. The graph of this measure however, represents the same using the numeric equivalents – 0 or 1.</td></tr>
<tr><td>Collection count</td><td>Indicates the number of time in the last measurement pool garbage collection was started on this memory pool.</td><td>Number</td><td></td></tr>
<tr><td>Initial memory before GC</td><td>Indicates the initial amount of memory (in MB) that this memory pool requests from the operating system for memory management during startup, before GC process.</td><td>MB</td><td>Comparing the value of these two measures for a memory pool will give you a fair idea of the effectiveness of the garbage collection activity.

If garbage collection reclaims a large amount of memory from the memory pool, then the Initial memory after GC will drop. On the other hand, if the garbage collector does not reclaim much memory from a memory pool, or if the Java application suddenly runs a</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
Initial memory after GC	Indicates the initial amount of memory (in MB) that this memory pool requests from the operating system for memory management during startup, after GC process	MB	memory-intensive process when GC is being performed, then the Initial memory after GC may be higher than the Initial memory before GC.
Max memory before GC	Indicates the maximum amount of memory that can be used for memory management by this memory pool, before GC process.	MB	Comparing the value of these two measures for a memory pool will provide you with insights into the effectiveness of the garbage collection activity. If garbage collection reclaims a large amount of memory from the memory pool, then the Max memory after GC will drop. On the other hand, if the garbage collector does not reclaim much memory from a memory pool, or if the Java application suddenly runs a memory-intensive process when GC is being performed, then the Max memory after GC value may exceed the Max memory before GC.
Max memory after GC	Indicates the maximum amount of memory (in MB) that can be used for memory management by this pool, after the GC process.	MB	
Committed memory before GC	Indicates the amount of memory that is guaranteed to be available for use by this memory pool, before the GC process.	MB	
Committed memory after GC	Indicates the amount of memory that is guaranteed to be available for use by this memory pool, after the GC process.	MB	
Used memory before GC	Indicates the amount of memory used by this memory pool before GC.	MB	Comparing the value of these two measures for a memory pool will provide you with insights into the effectiveness of the garbage collection

Measurement	Description	Measurement Unit	Interpretation
			activity. If garbage collection reclaims a large amount of memory from the memory pool, then the Used memory after GC may drop lower than the Used memory before GC. On the other hand, if the garbage collector does not reclaim much memory from a memory pool, or if the Java application suddenly runs a memory-intensive process when GC is being performed, then the Used memory after GC value may exceed the Used memory before GC.
Used memory after GC	Indicates the amount of memory used by this memory pool after GC.	MB	
Percentage memory collected	Indicates the percentage of memory collected from this pool by the GC activity.	Percent	A high value for this measure is indicative of a large amount of unused memory in the pool. A low value on the other hand indicates that the memory pool has been over-utilized. Compare the value of this measure across pools to identify the pools that have very little free memory. If too many pools appear to be running short of memory, it could indicate that the target application is consuming too much memory, which in the long run, can slow down the application significantly.
Collection duration	Indicates the time taken by this garbage collector for collecting unused memory from this pool.	Mins	Ideally, the value of this measure should be low. This is because, the garbage collection (GC) activity tends to suspend the operations of the application until such time that GC ends. Longer the GC time, longer it would take for the application to resume its functions. To minimize the impact of GC on application performance, it is best to ensure that GC activity does not take too long to complete.

3.3.10 JMX Connection to JVM Test

This test reports the availability of the target Java application, and also indicates whether JMX is enabled on the application or not. In addition, the test promptly alerts you to slowdowns experienced by the application, and also reveals whether the application was recently restarted or not.

Target of the test : A JAVA Application

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for the Java application being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
JMX Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the *management.properties* file in the <JAVA_HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
JNDIName	The JNDIName is a lookup name for connecting to the JMX connector. By default, this is JMXRMI. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
User, Password, and Confirm Password	These parameters appear only if the Mode is set to **JMX**. If JMX requires **authentication only** (but no security), then ensure that the User and Password parameters are configured with the credentials of a user with *read-write* access to JMX. To know how to create this user, refer to the *Monitoring Java Applications* document. Confirm the password by retyping it in the Confirm Password text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
JMX availability	Indicates whether the target application is available or not and	Percent	If the value of this measure is 100%, it indicates that the Java application is available with JMX enabled. The value

Measurement	Description	Measurement Unit	Interpretation
	whether JMX is enabled or not on the application.		0 on the other hand, could indicate one/both the following: • The Java application is unavailable; • The Java application is available, but JMX is not enabled;
JMX response time	Indicates the time taken to connect to the JMX agent of the Java application.	Secs	A high value could indicate a connection bottleneck.
Has the PID changed?	Indicates whether/not the process ID that corresponds to the Java application has changed.		This measure will report the value Yes if the PID of the target application has changed; such a change is indicative of an application restart. If the application has not restarted - i.e., if the PID has not changed - then this measure will return the value No.

3.3.11 JVM File Descriptors Test

This test reports useful statistics pertaining to file descriptors.

Target of the test : A Java Application

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for the Java application being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
JMX Port	This parameter appears only if the Measure Mode is set to **JMX**. Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the *management.properties* file in the <JAVA_

Parameter	Description
	HOME>\jre\lib\management folder used by the target application (refer to the Monitoring Java Applications document).
JNDIName	This parameter appears only if the Measure Mode is set to **JMX**. The JNDIName is a lookup name for connecting to the JMX connector. By default, this is JMXRMI. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
User, Password, and Confirm Password	These parameters appear only if the Measure Mode is set to **JMX**. If JMX requires **authentication only** (but no security), then ensure that the User and Password parameters are configured with the credentials of a user with *read-write* access to JMX. To know how to create this user, refer to the *Monitoring Java Applications* document. Confirm the password by retyping it in the Confirm Password text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Open file descriptors in JVM	Indicates the number of file descriptors currently open for the application.	Number	
Maximum file descriptors in JVM	Indicates the maximum number of file descriptors allowed for the application.	Number	
File descriptor usage by JVM	Indicates the file descriptor usage in percentage.	Percent	

3.4 The WebLogic Container Layer

The tests mapped to this layer measure the health of the WebLogic Container. In the process, the layer reveals:

- Processing bottlenecks in the WebLogic server;
- Excessive usage of threads by the execute queues/work managers on the WebLogic server;
- Work managers that are slow in processing requests;
- The availability and responsiveness of the Web server component of WebLogic;

- Security violations on the WebLogic server and the number of users who were locked out as a result;
- The transaction load on the WebLogic server and how well the server handles the load;
- Whether the JMS queues and topics on the WebLogic server have been sized right to handle the load on them.

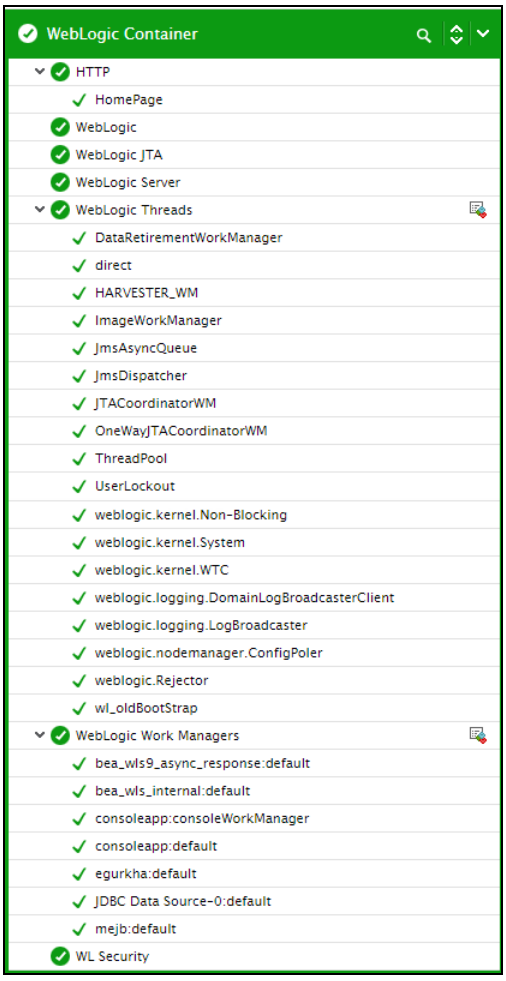


Figure 3.15: The tests mapped to the WebLogic Container layer

3.4.1 WebLogic Test

This test monitors the performance of a WebLogic server by tracking the rate of requests processed by the server, the number of requests waiting for processing, and the percentage of heap usage by the server. While the rate of requests processed and the number of queued requests can be indicative of performance problems with the WebLogic server, the percentage heap usage can be indicative of the reason for the problem.

The heap size determines how often, and for how long garbage collection is performed by the Java Virtual Machine (JVM) that hosts the WebLogic server. The Java heap is a repository for live objects, dead objects, and free memory. When the JVM runs out of memory in the heap, all execution in the JVM stops while a Garbage Collection (GC) algorithm goes through memory and frees space that is no longer required by an application. This is an obvious performance hit because users accessing a WebLogic server must wait while GC happens. No server-side work can be done during GC. Consequently, the heap size must be tuned to minimize the amount of time that the JVM spends in garbage collection, while at the same time maximizing the number of clients that the server can handle at a given time. For Java 2 environments, it is recommended that the heap size be set to be as possible without causing the host system to "swap" pages to disk (use the output of eG Enterprise's SystemTest to gauge the amount of swapping being performed by the operating system).

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each WebLogic application server.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
SNMPPort	The port number on which the WebLogic server is exposing its SNMP MIB (relevant to WebLogic server 5.1 only). For version 6.0 and above, enter "*none*" in this text box.
SNMPVersion	By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the SNMPversion list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list.
SNMPCommunity	The SNMP community name that the test uses to communicate with the firewall. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the SNMPVersion chosen is **v3**, then this parameter will not appear.
UserName	This parameter appears only when **v3** is selected as the SNMPVersion. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from

Parameter	Description
	the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against this parameter.
Context	This parameter appears only when v3 is selected as the SNMPVersion. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the SNMPEngineID value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a contextName). If the Username provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the Username in the Context text box. By default, this parameter is set to *none*.
AuthPass	Specify the password that corresponds to the above-mentioned Username. This parameter once again appears only if the SNMPVersion selected is **v3**.
Confirm Password	Confirm the AuthPass by retyping it here.
AuthType	This parameter too appears only if **v3** is selected as the SNMPVersion. From the Authtype list box, choose the authentication algorithm using which SNMP v3 converts the specified username and password into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm
EncryptFlag	This flag appears only when **v3** is selected as the SNMPVersion. By default, the eG agent does not encrypt SNMP requests. Accordingly, the this flag is set to **No** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **Yes** option.
EncryptType	If this EncryptFlag is set to **Yes**, then you will have to mention the encryption type by selecting an option from the EncryptType list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard
EncryptPassword	Specify the encryption password here.

Parameter	Description
Confirm Password	Confirm the encryption password by retyping it here.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
Timeout	Specify the duration (in seconds) within which the SNMP query executed by this test should time out in this text box. The default is 10 seconds.
Data Over TCP	By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic – for instance, certain types of data traffic or traffic pertaining to specific components – to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the monitored target over TCP (and not UDP). For this, set this flag to **Yes**. By default, this flag is set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the

Parameter	Description
	admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Throughput	Rate of requests processed by the WebLogic server.	Reqs/Sec	A high request rate is an indicator of server overload. By comparing the request rates across application servers, an operator can gauge the effectiveness of load balancers (if any) that are in use.
Heap usage percent	Percentage of heap space currently in use by the WebLogic server.	Percent	When the heap used percent reaches 100%, the server will start garbage

Measurement	Description	Measurement Unit	Interpretation
			collection rather than processing requests. Hence, a very high percentage of heap usage (close to 100%) will dramatically lower performance. In such a case, consider increasing the heap size to be used.
Requests queued	Number of requests currently waiting to be processed by the server.	Number	An increase in number of queued requests can indicate a bottleneck on the WebLogic server. One of the reasons for this could be a bottleneck at the server or due one or more of the applications hosted on the server.
Total heap size	Current heap size of the WebLogic server's Java Virtual Machine.	MB	
Free heap size	The currently unused portion of the WebLogic server's Java Virtual Machine.	MB	

3.4.2 WebLogic Threads Test

A WebLogic server (prior to version 9.x) may be configured with different execute queues. By default, a WebLogic server is configured with one thread queue that is used for execution by all applications running on a server instance. A common way of improving a WebLogic server's performance is by configuring multiple thread execute queues. For eg., a mission-critical application can be assigned to a specific thread execute queue, thereby guaranteeing it a fixed number of execute threads. Other, less critical applications may compete for threads in the default execute queue. While using different thread execute queues can significantly improve performance, if the thread execute queues are not properly configured or maintained, this could result in less than optimal performance. For eg., you may find that while one thread queue has a number of idle threads, applications in another thread execute queue could be waiting for execute threads to become available. In case of the WebLogic server prior to version 9.x, the WebLogic Threads test monitors the different thread execute queues configured for the server.

From WebLogic server 9.x onwards however, execute queues are replaced by 'work managers'. Therefore, while monitoring WebLogic server 9.x or above, the WebLogic Threads test will report

one set of metrics for every 'work manager' configured for the server. Also, the test will take an additional **ThreadPool** descriptor, which will report the extent of usage of the thread pool.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each thread execute queue of a WebLogic application server.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use.

Parameter	Description
	Note: If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Idle threads	Indicates the number of idle threads assigned to a queue.	Number	If the value of this measure is close to 0, it indicates a probable delay in the processing of subsequent requests. In case of WebLogic 9.x or higher, this measure will be available for the ThreadPool descriptor only, and not the individual work managers.

Measurement	Description	Measurement Unit	Interpretation
Thread utilization	Indicates the percentage of threads utilized in a queue	Percent	When this value becomes 100 %, it indicates a heavy load on the server and that it cannot process further requests until a few threads become idle. Typically, this value should be less than 90%. In case of WebLogic 9.x or higher, this measure will be available for the ThreadPool descriptor only, and not the individual work managers.
Pending requests	Indicates the number of requests waiting in the queue	Number	A high value of this measure can result in significant request processing delays.
Requests	Indicates the number of requests that are processed by the server per second	Reqs/sec	While a high value of this measure is indicative of the good health of the server, a low value indicates a processing bottleneck.

3.4.3 WebLogic Work Managers Test

The WebLogic Server allows you to configure how your application prioitizes the execution of its work based on rules you define and by monitoring actual runtime performance. You define the rules and constraints for your application by defining a Work Manager and applying it either globally to WebLogic Server domain or to a specific application component.

This test monitors the requests to applications, and helps analyze how the work manager mapped to each application is managing the requests. By closely observing the variations to the measures reported by this test, you can quickly identify current/potential application slowdowns, and figure out whether changes in the corresponding work manager specification can improve application performance.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every work manager on the WebLogic server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version).

Parameter	Description
	This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Completed requests	Indicates the number of requests that were successfully serviced by the work manager mapped to this application.	Number	
Pending requests	Indicates the number of requests to this application that are waiting in the queue.	Number	A large number of pending requests to an application could indicate a bottleneck in the request processing ability of that application. If too many applications on the server support long-winding request queues, it can ultimately overload the server, and eventually choke its performance. It is therefore essential to quickly isolate

Measurement	Description	Measurement Unit	Interpretation
			those applications that could be experiencing issues with request processing, and then initiate the relevant remedial action on them. Comparing the value of this measure across applications will enable you to accurately identify which application has the maximum number of pending requests. Once the application is spotted, you may want to observe the variations in the pending requests count over time for that application. If you find that the value of this measure keeps increasing with time for that application, further investigation may be necessary to determine the reasons for the same. One of the possible reasons for this could be the lack of sufficient threads. Incoming requests to an application cannot be processed if adequate threads are unavailable; such requests will hence be in queue until such time that the server allocates more threads to the application. As already stated, the WebLogic server prioritizes work and allocates threads to an application based on the rules and constraints defined within the work manager that is either defined globally or mapped specifically to that application. Therefore, in the event of a slow down in the request processing rate of an application, you can consider fine-tuning its associated work manager definition, so as to ensure the uninterrupted processing of requests. A typical work manager definition

Measurement	Description	Measurement Unit	Interpretation
			should include one request class and one/more thread constraints. A request class expresses a scheduling guideline that WebLogic Server uses to allocate threads to requests. Request classes help ensure that high priority work is scheduled before less important work, even if the high priority work is submitted after the lower priority work. A work manager can specify any one of the below-mentioned request classes: • **Fair share request class**: This specifies the average thread-use time required to process requests. For example, assume that WebLogic Server is running two modules. The Work Manager for ModuleA specifies a fair-share-request-class of 80 and the Work Manager for ModuleB specifies a fair-share-request-class of 20. During a period of sufficient demand, with a steady stream of requests for each module such that the number requests exceed the number of threads, WebLogic Server will allocate 80% and 20% of the thread-usage time to ModuleA and ModuleB, respectively. • **Response time request class**: This type of request class specifies a response time goal in

Measurement	Description	Measurement Unit	Interpretation
			milliseconds. Response time goals are not applied to individual requests. Instead, WebLogic Server computes a tolerable waiting time for requests with that class by subtracting the observed average thread use time from the response time goal, and schedules requests so that the average wait for requests with the class is proportional to its tolerable waiting time. • **Context request class**: This type of request class assigns request classes to requests based on context information, such as the current user or the current user's group. A constraint defines minimum and maximum numbers of threads allocated to execute requests and the total number of requests that can be queued or executing before WebLogic Server begins rejecting requests. You can define the following types of constraints: • **max-threads-constraint** - This constraint limits the number of concurrent threads executing requests from the constrained work set. The default is unlimited. For example, consider a constraint

Measurement	Description	Measurement Unit	Interpretation
			defined with maximum threads of 10 and shared by 3 entry points. The scheduling logic ensures that not more than 10 threads are executing requests from the three entry points combined. • **min-threads-constraint** - This constraint guarantees a number of threads the server will allocate to affected requests to avoid deadlocks. The default is zero. A min-threads-constraint value of one is useful, for example, for a replication update request, which is called synchronously from a peer. • **capacity** - This constraint causes the server to reject requests only when it has reached its capacity. The default is -1. Note that the capacity includes all requests, queued or executing, from the constrained work set. Work is rejected either when an individual capacity threshold is exceeded or if the global capacity is exceeded. This constraint is independent of the global queue threshold.
Stuck threads	Indicates the number of threads that are considered to be stuck on the basis of any thread constraints.	Number	WebLogic Server diagnoses a thread as stuck if it is continually working (not idle) for a set period of time. You can tune a server's thread detection

Measurement	Description	Measurement Unit	Interpretation
			behavior by changing the length of time before a thread is diagnosed as stuck, and by changing the frequency with which the server checks for stuck threads. In response to stuck threads, you can define a Stuck Thread Work Manager component that can shut down the Work Manager, move the application into admin mode, or mark the server instance as failed.

3.4.4 HTTP Test

The details of the HTTP test that emulates a user accessing the web server component of a WebLogic server, are provided below. Since this test can be executed from a location external to the WebLogic server, this test presents an unbiased external perspective of the state of the web server component. This test uses the GET command to submit its parameters.

Target of the test : A WebLogic Application Server

Agent deploying this test : An external agent executing on an eG server

Outputs of the test : One set of outputs for every URL being monitored.

Configurable parameters for the test

Parameters	Description
Test period	This indicates how often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port to which the specified host listens.
URL	The web page being accessed. While multiple URLs (separated by commas) can be provided, each URL should be of the format URL name:URL value. URL name is a unique name assigned to the URL, and the URL value is the value of the URL. For example, a URL can be specified as *HomePage:http://192.168.10.12:7077/*, where HomePage is the URL name and http://192.168.10.12:7077/ is the URL value.

Parameters	Description
Cookiefile	Whether any cookies being returned by the web server need to be saved locally and returned with subsequent requests.
ProxyHost	The host on which a web proxy server is running (in case a proxy server is to be used).
ProxyPort	The port number on which the web proxy server is listening.
Proxyusername	The user name of the proxy server.
Proxypassword	The password of the proxy server.
Confirm password	Confirm the password by retyping it here.
Content	This parameter is a set of instruction:value pairs that are used to validate the content being returned by the test. If the Content value is *none:none*, no validation is performed. The number of pairs specified in this text box, must be equal to the number of URLs being monitored. The instruction should be one of **Inc** or **Exc**. **Inc** tells the test that for the content returned by the web server to be valid, the content must include the specified value (a simple string search is done in this case). An instruction of **Exc** instructs the test that the server's output is valid if it does not contain the specified value. In both cases, the content specification can include wild card patterns. For example, an Inc instruction can be *Inc:*Home page**. An Inc and an Exc instruction can be provided in quick succession in the following format: *Inc:*Home Page*,Exc:*home*.
Credentials	This test supports HTTP authentication. The Credentials parameter is to be set if a specific user name / password has to be specified to login to a page. Against this parameter, the URLname of every configured URL will be displayed; corresponding to each listed URLname, a Username text box and a Password text box will be made available. If the web server on which HttpTest executes supports 'Anonymous user access', then this parameter will take either of the following values: • a valid Username and Password for every configured URLname • *none* in both the Username and Password text boxes of all configured URLnames (the default setting), if no user authorization is required Some IIS web servers however, support NTLM (Integrated Windows) authentication, where valid Credentials are mandatory. In other words, a *none* specification will not be supported by such IIS web servers. Therefore, in this case, against each configured URLname, you will have to provide a valid Username in the format: *domainname\username,* followed by a valid Password. Please be sure to check if your web site requires HTTP authentication while configuring this parameter. HTTP authentication typically involves a separate pop-up

Parameters	Description
	window when you try to access the page. Many sites use HTTP POST for obtaining the user name and password and validating the user login. In such cases, the username and password have to be provided as part of the POST information and NOT as part of the Credentials specification for the HTTP test.
Timeout	Here, specify the maximum duration (in seconds) for which the test will wait for a response from the server. The default Timeout period is 30 seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Availability	This measurement indicates whether the server was able to respond successfully to the query made by the test.	Percent	Availability failures could be caused by several factors such as the web server process(es) being down, the web server being misconfigured, a network failure, etc. Temporary unavailability may also occur if the web server is overloaded. Availability is determined based on the response code returned by the server. A response code between 200 to 300 indicates that the server is available.
Total response time	This measurement indicates the time taken by the server to respond to the requests it receives.	Secs	Response time being high denotes a problem. Poor response times may be due to the server being overloaded or misconfigured. If the URL accessed involves the generation of dynamic content by the server, backend problems (e.g., an overload at the application server or a database failure) can also result in an increase in response time.
Tcp connection availability	This measure indicates whether the test managed to establish a TCP connection to the server.	Percent	Failure to establish a TCP connection may imply that either the web server process is not up, or that the process is not operating correctly. In some cases of extreme overload, the failure to establish a TCP connection may be

Measurement	Description	Measurement Unit	Interpretation
			a transient condition. As the load subsides, the server may start functioning properly again.
Tcp connect time	This measure quantifies the time for establishing a TCP connection to the web server host.	Secs	Typically, the TCP connection establishment must be very small (of the order of a few milliseconds). Since TCP connection establishment is handled at the OS-level, rather than by the application, an increase in this value signifies a system-level bottleneck on the host that supports the web server.
Server response time	This measure indicates the time period between when the connection was established and when the server sent back a HTTP response header to the client.	Secs	While the total response time may depend on several factors, the server response time is typically, a very good indicator of a server bottleneck (e.g., because all the available server threads or processes are in use).
Response code	The response code returned by the server for the simulated request	Number	A value between 200 and 300 indicates a good response. A 4xx value indicates a problem with the requested content (eg., page not found). A 5xx value indicates a server error.
Content length	The size of the content returned by the server	Kbytes	Typically the content length returned by the server for a specific URL should be the same across time. Any change in this metric may indicate the need for further investigation on the server side.
Content validity	This measure validates whether the server was successful in executing the request made to it.	Percent	A value of 100% indicates that the content returned by the test is valid. A value of 0% indicates that the content may not be valid. This capability for content validation is especially important for multi-tier web applications. For example, a user may

Measurement	Description	Measurement Unit	Interpretation
			not be able to login to the web site but the server may reply back with a valid HTML page where in the error message, say, "Invalid Login" is reported. In this case, the availability will be 100 % (since we got a valid HTML response). If the test is configured such that the content parameter should exclude the string "Invalid Login," in the above scenario content validity would have a value 0.

3.4.5 WL Security Test

This test reports security-related statistics for a WebLogic server. This test will work on WebLogic 7.x and above only. While monitoring WebLogic 6.0, all the measures of this test will return 0.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every WebLogic server instance being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:**

Parameter	Description
	If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.

Parameter	Description
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Invalid login attempts	Returns the cumulative number of invalid logins attempted on this server.	Attempts / Sec	Look for an unusual number of invalid logins.
Invalid users count high water mark	Returns the high water mark of the number of users with outstanding invalid login attempts for this server.	Number	
Current locked users	Returns the number of currently locked users on this server.	Number	Locked users cannot login to the server. Hence, configure the thresholds, so as to be alerted when this value is greater than 0.
Locked user logins	Returns the cumulative number of invalid logins attempted during the last measurement period on this server while the user was locked.	Number	
Unlocked users	Returns the number of times a user has been unlocked on this server during the last measurement period.	Number	
Users lockedout	Returns the cumulative number of user lockouts done on this server during the last measurement period.	Number	

3.4.6 WebLogic JTA Test

This test reports transaction-related statistics for a WebLogic server.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

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

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use.

Parameter	Description
	Note: If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total active transactions	Returns the number of active transactions on the server	Number	This metric gives an idea of the server load.
Transaction aborts	Returns the rate of transactions that were abandoned	Trans/Sec	Typically, the abandoned transaction rate should be low.

Measurement	Description	Measurement Unit	Interpretation
Application rollbacks	Returns the rate of transactions that were rolled back due to an application error	Trans/Sec	Rollbacks can occur due to various reasons. Correlate the rollbacks on the application server with that on the database to isolate the cause of the rollbacks.
Resource rollbacks	Returns the rate of transactions that were rolled back due to a resource error	Trans/Sec	
System rollbacks	Returns the rate of transactions that were rolled back due to an internal system error	Trans/Sec	
Rollback timeouts	Returns the rate of transactions that were rolled back due to a timeout expiration	Trans/Sec	
Commits	Returns the rate of committed transactions	Trans/Sec	
Heuristic transactions	Returns the rate of transactions that completed with a heuristic status	Trans/Sec	
Total rollbacks	Returns the rate of transactions that were rolled back	Trans/Sec	A high value (rate) here would mean that more number of transactions are being rolled back.
Total transactions	Returns the total rate of transactions processed. This total includes all committed, rolled back and heuristic transaction completions	Trans/Sec	

3.4.7 WebLogic Server Test

This test reports key run-time statistics pertaining to the instances of a WebLogic server.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every WebLogic server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the

Parameter	Description
	admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Server availability	Indicates the state of the WebLogic server instance.	Percent	A server instance can assume different states. A table listing states of the server instance gives you a clear view on the various states of the server instance and what does each status mean. If the state of the server instance is STARTING, RUNNING, or RESUMING, then the value of this

Measurement	Description	Measurement Unit	Interpretation
			measure will be 100. For any other state, the value of this measure will be 0.
Current sockets count	Indicates the number of sockets registered for socket muxing on this server instance.	Number	
Sockets opened	Indicates the total number of registrations for socket muxing on this server.	Number	

A table listing states of the server instance

State	Description
ACTIVE_LATER	Indicates that MaxRestart restart attempts have been made in current RestartInterval, and Node Manager will attempt additional restarts
FAILED	A critical subsystem is not functioning
FAILED_NOT_ RESTARTABLE	Indicates that the Managed Server has failed or was killed by Node Manager as a result of the Managed Server's AutoKillIfFailed attribute being set to True, but Node Manager cannot restart the Managed Server because its AutoRestart attribute is set to False.
FAILED_RESTARTING	Indicates that Node Manager is currently restarting a failed Managed Server
RESUMING	The server is transitioning from the STANDBY state to RUNNING.
RUNNING	The server can receive and process requests from external clients as well as administrative requests.
SHUTDOWN	The server is configured but inactive.
SHUTDOWN_IN_PROCESS	Indicates that the shutdown is in progress
SHUTDOWN_PENDING	Indicates that a shutdown request has been sent, but the actual shutdown process is yet to begin
SHUTTING_DOWN	The server is transitioning from either the RUNNING or STANDBY state to SHUTDOWN.
STANDBY	The server can receive and process only administrative requests.
STARTING	The server is initializing all it's subsystems.
SUSPENDING	The server is transitioning from either the SHUTDOWN or RUNNING state to STANDBY.
UNKNOWN	The state of the server cannot be determined, perhaps because it cannot be

State	Description
	contacted.

3.4.8 WebLogic Topics Test

The publish/subscribe (pub/sub) messaging model enables an application to send a message to multiple applications. Pub/sub messaging applications send and receive messages by subscribing to a topic. A topic publisher (producer) sends messages to a specific topic. A topic subscriber (consumer) retrieves messages from a specific topic.

This test auto-discovers the topics on a WebLogic server, and monitors each topic for the size, number, and type of messages it holds, so that impending overloads and probable delivery bottlenecks can be proactively isolated and corrected.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every topic auto-discovered on the monitored WebLogic server.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the

Parameter	Description
	WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Messages count	Indicates the current number of messages in this topic.	Number	This count does not include the messages that are pending.
Messages pending count	Indicates the number of pending messages in this topic.	Number	While momentary spikes in the number of pending messages in a topic is normal, if the number is allowed to grow consistently over time, it is bound to increase the total number of messages in the topic. Typically, the sum of the values of the MessagesCurrentCount and the MessagesPendingCount measures equals the total number of messages in the topic. If this sum is equal to or is very close to the Messages Maximum setting for the quota resource that is mapped to this topic, it implies that the topic has filled up or is rapidly filling up with messages and cannot handle any more. When this happens, JMS prevents further sends with a ResourceAllocationException. Furthermore, such quota failures will force multiple producers to contend for space in the topic, thereby degrading application performance. To avoid this, you can do one/more of the following: ➢ Increase the Messages Maximum setting of the quota resource mapped to the topic; ➢ If a quota has not been configured for the topic, then increase the quota of the JMS server where the topic is deployed;

Measurement	Description	Measurement Unit	Interpretation
			➢ Regulate the flow of messages into the topic using one/more of the following configurations: ◦ Blocking senders during quota conditions: The Send Timeout feature provides more control over message send operations by giving message producers the option of waiting a specified length of time until space becomes available on a destination. ◦ Specifying a Blocking Send Policy on JMS Servers : The Blocking Send policies enable you to define the JMS server's blocking behavior on whether to deliver smaller messages before larger ones when multiple message producers are competing for space on a destination that has exceeded its message quota. ◦ Using the Flow Control feature: With the Flow Control feature, you can direct a JMS server or destination to slow down message producers when it determines that it is becoming overloaded. Specifically, when either a JMS server or it's destinations exceeds its specified

Measurement	Description	Measurement Unit	Interpretation
			byte or message threshold, it becomes armed and instructs producers to limit their message flow (messages per second). Producers will limit their production rate based on a set of flow control attributes configured for producers via the JMS connection factory. Starting at a specified flow maximum number of messages, a producer evaluates whether the server/destination is still armed at prescribed intervals (for example, every 10 seconds for 60 seconds). If at each interval, the server/destination is still armed, then the producer continues to move its rate down to its prescribed flow minimum amount. As producers slow themselves down, the threshold condition gradually corrects itself until the server/destination is unarmed. At this point, a producer is allowed to increase its production rate, but not necessarily to the maximum possible rate. In fact, its message flow continues to be controlled (even though the server/destination is no longer armed) until it reaches its prescribed flow maximum, at which point it is no longer flow

Measurement	Description	Measurement Unit	Interpretation
			controlled. By tuning Message Performance Preference: The Messaging Performance Preference tuning option on JMS destinations enables you to control how long a destination should wait (if at all) before creating full batches of available messages for delivery to consumers. At the minimum value, batching is disabled. Tuning above the default value increases the amount of time a destination is willing to wait before batching available messages. The maximum message count of a full batch is controlled by the JMS connection factory's Messages Maximum per Session setting. It may take some experimentation to find out which value works best for your system. For example, if you have a topic with many concurrent message consumers, by selecting the Administration Console's Do Not Batch Messages value (or specifying "0" on the DestinationBean MBean), the topic will make every effort to promptly push messages out to its consumers as soon as they are available. Conversely, if you have a topic with only one message consumer that does not require fast response times, by selecting the console's High Waiting Threshold for Message Batching value (or specifying "100" on the

Measurement	Description	Measurement Unit	Interpretation
			DestinationBean MBean), you can ensure that the topic only pushes messages to that consumer in batches.
Messages moved count	Indicates the number of messages that have been moved from this topic destination to another.	Number	
Bytes count	Indicates the current size of the message that is stored in the topic destination in bytes.	KB	This count does not include the pending bytes.
Bytes pending count	Indicates the current size of the pending message that is stored in the topic destination in bytes.	KB	While momentary spikes in the size of pending messages in a topic is acceptable, if the size is allowed to grow consistently over time, it is bound to increase the total size of all messages in the topic. Typically, the sum of the values of the BytesCurrentCount and the BytesPendingCount measures indicates the total size of all messages in the topic. If this sum is equal to or is very close to the Bytes Maximum setting for the quota resource that is mapped to this topic, it implies that the topic has filled up or is rapidly filling up with messages and cannot handle any more. When this happens, JMS prevents further sends with a ResourceAllocationException. Furthermore, such quota failures will force multiple producers to contend for space in the topic, thereby degrading application performance. To avoid this, you can do one/more of the following: ➢ Increase the Bytes Maximum setting of the quota resource mapped to the

Measurement	Description	Measurement Unit	Interpretation
			topic; ➢ If a quota has not been configured for the topic, then increase the quota of the JMS server where the topic is deployed; ➢ Regulate the flow of messages into the topic using one/more of the following configurations: ◦ Blocking senders during quota conditions: The Send Timeout feature provides more control over message send operations by giving message producers the option of waiting a specified length of time until space becomes available on a destination. ◦ Specifying a Blocking Send Policy on JMS Servers : The Blocking Send policies enable you to define the JMS server's blocking behavior on whether to deliver smaller messages before larger ones when multiple message producers are competing for space on a destination that has exceeded its message quota. ◦ Using the Flow Control feature: With the Flow Control feature, you can direct a JMS server or

Measurement	Description	Measurement Unit	Interpretation
			destination to slow down message producers when it determines that it is becoming overloaded. Specifically, when either a JMS server or it's destinations exceeds its specified byte or message threshold, it becomes armed and instructs producers to limit their message flow (messages per second). Producers will limit their production rate based on a set of flow control attributes configured for producers via the JMS connection factory. Starting at a specified flow maximum number of messages, a producer evaluates whether the server/destination is still armed at prescribed intervals (for example, every 10 seconds for 60 seconds). If at each interval, the server/destination is still armed, then the producer continues to move its rate down to its prescribed flow minimum amount. As producers slow themselves down, the threshold condition gradually corrects itself until the server/destination is unarmed. At this point, a producer is allowed to

Measurement	Description	Measurement Unit	Interpretation
			essage flow continues to be controlled (even though the server/destination is no longer armed) until it reaches its prescribed flow maximum, at which point it is no longer flow controlled. ◦ By Tuning the MessageMaximum configuration: WebLogic JMS pipelines messages that are delivered to asynchronous consumers (otherwise known as message listeners) or prefetch-enabled synchronous consumers. The messages backlog (the size of the pipeline) between the JMS server and the client is tunable by configuring the MessagesMaximum setting on the connection factory. In some circumstances, tuning this setting may improve performance dramatically, such as when the JMS application defers acknowledges or commits. In this case, BEA suggests setting the MessagesMaximum value to: 2 * (ack or commit interval) + 1. For example, if the JMS application acknowledges 50 messages at a time, set the MessagesMaximum

Measurement	Description	Measurement Unit	Interpretation
			value to 101. You may also need to configure WebLogic clients in addition to the WebLogic Server instance, when sending and receiving large messages. ◦ By compressing messages: You may improve the performance of sending large messages traveling across JVM boundaries and help conserve disk space by specifying the automatic compression of any messages that exceed a user-specified threshold size. Message compression can help reduce network bottlenecks by automatically reducing the size of messages sent across network wires. Compressing messages can also conserve disk space when storing persistent messages in file stores or databases. ◦ By paging out messages: With the message paging feature, JMS servers automatically attempt to free up virtual memory during peak message load periods. This feature can greatly benefit applications with large message spaces.

Measurement	Description	Measurement Unit	Interpretation
			◦ By tuning the Message Buffer Size: The Message Buffer Size option specifies the amount of memory that will be used to store message bodies in memory before they are paged out to disk. The default value of Message Buffer Size is approximately one-third of the maximum heap size for the JVM, or a maximum of 512 megabytes. The larger this parameter is set, the more memory JMS will consume when many messages are waiting on topics or topics. Once this threshold is crossed, JMS may write message bodies to the directory specified by the Paging Directory option in an effort to reduce memory usage below this threshold.
Consumers count	Indicates the current number of consumers accessing the topic destination.	Number	
Messages deleted count	Indicates the number of messages that have been deleted from the topic destination.	Number	While you can design a QueueBrowser on your JMS server to view and delete specific queue messages, some messages are automatically deleted by the server. For instance, one-way messages that exceed quota are silently deleted without immediately throwing exceptions back to the client.

3.4.9 WebLogic JMS Test

This test extracts performance statistics pertaining to the Java Message Service (JMS) provided by the WebLogic server. 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 *Oracle WebLogic* 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 WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every WebLogic server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the

Parameter	Description
	metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Data received	Returns the number of bytes received on this JMS server since the last reset	KB/Sec	This is an indicator of the workload on the JMS server.
Destination count high water mark	Returns the peak number of destinations on this JMS server since the last reset	Number	
Session pool count	Returns the current number of session pools instantiated on this JMS server	Number	
Session pool count high water mark	Returns the peak number of session pools instantiated on this JMS server since the last reset	Number	
Message received	Returns the number of messages received on this destination since the last reset	Msgs/Sec	
Current data count	Returns the current number of bytes stored on this JMS server. This does not include the pending bytes.	KB	
Data pending count	Returns the current number of bytes pending (unacknowledged or uncommitted) stored on this JMS server. Pending bytes are over and above the current number of bytes.	KB	
Data count high water mark	Returns the peak number of bytes stored in the JMS server since the last reset	KB	
Current messages	Returns the current number	Number	

Measurement	Description	Measurement Unit	Interpretation
	of messages stored on this JMS server. This does not include the pending messages.		
Pending messages	Returns the current number of messages pending (unacknowledged or uncommitted) stored on this JMS server. Pending messages are over and above the current number of messages.	Number	Ideally, the count of pending messages should be low.
Messages count high water mark	Returns the peak number of messages stored in the JMS server since the last reset	Number	
Destination current count	Returns the current number of destinations for this JMS server	Number	

3.4.10 WebLogic Queues Test

A JMS queue represents the point-to-point (PTP) messaging model, which enables one application to send a message to another. PTP messaging applications send and receive messages using named queues. A queue sender (producer) sends a message to a specific queue. A queue receiver (consumer) receives messages from a specific queue.

This test auto-discovers the queues on a WebLogic server, and monitors each queue for the size, number, and type of messages it holds, so that impending overloads and probable delivery bottlenecks can be proactively isolated and corrected.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every queue auto-discovered on the monitored WebLogic server.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version).

Parameter	Description
	This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Messages count	Indicates the current number of messages in this queue.	Number	This count does not include the messages that are pending.
Messages pending count	Indicates the number of pending messages in this queue.	Number	A message is considered to be in pending state when it is: • sent in a transaction but not committed. • received and not acknowledged • received and not committed • subject to a redelivery delay (as of WebLogic JMS 6.1 or later)

Measurement	Description	Measurement Unit	Interpretation
			• subject to a delivery time (as of WebLogic JMS 6.1 or later)
			While momentary spikes in the number of pending messages in a queue is normal, if the number is allowed to grow consistently over time, it is bound to increase the total number of messages in the queue. Typically, the sum of the values of the *Messages count* and the *Messages pending count* measures equals the total number of messages in the queue. If this sum is equal to or is very close to the *Messages Maximum* setting for the quota resource that is mapped to this queue, it implies that the queue has filled up or is rapidly filling up with messages and cannot handle any more. When this happens, JMS prevents further sends with a ResourceAllocationException. Furthermore, such quota failures will force multiple producers to contend for space in the queue, thereby degrading application performance. To avoid this, you can do one/more of the following: ➢ Increase the Messages Maximum setting of the quota resource mapped to the queue; ➢ If a quota has not been configured for the queue, then increase the quota of the JMS server where the queue is deployed; ➢ Regulate the flow of messages into the queue using one/more of the following configurations:

Measurement	Description	Measurement Unit	Interpretation
			• Blocking senders during quota conditions: The Send Timeout feature provides more control over message send operations by giving message producers the option of waiting a specified length of time until space becomes available on a destination. • Specifying a Blocking Send Policy on JMS Servers : The Blocking Send policies enable you to define the JMS server's blocking behavior on whether to deliver smaller messages before larger ones when multiple message producers are competing for space on a destination that has exceeded its message quota. • Using the Flow Control feature: With the Flow Control feature, you can direct a JMS server or destination to slow down message producers when it determines that it is becoming overloaded. Specifically, when either a JMS server or it's destinations exceeds its specified byte or message threshold, it becomes armed and instructs

Measurement	Description	Measurement Unit	Interpretation
			producers to limit their message flow (messages per second). Producers will limit their production rate based on a set of flow control attributes configured for producers via the JMS connection factory. Starting at a specified flow maximum number of messages, a producer evaluates whether the server/destination is still armed at prescribed intervals (for example, every 10 seconds for 60 seconds). If at each interval, the server/destination is still armed, then the producer continues to move its rate down to its prescribed flow minimum amount. As producers slow themselves down, the threshold condition gradually corrects itself until the server/destination is unarmed. At this point, a producer is allowed to increase its production rate, but not necessarily to the maximum possible rate. In fact, its message flow continues to be controlled (even though the server/destination is no longer armed) until it reaches its prescribed flow maximum, at

Measurement	Description	Measurement Unit	Interpretation
			• By tuning Message Performance Preference: The Messaging Performance Preference tuning option on JMS destinations enables you to control how long a destination should wait (if at all) before creating full batches of available messages for delivery to consumers. At the minimum value, batching is disabled. Tuning above the default value increases the amount of time a destination is willing to wait before batching available messages. The maximum message count of a full batch is controlled by the JMS connection factory's Messages Maximum per Session setting. It may take some experimentation to find out which value works best for your system. For example, if you have a queue with many concurrent message consumers, by selecting the Administration Console's Do Not Batch Messages value (or specifying "0" on the DestinationBean MBean), the queue will make every effort to promptly push messages out to its consumers as soon as they are available. Conversely, if you have a queue with only one message consumer that does not require fast response times, by selecting the console's High Waiting Threshold

Measurement	Description	Measurement Unit	Interpretation
			for Message Batching value (or specifying “100” on the DestinationBean MBean), you can ensure that the queue only pushes messages to that consumer in batches.
Bytes count	Indicates the current size of the message that is stored in the queue destination in bytes.	KB	This count does not include the pending bytes.
Bytes pending count	Indicates the current size of the pending message that is stored in the queue destination in bytes.	KB	While momentary spikes in the size of pending messages in a queue is acceptable, if the size is allowed to grow consistently over time, it is bound to increase the total size of all messages in the queue. Typically, the sum of the values of the BytesCurrentCount and the BytesPendingCount measures indicates the total size of all messages in the queue. If this sum is equal to or is very close to the Bytes Maximum setting for the quota resource that is mapped to this queue, it implies that the queue has filled up or is rapidly filling up with messages and cannot handle any more. When this happens, JMS prevents further sends with a ResourceAllocationException. Furthermore, such quota failures will force multiple producers to contend for space in the queue, thereby degrading application performance. To avoid this, you can do one/more of the following: ➢ Increase the Bytes Maximum setting of the quota resource mapped to the queue; ➢ If a quota has not been configured for

Measurement	Description	Measurement Unit	Interpretation
			the queue, then increase the quota of the JMS server where the queue is deployed; ➢ Regulate the flow of messages into the queue using one/more of the following configurations: • Blocking senders during quota conditions: The Send Timeout feature provides more control over message send operations by giving message producers the option of waiting a specified length of time until space becomes available on a destination. • Specifying a Blocking Send Policy on JMS Servers : The Blocking Send policies enable you to define the JMS server's blocking behavior on whether to deliver smaller messages before larger ones when multiple message producers are competing for space on a destination that has exceeded its message quota. • Using the Flow Control feature: With the Flow Control feature, you can direct a JMS server or destination to slow down message producers when it determines that it is becoming overloaded. Specifically, when

Measurement	Description	Measurement Unit	Interpretation
			either a JMS server or it's destinations exceeds its specified byte or message threshold, it becomes armed and instructs producers to limit their message flow (messages per second). Producers will limit their production rate based on a set of flow control attributes configured for producers via the JMS connection factory. Starting at a specified flow maximum number of messages, a producer evaluates whether the server/destination is still armed at prescribed intervals (for example, every 10 seconds for 60 seconds). If at each interval, the server/destination is still armed, then the producer continues to move its rate down to its prescribed flow minimum amount. As producers slow themselves down, the threshold condition gradually corrects itself until the server/destination is unarmed. At this point, a producer is allowed t o increase its production rate, but not necessarily to the maximum possible rate. In fact, its message flow continues to be controlled (even though the server/destination is no longer armed) until it reaches its prescribed flow maximum, at which point it is no longer flow controlled. • By Tuning the MessageMaximum

Measurement	Description	Measurement Unit	Interpretation
			configuration: WebLogic JMS pipelines messages that are delivered to asynchronous consumers (otherwise known as message listeners) or prefetch-enabled synchronous consumers. The messages backlog (the size of the pipeline) between the JMS server and the client is tunable by configuring the MessagesMaximum setting on the connection factory. In some circumstances, tuning this setting may improve performance dramatically, such as when the JMS application defers acknowledges or commits. In this case, BEA suggests setting the MessagesMaximum value to: 2 * (ack or commit interval) + 1. For example, if the JMS application acknowledges 50 messages at a time, set the MessagesMaximum value to 101. You may also need to configure WebLogic clients in addition to the WebLogic Server instance, when sending and receiving large messages. • By compressing messages: You may improve the performance of sending large messages traveling across JVM boundaries and help conserve disk space by specifying the automatic compression of any messages that exceed a user-specified

Measurement	Description	Measurement Unit	Interpretation
			threshold size. Message compression can help reduce network bottlenecks by automatically reducing the size of messages sent across network wires. Compressing messages can also conserve disk space when storing persistent messages in file stores or databases. • By paging out messages: With the message paging feature, JMS servers automatically attempt to free up virtual memory during peak message load periods. This feature can greatly benefit applications with large message spaces. • By tuning the Message Buffer Size: The Message Buffer Size option specifies the amount of memory that will be used to store message bodies in memory before they are paged out to disk. The default value of Message Buffer Size is approximately one-third of the maximum heap size for the JVM, or a maximum of 512 megabytes. The larger this parameter is set, the more memory JMS will consume when many messages are waiting on queues or topics. Once this threshold is crossed, JMS may write message bodies to the

Measurement	Description	Measurement Unit	Interpretation
			directory specified by the Paging Directory option in an effort to reduce memory usage below this threshold.
Messages deleted count	Indicates the number of messages that have been deleted from this queue.	Number	While you can design a QueueBrowser on your JMS server to view and delete specific queue messages, some messages are automatically deleted by the server. For instance, one-way messages that exceed quota are silently deleted without immediately throwing exceptions back to the client.
Messages moved count	Indicates the number of messages that have been moved from one queue destination to the other.	Number	
Consumers count	Indicates the current number of consumers accessing the queue destination.	Number	

3.4.11 WebLogic Clusters Test

This test extracts cluster related statistics from a managed WebLogic server that is part of a cluster. Server instances in a cluster communicate with each other using multicast—multicasts help the server instances in a cluster announce their services, and issue periodic heartbeats that indicate continued availability. 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 *Oracle WebLogic* as 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 WebLogic Application Server

Agent deploying the test : An internal agent

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

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version).

Parameter	Description
	This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Alive servers	Number of servers in the clusters that are running currently	Number	Ideally this number must be equal to the actual number of servers in the cluster.
Fragments sent	The rate of multicast fragments sent from this server into the cluster	Fragments/Sec	
Fragments received	Returns the rate of multicast messages received on this server from the cluster	Fragments/Sec	
Fragments dropped	Indicates the rate of fragments that originated in foreign domains/clusters that use the same multicast address	Fragments/Sec	

Measurement	Description	Measurement Unit	Interpretation
Messages lost	Returns the rate at which incoming multicast messages that were lost	Msgs/Sec	
Request resends	Returns the rate of state-delta messages that had to be resent because a receiving server in the cluster missed a message	Reqs/sec	
Primary count	Returns the number of objects that the local server hosts as primaries	Number	

3.4.12 File Descriptors Test

A file descriptor is a handle created by a process when a file is opened. A new descriptor is created each time the file is opened. The File Descriptor test monitors the descriptors created by an application. 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 *Oracle WebLogic* as 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 WebLogic Application Server

Agent deploying the test : An internal agent

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

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
PSPath	Specify the path to the **ps** command (The default location is /usr/ucb). The eG user must be vested with execute permissions in order to execute the **ps** command.

Parameter	Description
PFilesPath	Specify the path to the pfiles command (The default is /usr/ucb).
Process	Takes a process name and process pattern in the format: *processName:processPattern*. While the *processName* is a display name, the *processPattern* should uniquely identify the application's process ID (PID of the application). The *processPattern* can be of the form - *expr* or expr or *expr or expr* or *expr1*expr2*... or expr1*expr2, etc. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters. For instance, the Process parameter can be configured as: *iplanet:*Xms**, where *iplanet* is the name that will be displayed in the eG monitor interface, and **Xms** is the process pattern that needs to be monitored. **Xms** will monitor only those processes which contain the string "Xms". Multiple processes can be defined as a comma-separated list.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Rlimit fd current	The current limit of file descriptors associated with a particular application. Each application has a limit on the number of file descriptors which it can use. The default is 256.	Number	
Number of file descriptors open	The number of file descriptors which are open at present	Number	A consistent increase in the value of this measure over a period of time could indicate that file handles are not being released properly. If the problem is not addressed soon, it could seriously hamper application performance.
Usage of file descriptors	The percentage usage of the file descriptors for the application	Percent	

3.4.13 WebLogic Connectors Test

This test extracts connector related statistics from a managed WebLogic server. 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 *Oracle WebLogic* as 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 WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every WebLogic server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can

Parameter	Description
	be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Connection created	Returns the total number	Conns/sec	

Measurement	Description	Measurement Unit	Interpretation
	of connector connections created in this connector pool since the pool was instantiated		
Connections destroyed	Returns the total number of connector connections destroyed in this connector pool since the pool was instantiated	Conns/sec	
Connections matched	Returns the total number of times a request for a connector connection was satisfied via the use of an existing created connection, since the pool was instantiated	Conns/Sec	
Connection rejects	Returns the total number of rejected requests for a connector connection in this connector pool since the pool was instantiated	Conns/Sec	
Connections recycled	Returns the total number of connector connections that have been recycled in this connector pool since the pool was instantiated	Conns/Sec	
Current active connections	Returns the current total number of active connections	Number	
Active connections high water mark	Returns the high water mark of active connections in this connector pool since the pool was instantiated	Number	
Current free	Returns the current total	Number	

Measurement	Description	Measurement Unit	Interpretation
connections	free connections		
Free connections high water mark	Returns the high water mark of free connections in this connector pool since the pool was instantiated	Number	
Max capacity	Returns the maximum capacity configured for this connector connection pool	Number	

3.4.14 Web Service Test

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. A complete web service is, therefore, any service that:

- Is available over the Internet or private (intranet) networks
- Uses a standardized XML messaging system
- Is not tied to any one operating system or programming language
- Is self-describing via a common XML grammar
- Is discoverable via a simple find mechanism

The basic web services platform is XML + HTTP. All the standard web services work using the following components:

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of the following:

- XML to tag the data
- SOAP to transfer a message

- WSDL to describe the availability of service.

The following are the major uses of the Web Services:

- **Reusable application-components**: Often applications need repeated access to application-components like currency conversion, weather reports, or even language translation. In such cases, the web services can be used to offer the application-components as services with ease.
- **Connect existing software**: Web services can help to solve the interoperability problem by giving different applications a way to link their data. With Web services you can exchange data between different applications and different platforms. Any application can have a Web Service component. Web Services can be created regardless of programming language.

In certain environments, administrators are required to keep an eye on the web services that offer repeated access to the application-components i.e., operations so that the work load on the users using those applicaiton components can be minimized. If for some reason the web service takes too long to respond or is unavailable to cater to the needs of the users, then the users will be deprived of access to the application-components involved in that particular web service. To avoid such inconvenience caused to the users, administrators are required to continuously monitor the web services. The **Web Service** test helps administrators to perform this task perfectly. By continuously monitoring each operation i.e., application component of a web service that is offered, using the SOAP commands, this test helps administrators identify the availability, response time and response code of the web service and quickly figure out discrepancies if any web service is deemed unavailable. This way, the web services can be kept available round the clock thus helping the users perform their tasks without any difficulty.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each *WebService:Operation* i.e., application-component performed on the target server that is being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port number at which the specified host listens.

Parameter	Description
WSDL URL	This test emulates a user accessing a specific web service(s) on the target server to determine the availability and responsiveness of the server. to enable this emulation, you need to configure the test with the url of the web service that it should access. specify this url against the WSDL URL parameter. if required, you can even configure multiple WSDL URLs - one each for every web service that the test should attempt to access. if each WSDL URL configured requires special permissions for logging in, then, you need to configure the test with separate credentials for logging into every WSDL URL. likewise, you need to provide instructions to the test on how to validate the content returned by every WSDL URL, and also set an encoding format for each wsdl url. to enable administrators to easily configure the above per WSDL URL, eg enterprise provides a special interface. to access this interface, click on the encircled '+' button alongside the url text box in the test configuration page. alternatively, you can even click on the encircled '+' button adjacent to the WSDL URL parameter in the test configuration page. to know how to use this special interface, refer to Section **3.4.14.1**.
Operations	Once the WSDL URL(s) are specified, the operations that are offered by the web services and those that are to be monitored have to be configured. To select the required operations for monitoring, eG Enterprise provides a special interface. TO access this interface, click on the encircled '+' button alongside the Operations text box in the test configuration page. Alternatively, you can even click on the encircled '+' button adjacent to the Operations parameter in the test configuration page. To know how to use this special interface, refer to Section **3.4.14.2**.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from the server. If there is no response from the server beyond the configured duration, the test will timeout. By default, this is set to *30* seconds.
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

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td>WSDL URL availability</td><td>Indicates whether the web service was able to respond successfully to the query made by the test.</td><td>Percent</td><td>Availability failures could be caused by several factors such as the web service process(es) being down, the web service being misconfigured, a network failure, etc. Temporary unavailability may also occur if the web service is overloaded. Availability is determined based on the response code returned by the service. A response code between 200 to 300 indicates that the service is available.</td></tr>
<tr><td>WSDL response time</td><td>Indicates the time taken by the eG agent to get the configured web service.</td><td>Secs</td><td>Response time being high denotes a problem. Poor response times may be due to the service being overloaded or misconfigured. If the URL accessed involves the generation of dynamic content by the service, backend problems (e.g., an overload at the application server or a database failure) can also result in an increase in response time.</td></tr>
<tr><td>Port status</td><td>Indicates whether/not the port of the web server is reachable.</td><td></td><td>The values reported by this measure and the corresponding numeric equivalents are listed in the table below:
<table><tr><th>Measure Values</th><th>Numeric Values</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>0</td></tr></table>
Note:
By default, this measure reports the above-mentioned Measure Values to indicate whether the server has been rebooted or not. In the graph of this</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
			measure however, the Measure Values are represented using the numeric equivalents only.
TCP connection availability	Indicates whether the test managed to establish a TCP connection to the server.	Percent	Failure to establish a TCP connection may imply that either the web server process is not up, or that the process is not operating correctly. In some cases of extreme overload, the failure to establish a TCP connection may be a transient condition. As the load subsides, the server may start functioning properly again.
TCP connect time	This measure quantifies the time for establishing a TCP connection to the web server host.	Secs	Typically, the TCP connection establishment must be very small (of the order of a few milliseconds). Since TCP connection establishment is handled at the OS-level, rather than by the application, an increase in this value signifies a system-level bottleneck on the host that supports the web server.
Server response time	Indicates the time period between when the connection was established and when the web server sent back a response header to the client.	Secs	While the total response time may depend on several factors, this measure is typically, a very good indicator of a server bottleneck (e.g., because all the available server threads or processes are in use).
Response code	The response code returned by the web server for the simulated request.	Number	A value between 200 and 300 indicates a good response. A 4xx value indicates a problem with the requested content (eg., page not found). A 5xx value indicates a server error.
Service availability	Indicates whether/not the web service is available.	Percent	A value of 100 indicates that the web service is available and a value of 0 indicates that the web service is not available.

Measurement	Description	Measurement Unit	Interpretation
Operation status	Indicates whether/not the configured operation is present in the web service.		This measure will not report metrics if the OPERATION parameter in the test configuration page is none in the test configuration page.
Operation Content length	Indicates the response code returned by the server for the simulated request.	Number	A value between 200 and 300 indicates a good response. A 4xx value indicates a problem with the requested content (e.g., page not found). A 5xx value indicates a server error. This measure will not report metrics if the Operation parameter in the test configuration page is none or if an invalid Value is specified or if the Value is not specified in the HTML View tab while configuring the operation for monitoring in the test configuration page.
Operation Content validity	This measure validates whether the operation was successful in executing the request made to it.	Percent	A value of 100% indicates that the content returned by the test is valid. A value of 0% indicates that the content may not be valid. This capability for content validation is especially important for multi-tier web applications. For example, a user may not be able to login to the web site but the server may reply back with a valid HTML page where in the error message, say, “Invalid Login” is reported. In this case, the availability will be 100 % (since we got a valid HTML response). If the test is configured such that the content parameter should exclude the string “Invalid Login”, in the above scenario content validity would have a value 0. This measure will not report metrics if the OPERATION parameter in the test configuration page is none or if an

Measurement	Description	Measurement Unit	Interpretation
			invalid Value is specified or if the Value is not specified in the HTML View tab while configuring the operation for monitoring in the test configuration page.
Operation execution time	Indicates the time taken to invoke the configured operation in the web service.	Secs	This measure will not report metrics if the Operation parameter in the test configuration page is none or if an invalid Value is specified or if the Value is not specified in the HTML View tab while configuring the operation for monitoring in the test configuration page.

3.4.14.1 Configuring Multiple WSDL URLs for Monitoring

In order to enable the eG agent to connect to multiple WSDL URLs and pull out the required metrics from them, the eG administrative interface provides a special page using which different WSDL URLs and their corresponding operations that need to be monitored can be specified. To configure the WSDL URLs, do the following:

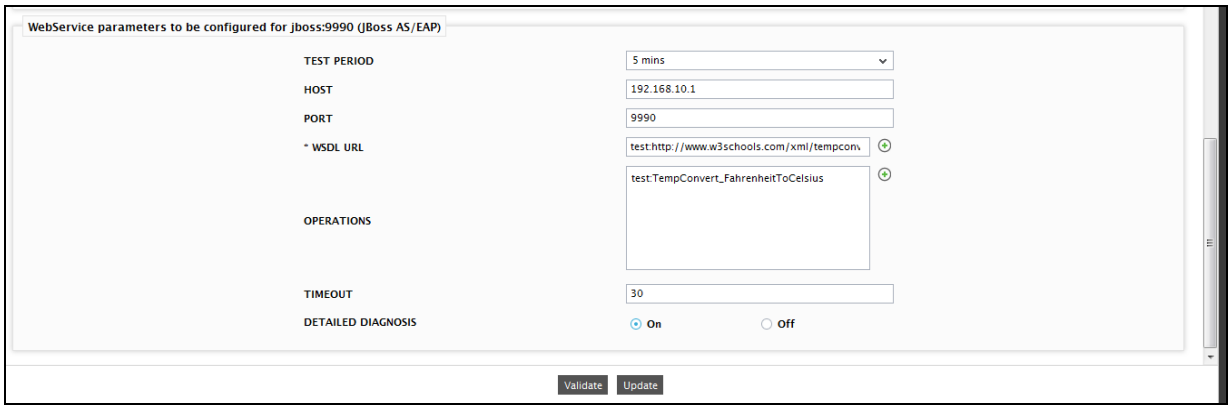


Figure 3.16: Configuring the WebService test

1. Click on the encircled '+' button alongside the **WSDL URL** text box in the . will then appear.

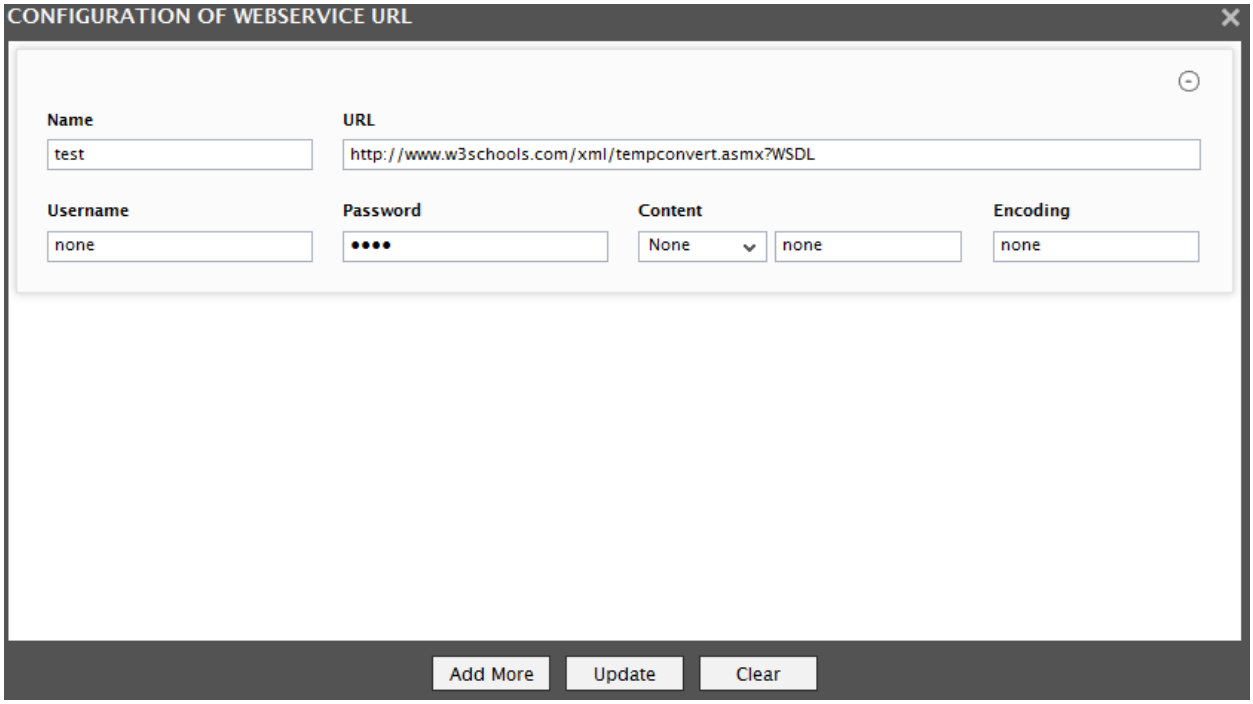


Figure 3.17: The WebService URL Configuration page

2. Specify the following in Figure 3.17:

   - **Name:** Specify a unique name by which the WSDL URL you will be specifiying shortly will be referred to across the eG user interface. This is the name that will appear as the descriptor of this test.
   - **URL:** Enter the WSDL URL of the web service that this test should access.
   - **Username** and **Password:** These parameters are to be set only if a specific user name / password has to be specified to login to the web service (i.e., WSDL URL) that you have configured for monitoring. In this case, provide valid login credentials using the **Username** and **Password** text boxes. If the server on which **WebService** test executes supports 'Anonymous user access', then these parameters will take either of the following values:

     - A valid **Username** and **Password** for the configured **WSDL URL**
     - *none* in both the **Username** and **Password** text boxes of the configured WSDL URL, if no user authorization is required
     - Some servers however, support NTLM (Integrated Windows) authentication, where valid login credentials are mandatory. In other words, a *none* specification will not be

supported by such servers. Therefore, in this case, against each configured WSDL URL, you will have to provide a valid **Username** in the format: *domainname**username*, followed by a valid **Password**.

- Please be sure to check if your web service requires HTTP authentication while configuring this parameter. HTTP authentication typically involves a separate pop-up window when you try to access the page. Many services use HTTP POST for obtaining the user name and password and validating the user login. In such cases, the username and password have to be provided as part of the POST information and NOT as part of the CREDENTIALS specification for the WebService test.

➢ **Content**: The **Content** parameter has to be configured with an instruction:value pair that will be used to validate the content being returned by the test. If the **Content** value is *None*, no validation is performed. On the other hand, if you pick the *Include* option from the **Content** list, it indicates to the test that for the content returned by the web server to be valid, the content must include the specified value (a simple string search is done in this case). This value should be specified in the adjacent text box. Similarly, if the *Exclude* option is chosen from the **Content** drop-down, it indicates to the test that the server's output is valid if it does not contain the value specified in the adjacent text box. The *Include* or *Exclude* value you specify in the text box can include wildcard characters. For example, an *Include* instruction can be **Home page**.

➢ **Encoding**: Sometimes the eG agent has to parse the WSDL URL content with specific encoding other than the default (ISO-8859-1) encoding. In such a case, specify the type of encoding using which the eG agent can parse the WSDL URL content in the **Encoding** text box. By default, this value is *none*.

3. Similarly, you can add multiple URL specifications by clicking the **Add More** button. To remove a WSDL URL specification, click on the encircled '-' button corresponding to it. To clear all WSDL URL specifications, click the **Clear** button. To update all the changes you made, click the **Update** button.
4. Once **Update** is clicked, you will return to the test configuration page as shown in Figure 3.16. The **WSDL URL** text box in the test configuration page will display just the **Name**s - i.e., the unique display names - that you may have configured for the multiple WSDL URLs, as a comma-separated list. To view the complete WSDL URL specification, click the encircled '+' button alongside the **WSDL URL** text box, once again.

3.4.14.2 Configuring Multiple Operations for Monitoring - WebServiceTest

By default, the **WebServiceTest** test will be configured with the WSDL URLs that offer the web services that are to be monitored. To configure the operations that are offered by the WSDL URLs,

do the following:

1. Click on the encircled '+' button alongside the **OPERATIONS** text box as shown in Figure 3.16. Figure 3.17 will then appear.

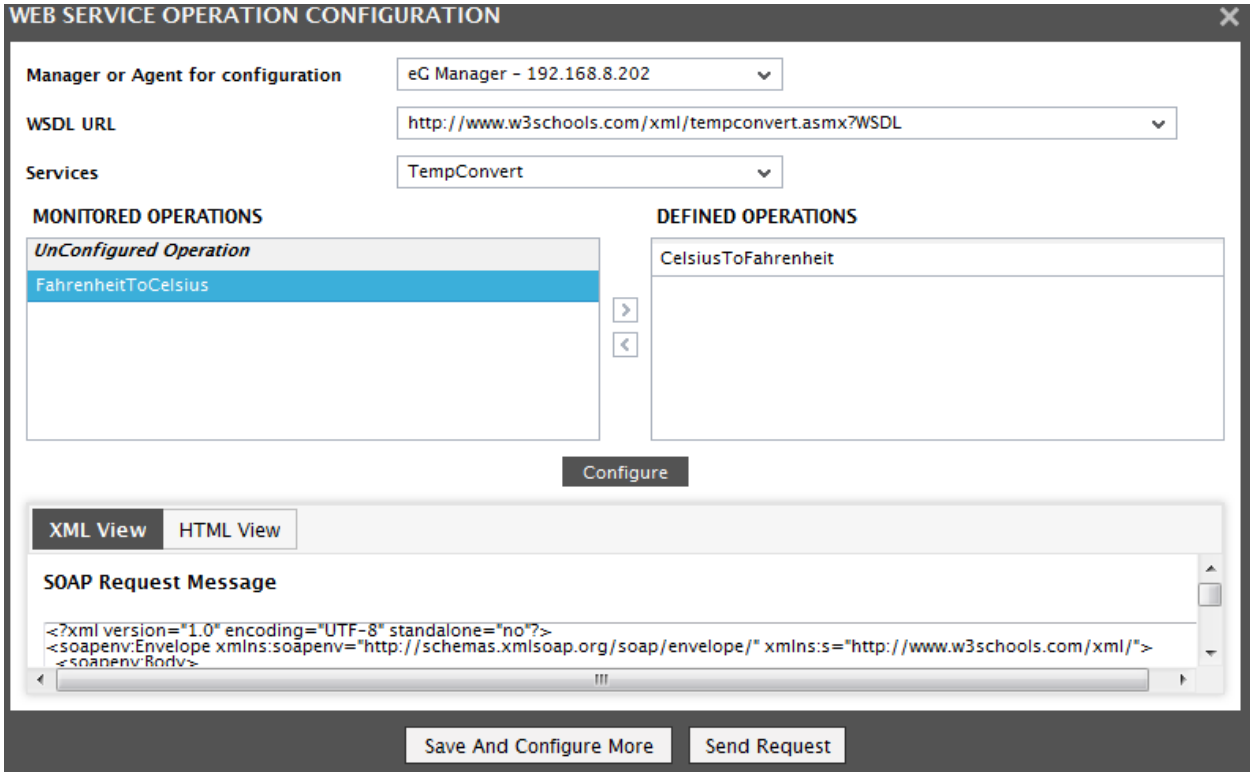


Figure 3.18: Configuring the Web Service Operation

2. Specify the following in Figure 3.18:

   - **Manager/Agent for accessing WSDL URL:** Select the eG agent or the eG Manager that is authorized to access the configured WSDL URL from this list.
   - **WSDL URL:** Once the eG agent/eG Manager is chosen from the Manager/Agent for accessing WSDL URL list, this list will be populated automatically with all the WSDL URLs specified in the WSDL URL text box (see Figure 3.17). Select the WSDL URL of your choice from this list.
   - **Services:** The web services offered by the chosen WSDL URL will then be populated in this list. Select a service of your choice from this list.
     - The operations that are offered by the chosen service will then be populated in the **DEFINED OPERATIONS** list. To monitor a chosen operation, select the operation and click the **<** button. This will move the chosen operation to the **MONITORED**

OPERATIONS list.

- Click the Configure button to save the changes.
- The eG agent uses SOAP requests to obtain the necessary metrics from the web service. Once the operation is configured, the XML View of the SOAP Request corresponding to the chosen operation will be generated and listed in the **XML View** tab. Likewise, the **HTML View** tab lists the **SOAP Parameter** that is passed to collect the required metrics for the chosen operation.
- To obtain operation-level statistics, it is important to specify a valid value in the VALUE text box of the HTML View tab as shown in Figure 3.19. Each time the test is executed, this value will be provided as an input to the chosen operation.

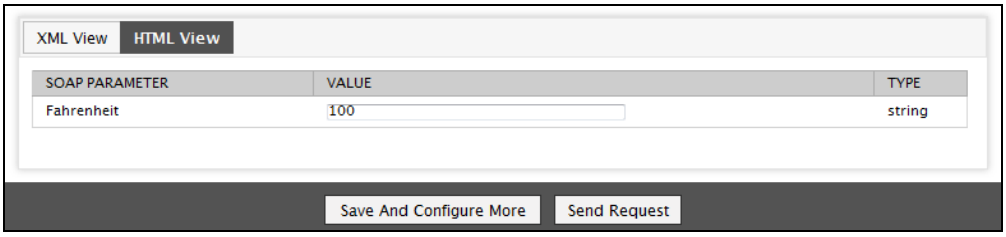


Figure 3.19: Specifying the value for the chosen operation

- Click the Save and Configure More button to save the changes made.
- If you wish to verify if the **VALUE** specified in the **HTML View** tab is valid, then you can do so by clicking the **Send Request** button. Figure 3.19 will then appear. If the value specified in the **VALUE** text box is indeed valid, then the operation will be performed on the value and the result will be specified. For example, if your chosen operation is *FahrenheittoCelsius*, the **SOAP Parameter** is *Farenheit* and the value that you wish to convert is *100*, the result will be specified in the **WEB SERVICE RESPONSE** pop up window as below: *<FahrenheitToCelsiusResult>37.7777777777778</FahrenheitToCelsiusResult>*

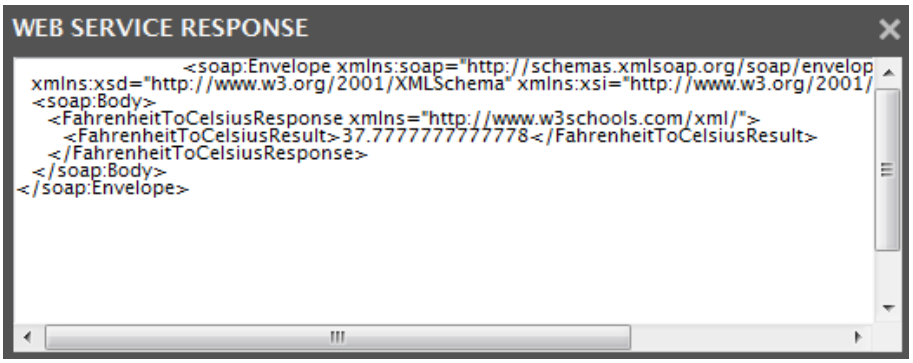


Figure 3.20: The value that appears when the operation is performed successfully

- If you have specified an invalid value, then a message as follows will be displayed in the pop up window: *<FahrenheitToCelsiusResult>Error</FahrenheitToCelsiusResult>*

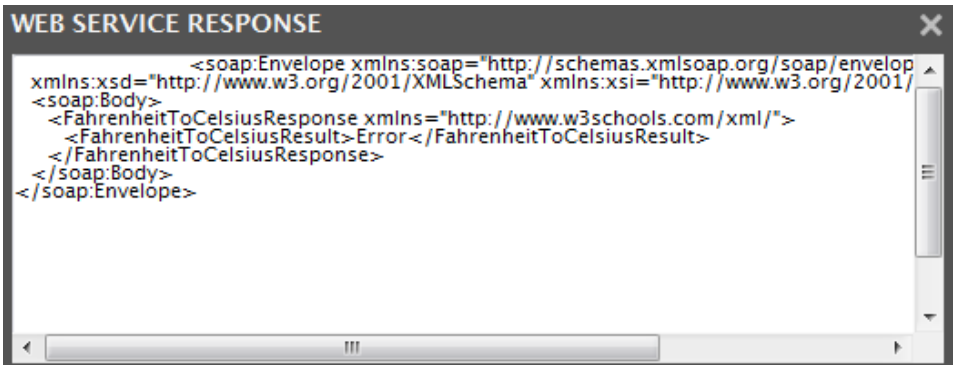


Figure 3.21: An Error appearing during value conversion

- If you do not specify a **VALUE** or specify an invalid value, operation-level statistics will not be collected by the eG agent and such metrics will not be available in the eG monitoring interface.

➢ Similarly, you can configure multiple Operations by clicking the **Configure** button in Figure 3.18. To remove an operation, select the operation from the **MONITORED OPERATION** list and click the **>** button.

Once **Save and Configure More** button is clicked, you will return to the test configuration page (see Figure 3.18). The Operations text box in the test configuration page will display just the operations, as a comma-separated list. To view the complete operation specification, click the encircled '+' button alongside the Operations text box, once again.

3.5 The WebLogic Databases Layer

The status of the database connectivity from the WebLogic server to one or more backend database servers is assessed by the **WebLogic Databases** layer. The status of the **WebLogic Databases** layer is determined based on the results of a WebLogicJdbc test that is shown in Figure 3.22.

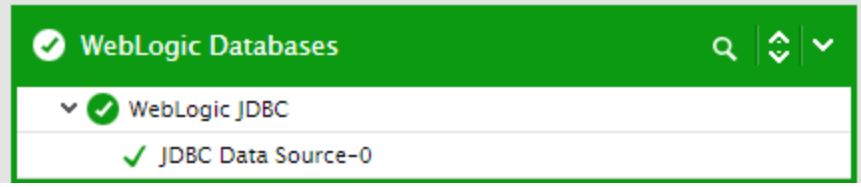


Figure 3.22: Tests mapping to the WebLogic Databases layer

3.5.1 WebLogic JDBC Test

This test collects measures related to JDBC pools created on the WebLogic server.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each connection pool used by a WebLogic application server.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
SNMPPort	The port number on which the WebLogic server is exposing its SNMP MIB (relevant to WebLogic server 5.1 only). For version 6.0 and above, enter "*none*" in this text box.
SNMPVersion	By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the SNMPversion list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list.
SNMPCommunity	The SNMP community name that the test uses to communicate with the firewall. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the SNMPVersion chosen is **v3**, then this parameter will not appear.

Parameter	Description
UserName	This parameter appears only when **v3** is selected as the SNMPVersion. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against this parameter.
Context	This parameter appears only when v3 is selected as the SNMPVersion. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the SNMPEngineID value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a contextName). If the Username provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the Username in the Context text box. By default, this parameter is set to *none*.
AuthPass	Specify the password that corresponds to the above-mentioned Username. This parameter once again appears only if the SNMPVersion selected is **v3**.
Confirm Password	Confirm the AuthPass by retyping it here.
AuthType	This parameter too appears only if **v3** is selected as the SNMPVersion. From the Authtype list box, choose the authentication algorithm using which SNMP v3 converts the specified username and password into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm
EncryptFlag	This flag appears only when **v3** is selected as the SNMPVersion. By default, the eG agent does not encrypt SNMP requests. Accordingly, the this flag is set to **No** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **Yes** option.
EncryptType	If this EncryptFlag is set to **Yes**, then you will have to mention the encryption type by selecting an option from the EncryptType list. SNMP v3 supports the following encryption types:

Parameter	Description
	• **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard
EncryptPassword	Specify the encryption password here.
Confirm Password	Confirm the encryption password by retyping it here.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
Timeout	Specify the duration (in seconds) within which the SNMP query executed by this test should time out in this text box. The default is 10 seconds.
Data Over TCP	By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic – for instance, certain types of data traffic or traffic pertaining to specific components – to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the monitored target over TCP (and not UDP). For this, set this flag to **Yes**. By default, this flag is set to **No**.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration

Parameter	Description
	to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Pool availability	The current state of a database connection pool – whether it is available or not	Percent	A value of 100 denotes that the pool is available; a value of 0 denotes unavailability.

Measurement	Description	Measurement Unit	Interpretation
Percent of connections used	Percentage of database connection allocated to a connection pool that are in use	Percent	When this value reaches 100%, connections to the database will either have to wait for access or will time out. Consider increasing the size of the connection pool in this case. A very high percentage of use can also result if one or more of the applications that use the connection pool are not releasing the connections after their use.
Max capacity	The maximum number of connections configured for a JDBC connection pool	Number	
Active connections current	The number of connections currently in use for a JDBC connection pool	Number	
Waiting for connections	The number of requests for connections to the database that are currently pending	Number	A high value of pending connections is indicative of a bottleneck during database access. Reasons for this could either be that the connection pool capacity is insufficient, or that the database server has slowed down, causing requests to take longer to execute their queries.
Active connections max	The high water mark of active connections in a pool	Number	Note the changes in the high water mark. This can give you an idea about the times when the JDBC pool was most heavily used.
Waiting for connections max	The high water mark of number of waiters for a connection from the pool	Number	Note the changes in the high water mark. This indicates periods when the database connection pool could have been a bottleneck.
Connections added to pool	The number of JDBC connections added to the pool in the last measurement period	Number	

Measurement	Description	Measurement Unit	Interpretation
Leaked connections	The number of connections tagged as a leaked connection during the last measurement period. A leaked connection is a connection that was checked out from the connection pool but was not returned to the pool by calling close().	Number	A non-zero value indicates that the application may not be releasing connections after use. This can result in inefficient management of the database connections in the pool.
Failures to reconnect	The number of cases during the last measurement period when a connection pool attempted to refresh a connection to the database and failed.	Number	Failures could happen if the database is unavailable, or, the connection was terminated.
Connection delay time	Avg. time taken to get a connection from the database (in seconds)	Secs	An increase in this metric could indicate a bottleneck in the database tier.
Max wait to get connection	The high water mark of the time a thread had to wait in order to get a connection from the pool	Secs	

3.5.2 WL JDBC Test

This test also measures statistics pertaining to the JDBC database connection pools in use by the WebLogic server. This test is disabled by default and is included for backward compatibility only. 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 *Oracle WebLogic* as 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 WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each connection pool used by a WebLogic application server.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
SNMPPort	The port number on which the WebLogic server is exposing its SNMP MIB (relevant to WebLogic server 5.1 only). For version 6.0 and above, enter "*none*" in this text box.
SNMPVersion	By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the SNMPversion list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list.
SNMPCommunity	The SNMP community name that the test uses to communicate with the firewall. This parameter is specific to SNMP **v1** and **v2** only. Therefore, if the SNMPVersion chosen is **v3**, then this parameter will not appear.
UserName	This parameter appears only when **v3** is selected as the SNMPVersion. SNMP version 3 (SNMPv3) is an extensible SNMP Framework which supplements the SNMPv2 Framework, by additionally supporting message security, access control, and remote SNMP configuration capabilities. To extract performance statistics from the MIB using the highly secure SNMP v3 protocol, the eG agent has to be configured with the required access privileges – in other words, the eG agent should connect to the MIB using the credentials of a user with access permissions to be MIB. Therefore, specify the name of such a user against this parameter.
Context	This parameter appears only when v3 is selected as the SNMPVersion. An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context and an SNMP entity potentially has access to many contexts. A context is identified by the SNMPEngineID value of the entity hosting the management information (also called a contextEngineID) and a context name that identifies the specific context (also called a contextName). If the Username provided is associated with a context name, then the eG agent will be able to poll the MIB and collect metrics only if it is configured with the context name as well. In such cases therefore, specify the context name of the Username in the Context text box. By default, this parameter is set to *none*.
AuthPass	Specify the password that corresponds to the above-mentioned Username. This

Parameter	Description
	parameter once again appears only if the SNMPVersion selected is **v3**.
Confirm Password	Confirm the AuthPass by retyping it here.
AuthType	This parameter too appears only if **v3** is selected as the SNMPVersion. From the Authtype list box, choose the authentication algorithm using which SNMP v3 converts the specified username and password into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • **MD5** – Message Digest Algorithm • **SHA** – Secure Hash Algorithm
EncryptFlag	This flag appears only when **v3** is selected as the SNMPVersion. By default, the eG agent does not encrypt SNMP requests. Accordingly, the this flag is set to **No** by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the **Yes** option.
EncryptType	If this EncryptFlag is set to **Yes**, then you will have to mention the encryption type by selecting an option from the EncryptType list. SNMP v3 supports the following encryption types: • **DES** – Data Encryption Standard • **AES** – Advanced Encryption Standard
EncryptPassword	Specify the encryption password here.
Confirm Password	Confirm the encryption password by retyping it here.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
Timeout	Specify the duration (in seconds) within which the SNMP query executed by this test should time out in this text box. The default is 10 seconds.
Data Over TCP	By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic – for instance, certain types of data traffic or traffic pertaining to specific components – to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the monitored target over TCP (and not UDP). For this, set this flag to **Yes**. By default, this flag is set to **No**.

Parameter	Description
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well.

Parameter	Description
	When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Connection usage percent	Percentage of database connections allocated to a connection pool that is in use	Percent	When this value reaches 100%, connections to the database will either have to wait for access or will time out. Consider increasing the size of the connection pool in this case. A very high percentage of use can also result if one or more of the applications that use the connection pool are not releasing the connections after their use.
Pending connections	Number of requests for connections to the database that are currently pending	Number	A high value of pending connections is indicative of a bottleneck during database access. Reasons for this could either be that the connection pool capacity is insufficient, or that the database server has slowed down, causing requests to take longer to execute their queries.
Max connections	The maximum number of connections configured for a JDBC connection pool.	Number	
Current connections	The current number of connections in use for a JDBC connection pool.	Number	

Measurement	Description	Measurement Unit	Interpretation
Pool availability	The current state of a database connection pool – whether it is available or not.	Percent	A value of 100 denotes that the pool is available; a value of 0 denotes unavailability.
Failed reconnects	The number of cases during the last measurement period when a connection pool attempted to refresh a connection to the database and failed.	Number	Failures could happen if the database is unavailable, or, the connection was terminated.
Connections to pool	The number of JDBC connections added to the pool in the last measurement period.	Number	
Avg connection delay	Avg. time taken to get a connection from the database (in seconds).	Secs	An increase in this metric could indicate a bottleneck in the database tier.
Leaked connections	Number of connections tagged as a leaked connection during the last measurement period. A leaked connection is a connection that was checked out from the connection pool but was not returned to the pool by calling close().	Number	A non-zero value indicates that the application may not be releasing connections after use. This can result in inefficient management of the database connections in the pool.
Active connections high mark	The high water mark of active connections in a pool.	Number	Note the changes in the high water mark. This can provide an idea of the times when the JDBC pool was most heavily used.
Waiting connections high mark	The high water mark of number of waiters for a connection from the pool.	Number	Note the changes in the high water mark. This indicates periods when the database connection pool could have been a bottleneck.

Note:

The *Failed reconnects*, *Avg connection delay* and *Leaked connections* measures will always report 0 for WebLogic 6.0 with Service Pack 2.0.

3.5.3 Jdbc Stats Test

This test collects measures like how many JDBC queries have been executed, how long those queries took to execute and what those statements are. This test is key to determining whether the database/database queries are the cause of any slowdowns with applications executing on the WebLogic server. 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 *Oracle WebLogic* as the **Component type**, *Performance* as the **Test type**, choose this 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 WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for a WebLogic application server.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.

Parameter	Description
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
Total query rate	The rate of JDBC queries that have been executed on the database.	Queries/Sec	This is an indicator of the JDBC workload.
Avg query time	The average time taken by the queries to execute on the database.	Secs	A higher average query time signifies a performance bottleneck at the data abase tier.
Select query rate	The rate of select queries executed on the database.	Queries/Sec	Comparing the types of queries to the database provides an idea about the nature of workload on the database
Avg Select query time	The average time taken by the select queries for executing on the database.	Secs	Unexpectedly large query times can indicate that there is sufficient scope to optimize database access using better indexes.
Insert query rate	The rate of insert queries executed on the database.	Queries/Sec	
Avg insert query time	The average time taken for executing the insert queries on the database.	Secs	
Update query rate	The rate of update queries executed on the database.	Queries/Sec	

Measurement	Description	Measurement Unit	Interpretation
Avg update query time	The average time taken for executing the update queries on the database.	Secs	
Delete query rate	The rate of delete queries executed on the database.	Queries/Sec	
Avg delete query time	The average time taken by the delete queries to execute on the database.	Secs	
Commit queries rate	The rate of commit queries executed on the database.	Queries/Sec	
Avg commit query time	The average time taken by the commit queries to execute on the database.	Secs	
Rollback queries rate	The rate of rollback queries executed on the database.	Queries/Sec	Typically, the rate of rollbacks should be low.
Avg rollback query time	The average time taken by the rollback queries to execute on the database.	Secs	
Jdbc query error rate	The rate of queries that have resulted in an error.	Errors/Sec	

3.5.4 WebLogic Applications Layer

Using the tests mapped to this layer, administrators can identify the web applications that are running on the WebLogic server, and the current session load on each web application. Additionally, the layer also monitors the EJB activity on the server and even measures the how quickly the servlets are processing requests they receive.

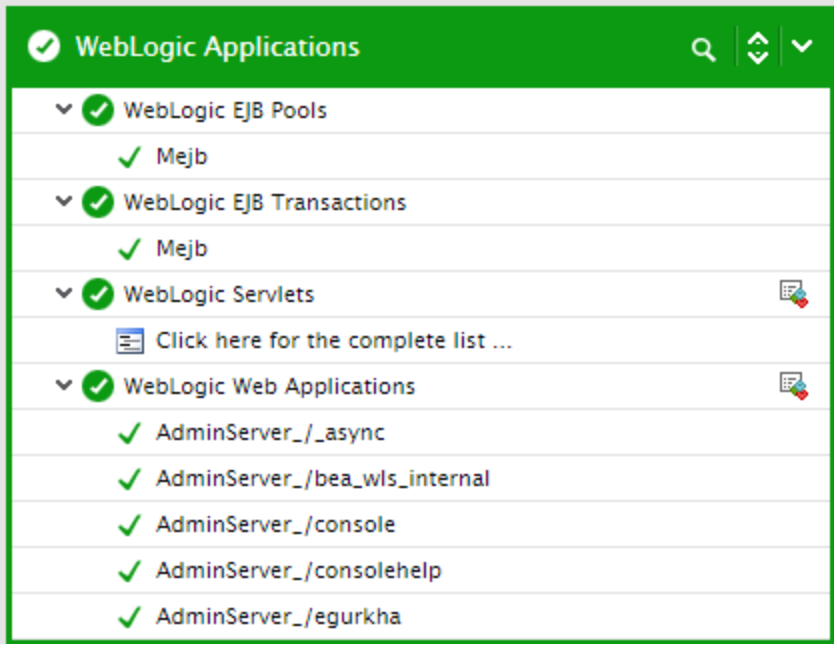


Figure 3.23: The tests mapped to the WebLogic Applications layer

3.5.5 WebLogic EJB Locks Test

The WebLogic EJB Lock test monitors the EJB locking activity performed by the WebLogic server. Use the Click here hyperlink in the test configuration page to configure the EJB groups that need to be monitored by the eG Enterprise suite. By default, the eG Enterprise system will monitor only those EJBs that are part of a group.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every EJB group configured using the eG administrative interface.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.

Parameter	Description
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is**

Parameter	Description
	available for WebLogic servers ver.9 and above. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.
AutoDiscovery	By default, eG Enterprise allows administrators to configure EJB groups using the eG administrative interface, and reports metrics pertaining to every group so created. Accordingly, by default, AutoDiscovery is set to **No**. If you want EJBs to be discovered and monitored automatically, then select the **Yes**option against AutoDiscovery. When this is done, the eG agent automatically discovers all the EJBs on the WebLogic server, and reports one set of measures for every EJB so discovered.
ShowServerName	WebLogic version 8.0 and above prefixes the names of the EJBs it hosts, with the server name specified in the Server text box. For example, if the Server text box contains *MedRecServer*, then an EJB named MedRecEAR_EntityEJB_PatientEJB on that server will be renamed by WebLogic as MedRecServer_MedRecEAR_ EntityEJB_PatientEJB. This could pose a challenge while applying EJB groups configured (using eG) on a particular WebLogic server to another server. This is because, when the eG Enterprise suite lists the EJBs available for grouping, each of the displayed EJBs will be prefixed by the corresponding Server name. When such EJBs are grouped and the group is then applied to another WebLogic server, the eG Enterprise system will not be able to extract metrics for the EJB grouping applied to the new server due to the server name mismatch. Therefore, by default, the eG Enterprise system sets the ShowServerName flag to **No**. This ensures that the server name specified in the Server text box does not prefix the EJB listing during EJB group configuration. To enable the prefix, select the **Yes** option against ShowServerName. **Note:** If you modify the ShowServerName setting after configuring the EJB groups, then wait for the test to run once so that the eG agent rediscovers the EJBs on that server. Then, delete the existing EJB groups, and repeat the group configuration.

Parameter	Description
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
Locked beans count	The sum total of the beans from within an EJB group that are currently locked	Number	
Attempts to lock rate	The sum total of the rate at which attempts were made to obtain a lock on every bean within an EJB group	Attempts/Sec	
Timeout rate	The sum total of the rate at which threads have timed out waiting for a lock on every bean within an EJB group	Timeouts/Sec	
Threads waiting rate	The sum total of the rate at which threads have been waiting for a lock on every bean in an EJB group	Threads/Sec	The detailed diagnosis of this measure, if enabled, provides locking information pertaining to the individual beans in the monitored EJB group. The details displayed include: The Bean name, the number of locked beans, the number of attempts made to lock, the timeout rate, and the threads waiting

Measurement	Description	Measurement Unit	Interpretation
			rate of the bean. **Note that the detailed diagnosis information will not be available if the AutoDiscovery parameter is set to 'Yes'**.

3.5.6 WebLogic Servlets Test

This test periodically tracks the servlets invoked and reloaded, and measures the minimum, maximum, and average response times of a specific server instance. Use the **Click here** hyperlink in the test configuration page to configure the servlet groups that need to be monitored by the eG Enterprise suite. By default, eG Enterprise system monitors only those servlets that are part of a group.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every servlet group configured.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.

Parameter	Description
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-**

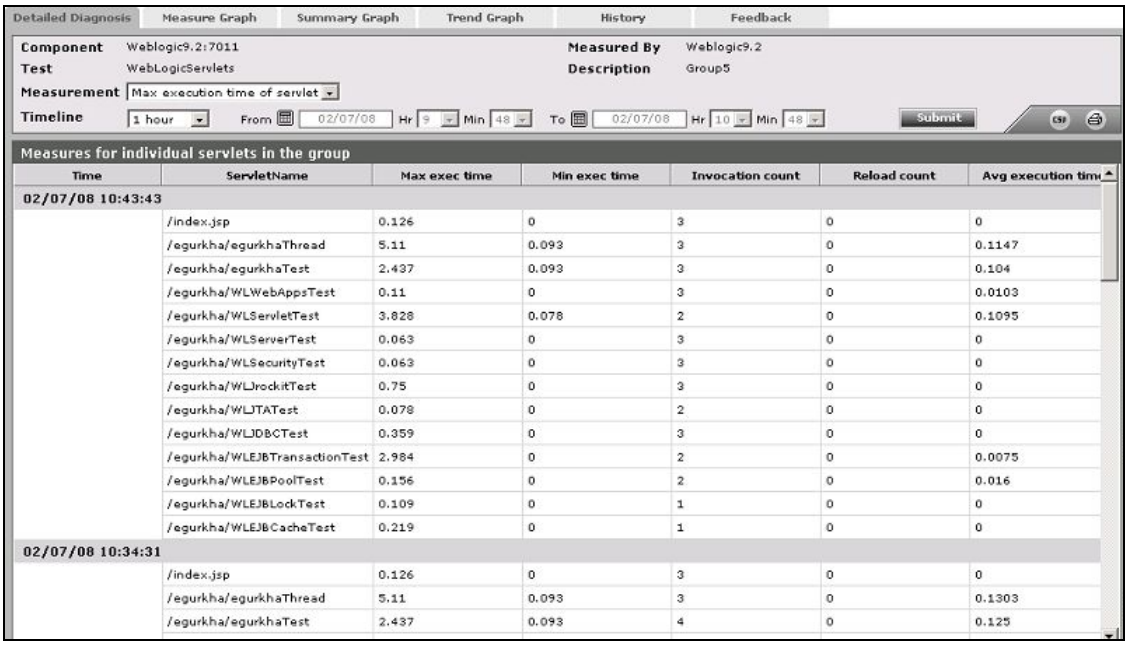
Parameter	Description
	based support is available for WebLogic servers ver.9 and above.
AutoDiscovery	By default, eG Enterprise allows administrators to configure EJB groups using the eG administrative interface, and reports metrics pertaining to every group so created. Accordingly, by default, AutoDiscovery is set to **No**. If you want EJBs to be discovered and monitored automatically, then select the **Yes**option against AutoDiscovery. When this is done, the eG agent automatically discovers all the EJBs on the WebLogic server, and reports one set of measures for every EJB so discovered.
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
Max execution time of servlet	The average duration for which the single longest invocation of all the servlets within a servlet group executed since creation	Number	
Min execution time of servlet	The average duration for which the single shortest invocation of all the servlets in a servlet group executed since creation	Secs	
Invocation count of servlet	The total number of times the servlets within a	Number	Comparing this value across servlets

Measurement	Description	Measurement Unit	Interpretation
	servlet group were invoked		can provide an indication of the relative popularity of the servlets. An administrator can use information about invocations to determine the servlet(s), by tuning which the performance of the service can be significantly improved.
Reloads of servlet	The total number of times for which the servlets within a servlet group were reloaded	Number	
Avg execution of servlet	The average of response time of the servlets in a servlet group	Secs	This measure indicates which servlet (s) are providing slow response. An increased execution time can be attributed to poor design of the servlet code, slow database access or non-optimal queries.

The detailed diagnosis of the all the above measures, if enabled, reveals the maximum execution time, minimum execution time, invocation count, reload count, and average execution time for every servlet in the configured servlet groups. **Note that the detailed diagnosis information will not be available if the AutoDiscovery parameter is set to 'Yes'**.



Time	ServletName	Max exec time	Min exec time	Invocation count	Reload count	Avg execution time
02/07/08 10:43:43						
	/index.jsp	0.126	0	3	0	0
	/egurkha/egurkhaThread	5.11	0.093	3	0	0.1147
	/egurkha/egurkhaTest	2.437	0.093	3	0	0.104
	/egurkha/WLWebAppsTest	0.11	0	3	0	0.0103
	/egurkha/WLServletTest	3.828	0.078	2	0	0.1095
	/egurkha/WLServerTest	0.063	0	3	0	0
	/egurkha/WLSecurityTest	0.063	0	3	0	0
	/egurkha/WLJrockitTest	0.75	0	3	0	0
	/egurkha/WLJTATest	0.078	0	2	0	0
	/egurkha/WLJDBCTest	0.359	0	3	0	0
	/egurkha/WLEJBTransactionTest	2.984	0	2	0	0.0075
	/egurkha/WLEJBPoolTest	0.156	0	2	0	0.016
	/egurkha/WLEJBLockTest	0.109	0	1	0	0
	/egurkha/WLEJBCacheTest	0.219	0	1	0	0
02/07/08 10:34:31						
	/index.jsp	0.126	0	3	0	0
	/egurkha/egurkhaThread	5.11	0.093	3	0	0.1303
	/egurkha/egurkhaTest	2.437	0.093	4	0	0.125

Figure 3.24: The detailed diagnosis of the Max execution time measure

3.5.7 WebLogic Web Applications Test

This test automatically discovers all the Web application components deployed on the WebLogic server, and reports real-time performance data pertaining to each of the components.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Web application discovered on the WebLogic server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.

Parameter	Description
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well.

Parameter	Description
	When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Sessions open current	Indicates the total number of open sessions in this component currently.	Number	
Sessions open high water mark	Returns the high water mark of the total number of open sessions in this component.	Number	
Sessions open total	Returns the total number of open sessions in this component since the last measurement period.	Number	This measure is indicative of the workload on the application component. By observing the variations to this measure over time, you can determine usage trends such as the time of day when an application is getting the most amount of traffic.
Sessions invalid time	Returns the invalidation check timer interval configured for http sessions.	Secs	

3.5.8 WebLogic EJB Cache Test

To improve the performance and the scalability of the EJB container, WebLogic server caches EJBs. The WebLogicEJBCache test collects measures related to EJB caching activity by the WebLogic server. Use the **Click here** hyperlink in the test configuration page to configure the EJB

groups that need to be monitored by the eG Enterprise suite. **By default, the eG Enterprise system will monitor only those EJBs that are part of a group.**

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every EJB group configured using the eG administrative interface.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use.

Parameter	Description
	Note: If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.
AutoDiscovery	By default, eG Enterprise allows administrators to configure EJB groups using the eG administrative interface, and reports metrics pertaining to every group so created. Accordingly, by default, AutoDiscovery is set to **No**. If you want EJBs to be discovered and monitored automatically, then select the **Yes**option against AutoDiscovery. When this is done, the eG agent automatically discovers all the EJBs on the WebLogic server, and reports one set of measures for every EJB so discovered.
ShowServerName	WebLogic version 8.0 and above prefixes the names of the EJBs it hosts, with the server name specified in the Server text box. For example, if the Server text box contains *MedRecServer*, then an EJB named MedRecEAR_EntityEJB_PatientEJB on that server will be renamed by WebLogic as MedRecServer_MedRecEAR_

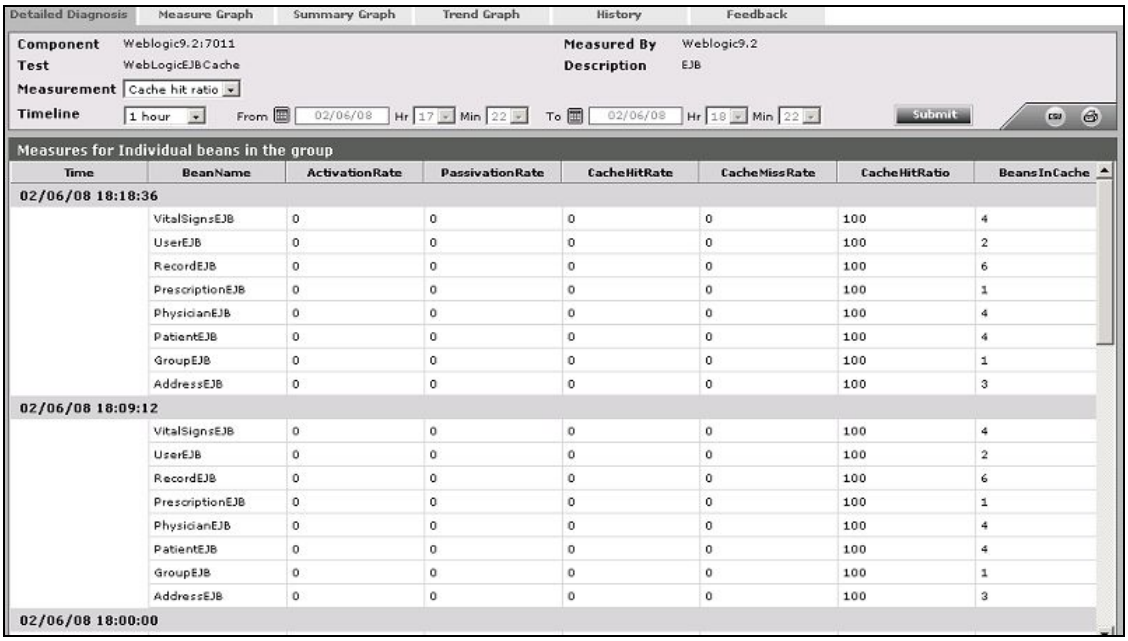
Parameter	Description
	EntityEJB_PatientEJB. This could pose a challenge while applying EJB groups configured (using eG) on a particular WebLogic server to another server. This is because, when the eG Enterprise suite lists the EJBs available for grouping, each of the displayed EJBs will be prefixed by the corresponding Server name. When such EJBs are grouped and the group is then applied to another WebLogic server, the eG Enterprise system will not be able to extract metrics for the EJB grouping applied to the new server due to the server name mismatch. Therefore, by default, the eG Enterprise system sets the ShowServerName flag to **No**. This ensures that the server name specified in the Server text box does not prefix the EJB listing during EJB group configuration. To enable the prefix, select the **Yes** option against ShowServerName. **Note:** If you modify the ShowServerName setting after configuring the EJB groups, then wait for the test to run once so that the eG agent rediscovers the EJBs on that server. Then, delete the existing EJB groups, and repeat the group configuration.
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
Activation rate	The sum total of the rate at which every bean from a particular EJB group have been activated	Beans/Sec	
Passivation rate	The sum total of the rate at which the every bean from	Beans/Sec	

Measurement	Description	Measurement Unit	Interpretation
	a specific EJB group have been passivated		
Cache hit rate	The sum total of the rate at which attempts to access every bean within an EJB group, from the cache succeeded	Hits/Sec	
Cache miss rate	The sum total of the rate at which attempts to access every bean within an EJB group, from the cache failed	Misses/Sec	
Cache hit ratio	The average of the percentage of time every bean in an EJB group was successfully accessed from the cache	Percent	
Beans in cache	The number of beans in the cache	Number	

The detailed diagnosis of the *Cache hit ratio* measure, provides the details pertaining to the EJB caching activity of the individual beans in the EJB group being monitored (see Figure 3.25). This information enables administrators to assess the effectiveness of the caching activity performed by the WebLogic server. **Note that the detailed diagnosis information will not be available if the AutoDiscovery parameter is set to 'Yes'**.



Time	BeanName	ActivationRate	PassivationRate	CacheHitRate	CacheMissRate	CacheHitRatio	BeansInCache
02/06/08 18:18:36							
	VitalSignsEJB	0	0	0	0	100	4
	UserEJB	0	0	0	0	100	2
	RecordEJB	0	0	0	0	100	6
	PrescriptionEJB	0	0	0	0	100	1
	PhysicianEJB	0	0	0	0	100	4
	PatientEJB	0	0	0	0	100	4
	GroupEJB	0	0	0	0	100	1
	AddressEJB	0	0	0	0	100	3
02/06/08 18:09:12							
	VitalSignsEJB	0	0	0	0	100	4
	UserEJB	0	0	0	0	100	2
	RecordEJB	0	0	0	0	100	6
	PrescriptionEJB	0	0	0	0	100	1
	PhysicianEJB	0	0	0	0	100	4
	PatientEJB	0	0	0	0	100	4
	GroupEJB	0	0	0	0	100	1
	AddressEJB	0	0	0	0	100	3
02/06/08 18:00:00							

Figure 3.25: The detailed diagnosis of the Cache hit ratio measure

3.5.9 WebLogic EJB Pools Test

Following are three different kinds of EJB components that can be hosted on a WebLogic server:

- A stateful session bean is an enterprise bean that acts as a server-side extension of the client that uses it. The stateful session bean is created by a client and will work for only that client until the client connection is dropped or the bean is explicitly removed.
- A stateless session bean is an enterprise bean that provides a stateless service to the client
- An entity bean represents a business object in a persistent storage mechanism such as a database. For example, an entity bean could represent a customer, which might be stored as a row in the customer table of a relational database

This test collects measures related to the bean pooling activity performed by the WebLogic server. Use the **Click here** hyperlink in the test configuration page to configure the EJB groups that need to be monitored by the eG Enterprise suite. **By default, the eG Enterprise system monitors only those EJBs that are part of a group**.

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every EJB group configured using the eG administrative interface.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The

Parameter	Description
	default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.
AutoDiscovery	By default, eG Enterprise allows administrators to configure EJB groups using the eG administrative interface, and reports metrics pertaining to every group so created. Accordingly, by default, AutoDiscovery is set to **No**. If you want EJBs to be discovered and monitored automatically, then select the **Yes**option against AutoDiscovery. When this is done, the eG agent automatically discovers all the EJBs on the WebLogic server, and reports one set of measures for every EJB so discovered.
ShowServerName	WebLogic version 8.0 and above prefixes the names of the EJBs it hosts, with the server name specified in the Server text box. For example, if the Server text box contains *MedRecServer*, then an EJB named MedRecEAR_EntityEJB_PatientEJB on that server will be renamed by WebLogic as MedRecServer_MedRecEAR_ EntityEJB_PatientEJB. This could pose a challenge while applying EJB groups configured (using eG) on a particular WebLogic server to another server. This is because, when the eG Enterprise suite lists the EJBs available for grouping, each of the displayed EJBs will be prefixed by the corresponding Server name. When such EJBs are grouped and the group is then applied to another WebLogic server, the eG Enterprise system will not be able to extract metrics for the EJB grouping applied to

Parameter	Description
	the new server due to the server name mismatch. Therefore, by default, the eG Enterprise system sets the ShowServerName flag to **No**. This ensures that the server name specified in the Server text box does not prefix the EJB listing during EJB group configuration. To enable the prefix, select the **Yes** option against ShowServerName. **Note:** If you modify the ShowServerName setting after configuring the EJB groups, then wait for the test to run once so that the eG agent rediscovers the EJBs on that server. Then, delete the existing EJB groups, and repeat the group configuration.
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
Beans in use	The sum total of instances of every bean in an EJB group that are currently being used from the free pool	Number	
Idle beans	The sum total of the instances of every bean within an EJB group that are currently available in the free pool	Number	
Waiting threads	The sum total of threads	Number	

Measurement	Description	Measurement Unit	Interpretation
	currently waiting for every available bean instance in an EJB group		
Thread timeouts	The sum total of the number of threads that have timed out waiting for an available instance of every bean in an EJB group	Threads/Sec	
Access attempts	An access attempt is an attempt made to get an instance of a bean from the free pool. The Access_ attempts measure reports the sum total of all such attempts made for every bean in an EJB group.	Number	This measure will be available on WebLogic servers 8.0 and above only.
Missed attempts	A missed attempt is a failed attempt made to get an instance of a bean from the free pool. The Missed_ attempts measure reports the sum total of all such failed attempts made for every bean in an EJB group.	Number	This measure will be available on WebLogic servers 8.0 and above only.
Destroyed beans	If an instance of a bean (in an EJB group) from a pool was destroyed due to a non-application Exception being thrown from it, then the bean is deemed destroyed. The Destroyed_beans measure reports the sum total of all the beans in an EJB group that threw a non-application Exception.	Number	This measure will be available on WebLogic servers 8.0 and above only.

The detailed diagnosis of the *Waiting threads* (see Figure 3.26) and the *Thread timeouts* measures provides the details related to the usage of the individual beans present in the EJB group being monitored. This information enables administrators to evaluate the effectiveness of the bean pooling activity of the WebLogic server. **Note that the detailed diagnosis information will not be available if the AutoDiscovery parameter is set to 'Yes'**.

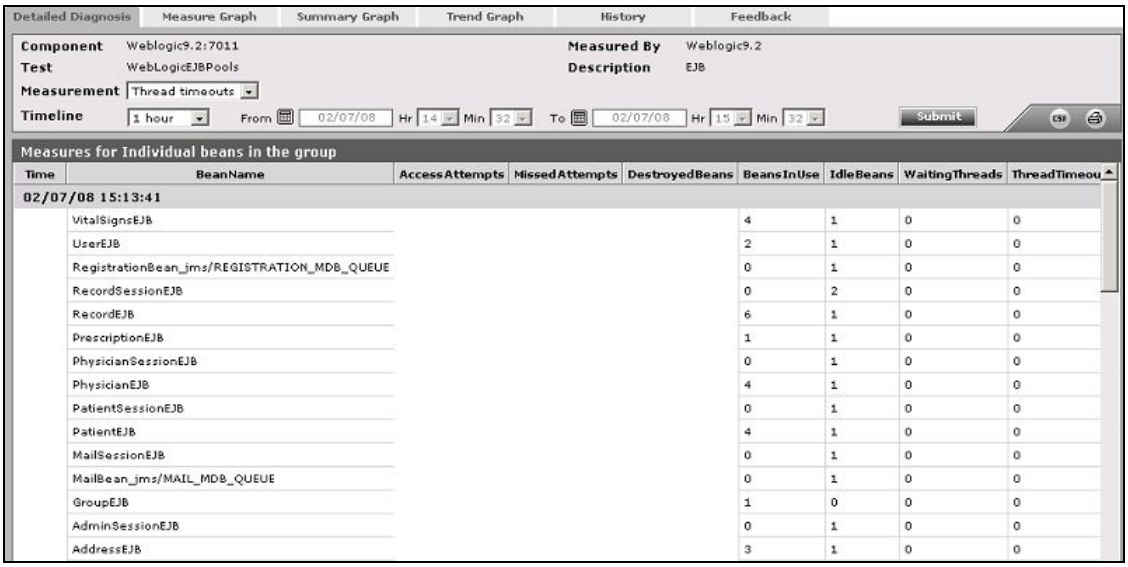


Figure 3.26: The detailed diagnosis of the Threads timeout measure

3.5.10 WebLogic EJB Pool Test

The WebLogic EJB PoolTest collects measures related to the bean pooling activity performed by the WebLogic server. This test will be disabled by default. You can enable this test by clicking on the check box corresponding to the test in the **AGENTS – TESTS CONFIGURATION** page, and then clicking on the **Update** button. Use the **Click here** hyperlink in the test configuration page to configure the EJB groups that need to be monitored by the eG Enterprise suite. **By default, the eG Enterprise system monitors only those EJBs that are part of a group.**

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every EJB group configured using the eG administrative interface.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version).

Parameter	Description
	This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.
AutoDiscovery	By default, eG Enterprise allows administrators to configure EJB groups using the eG administrative interface, and reports metrics pertaining to every group so created. Accordingly, by default, AutoDiscovery is set to **No**. If you want EJBs to be discovered and monitored automatically, then select the **Yes**option against AutoDiscovery. When this is done, the eG agent automatically discovers all the EJBs on the WebLogic server, and reports one set of measures for every EJB so discovered.
ShowServerName	WebLogic version 8.0 and above prefixes the names of the EJBs it hosts, with the server name specified in the Server text box. For example, if the Server text box contains *MedRecServer*, then an EJB named MedRecEAR_EntityEJB_PatientEJB on that server will be renamed by WebLogic as MedRecServer_MedRecEAR_ EntityEJB_PatientEJB. This could pose a challenge while applying EJB groups configured (using eG) on a particular WebLogic server to another server. This is because, when the eG Enterprise suite lists the EJBs available for grouping, each of the displayed EJBs will be prefixed by the corresponding Server name. When such EJBs are grouped and the group is then applied to another WebLogic server, the eG Enterprise system will not be able to extract metrics for the EJB grouping applied to the new server due to the server name mismatch. Therefore, by default, the eG Enterprise system sets the ShowServerName flag to **No**. This ensures that the server name specified in the Server text box does not prefix the EJB listing during

Parameter	Description
	EJB group configuration. To enable the prefix, select the **Yes** option against ShowServerName. **Note:** If you modify the ShowServerName setting after configuring the EJB groups, then wait for the test to run once so that the eG agent rediscovers the EJBs on that server. Then, delete the existing EJB groups, and repeat the group configuration.
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
Beans in use	The sum total of instances of every bean in an EJB group that are currently being used from the free pool	Number	
Idle beans	The sum total of the instances of every bean within an EJB group that are currently available in the free pool	Number	
Waiting threads	The sum total of threads currently waiting for every available bean instance in an EJB group	Number	

Measurement	Description	Measurement Unit	Interpretation
Threads timeout rate	The sum total of the rate at which threads have timed out waiting for an available instance of every bean in an EJB group	Threads/Sec	

3.5.11 WebLogic EJB Transactions Test

This test collects measures related to the EJB transaction activity performed by the WebLogic server. Use the **Click here** hyperlink in the test configuration page to configure the EJB groups that need to be monitored by the eG Enterprise suite. **By default, the eG Enterprise system monitors only those EJBs that are part of a group.**

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every EJB group configured using the eG administrative interface.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.

Parameter	Description
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-**

Parameter	Description
	based support is available for WebLogic servers ver.9 and above.
AutoDiscovery	By default, eG Enterprise allows administrators to configure EJB groups using the eG administrative interface, and reports metrics pertaining to every group so created. Accordingly, by default, AutoDiscovery is set to **No**. If you want EJBs to be discovered and monitored automatically, then select the **Yes**option against AutoDiscovery. When this is done, the eG agent automatically discovers all the EJBs on the WebLogic server, and reports one set of measures for every EJB so discovered.
ShowServerName	WebLogic version 8.0 and above prefixes the names of the EJBs it hosts, with the server name specified in the Server text box. For example, if the Server text box contains *MedRecServer*, then an EJB named MedRecEAR_EntityEJB_PatientEJB on that server will be renamed by WebLogic as MedRecServer_MedRecEAR_ EntityEJB_PatientEJB. This could pose a challenge while applying EJB groups configured (using eG) on a particular WebLogic server to another server. This is because, when the eG Enterprise suite lists the EJBs available for grouping, each of the displayed EJBs will be prefixed by the corresponding Server name. When such EJBs are grouped and the group is then applied to another WebLogic server, the eG Enterprise system will not be able to extract metrics for the EJB grouping applied to the new server due to the server name mismatch. Therefore, by default, the eG Enterprise system sets the ShowServerName flag to **No**. This ensures that the server name specified in the Server text box does not prefix the EJB listing during EJB group configuration. To enable the prefix, select the **Yes** option against ShowServerName. **Note:** If you modify the ShowServerName setting after configuring the EJB groups, then wait for the test to run once so that the eG agent rediscovers the EJBs on that server. Then, delete the existing EJB groups, and repeat the group configuration.
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

Parameter	Description
	measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Transaction commits	Commit rate is the rate at which transactions are committed for a particular bean. Tx_commit_rate reveals the sum total of the commit rates of the individual beans in an EJB group.	Trans/Sec	Comparing this value across all the deployed groups can give an idea of the relative importance of the beans within the groups, in supporting user accesses. A sudden change in user access patterns can be indicative of a change in the user workload.
Transaction rollbacks	Rollback rate is the rate at which the transactions are rolled back for a particular bean. Tx_rollback_rate reveals the sum total of the rollback rates of the individual beans in an EJB group.	Trans/Sec	A high rollback rate indicates a problem with specific beans within a group. Possible reasons for this could be problems with the design and implementation of the specific bean or problems with any of the dependent servers of the bean (e.g., database server).
Transaction timeouts	The timeout rate is the rate at which the transactions are timed out by the bean. Tx_timeout_rate reveals the sum total of the timeout rates of the individual beans in an EJB group.	Trans/Sec	A significantly high value may denote a load on the specific bean. This may indicate that specific transactions are taking too long to process requests. The detailed diagnosis of the *Transaction rollbacks* and *Transaction timeouts* measures, if enabled, provides the commit rate, rollback rate, and timeout rate of the transactions conducted by each of the beans within the EJB group being monitored. **Note that the detailed diagnosis information will not be available if the AutoDiscovery parameter is set to 'Yes'**.

3.5.12 WebLogic EJBs Test

This test monitors the state of EJB component groups hosted on a WebLogic server 6.0 or higher using JMX. 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 *Oracle WebLogic* as 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.

Use the **Click here** hyperlink in the test configuration page to configure the EJB groups that need to be monitored by the eG Enterprise suite. **Note that the eG Enterprise system monitors only those EJBs that are part of a group.**

Target of the test : A WebLogic Application Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every EJB group configured.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port at which the specified host listens. By default, this is *NULL*.
User	The admin user name of the WebLogic server being monitored.
Password	The password of the specified admin user.
Confirm Password	Confirm the password by retyping it here.
EncryptPass	If the specified password needs to be encrypted, set the EncryptPass flag to **Yes**. Otherwise, set it to **No**. By default, the **Yes** option will be selected. **Note:** If the UseWarFile flag is set to **No**, then make sure that the EncryptPass flag is also set to **No**.
SSL	Indicate whether the SSL (Secured Socket Layer) is to be used to connect to the WebLogic server.
Server	The name of the specific server instance to be monitored for a WebLogic server (the

Parameter	Description
	default value is "localhome")
URL	The URL to be accessed to collect metrics pertaining to the WebLogic server. By default, this test connects to a managed WebLogic server and attempts to obtain the metrics of interest by accessing the local Mbeans of the server. This parameter can be changed to a value of *http://<adminserverIP>:<adminserverPort>*. In this case, the test connects to the WebLogic admin server to collect metrics pertaining to the managed server (specified by the Host and Port). The URL setting provides the administrator with the flexibility of determining the WebLogic monitoring configuration to use. **Note:** If the admin server is to be used for collecting measures for all the managed WebLogic servers, then it is mandatory that the **egurkha war file** is deployed to the admin server, and it is up and running.
Version	The Version text box indicates the version of the Weblogic server to be managed. The default value is "*none*", in which case the test auto-discovers the weblogic version. If the value of this parameter is not "*none*", the test uses the value provided (e.g., 7.0) as the weblogic version (i.e., it does not auto-discover the weblogic server version). This parameter has been added to address cases when the eG agent is not able to discover the WebLogic server version.
UseWarFile	This flag indicates whether/not monitoring is to be done using a Web archive file deployed on the WebLogic server (in which case, HTTP/HTTPS is used by the server to connect to the server). If this flag is set to **No**, the agent directly connects to the WebLogic server using the T3 protocol (no other file needs to be deployed on the WebLogic server for this to work). **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**. Also, if the UseWarFile parameter is set to **No**, make sure that the EncryptPass parameter is set to **No** as well. When monitoring a WebLogic server deployed on a Unix platform particularly, if the UseWarFile parameter is set to **No**, you have to make sure that the eG agent install user is added to the WebLogic users group.
WeblLgicJARLocation	Specify the location of the WebLogic server's java archive (Jar) file. If the UseWarFile flag is set to **No**, then the weblogic.jar file specified here is used to connect to the corresponding WebLogic server using the T3 protocol. **Note that the T3 protocol-based support is available for WebLogic servers ver.9 and above**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Transaction commit rate	Indicates the rate at which transactions are committed for a particular bean.	Trans/Sec	Comparing this value across all the deployed beans can give an idea of the relative importance of the beans in supporting user accesses. A sudden change in user access patterns can be indicative of a change in the user workload.
Transaction rollback rate	Indicates the rate at which the transactions are rolled back for a particular bean.	Trans/Sec	A high rollback rate indicates a problem with specific beans. Possible reasons for this could be problems with the design and implementation of the specific bean or problems with any of the dependent servers of the bean (e.g., database server).
Transactions inflight	Number of transactions currently in progress through a particular bean.	Number	A significantly high value may denote a load on the specific bean. This may indicate that specific transactions are taking too long to process requests.
Num waiting rate	This measure corresponds to only stateless session bean. This indicates the rate at which connections are established by the client with the instances of the bean.	Conns/Sec	A high value signifies that it is taking more time for the clients to access an instance of the bean from the pool.
Timeout rate	This measure also corresponds to the stateless session beans. The measure indicates the rate at which instances are timed out by the bean.	Conns/Sec	A high value indicates a problem with a specific bean. Again, the problem can be specific to the bean implementation or with any of the dependent servers of the bean.
Idle bean percent	The percentage of beans those are idle in the cache.	Percent	A high percentage indicates a memory bottleneck. The timeout of the session beans could be one of the possible reasons.

3.6 Troubleshooting

If the WLJDBC test is blue, it could indicate that no JDBC connection pools have been configured in the WebLogic application server. To verify this, do the following:

1. Open the Internet Explorer and type the following URL: *http://<weblogic_server_ip>:<WebLogic_Server_Port>/console*, to open the login screen of the WebLogic server administration console (see Figure 3.27).

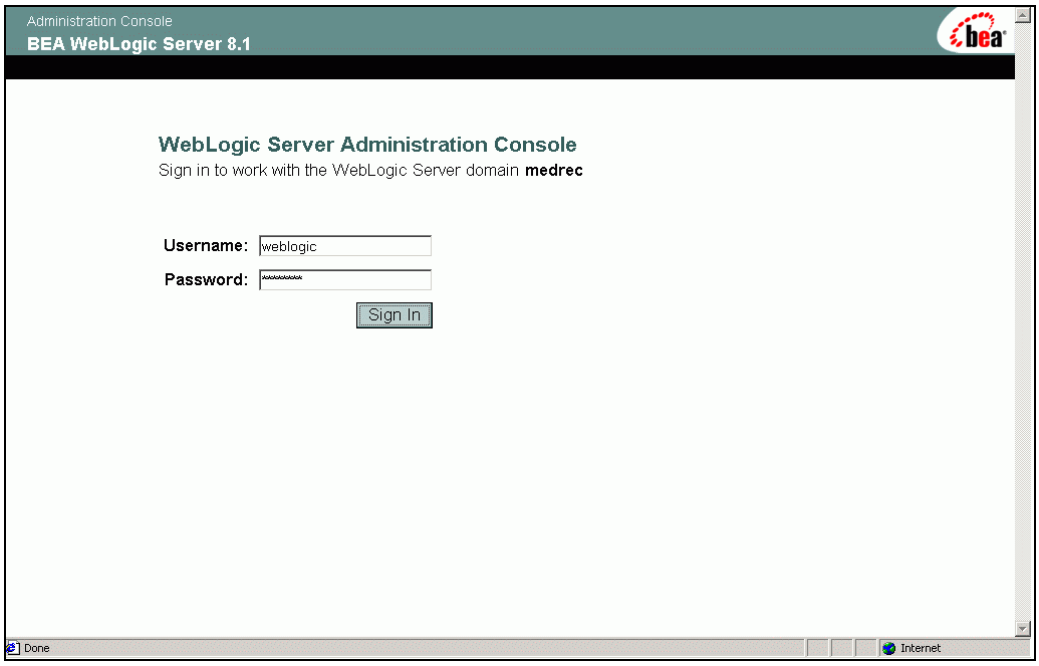


Figure 3.27: Logging into the WebLogic Server Administration Console

2. Next, expand the **JDBC** node in the tree-structure in the left pane of the console, and then, expand the **Connection Pools** sub-node within. This sub-node will contain the list of connection pools that have been conifigured in the WebLogic server (see Figure 3.28).

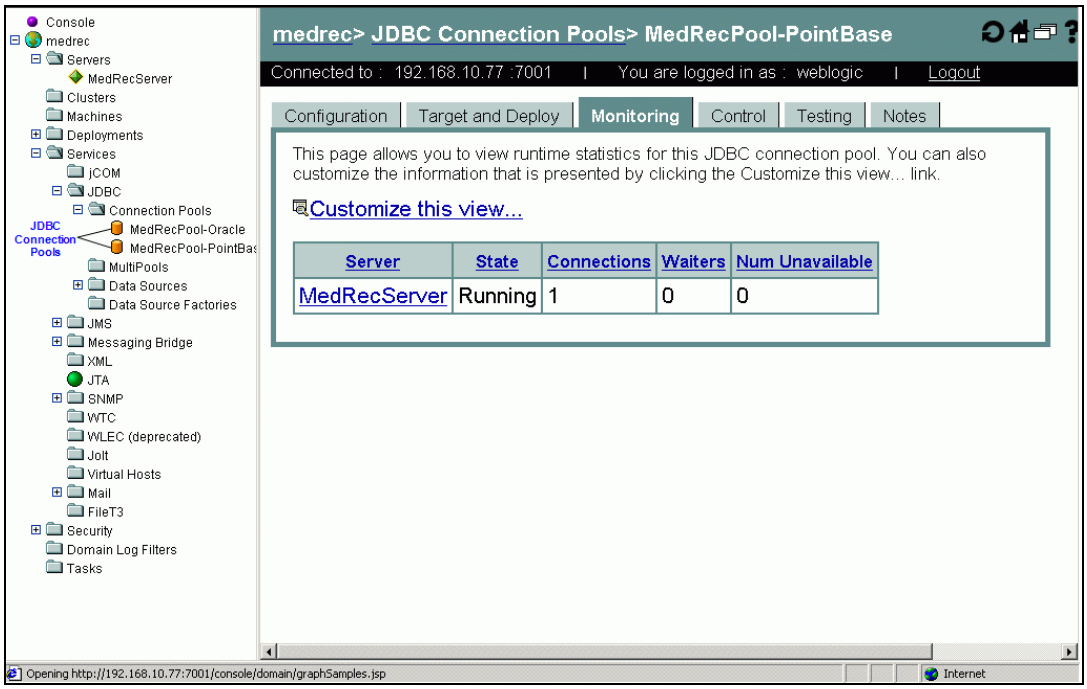


Figure 3.28: Viewing the connection pools configured in the WebLogic server

3. If no connection pools exist, then this sub-node will not list any. This would hence be a clear indicator of the non-existence of connection pools.

If all the WebLogic-related tests are in blue, it could indicate that the **egurkha.war** file has not been deployed on the WebLogic server. To verify this, do the following:

1. Open the Internet Explorer and specify the following URL in its Address bar: *http://<weblogic_server_ip>:<WebLogic_Server_Port>/egurkha/Hello.jsp*.
2. If the **egurkha.war** file has been deployed properly, then the following page will appear:

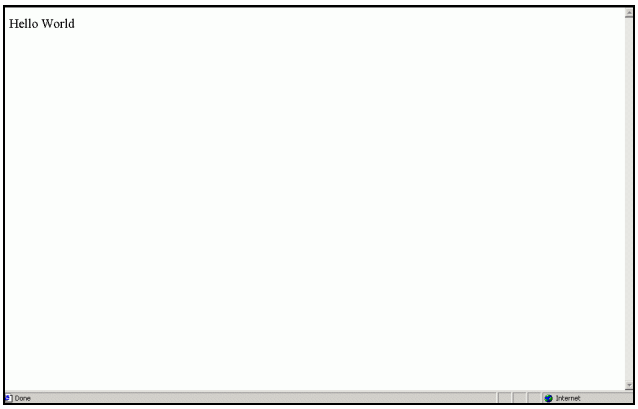


Figure 3.29: The page that appears upon typing the specified URL in the Internet Explorer

3. If the war file has not been deployed, then an error message will appear instead.

Chapter 4: Monitoring the WebLogic Server Ver. 6/7/8

The special WebLogic (6/7/8) monitoring model (see Figure 3.1) that eG Enterprise offers uses JMX (Java Management extension), the new standard for managing java components, for monitoring version 6, 7, and 8 of the WebLogic server. JMX allows users to instrument their applications and control or monitor them using a management console.

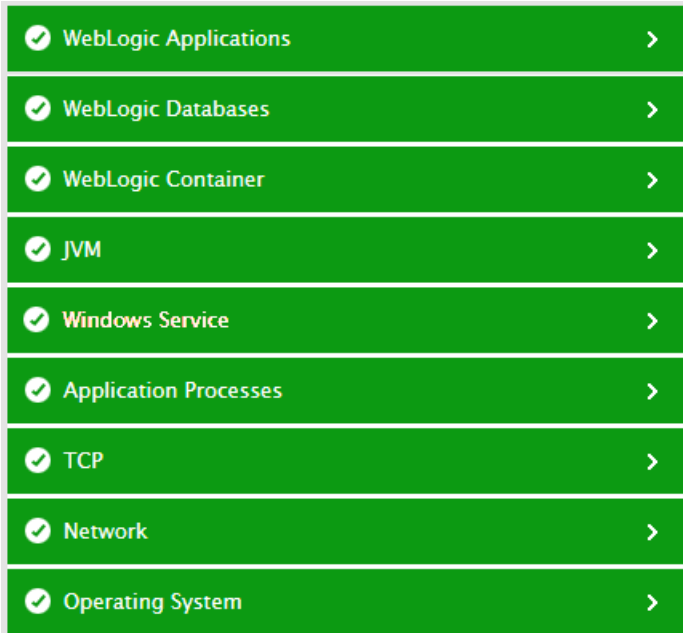


Figure 4.1: Layer model of the WebLogic Application server

The sections that will follow discuss the top 4 layers of Figure 3.1, and the metrics they report. The remaining layers have been extensively dealt with in the *Monitoring Unix and Windows Servers* document.

4.1 The JVM Layer

A Java virtual machine (JVM), an implementation of the Java Virtual Machine Specification, interprets compiled java binary code for a computer's processor (or "hardware platform") so that it can perform a Java program's instructions. The Java Virtual Machine Specification defines an abstract -- rather than a real -- machine or processor. The Specification specifies an instruction set, a set of registers, a stack, a garbage heap, and a method area.

The tests associated with the **JVM** layer of WebLogic enables administrators to perform the following functions:

- Assess the effectiveness of the garbage collection activity performed on the JVM heap
- Monitor WebLogic thread usage
- Evaluate the performance of the BEA JRockit JVM

Figure 4.2: The tests associated with the JVM layer

Note that the **WebLogic Work Manager** test is not mapped to this layer, indicating that this test is not available for WebLogic versions 6, 7, and 8.

All the tests listed above are similar to the tests mapped to Section **3.2** of this document. Therefore, let us proceed to the next layer.

4.2 The WebLogic Container Layer

This layer tracks the health of the WebLogic service. The status of this layer is determined by the results of the tests shown in Figure 4.3.

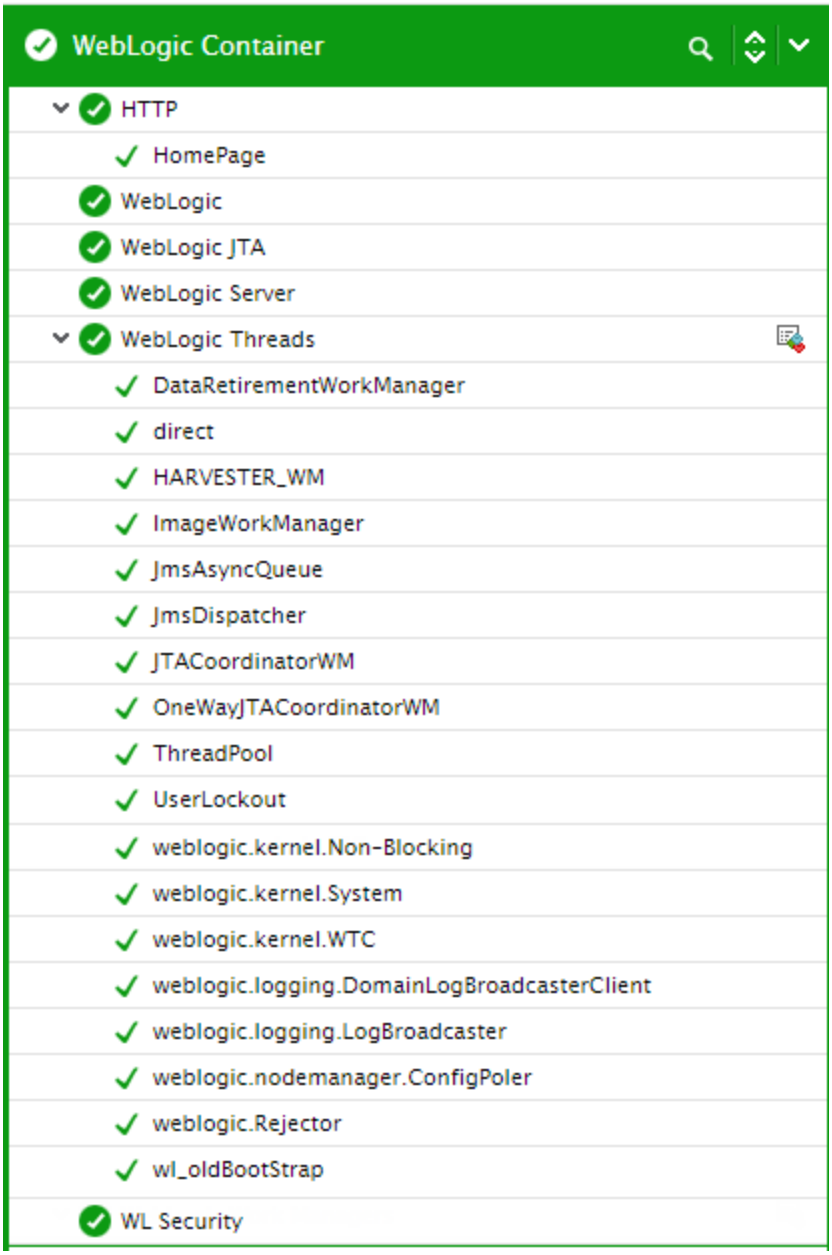


Figure 4.3: Tests mapped to the WebLogic Container layer

Note that the **WebLogic Work Managers**, **WebLogic Queues** and **WebLogic Topics** tests are not mapped to this layer, indicating that these tests are not available for WebLogic versions 6, 7, and 8. All other tests depicted by Figure 3.11 have already been discussed in Section **3.3.8** of this document.

4.3 The WebLogic Databases Layer

The status of the database connectivity from the WebLogic server to one or more backend database servers is assessed by the WebLogic Databases layer. The status of the **WebLogic Databases** layer is determined based on the results of a WebLogicJdbc test that is shown in Figure 3.22.

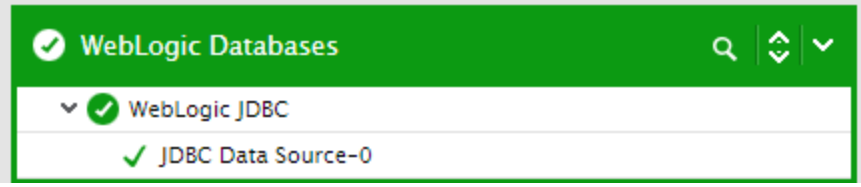


Figure 4.4: Tests mapping to the WebLogic Databases layer

The **WebLogicJDBC** test mapped to this layer has already been discussed in Section **3.5.1**. Therefore, let us move on to the next layer.

4.4 The WebLogic Applications Layer

Using the tests mapped to this layer, administrators can identify the web applications that are running on the WebLogic server, and the current session load on each web application. Additionally, the layer also monitors the EJB activity on the server and even measures the how quickly the servlets are processing requests they receive.

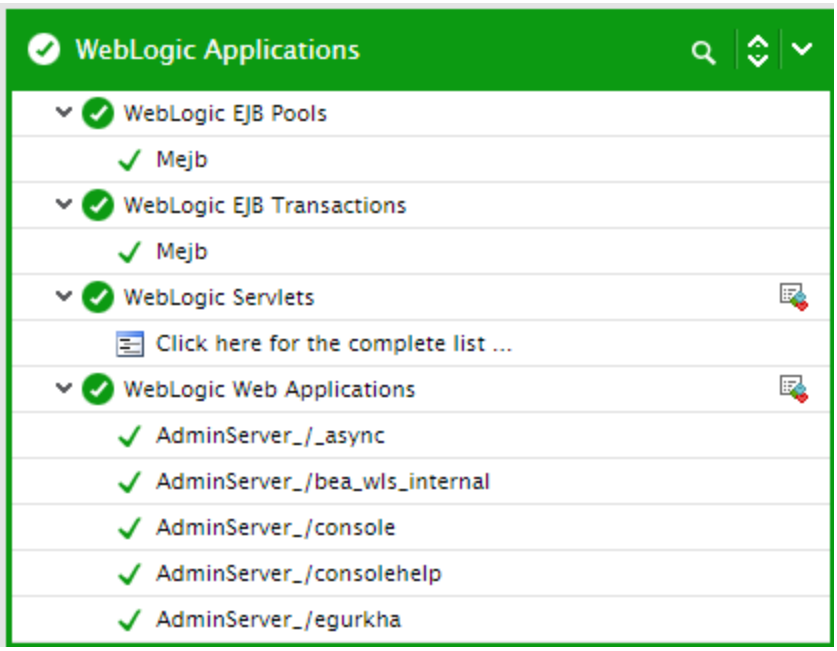


Figure 4.5: Tests mapping to the WebLogic Applications layer

These tests, once again, have already been discussed under Section **3.5.4**.

5

Chapter 5: Conclusion

This document has described in detail the monitoring paradigm used and the measurement capabilities of the eG Enterprise suite of products with respect to **Oracle Web Logic Application Servers**. For details of how to administer and use the eG Enterprise suite of products, refer to the user manuals.

We will be adding new measurement capabilities into the future versions of the eG Enterprise suite. If you can identify new capabilities that you would like us to incorporate in the eG Enterprise suite of products, please contact support@eginnovations.com. We look forward to your support and cooperation. Any feedback regarding this manual or any other aspects of the eG Enterprise suite can be forwarded to feedback@eginnovations.com.