Monitoring Tomcat Server

eG Innovations Product Documentation

www.eginnovations.com

Total Performance Visibility

Table of Contents

Table of Figures

Chapter 1: Introduction

The Tomcat server is a Java based Web Application container that was created to run Servlets and JavaServer Pages (JSP) in Web applications. As part of Apache's open source Jakarta project, it has nearly become the industry accepted standard reference implementation for both the Servlets and JSP API.

Application developers capitalize on the capabilities of the Tomcat container to build robust web applications that provide mission-critical services to end users. Naturally, the 24 x 7 availability and speedy responsiveness of the Tomcat server is imperative for the applications and the dependent businesses to remain afloat.

The application server middleware that hosts and supports the business logic components is often the most complex of the multi-tier infrastructure. To offer peak performance, an application server provides a host of complex functions and features including database connection pooling, thread pooling, database result caching, session management, bean caching and management etc. To ensure that the application server is functioning effectively at all times, all of these functions have to be monitored and tracked proactively and constantly.

eG Enterprise offers specialized monitoring models for each of the most popular application servers such as WebLogic, WebSphere, ColdFusion, Oracle 9i/10G, etc. A plethora of metrics relating to the health of the application servers can be monitored in real-time and alerts can be generated based on user-defined thresholds or auto-computed baselines. These metrics enable administrators to quickly and accurately determine server availability and responsiveness, resource usage at the host-level and at the application server level, how well the application server processes requests, how quickly the server completes transactions, overall server security, etc.

This document engages you in an elaborate discussion on how eG Enterprise monitors each of the popular web application servers in the market.

Chapter 2: How does eG Enterprise Monitor Tomcat Servers?

The eG agent is capable of monitoring Tomcat in an agent-based manner and an agentless manner.

2.1 Pre-requsites for monitoring the Tomcat Servers

To enable the eG agent to start monitoring the server, a set of pre-requisites should be fulfilled. These requirements are discussed in the following sections.

2.1.1 Configuring a Tomcat server to work with the eG Agent

The performance data pertaining to a Tomcat server are captured by certain JSP files - one each for every test that is executed on the Tomcat server. The eG agent on the Tomcat server, while running tests on the server, contacts these JSP files and pulls out the reported metrics from them. To deploy these JSP files on the Tomcat server, you will first have to install a special **egtomcat** application (i.e., the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory) on the server.

Note:

Prior to war deployment, ensure that the Tomcat server to be monitored executes on JDK 1.5 or above; the eG agent can monitor only those Tomcat servers that are using JDK 1.5 or above.

To deploy the war, do the following:

1. From the browser, connect to the Tomcat server using the URL: **http://<TomcatIP>:<Tomcatport>.**
2. Next, login to the Tomcat server by providing valid credentials at Figure 2.1.

Figure 2.1: Logging into Tomcat

3. Upon a successful login, Figure 2.2 will appear.

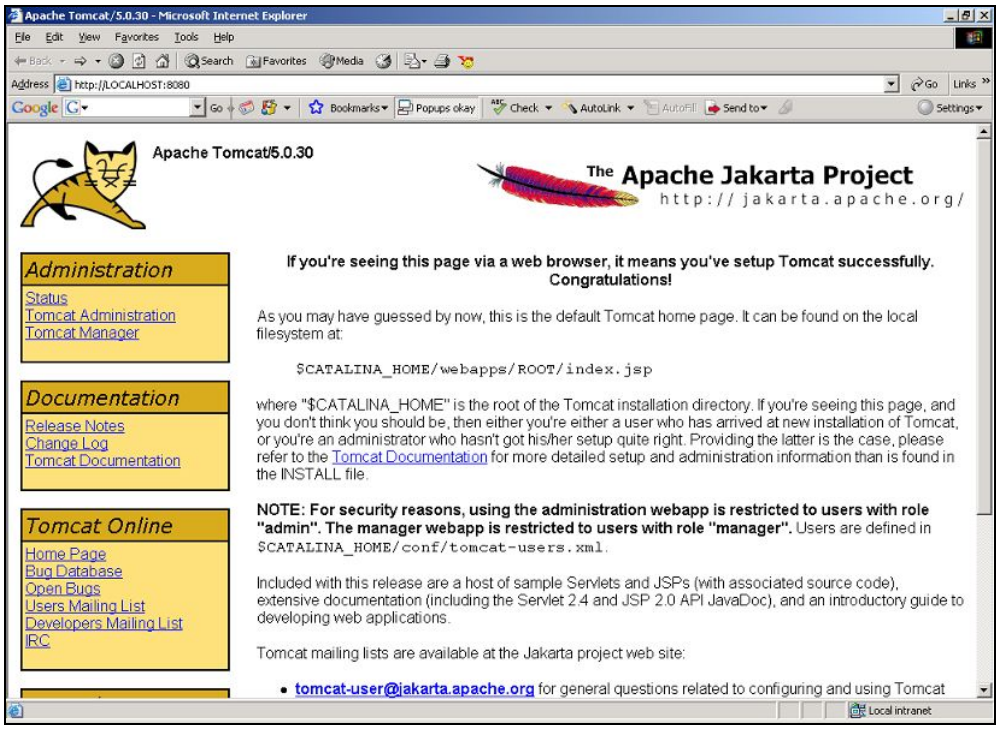

Figure 2.2: The default Tomcat home page

4. To begin installing a new application, click on the **Tomcat Manager** hyperlink under the **Administration** section in the left panel of Figure 2.2. Doing so invokes Figure 2.3 that displays all the applications that pre-exist on the Tomcat server.

Figure 2.3: List of applications on the Tomcat server

5. In order to deploy a new application, scroll down the applications list (see Figure 2.3) until a **WAR file to deploy** section (see Figure 2.4) becomes visible. Use the **Browse** button in that section to quickly specify the full path to the **egtomcat.war** file to be deployed on the server. Finally, click the **Deploy** button.

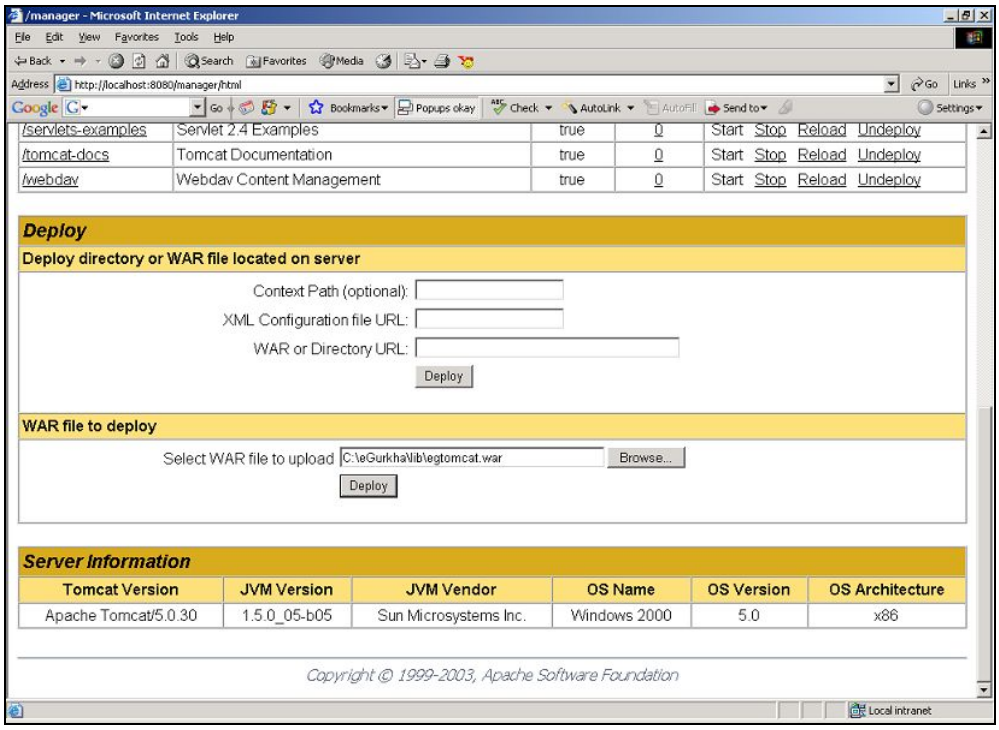

Figure 2.4: Deploying the egtomcat.war file

6. If the war file is successfully deployed, then an entry for the **egtomcat** application will be automatically added to the applications list, as depicted by Figure 2.5.

Figure 2.5: Tomcat application successfully deployed

2.2 Configuring the eG Agent to Collect JVM-related Metrics from the Tomcat Server

The **JVM** layer of the *Tomcat* monitoring model is associated with tests that report critical statistics related to the Tomcat 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 tests connect to the JRE used by the Tomcat application server to pull out the above-mentioned metrics. For these test to execute, you should configure the eG agent to connect to JRE and collect the required metrics using either of the following methodologies:

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

Since both JMX and SNMP support are available for JRE 1.5 and above only, these **tests will work only if the Tomcat server being monitored uses JRE 1.5 and above**.

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

1. 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:
   - Login to the target Tomcat server.
2. Edit the *catalina.sh* file (on Unix; on Windows, this will be *catalina.bat*) in the <CATALINA_HOME_DIR>\bin directory (on Unix; on Windows, this will be <CATALINE_HOME_DIR>\bin) of the target Tomcat server, and configure the following in it:
   - The JMX remote port
   - Whether JMX is SSL-enabled or not
3. Indicate that JMX does not require authentication
   - The IP address using which the Tomcat server has been managed in the eG Enterprise system
4. To configure the above, append the following lines to the *catalina.sh* (or *catalina.bat* file, as the case may be) of Tomcat versions prior to v7.0:

```
-Dcom.sun.management.jmxremote.port=<Port No>
-Dcom.sun.management.jmxremote.ssl=<true/false>
-Dcom.sun.management.jmxremote.authenticate=false
-Djava.rmi.server.hostname=<IP address using which the Tomcat server is managed in the eG Entperprise system>
```

   In case of Tomcat 7.0 (or above), append the following lines to the *catalina.sh* (or *catalina.bat*) file:

```
JAVA_OPTS="-Dcom.sun.management.jmxremote.port=<PortNo> -
Dcom.sun.management.jmxremote.ssl=<true/false> -
Dcom.sun.management.jmxremote.authenticate=false -Djava.rmi.server.hostname=<IP
address using which the Tomcat server is managed in the eG Enterprise system>
```

Note:

- The catalina.sh (or catalina.bat) file is used for both starting and stopping a Tomcat server instance. However, when enabling JMX for a Tomcat server instance, you need to make sure that the related JAVA_OPTS setting is invoked only when starting the server instance, and not when stopping it. To ensure this, follow the steps below:

  1. In the catalina.sh (or catalina.bat) file, look for the following entry:

     ```
     JAVA_OPTS=$JAVA_OPTS $JSSE_OPTS
     ```

  2. Once it is found, replace the entry with the following script:

     ```
     If ["$1" = start ]; then
     JAVA_OPTS=$JAVA_OPTS -Dcom.sun.management.jmxremote.port=1234 -
     Dcom.sun.management.jmxremote.authenticate=false -
     Dcom.sun.management.jmxremote.ssl=false $JSSE_OPTS
     else
     JAVA_OPTS=$JAVA_OPTS $JSSE_OPTS
     fi
     ```

  3. Finally, save the file.

- The Java Service Wrapper provides a pair of Java Management Extensions (JMX) MBean interfaces (J2SE 5.0 JMX, JavaSE6 JMX) which make it possible to control the Wrapper using a JMX interface.

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

1. Login to the target Tomcat server. If the server is executing on a Windows host, then, login to the host as a local/domain administrator.
2. 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.
3. 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.
4. 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.

5. Edit the *catalina.sh* file (on Unix; on Windows, this will be *catalina.bat*) in the <CATALINA_HOME_DIR>/bin directory (on Unix; on Windows, this will be <CATALINA_HOME_dir>\bin) of the target Tomcat server, and configure the following in it:

   - The JMX remote port
   - Whether JMX is SSL-enabled or not
   - Indicate that JMX **requires authentication**
   - The full path to the *jmxremote.access* file
   - The full path to the *jmxremote.password* file
   - The IP address using which the Tomcat server has been managed in the eG Enterprise system
   - To configure the above, append the following lines to the *catalina.sh* (or *catalina.bat* file, as the case may be) file of Tomcat server prior to v7.0:

```
-Dcom.sun.management.jmxremote.port=<Port No>
-Dcom.sun.management.jmxremote.ssl=<true/false>
-Dcom.sun.management.jmxremote.authenticate=true
-Dcom.sun.management.jmxremote.access.file=<Path of jmxremote.access>
-Dcom.sun.management.jmxremote.password.file=<Path of jmxremote.password>
-Djava.rmi.server.hostname=<IP address using which the Tomcat server is managed in the eG Entperprise system>
```

   - In case of Tomcat 7.0 (or above), append the following lines to the *catalina.sh* (or *catalina.bat*) file:

```
JAVA_OPTS="-Dcom.sun.management.jmxremote.port=<PortNo> -
Dcom.sun.management.jmxremote.ssl=<true/false> -
Dcom.sun.management.jmxremote.authenticate=false -Djava.rmi.server.hostname=<IP address using which the Tomcat server is managed in the eG Enterprise system> -
Dcom.sun.management.jmxremote.access.file=<Path to the jmxremote.access file> -
Dcom.sun.management.jmxremote.password.file=<Path to the jmxremote.password file>"
```

6. Then, **Save** the file.

Chapter 2: How does eG Enterprise Monitor Tomcat Servers?

The eG agent is capable of monitoring Tomcat in an agent-based manner and an agentless manner.

2.3 Pre-requsites for monitoring the Tomcat Servers

To enable the eG agent to start monitoring the server, a set of pre-requisites should be fulfilled. These requirements are discussed in the following sections.

2.3.1 Configuring a Tomcat server to work with the eG Agent

The performance data pertaining to a Tomcat server are captured by certain JSP files - one each for every test that is executed on the Tomcat server. The eG agent on the Tomcat server, while running tests on the server, contacts these JSP files and pulls out the reported metrics from them. To deploy these JSP files on the Tomcat server, you will first have to install a special **egtomcat** application (i.e., the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory) on the server.

Note:

Prior to war deployment, ensure that the Tomcat server to be monitored executes on JDK 1.5 or above; the eG agent can monitor only those Tomcat servers that are using JDK 1.5 or above.

To deploy the war, do the following:

1. From the browser, connect to the Tomcat server using the URL: **http://<TomcatIP>:<Tomcatport>.**
2. Next, login to the Tomcat server by providing valid credentials at Figure 2.6.

Figure 2.6: Logging into Tomcat

3. Upon a successful login, Figure 2.7 will appear.

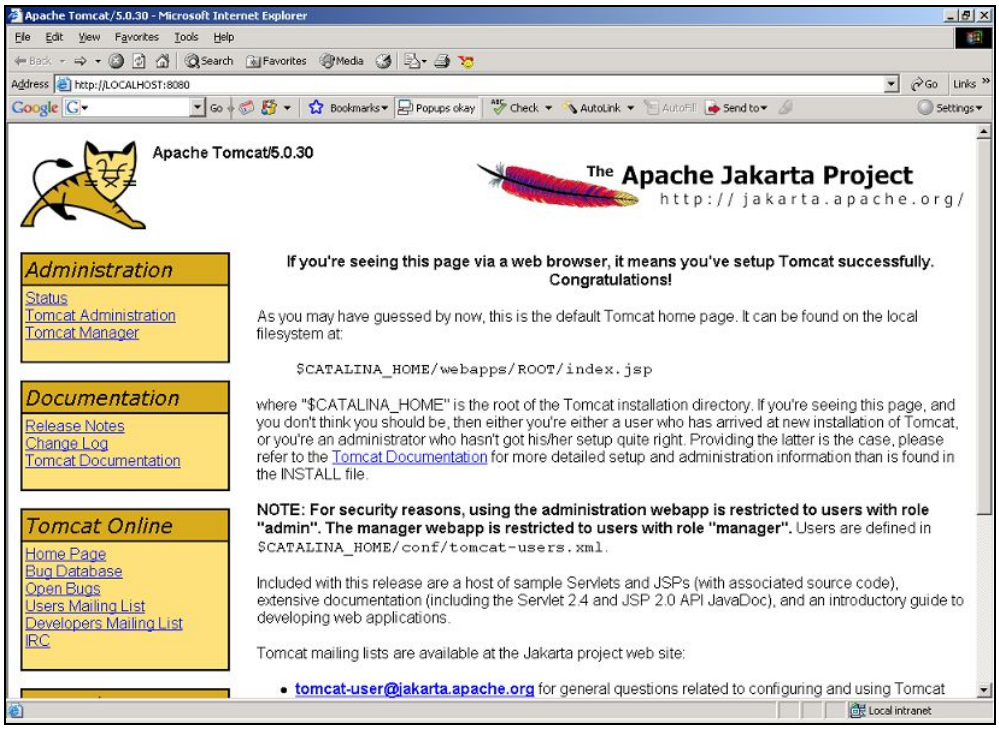

Figure 2.7: The default Tomcat home page

4. To begin installing a new application, click on the **Tomcat Manager** hyperlink under the **Administration** section in the left panel of Figure 2.7. Doing so invokes Figure 2.8 that displays all the applications that pre-exist on the Tomcat server.

Figure 2.8: List of applications on the Tomcat server

5. In order to deploy a new application, scroll down the applications list (see Figure 2.8) until a **WAR file to deploy** section (see Figure 2.9) becomes visible. Use the **Browse** button in that section to quickly specify the full path to the **egtomcat.war** file to be deployed on the server. Finally, click the **Deploy** button.

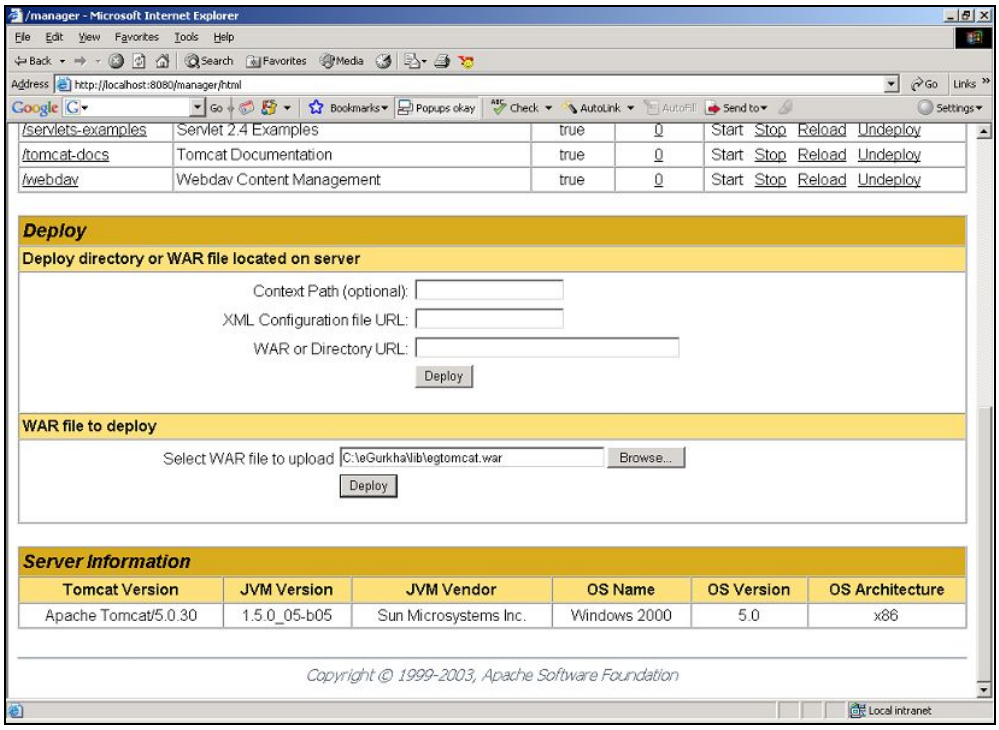

Figure 2.9: Deploying the egtomcat.war file

6. If the war file is successfully deployed, then an entry for the **egtomcat** application will be automatically added to the applications list, as depicted by Figure 2.10.

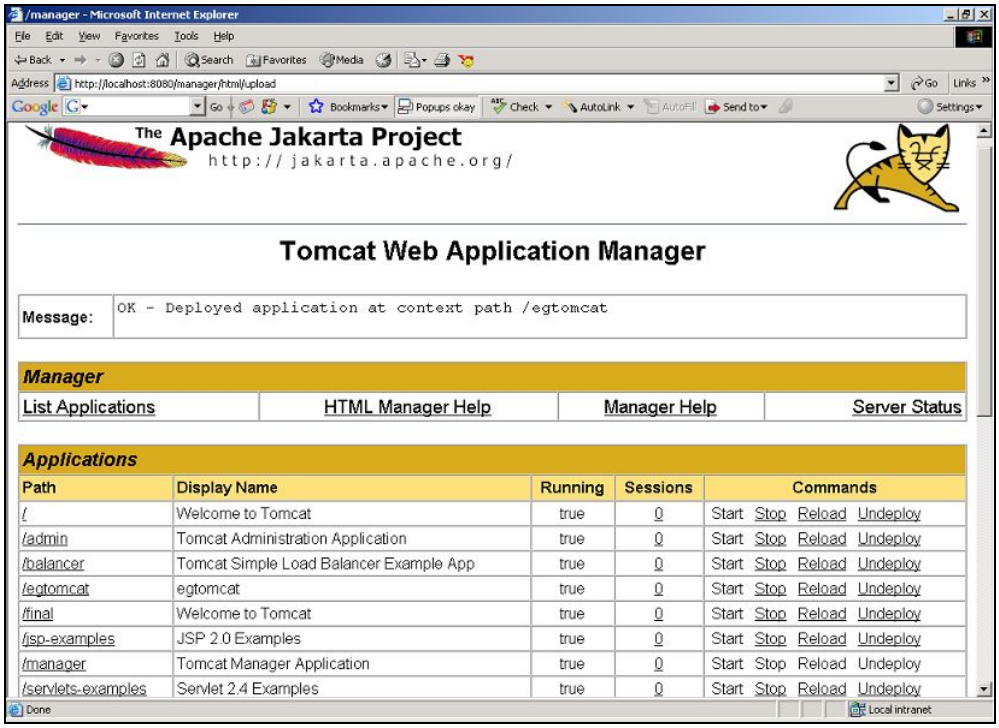

Figure 2.10: Tomcat application successfully deployed

2.4 Configuring the eG Agent to Collect JVM-related Metrics from the Tomcat Server

The **JVM** layer of the *Tomcat* monitoring model is associated with tests that report critical statistics related to the Tomcat 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 tests connect to the JRE used by the Tomcat application server to pull out the above-mentioned metrics. For these test to execute, you should configure the eG agent to connect to JRE and collect the required metrics using either of the following methodologies:

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

Since both JMX and SNMP support are available for JRE 1.5 and above only, these **tests will work only if the Tomcat server being monitored uses JRE 1.5 and above**.

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

1. 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:
    - Login to the target Tomcat server.
2. Edit the *catalina.sh* file (on Unix; on Windows, this will be *catalina.bat*) in the <CATALINA_HOME_DIR>\bin directory (on Unix; on Windows, this will be <CATALINE_HOME_DIR>\bin) of the target

Tomcat server, and configure the following in it:

- The JMX remote port
- Whether JMX is SSL-enabled or not

3. Indicate that JMX does not require authentication
   - The IP address using which the Tomcat server has been managed in the eG Enterprise system
4. To configure the above, append the following lines to the *catalina.sh* (or *catalina.bat* file, as the case may be) of Tomcat versions prior to v7.0:

```
-Dcom.sun.management.jmxremote.port=<Port No>
-Dcom.sun.management.jmxremote.ssl=<true/false>
-Dcom.sun.management.jmxremote.authenticate=false
-Djava.rmi.server.hostname=<IP address using which the Tomcat server is managed in the eG Entperprise system>
```

In case of Tomcat 7.0 (or above), append the following lines to the *catalina.sh* (or *catalina.bat*) file:

```
JAVA_OPTS="-Dcom.sun.management.jmxremote.port=<PortNo> -
Dcom.sun.management.jmxremote.ssl=<true/false> -
Dcom.sun.management.jmxremote.authenticate=false -Djava.rmi.server.hostname=<IP address using which the Tomcat server is managed in the eG Enterprise system>
```

Note:

- The catalina.sh (or catalina.bat) file is used for both starting and stopping a Tomcat server instance. However, when enabling JMX for a Tomcat server instance, you need to make sure that the related JAVA_OPTS setting is invoked only when starting the server instance, and not when stopping it. To ensure this, follow the steps below:
  1. In the catalina.sh (or catalina.bat) file, look for the following entry:

```
JAVA_OPTS=$JAVA_OPTS $JSSE_OPTS
```

  2. Once it is found, replace the entry with the following script:

```
If ["$1" = start ]; then
JAVA_OPTS=$JAVA_OPTS -Dcom.sun.management.jmxremote.port=1234 -
Dcom.sun.management.jmxremote.authenticate=false -
Dcom.sun.management.jmxremote.ssl=false $JSSE_OPTS
else
```

```
JAVA_OPTS=$JAVA_OPTS $JSSE_OPTS
fi
```

3. Finally, save the file.

- The Java Service Wrapper provides a pair of Java Management Extensions (JMX) MBean interfaces (J2SE 5.0 JMX, JavaSE6 JMX) which make it possible to control the Wrapper using a JMX interface.

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

1. Login to the target Tomcat server. If the server is executing on a Windows host, then, login to the host as a local/domain administrator.
2. 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.
3. 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.
4. 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.
5. Edit the *catalina.sh* file (on Unix; on Windows, this will be *catalina.bat*) in the <CATALINA_HOME_DIR>/bin directory (on Unix; on Windows, this will be <CATALINA_HOME_dir>\bin) of the target Tomcat server, and configure the following in it:
   - The JMX remote port
   - Whether JMX is SSL-enabled or not
   - Indicate that JMX **requires authentication**
   - The full path to the *jmxremote.access* file
   - The full path to the *jmxremote.password* file
   - The IP address using which the Tomcat server has been managed in the eG Enterprise system
   - To configure the above, append the following lines to the *catalina.sh* (or *catalina.bat* file, as the case may be) file of Tomcat server prior to v7.0:

```
-Dcom.sun.management.jmxremote.port=<Port No>
```

```
-Dcom.sun.management.jmxremote.ssl=<true/false>
-Dcom.sun.management.jmxremote.authenticate=true
-Dcom.sun.management.jmxremote.access.file=<Path of jmxremote.access>
-Dcom.sun.management.jmxremote.password.file=<Path of jmxremote.password>
-Djava.rmi.server.hostname=<IP address using which the Tomcat server is managed in the eG Entperprise system>
```

- In case of Tomcat 7.0 (or above), append the following lines to the *catalina.sh* (or *catalina.bat*) file:

```
JAVA_OPTS="-Dcom.sun.management.jmxremote.port=<PortNo> -
Dcom.sun.management.jmxremote.ssl=<true/false> -
Dcom.sun.management.jmxremote.authenticate=false -Djava.rmi.server.hostname=<IP address using which the Tomcat server is managed in the eG Enterprise system> -
Dcom.sun.management.jmxremote.access.file=<Path to the jmxremote.access file> -
Dcom.sun.management.jmxremote.password.file=<Path to the jmxremote.password file>"
```

6. Then, **Save** the file.

Chapter 2: How does eG Enterprise Monitor Tomcat Servers?

The eG agent is capable of monitoring Tomcat in an agent-based manner and an agentless manner.

2.5 Pre-requsites for monitoring the Tomcat Servers

To enable the eG agent to start monitoring the server, a set of pre-requisites should be fulfilled. These requirements are discussed in the following sections.

2.5.1 Configuring a Tomcat server to work with the eG Agent

The performance data pertaining to a Tomcat server are captured by certain JSP files - one each for every test that is executed on the Tomcat server. The eG agent on the Tomcat server, while running tests on the server, contacts these JSP files and pulls out the reported metrics from them. To deploy these JSP files on the Tomcat server, you will first have to install a special **egtomcat** application (i.e., the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory) on the server.

Note:

Prior to war deployment, ensure that the Tomcat server to be monitored executes on JDK 1.5 or above; the eG agent can monitor only those Tomcat servers that are using JDK 1.5 or above.

To deploy the war, do the following:

1. From the browser, connect to the Tomcat server using the URL: **http://<TomcatIP>:<Tomcatport>.**
2. Next, login to the Tomcat server by providing valid credentials at Figure 2.11.

Figure 2.11: Logging into Tomcat

3. Upon a successful login, Figure 2.12 will appear.

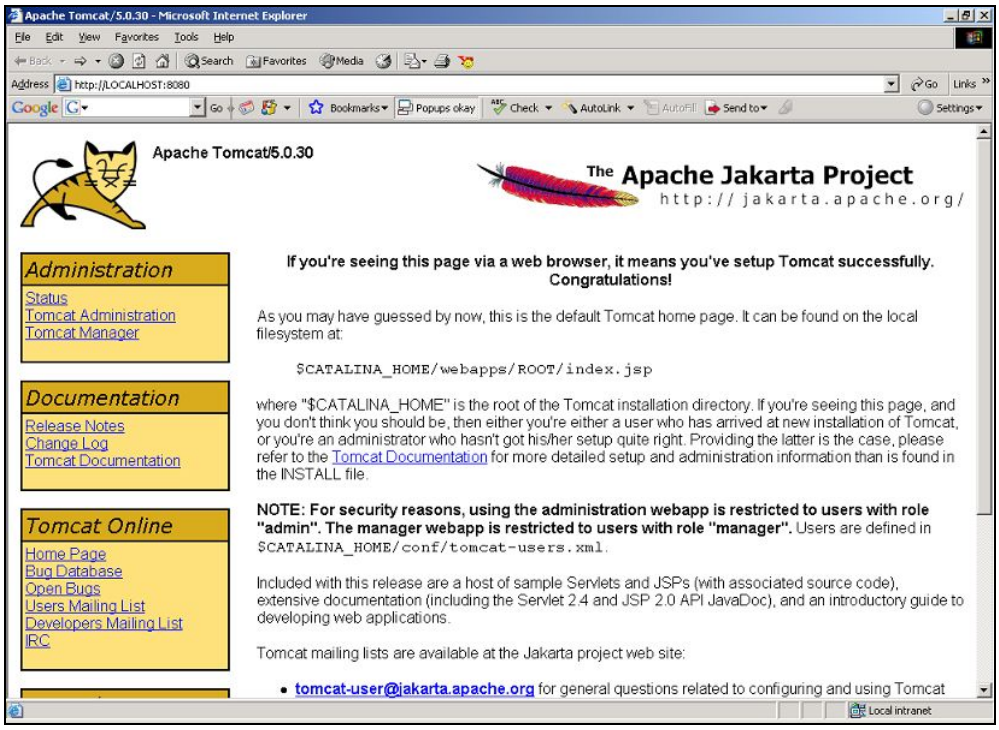

Figure 2.12: The default Tomcat home page

4. To begin installing a new application, click on the **Tomcat Manager** hyperlink under the **Administration** section in the left panel of Figure 2.12. Doing so invokes Figure 2.13 that displays all the applications that pre-exist on the Tomcat server.

Figure 2.13: List of applications on the Tomcat server

5. In order to deploy a new application, scroll down the applications list (see Figure 2.13) until a **WAR file to deploy** section (see Figure 2.14) becomes visible. Use the **Browse** button in that section to quickly specify the full path to the **egtomcat.war** file to be deployed on the server. Finally, click the **Deploy** button.

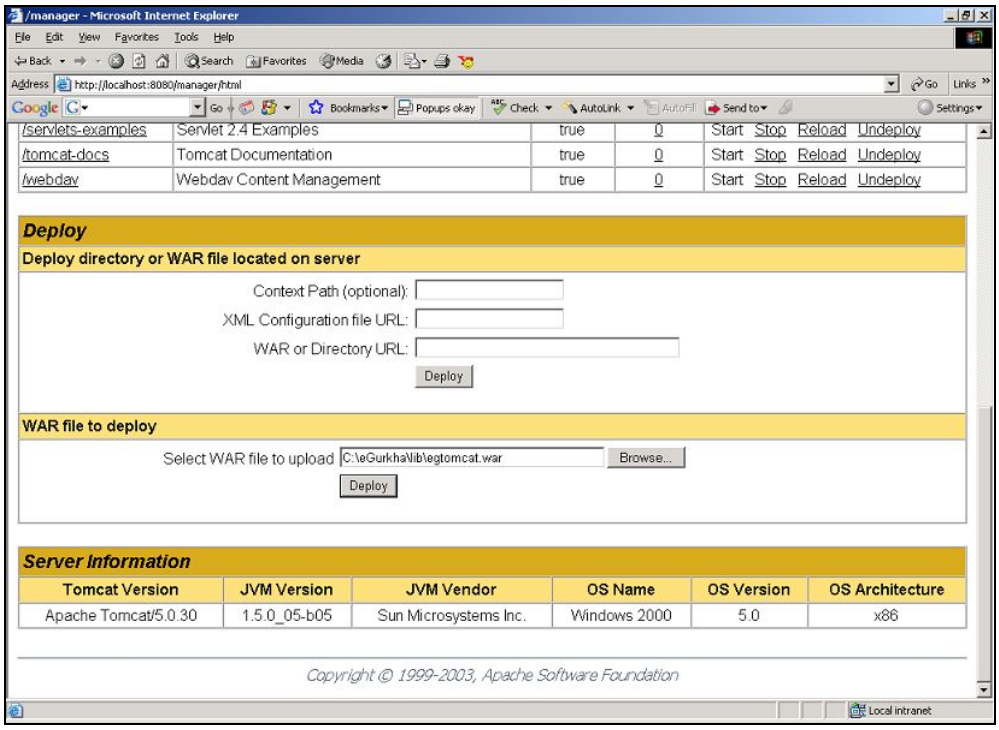

Figure 2.14: Deploying the egtomcat.war file

6. If the war file is successfully deployed, then an entry for the **egtomcat** application will be automatically added to the applications list, as depicted by Figure 2.15.

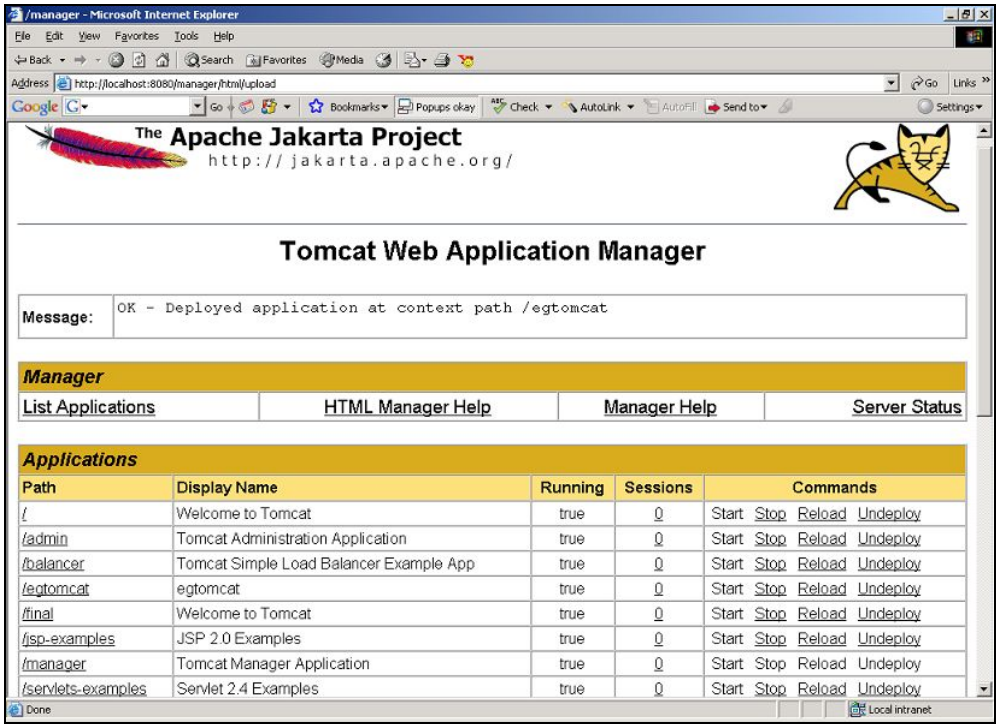

Figure 2.15: Tomcat application successfully deployed

2.6 Configuring the eG Agent to Collect JVM-related Metrics from the Tomcat Server

The **JVM** layer of the *Tomcat* monitoring model is associated with tests that report critical statistics related to the Tomcat 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 tests connect to the JRE used by the Tomcat application server to pull out the above-mentioned metrics. For these test to execute, you should configure the eG agent to connect to JRE and collect the required metrics using either of the following methodologies:

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

Since both JMX and SNMP support are available for JRE 1.5 and above only, these **tests will work only if the Tomcat server being monitored uses JRE 1.5 and above**.

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

1. 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:
    - Login to the target Tomcat server.
2. Edit the *catalina.sh* file (on Unix; on Windows, this will be *catalina.bat*) in the <CATALINA_HOME_DIR>\bin directory (on Unix; on Windows, this will be <CATALINE_HOME_DIR>\bin) of the target

Tomcat server, and configure the following in it:

- The JMX remote port
- Whether JMX is SSL-enabled or not

3. Indicate that JMX does not require authentication
   - The IP address using which the Tomcat server has been managed in the eG Enterprise system
4. To configure the above, append the following lines to the *catalina.sh* (or *catalina.bat* file, as the case may be) of Tomcat versions prior to v7.0:

```
-Dcom.sun.management.jmxremote.port=<Port No>
-Dcom.sun.management.jmxremote.ssl=<true/false>
-Dcom.sun.management.jmxremote.authenticate=false
-Djava.rmi.server.hostname=<IP address using which the Tomcat server is managed in the eG Entperprise system>
```

In case of Tomcat 7.0 (or above), append the following lines to the *catalina.sh* (or *catalina.bat*) file:

```
JAVA_OPTS="-Dcom.sun.management.jmxremote.port=<PortNo> -
Dcom.sun.management.jmxremote.ssl=<true/false> -
Dcom.sun.management.jmxremote.authenticate=false -Djava.rmi.server.hostname=<IP address using which the Tomcat server is managed in the eG Enterprise system>
```

Note:

- The catalina.sh (or catalina.bat) file is used for both starting and stopping a Tomcat server instance. However, when enabling JMX for a Tomcat server instance, you need to make sure that the related JAVA_OPTS setting is invoked only when starting the server instance, and not when stopping it. To ensure this, follow the steps below:
  1. In the catalina.sh (or catalina.bat) file, look for the following entry:

     ```
     JAVA_OPTS=$JAVA_OPTS $JSSE_OPTS
     ```

  2. Once it is found, replace the entry with the following script:

     ```
     If ["$1" = start ]; then
     JAVA_OPTS=$JAVA_OPTS -Dcom.sun.management.jmxremote.port=1234 -
     Dcom.sun.management.jmxremote.authenticate=false -
     Dcom.sun.management.jmxremote.ssl=false $JSSE_OPTS
     else
     ```

```
JAVA_OPTS=$JAVA_OPTS $JSSE_OPTS
fi
```

3. Finally, save the file.

- The Java Service Wrapper provides a pair of Java Management Extensions (JMX) MBean interfaces (J2SE 5.0 JMX, JavaSE6 JMX) which make it possible to control the Wrapper using a JMX interface.

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

1. Login to the target Tomcat server. If the server is executing on a Windows host, then, login to the host as a local/domain administrator.
2. 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.
3. 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.
4. 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.
5. Edit the *catalina.sh* file (on Unix; on Windows, this will be *catalina.bat*) in the <CATALINA_HOME_DIR>/bin directory (on Unix; on Windows, this will be <CATALINA_HOME_dir>\bin) of the target Tomcat server, and configure the following in it:
   - The JMX remote port
   - Whether JMX is SSL-enabled or not
   - Indicate that JMX **requires authentication**
   - The full path to the *jmxremote.access* file
   - The full path to the *jmxremote.password* file
   - The IP address using which the Tomcat server has been managed in the eG Enterprise system
   - To configure the above, append the following lines to the *catalina.sh* (or *catalina.bat* file, as the case may be) file of Tomcat server prior to v7.0:

```
-Dcom.sun.management.jmxremote.port=<Port No>
```

```
-Dcom.sun.management.jmxremote.ssl=<true/false>
```

```
-Dcom.sun.management.jmxremote.authenticate=true
```

```
-Dcom.sun.management.jmxremote.access.file=<Path of jmxremote.access>
```

```
-Dcom.sun.management.jmxremote.password.file=<Path of jmxremote.password>
```

```
-Djava.rmi.server.hostname=<IP address using which the Tomcat server is managed in the eG Entperprise system>
```

- In case of Tomcat 7.0 (or above), append the following lines to the *catalina.sh* (or *catalina.bat*) file:

```
JAVA_OPTS="-Dcom.sun.management.jmxremote.port=<PortNo> -
Dcom.sun.management.jmxremote.ssl=<true/false> -
Dcom.sun.management.jmxremote.authenticate=false -Djava.rmi.server.hostname=<IP address using which the Tomcat server is managed in the eG Enterprise system> -
Dcom.sun.management.jmxremote.access.file=<Path to the jmxremote.access file> -
Dcom.sun.management.jmxremote.password.file=<Path to the jmxremote.password file>"
```

6. Then, **Save** the file.

2.7 Managing a Tomcat Server

The Tomcat server can be automatically discovered by eG Enterprise. However, it can be added manually using the eG administrative interface. Remember that the eG Enterprise automatically manages the components that are added manually. Discovered components are managed using the **COMPONENTS - MANAGE/UNMANAGE** page. To manually add the Tomcat server component, do the following:

1. Log into the eG administrative interface.
2. If a Tomcat server is already discovered, , then directly proceed towards managing it using the **COMPONENTS - MANAGE/UNMANAGE** page (Infrastructure - > Components - > Manage/Unmanage).
3. If the Tomcat server is yet to be discovered, then run discovery (Infrastructure -> Components -> Discovery) to get it discovered or add the Tuxedo domain server manually using the **COMPONENTS**page (Infrastructure -> Components -> Add/Modify). Figure 2.16 clearly illustrates the process of adding a Tomcat server.

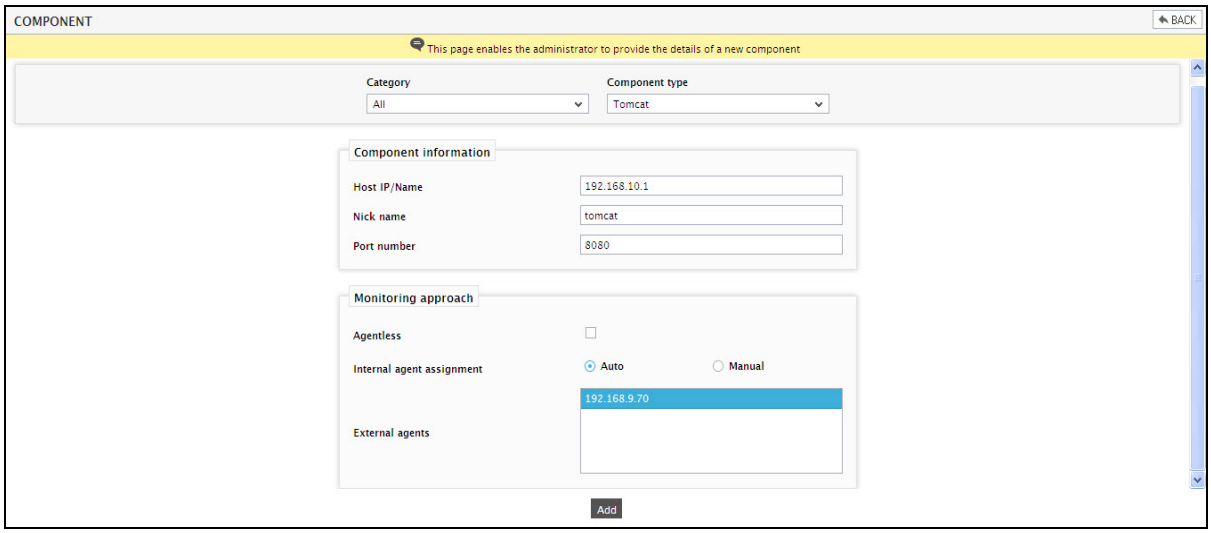

Figure 2.16: Adding a Tomcat server

4. Specify the **Host IP/Name** and the **Nick name** of the Tomcat server in 2.7. Then, click the **Add** button to register the changes.
5. Next, try to sign out of the eG administrative interface. Upon doing so, a list of unconfigured tests will appear prompting you to configure the tests pertaining to the Tomcat server (see).

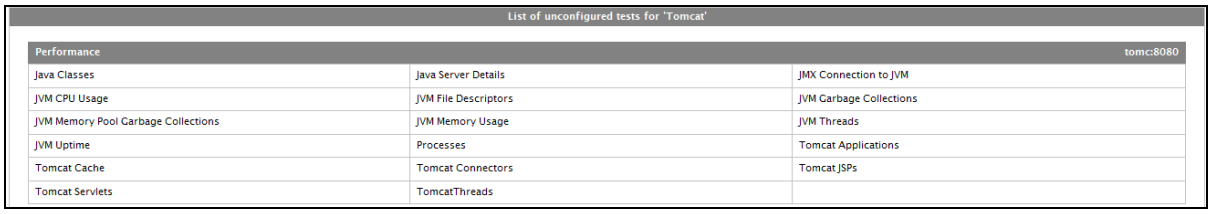

Figure 2.17: List of unconfigured tests for Tomcat server

6. To know procedure to configure the tests, refer to **Monitoring Tomcat Servers**.
7. Finally, sign out of the eG administrative interface.

Chapter 3: Monitoring Tomcat Servers

Using the specialized monitoring model that eG Enterprise offers for the Tomcat server (see Figure 3.1), one can extensively monitor every layer of the Tomcat server, automatically correlate performance across the layers, and accurately locate the problem source.

Note:

The eG agent can monitor a Tomcat server, only if the Tomcat server executes on JDK 1.5 or above.

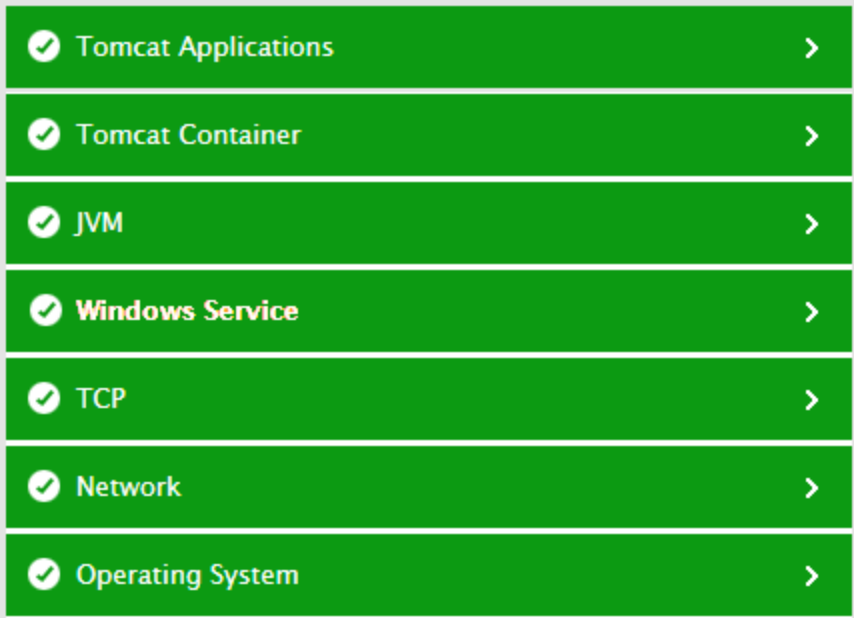

Figure 3.1: The layer model of the Tomcat server

Each layer depicted by Figure 3.1 performs a number of tests on the Tomcat server and evaluates the critical aspects of Tomcat performance such as thread usage, memory management, processing capabilities of the servlets, efficiency of the JSP engine, etc.

The eG agent begins monitoring the Tomcat Server once the pre-requisites discussed in the Section **2.5** are fulfilled. The statistics so collected enable administrators find quick and accurate answers to persistent performance-related queries, such as the following:

Availability monitoring	• Is the Tomcat server available? • How quickly does it respond to requests?

Thread pool usage monitoring	• Have adequate threads been allocated to the pool? • Are too many threads idle?
Servlet monitoring	• Is the server taking too long to process servlets? Which servlet is contributing to the delay?
Jasper monitoring	• How frequently do reloads occur? • Are the JSPs been updated often?
Connector monitoring	• How quickly are connectors processing requests? Is there a delay in request processing? If so, which connector is responsible for the delay? • How much traffic is generated by a connector? • Have any errors occurred during request processing?
JVM monitoring	• Has the JVM been up and running continuously? • How many classes have been loaded/unloaded by the JVM? • Is there a garbage collection bottleneck? • Does the JVM have sufficient free memory?
Session manager monitoring	• Is the server workload high? How many sessions are currently active on the server? • Were too many sessions rejected?
Cache monitoring	• Are cache hits optimal? • Are disk accesses low?

The sections to come discuss the top 3 layers of the layer model, as the remaining layers have already been discussed in the *Monitoring Unix and Windows Servers* document.

3.1 The JVM Layer

The tests associated with this layer provide users with valuable insight into the various aspects of Java Virtual Machines including:

- Class loading/unloading
- Server availability
- Garbage collection activity
- Memory/CPU management

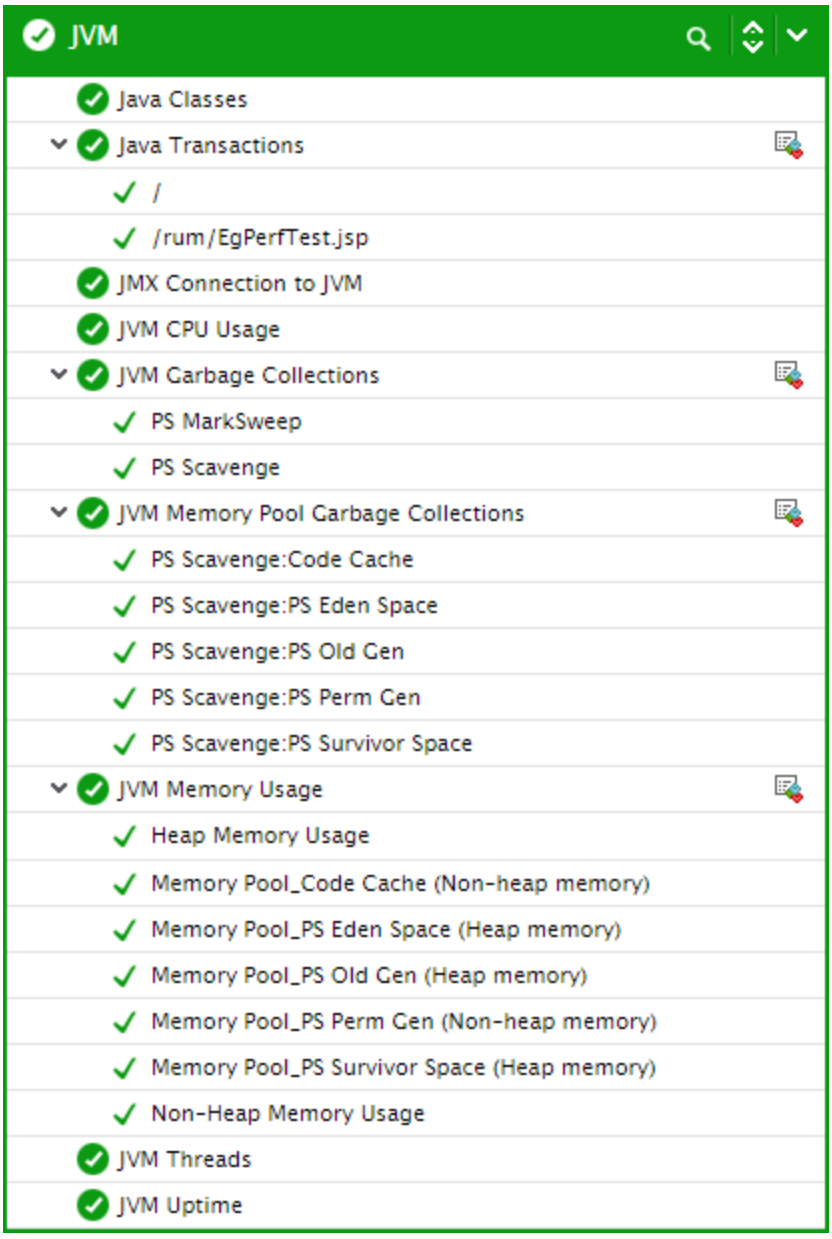

Figure 3.2: Tests associated with JVM layer

3.1.1 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 Tomcat Server

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	Refers to the port at which the specified host listens to.
JMX Remote 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 for details).
User, Password, and Confirm Password	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.
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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from the target Java application. If there is no response from the target beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
JMX availability	Indicates whether the target application is available or not and whether JMX is enabled or not on the application.	Percent	If the value of this measure is 100%, it indicates that the Java application is available with JMX enabled. The value 0 on the other hand, could indicate one/both the following: • The Java application is unavailable;

Measurement	Description	Measurement Unit	Interpretation
			• 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.

3.1.2 JVM File Descriptors Test

This test reports useful statistics pertaining to file descriptors.

Target of the test : A Tomcat Server

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	Refers to the port at which the specified host listens to.
JMX Remote 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 for details).
User, Password, and Confirm Password	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*

Parameter	Description
	Applications document. Confirm the password by retyping it in the Confirm Password text box.
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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from the target Java application. If there is no response from the target beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

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	

3.1.3 Java Classes Test

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

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	Refers to the port at which the specified host listens to.
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 Remote 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 for details).
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.
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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from the target Java application. If there is no response from the target beyond the configured duration, the test will timeout. By default, this is set to **240** seconds if the mode is **JMX**, and **10** seconds if the mode is **SNMP**.

Parameter	Description
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 for details).
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 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

Parameter	Description
	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.
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

Measurement	Description	Measurement Unit	Interpretation
			container, and the applications running on the container, to have access to different repositories of available classes and resources. A consistent 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	

3.1.4 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 **JVM Garbage Collections** test reports the performance statistics pertaining to the JVM's garbage collection.

Target of the test : A Java application

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for each garbage collector that is reclaiming the unused memory on the JVM of 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	Refers to the port at which the specified host listens to.
Mode	This test can extract metrics from the Java application using either of the following mechanisms:

Parameter	Description
	• 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 Remote 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 for details).
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.
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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from the target Java application. If there is no response from the target beyond the configured duration, the test will timeout. By default, this is set to **240** seconds if the mode is **JMX**, and **10** seconds if the mode is **SNMP**.
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 for details).
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 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

Parameter	Description
	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.
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 this garbage collector was started to release dead objects from memory during the last measurement period.	Number	
Time taken for garbage collection	Indicates the time taken to by this garbage collector to perform the current garbage collection operation.	Secs	Ideally, the value of both these 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

Measurement	Description	Measurement Unit	Interpretation
Percent of time spent by JVM for garbage collection	Indicates the percentage of time spent by this garbage collector on garbage collection during the last measurement period.	Percent	application performance, it is best to ensure that GC activity does not take too long to complete.

3.1.5 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 gauge 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 Tomcat 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 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	Refers to the port at which the specified host listens to.
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

Parameter	Description
	• Using GC logs To use JMX for metrics collections, set the measure modeto **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 Remote 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 for details).
User, Password, and Confirm Password	This parameter will be available 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.
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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
JREHome	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 path to the GC log file to be used for metrics collection.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from the target Java application. If there is no response from the target beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Has garbage collection happened	Indicates whether garbage collection occurred on this memory pool in the last measurement period.		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: State: Yes = 1; No = 0 **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.
Collection count	Indicates the number of time in the last measurement pool garbage collection was started on this memory pool.	Number	
Initial memory before GC	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.	MB	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

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 of 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.

3.1.6 JVM Threads Test

This test reports the status of threads running in the JVM. Details of this test can be used to identify resource-hungry threads.

Target of the test : A Java Application

Agent deploying the test : An internal 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first, select the **War File** option. Then, refer to Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the JMX option. Then, follow the procedure detailed in the Configuring and Monitoring Tomcat Servers document to configure the test to use JMX. By default, the JMX option is chosen here.
JMX Remote 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 for details).
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

Parameter	Description
	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.
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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from the target Java application. If there is no response from the target beyond the configured duration, the test will timeout. By default, this is set to **240** seconds if the mode is **JMX**, and **10** seconds if the mode is **SNMP**.
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 for details).
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 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.

Parameter	Description
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.
PCT Medium CPU Util Threads	By default, the PCT Medium CPU Util Threads parameter is set to 50. This implies that, by default, the threads for which the current CPU consumption is between 50% and 70% (the default value of the PCT High CPU Util Threads parameter) will be counted as medium CPU-consuming threads. The count of such threads will be reported as the value of the *Medium CPU threads* measure. This default setting also denotes that threads that consume less than 50% CPU will, by default, be counted as *Low CPU threads*. If need be, you can modify the value of this PCT Medium CPU Util threads parameter to change how much CPU should be used by a thread for it to qualify as a medium CPU-consuming thread. This will consequently alter the count of low CPU-consuming threads as well.
PCT High CPU Util Threads	By default, the PCT High CPU Util Threads parameter is set to 70. This implies that, by default, the threads that are currently consuming over 70% of CPU time are counted as high CPU consumers. The count of such threads will be reported as the value of the High CPU threads measure. If need be, you can modify the value of this parameter to change how much CPU should be used by a thread for it to qualify as a high CPU-consuming thread.
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

Parameter	Description
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.
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**.
Useps	**This flag is applicable only for AIX LPARs**. By default, on AIX LPARs, this test uses the **tprof** command to compute CPU usage. Accordingly, the UsePS flag is set to **No** by default. On some AIX LPARs however, the **tprof** command may not function properly (this is an AIX issue). While monitoring such AIX LPARs therefore, you can configure the test to use the **ps** command instead for metrics collection. To do so, set the Useps flag to **Yes**. **Note:** Alternatively, you can set the **AIXusePS** flag in the **[AGENT_SETTINGS]** section of the **eg_tests.ini** file (in the <EG_INSTALL_DIR>\manager\config directory) to **yes** (default: **no**) to enable the eg agent to use the **ps** command for CPU usage computations on AIX LPARs. If this global flag and the Useps flag for a specific component are both set to **no**, then the test will use the default **tprof** command to compute CPU usage for AIX LPARs. If either of these flags is set to **yes**, then the **ps** command will perform the CPU usage computations for monitored AIX LPARs. In some high-security environments, the **tprof** command may require some special privileges to execute on an AIX LPAR (eg., *sudo* may need to be used to run **tprof**). In

Parameter	Description
	such cases, you can prefix the **tprof** command with another command (like *sudo*) or the full path to a script that grants the required privileges to **tprof**. To achieve this, edit the **eg_tests.ini** file (in the <EG_INSTALL_DIR>\manager\config directory), and provide the prefix of your choice against the **AixTprofPrefix** parameter in the [AGENT_SETTINGS] section. Finally, save the file. For instance, if you set the **AixTprofPrefix** parameter to *sudo*, then the eG agent will call the **tprof** command as *sudo tprof*.
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 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

Measurement	Description	Measurement Unit	Interpretation
			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 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.

Measurement	Description	Measurement Unit	Interpretation
			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 threads that are currently consuming CPU lower than the value configured in the PCT Medium CPU Util Threads text box.	Number	
Medium CPU threads	Indicates the number of threads that are currently consuming CPU that is higher than the value configured in the PCT Medimum CPU Util Threads text box and is lower than or equal to the value specified in the PCT High CPU Util Threads text box.	Number	
High CPU threads	Indicates the number of threads that are currently	Number	Ideally, the value of this measure should be very low. A high value is

Measurement	Description	Measurement Unit	Interpretation
	consuming CPU that is greater than the percentage configured in the PCT High CPU Util Threads text box.		indicative of a resource contention at the JVM. Under such circumstances, you might want to identify the resource-hungry threads. 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	
Total threads	Indicates 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	
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:

- Login to the application host.
- Go to the <JAVA_HOME>\jre\lib\management folder used by the target application, and edit the *management.properties* file in that folder.
- Append the following line to the file:

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

- Finally, save the file.

3.1.7 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 threads that are contributing to the CPU spike. Detailed stack trace information provides insights to code level information that can highlight problems with the design of the Java application.

Note:

- If you want to collect metrics for this test from the JRE MIB – i.e, if the mode parameter of this test is set to SNMP - then ensure that the SNMP and SNMP Trap services are up and running on the application host.
- While monitoring a Java application executing on a Windows 2003 server using SNMP, ensure that the community string to be used during SNMP access is explicitly added when starting the SNMP service.

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	Refers to the port at which the specified host listens to.
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.

Parameter	Description
JMX Remote 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 for details).
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.
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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from the target Java application. If there is no response from the target beyond the configured duration, the test will timeout. By default, this is set to **240** seconds if the mode is **JMX**, and **10** seconds if the mode is **SNMP**.
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 for details).
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.

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.
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**.
Useps	**This flag is applicable only for AIX LPARs**. By default, on AIX LPARs, this test uses the **tprof** command to compute CPU usage. Accordingly, the UsePS flag is set to **No** by default. On some AIX LPARs however, the **tprof** command may not function properly (this is an AIX issue). While monitoring such AIX LPARs therefore, you can configure the test to use the **ps** command instead for metrics collection. To do so, set the Useps flag to **Yes**. **Note:** Alternatively, you can set the **AIXusePS** flag in the **[AGENT_SETTINGS]** section of the **eg_tests.ini** file (in the <EG_INSTALL_DIR>\manager\config directory) to **yes** (default: **no**) to enable the eG agent to use the **ps** command for CPU usage computations on AIX LPARs. If this global flag and the Useps flag for a specific component are both set to **no**, then the test will use the default **tprof** command to compute CPU usage for AIX LPARs. If either of these flags is set to **yes**, then the **ps** command will perform the CPU usage computations for monitored AIX LPARs. In some high-security environments, the **tprof** command may require some special privileges to execute on an AIX LPAR (eg., *sudo* may need to be used to run **tprof**). In such cases, you can prefix the **tprof** command with another command (like *sudo*) or the full path to a script that grants the required privileges to **tprof**. To achieve this, edit the **eg_tests.ini** file (in the <EG_INSTALL_DIR>\manager\config directory), and provide the prefix of your choice against the **AixTprofPrefix** parameter in the [AGENT_SETTINGS] section. Finally, save the file. For instance, if you set the **AixTprofPrefix** parameter to *sudo*, then the eG agent will call the **tprof** command as *sudo tprof*.

Parameter	Description
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
CPU utilization of JVM	Indicates the percentage of total available CPU time taken up by the JVM.	Percent	If a system has multiple processors, this value is the total CPU time used by the JVM divided by the number of processors on the system. 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 runnable threads are currently utilizing excessive CPU.

The detailed diagnosis of the *CPU utilization* of JVM measure lists all the CPU-consuming threads currently executing in the JVM, in the descending order of the Percentage Cpu Time of the threads; this way, you can quickly and accurately identify CPU-intensive threads in the JVM. In addition to CPU usage information, the detailed diagnosis also reveals the following information for every thread:

- The number of times the thread was blocked during the last measurement period, the total duration of the blocks, and the percentage of time for which the thread was blocked;
- The number of times the thread was in waiting during the last measurement period, the total duration waited, and the percentage of time for which the thread waited;
- The **Stacktrace** of the thread, using which you can nail the exact line of code causing the CPU consumption of the thread to soar;

Details of the threads

Time	Thread Name	ThreadID	Thread State	Cpu Time (Secs)	Percentage Cpu Time (%)	Blocked Count	Blocked Time (Secs)	Percentage Blocked Time (%)	Waited	Waited Time (Secs)	Percentage Waited Time (%)	Stacktrace
Jun 22, 2009 14:42:30												Stack Trace
	http7077-Processor2	19	RUNNABLE	2.296	0.6651	103	0.006	0	83	534.26	18.18	java.net.SocketInputStream.socketRead0(Native Method); java.net.SocketInputStream.read(SocketInputStream.java:129); org.apache.coyote.http11.InternalInputBuffer.fill(InternalInputBuffer.java:767); org.apache.coyote.http11.InternalInputBuffer.parseRequestLine (InternalInputBuffer.java:428); org.apache.coyote.http11.Http11Processor.process (Http11Processor.java:790); org.apache.coyote.http11.Http11Protocol$Http11ConnectionHandler.processConnection (Http11Protocol.java:700); org.apache.tomcat.util.net.TcpWorkerThread.runIt (PoolTcpEndpoint.java:584); org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run (ThreadPool.java:683); java.lang.Thread.run(Thread.java:619);
	http7077-Processor4	21	RUNNABLE	2.89	0.4811	231	0.045	0	403	654.029	0	java.net.SocketInputStream.socketRead0(Native Method); java.net.SocketInputStream.read(SocketInputStream.java:129); org.apache.coyote.http11.InternalInputBuffer.fill(InternalInputBuffer.java:767); org.apache.coyote.http11.InternalInputBuffer.parseRequestLine (InternalInputBuffer.java:428); org.apache.coyote.http11.Http11Processor.process (Http11Processor.java:790); org.apache.coyote.http11.Http11Protocol$Http11ConnectionHandler.processConnection (Http11Protocol.java:700); org.apache.tomcat.util.net.TcpWorkerThread.runIt (PoolTcpEndpoint.java:584); org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run (ThreadPool.java:683); java.lang.Thread.run(Thread.java:619);
	http7077-Processor1	18	RUNNABLE	3.984	0.4528	103	0.014	0	407	562.412	17.27	java.net.SocketInputStream.socketRead0(Native Method); java.net.SocketInputStream.read(SocketInputStream.java:129); org.apache.coyote.http11.InternalInputBuffer.fill(InternalInputBuffer.java:767); org.apache.coyote.http11.InternalInputBuffer.parseRequestLine (InternalInputBuffer.java:428); org.apache.coyote.http11.Http11Processor.process (Http11Processor.java:790);

Figure 3.3: The detailed diagnosis of the CPU utilization of JVM measure

3.1.8 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.

Note:

- This test works only on Windows platforms.
- This test can provide detailed diagnosis information for only those monitored Java applications that use JRE 1.6 or higher.

Target of the test : A Java application

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	Refers to the port at which the specified host listens to.
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 Remote 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 for details).
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.
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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from the target Java application. If there is no response from the target beyond the configured

Parameter	Description
	duration, the test will timeout. By default, this is set to **240** seconds if the mode is **JMX**, and **10** seconds if the mode is **SNMP**.
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 for details).
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 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.

Parameter	Description
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.
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**.
Heap Analysis	By default, this flag is set to **off**. This implies that the test will not provide detailed diagnosis information for memory usage, by default. To trigger the collection of detailed measures, set this flag to **On**. **Note**: If Heap Analysis is switched **On**, then the eG agent will be able to collect detailed measures only if the Java application being monitored uses JDK 1.6 or higher.
JAVA Home	This parameter appears only when the Heap Analysis flag is switched **On**. Here, provide the full path to the install directory of JDK 1.6 or higher on the application host.

Parameter	Description
	For example, *c:\JDK1.6.0*.
Exclude Packages	The detailed diagnosis of this test, if enabled, lists the Java classes/packages that are using the pool memory and the amount of memory used by each class/package. To enable administrators to focus on the memory consumed by those classes/packages that are specific to their application, without being distracted by the memory consumption of basic Java classes/packages, the test, by default, excludes some common Java packages from the detailed diagnosis. The packages excluded by default are as follows: • All packages that start with the string java or javax - in other words, *java.** and *javax.**. • Arrays of primitive data types - eg., [Z, which is a one-dimensional array of type boolean, [[B, which is a 2-dimensional array of type byte, etc. • A few class loaders - eg., *<symbolKlass>, <constantPoolKlass>, <instanceKlassKlass>, <constantPoolCacheKlass>*, etc. This is why, the Exclude Packages parameter is by default configured with the packages mentioned above. You can, if required, append more packages or patterns of packages to this comma-separated list. This will ensure that such packages also are excluded from the detailed diagnosis of the test. **Note that the exclude packages parameter is of relevance only if the Heap Analysis flag is set to 'Yes'**.
Include Packages	By default, this is set to *all*. This indicates that, by default, the detailed diagnosis of the test (if enabled) includes all classes/packages associated with the monitored Java application, regardless of whether they are basic Java packages or those that are crucial to the functioning of the application. However, if you want the detailed diagnosis to provide the details of memory consumed by a specific set of classes/packages alone, then, provide a comma-separated list of classes/packages to be included in the detailed diagnosis in the Include Packages text box. **Note that the include packages parameter is of relevance only if the heap analysis flag is set to 'Yes'**.
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

Parameter	Description
	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
Initial memory	Indicates the amount of memory initially allocated at startup.	MB	
Used memory	Indicates the amount of memory currently used.	MB	It includes the memory occupied by all objects, including both reachable and unreachable objects. Ideally, the value of this measure should be low. A high value or a consistent increase in the value could indicate gradual erosion of memory resources. In such a situation, you can take the help of the detailed diagnosis of this measure (if enabled), to figure out which class is using up memory excessively.
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. In such a situation, you can take the help of the detailed diagnosis of this measure (if enabled), to figure out which class is using up memory excessively.

The detailed diagnosis of the *Used memory* measure, if enabled, lists all the classes that are using the pool memory, the amount and percentage of memory used by each class, the number of instances of each class that is currently operational, and also the percentage of currently running instances of each class. Since this list is by default sorted in the descending order of the percentage memory usage, the first class in the list will obviously be the leading memory consumer.

Details of JVM Heap Usage

Time	Class Name	Instance Count	Instance Percentage	Memory used(MB)	Percentage memory used
Jun 17, 2009 12:11:01					
	com.ibc.object.SapBusinessObject	104003	11.5774	12.629	22.5521
	[Ljava.lang.Object;	23586	2.6255	7.4904	13.3759
	<constMethodKlas>	41243	4.5911	5.8357	10.4211
	java.lang.String	174044	19.3742	3.9836	7.1136
	[C	240000	26.7163	3.6621	6.5396
	[B	7336	0.8166	3.5868	6.4051
	<methodKlas>	41243	4.5911	3.1514	5.6275
	<symbolKlas>	69152	7.6979	2.8014	5.0025
	[I	26240	2.921	2.3018	4.1105
	<constantPoolKlas>	3097	0.3448	2.0491	3.6591
	<instanceKlassKlas>	3097	0.3448	1.296	2.3144
	<constantPoolCacheKlas>	2663	0.2964	1.2536	2.2386
	[S	5546	0.6174	0.4283	0.7649
	java.util.Hashtable$Entry	15908	1.7708	0.3641	0.6502
	<methodDataKlas>	870	0.0968	0.3594	0.6418
	java.lang.reflect.Method	4269	0.4752	0.3257	0.5816
	java.lang.Class	3383	0.3766	0.3097	0.5531
	java.util.Vector	13266	1.4767	0.3036	0.5422

Figure 3.4: The detailed diagnosis of the Used memory measure

3.1.9 JVM Uptime Test

This test tracks the uptime of a JVM. Using information provided by this test, administrators can determine whether the JVM was restarted. Comparing uptime across Java applications, an admin can determine the JVMs that have been running without any restarts for the longest time. One set of results for every Java application monitored

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	Refers to the port at which the specified host listens to.
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;

Parameter	Description
	• 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 Remote 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 for details).
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.
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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from the target Java application. If there is no response from the target beyond the configured duration, the test will timeout. By default, this is set to **240** seconds if the mode is **JMX**, and **10** seconds if the mode is **SNMP**.
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 for details).
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.

Parameter	Description
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 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.

Parameter	Description
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.
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**.
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

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td>Has JVM been restarted?</td><td>Indicates whether or not the JVM has restarted during the last measurement period.</td><td></td><td>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 restart states discussed above are listed in the table 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 JVM has restarted. The graph of this measure however, represents the same using the numeric equivalents – 0 or 1.</td></tr>
<tr><td>Uptime during the last measure period</td><td>Indicates the time period that the JVM has been up since the last time this test ran.</td><td>Secs</td><td>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</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
			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.

3.2 Tests Disabled by Default for a Tomcat Server

The tests discussed above are enabled by default for a Tomcat server. In addition to these tests, the eG agent can be optionally configured to execute a few other tests that report critical statistics related to CPU usage, thread usage, and the uptime of the Tomcat JVM. To enable one/more of these tests, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, select *Tomcat* as the **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.

To enable one/more of these tests, open the agents – tests configuration page using the Agents -> Tests -> Configure menu sequence, select Tomcat from the Component type list, scroll down the test list that appears to view the disabled tests section, select the check box corresponding to the JVM test of interest to you, and then, click the Update button. These tests have been discussed below.

3.2.1 Java Business Transactions Test

The responsiveness of a transaction is the key determinant of user experience with that transaction; if response time increases, user experience deteriorates. To make users happy, a Java business transaction should be rapidly processed by each of the JVM nodes in its path. Processing bottlenecks on a single JVM node can slowdown/stall an entire business transaction or can cause serious transaction errors. This in turn can badly scar the experience of users. To avoid this, administrators should promptly identify slow/stalled/errored transactions, isolate the JVM node on which the slowness/error occurred, and uncover what caused the aberration on that node – is it owing to SQL queries executed by the node? Or is it because of external calls – eg., async calls,

SAP JCO calls, HTTP calls, etc. - made by that node? The **Java Business Transactions** test helps with this!

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

For this test to run and report metrics on Tomcat, you first need to BTM-enable the Tomcat JVM. To know how, refer to the Installing eG Java BTM on an Apache Tomcat Server topic in the *Java Business Transaction Monitoring* document.

Then, proceed to configure this test. Refer to the Java Business Transactions Test topic in the *Java Business Transaction Monitoring* document to know how to configure this test and learn about the metrics it reports.

3.2.2 JVM Details Test

This reveals the health of the Tomcat class loaders and daemon threads, and also indicates JVM availability.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms:

Parameter	Description
	• By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName,	These parameters appear only if the Measurement Mode is set to **War File**. In the

Parameter	Description
Password, and Confirm Password	UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total classes loaded	Refers to the total number of classes that have been loaded since the Java virtual machine started execution.	Number	
Classes loaded	Indicates the number of classes that have been loaded in the Java virtual machine since the last measurement period.	Number	Classes are fundamental to the design of Java programming language. Like many server applications, Tomcat installs a variety of class loaders (that is, classes that implement java.lang.ClassLoader) to allow different portions of the container, and the web applications running on the container, to have access to different repositories of available classes and resources. A consistent 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 web applications, thereby severely affecting application performance.
Classes unloaded	Indicates the total number of classes that have been unloaded in the Java virtual machine since the last measurement period.	Number	

Measurement	Description	Measurement Unit	Interpretation
Total daemon threads	Indicates the current number of live daemon threads.	Number	The main function of the daemon threads is to provide service to other threads, running in the same process as the daemon thread. If the value of this measure is equal to that of the Live_ threads measure, then JVM would stop executing; in other words, when the only remaining threads are the daemon threads, then JVM shuts down and so does Tomcat.
Live threads	Indicates the current number of live daemon and non-daemon threads.	Number	
Deadlock threads	The current number of deadlock threads	Number	Ideally, the value of this measure should be 0. A non-zero value indicates the occurrence of a deadlock. When a thread attempts to acquire a resource that is already locked by another thread, a deadlock situation arises.
Server uptime	The uptime of Java virtual machine	Mins	To ensure that the JVM is up and running for a long time, you need to make sure that at least one non-daemon thread is executing at all times. This is because, without any non-daemon threads, the daemon threads have no recipients for the service it provides; it hence brings down both the JVM and Tomcat.

3.2.3 Tomcat JVM Garbage Collection 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 Tomcat'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". This test reports the performance statistics pertaining to the JVM's garbage collection.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to

Parameter	Description
	com.sun.jmx.remote.protocol.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
No of collections	Indicates the number of garbage collections that were performed by the JVM since the last measurement period.	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 reducing the performance for applications, 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.
Time taken	Indicates the time taken for	Secs	A shorter GC execution time is desired

Measurement	Description	Measurement Unit	Interpretation
	GC execution, since the last measurement period.		to avoid issues in application performance and database connection bottlenecks.
Elapsed time	Indicates the percentage of time spent on GC execution.	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, you can fine-tune the GC activity.

3.2.4 JVM Memory Test

The memory system of the Java Virtual machines manages two types of memory, which are Heap and Non Heap. To define Heap, the Java virtual machine is a heap that is the runtime data area from which memory for all class instances and arrays are allocated. It is created at the Java Virtual Machine start-up and the memory for the objects is reclaimed by an automatic memory management systems known as Garbage collector. The Heap may be of fixed size or may be expanded and shrunk.

The Java virtual machine has a method area that is shared among all threads. This method area is called non heap. It stores per class structures such as runtime constant pool field and method data, and the code for methods and constructors. It is created at the Java Virtual Machine start up.

In order to ensure the uninterrupted functioning of the Tomcat server, sufficient memory resources need to be made available to the JVM memory pools. Inadequacies in memory allocations could lead to slow start up issues or intermittent application failures. This test periodically reports usage statistics of the JVM memory pools, so that memory bottlenecks are promptly detected and resolved.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.

Parameter	Description
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http:///{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Used	Indicates the amount of memory currently in use.	MB	
Free	Indicates the amount of unused memory currently available in a server.	MB	Ideally, this value should be high. An unusually low value for the available memory can indicate a memory bottleneck. Check the memory utilization of individual processes to figure out the process(es) that has (have) maximum memory consumption and look to tune their memory usages and allocations accordingly.
Initial	Indicates the initial amount of memory that Java virtual machine requests	MB	An unusually high usage of memory by the Java Virtual machine from the OS is a cause of concern. Further analysis

Measurement	Description	Measurement Unit	Interpretation
	from the operating system (OS) for memory management during startup.		is required to determine if specific applications or queries are consuming excess memory.
Pending objects	Indicates the number of objects in the heap/non-heap memory for which finalization is pending.	Number	Finalization allows an object to gracefully clean up after itself when it is being collected. When the garbage collector detects that an object is garbage, the garbage collector calls the object's Finalize method (if it exists) and then the object's memory is reclaimed - in other words, the garbage collector frees the memory allocated to the object. This measure is available only for the Heap_Memory and Nonheap_Memory descriptors of this test. A high value for this measure indicates that a chunk of heap / non-heap memory (as the case may be) in the JVM is awaiting reclamation. Object finalization facilitates efficient memory re-use, and thus ensures that the JVM memory pools do not run out of memory. If the value of this measure grows continuously, it could imply that objects are not releasing the memory resources properly. This in turn could be because the Finalize method does not exist for some objects, or there could be a bottleneck in the invocation of the method. Either way, thorough investigation can only provide pointers to the source of the problem.

3.3 Tests Disabled by Default for a Tomcat Server

The tests discussed above are enabled by default for a Tomcat server. In addition to these tests, the eG agent can be optionally configured to execute a few other tests that report critical statistics related to CPU usage, thread usage, and the uptime of the Tomcat JVM. To enable one/more of these tests, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, select *Tomcat* as the **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.

To enable one/more of these tests, open the agents – tests configuration page using the Agents -> Tests -> Configure menu sequence, select Tomcat from the Component type list, scroll down the test list that appears to view the disabled tests section, select the check box corresponding to the JVM test of interest to you, and then, click the Update button. These tests have been discussed below.

3.3.1 Java Business Transactions Test

The responsiveness of a transaction is the key determinant of user experience with that transaction; if response time increases, user experience deteriorates. To make users happy, a Java business transaction should be rapidly processed by each of the JVM nodes in its path. Processing bottlenecks on a single JVM node can slowdown/stall an entire business transaction or can cause serious transaction errors. This in turn can badly scar the experience of users. To avoid this, administrators should promptly identify slow/stalled/errored transactions, isolate the JVM node on which the slowness/error occurred, and uncover what caused the aberration on that node – is it owing to SQL queries executed by the node? Or is it because of external calls – eg., async calls, SAP JCO calls, HTTP calls, etc. - made by that node? The **Java Business Transactions** test helps with this!

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

For this test to run and report metrics on Tomcat, you first need to BTM-enable the Tomcat JVM. To know how, refer to the Installing eG Java BTM on an Apache Tomcat Server topic in the *Java Business Transaction Monitoring* document.

Then, proceed to configure this test. Refer to the Java Business Transactions Test topic in the *Java Business Transaction Monitoring* document to know how to configure this test and learn about the metrics it reports.

3.3.2 JVM Details Test

This reveals the health of the Tomcat class loaders and daemon threads, and also indicates JVM availability.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement Mode is set to **JMX**. Here, specify the port at which the **JMX** listens for requests from remote hosts. Ensure that you specify

Parameter	Description
	the same port that you configured in the *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total classes loaded	Refers to the total number of classes that have been loaded since the Java virtual machine started execution.	Number	
Classes loaded	Indicates the number of classes that have been loaded in the Java virtual machine since the last measurement period.	Number	Classes are fundamental to the design of Java programming language. Like many server applications, Tomcat installs a variety of class loaders (that is, classes that implement java.lang.ClassLoader) to allow different portions of the container, and the web applications running on the container, to have access to different repositories of available classes and resources. A consistent 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 web applications, thereby severely affecting application performance.
Classes unloaded	Indicates the total number of classes that have been unloaded in the Java virtual machine since the last measurement period.	Number	
Total daemon threads	Indicates the current number of live daemon threads.	Number	The main function of the daemon threads is to provide service to other threads, running in the same process as the daemon thread. If the value of this measure is equal to that of the Live_ threads measure, then JVM would stop executing; in other words, when the only remaining threads are the daemon threads, then JVM shuts down and so does Tomcat.
Live threads	Indicates the current number of live daemon and non-daemon threads.	Number	
Deadlock threads	The current number of deadlock threads	Number	Ideally, the value of this measure should be 0. A non-zero value indicates

Measurement	Description	Measurement Unit	Interpretation
			the occurrence of a deadlock. When a thread attempts to acquire a resource that is already locked by another thread, a deadlock situation arises.
Server uptime	The uptime of Java virtual machine	Mins	To ensure that the JVM is up and running for a long time, you need to make sure that at least one non-daemon thread is executing at all times. This is because, without any non-daemon threads, the daemon threads have no recipients for the service it provides; it hence brings down both the JVM and Tomcat.

3.3.3 Tomcat JVM Garbage Collection 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 Tomcat'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". This test reports the performance statistics pertaining to the JVM's garbage collection.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms:

Parameter	Description
	• By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName,	These parameters appear only if the Measurement Mode is set to **War File**. In the

Parameter	Description
Password, and Confirm Password	UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
No of collections	Indicates the number of garbage collections that were performed by the JVM since the last measurement period.	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 reducing the performance for applications, 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.
Time taken	Indicates the time taken for GC execution, since the last measurement period.	Secs	A shorter GC execution time is desired to avoid issues in application performance and database connection bottlenecks.
Elapsed time	Indicates the percentage of time spent on GC execution.	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, you can fine-tune the GC activity.

3.3.4 JVM Memory Test

The memory system of the Java Virtual machines manages two types of memory, which are Heap and Non Heap. To define Heap, the Java virtual machine is a heap that is the runtime data area from which memory for all class instances and arrays are allocated. It is created at the Java Virtual Machine start-up and the memory for the objects is reclaimed by an automatic memory management systems known as Garbage collector. The Heap may be of fixed size or may be expanded and shrunk.

The Java virtual machine has a method area that is shared among all threads. This method area is called non heap. It stores per class structures such as runtime constant pool field and method data, and the code for methods and constructors. It is created at the Java Virtual Machine start up.

In order to ensure the uninterrupted functioning of the Tomcat server, sufficient memory resources need to be made available to the JVM memory pools. Inadequacies in memory allocations could lead to slow start up issues or intermittent application failures. This test periodically reports usage statistics of the JVM memory pools, so that memory bottlenecks are promptly detected and resolved.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target

Parameter	Description
	Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.

Parameter	Description
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Used	Indicates the amount of memory currently in use.	MB	
Free	Indicates the amount of unused memory currently available in a server.	MB	Ideally, this value should be high. An unusually low value for the available memory can indicate a memory bottleneck. Check the memory utilization of individual processes to figure out the process(es) that has (have) maximum memory consumption and look to tune their memory usages and allocations accordingly.
Initial	Indicates the initial amount of memory that Java virtual machine requests from the operating system (OS) for memory management during startup.	MB	An unusually high usage of memory by the Java Virtual machine from the OS is a cause of concern. Further analysis is required to determine if specific applications or queries are consuming excess memory.
Pending objects	Indicates the number of objects in the heap/non-heap memory for which finalization is pending.	Number	Finalization allows an object to gracefully clean up after itself when it is being collected. When the garbage collector detects that an object is garbage, the garbage collector calls the object's Finalize method (if it exists) and then the object's memory is reclaimed - in other words, the garbage collector frees the memory allocated to the object.

Measurement	Description	Measurement Unit	Interpretation
			This measure is available only for the Heap_Memory and Nonheap_Memory descriptors of this test. A high value for this measure indicates that a chunk of heap / non-heap memory (as the case may be) in the JVM is awaiting reclamation. Object finalization facilitates efficient memory re-use, and thus ensures that the JVM memory pools do not run out of memory. If the value of this measure grows continuously, it could imply that objects are not releasing the memory resources properly. This in turn could be because the Finalize method does not exist for some objects, or there could be a bottleneck in the invocation of the method. Either way, thorough investigation can only provide pointers to the source of the problem.

3.4 Tests Disabled by Default for a Tomcat Server

The tests discussed above are enabled by default for a Tomcat server. In addition to these tests, the eG agent can be optionally configured to execute a few other tests that report critical statistics related to CPU usage, thread usage, and the uptime of the Tomcat JVM. To enable one/more of these tests, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, select *Tomcat* as the **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.

To enable one/more of these tests, open the agents – tests configuration page using the Agents -> Tests -> Configure menu sequence, select Tomcat from the Component type list, scroll down the test list that appears to view the disabled tests section, select the check box corresponding to the JVM test of interest to you, and then, click the Update button. These tests have been discussed below.

3.4.1 Java Business Transactions Test

The responsiveness of a transaction is the key determinant of user experience with that transaction; if response time increases, user experience deteriorates. To make users happy, a Java business transaction should be rapidly processed by each of the JVM nodes in its path. Processing bottlenecks on a single JVM node can slowdown/stall an entire business transaction or can cause serious transaction errors. This in turn can badly scar the experience of users. To avoid this, administrators should promptly identify slow/stalled/errored transactions, isolate the JVM node on which the slowness/error occurred, and uncover what caused the aberration on that node – is it owing to SQL queries executed by the node? Or is it because of external calls – eg., async calls, SAP JCO calls, HTTP calls, etc. - made by that node? The **Java Business Transactions** test helps with this!

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

For this test to run and report metrics on Tomcat, you first need to BTM-enable the Tomcat JVM. To know how, refer to the Installing eG Java BTM on an Apache Tomcat Server topic in the *Java Business Transaction Monitoring* document.

Then, proceed to configure this test. Refer to the Java Business Transactions Test topic in the *Java Business Transaction Monitoring* document to know how to configure this test and learn about the metrics it reports.

3.4.2 JVM Details Test

This reveals the health of the Tomcat class loaders and daemon threads, and also indicates JVM availability.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.

Parameter	Description
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http:///{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total classes loaded	Refers to the total number of classes that have been loaded since the Java virtual machine started execution.	Number	
Classes loaded	Indicates the number of classes that have been loaded in the Java virtual machine since the last measurement period.	Number	Classes are fundamental to the design of Java programming language. Like many server applications, Tomcat installs a variety of class loaders (that is, classes that implement java.lang.ClassLoader) to allow different portions of the container, and the web applications running on the container, to have access to different repositories of available classes and resources. A consistent decrease in the

Measurement	Description	Measurement Unit	Interpretation
			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 web applications, thereby severely affecting application performance.
Classes unloaded	Indicates the total number of classes that have been unloaded in the Java virtual machine since the last measurement period.	Number	
Total daemon threads	Indicates the current number of live daemon threads.	Number	The main function of the daemon threads is to provide service to other threads, running in the same process as the daemon thread. If the value of this measure is equal to that of the Live_ threads measure, then JVM would stop executing; in other words, when the only remaining threads are the daemon threads, then JVM shuts down and so does Tomcat.
Live threads	Indicates the current number of live daemon and non-daemon threads.	Number	
Deadlock threads	The current number of deadlock threads	Number	Ideally, the value of this measure should be 0. A non-zero value indicates the occurrence of a deadlock. When a thread attempts to acquire a resource that is already locked by another thread, a deadlock situation arises.
Server uptime	The uptime of Java virtual machine	Mins	To ensure that the JVM is up and running for a long time, you need to make sure that at least one non-daemon thread is executing at all times. This is because, without any non-daemon threads, the daemon threads have no recipients for the service it provides; it hence brings down both the JVM and Tomcat.

3.4.3 Tomcat JVM Garbage Collection 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 Tomcat'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". This test reports the performance statistics pertaining to the JVM's garbage collection.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires

Parameter	Description
Password, and Confirm Password	**authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
No of collections	Indicates the number of garbage collections that	Number	If adequate memory is not allotted to the JVM, then the value of this

Measurement	Description	Measurement Unit	Interpretation
	were performed by the JVM since the last measurement period.		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 reducing the performance for applications, 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.
Time taken	Indicates the time taken for GC execution, since the last measurement period.	Secs	A shorter GC execution time is desired to avoid issues in application performance and database connection bottlenecks.
Elapsed time	Indicates the percentage of time spent on GC execution.	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, you can fine-tune the GC activity.

3.4.4 JVM Memory Test

The memory system of the Java Virtual machines manages two types of memory, which are Heap and Non Heap. To define Heap, the Java virtual machine is a heap that is the runtime data area from which memory for all class instances and arrays are allocated. It is created at the Java Virtual Machine start-up and the memory for the objects is reclaimed by an automatic memory management systems known as Garbage collector. The Heap may be of fixed size or may be expanded and shrunk.

The Java virtual machine has a method area that is shared among all threads. This method area is called non heap. It stores per class structures such as runtime constant pool field and method data, and the code for methods and constructors. It is created at the Java Virtual Machine start up.

In order to ensure the uninterrupted functioning of the Tomcat server, sufficient memory resources need to be made available to the JVM memory pools. Inadequacies in memory allocations could lead to slow start up issues or intermittent application failures. This test periodically reports usage statistics of the JVM memory pools, so that memory bottlenecks are promptly detected and resolved.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.

Parameter	Description
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Used	Indicates the amount of memory currently in use.	MB	
Free	Indicates the amount of unused memory currently available in a server.	MB	Ideally, this value should be high. An unusually low value for the available memory can indicate a memory bottleneck. Check the memory

Measurement	Description	Measurement Unit	Interpretation
			utilization of individual processes to figure out the process(es) that has (have) maximum memory consumption and look to tune their memory usages and allocations accordingly.
Initial	Indicates the initial amount of memory that Java virtual machine requests from the operating system (OS) for memory management during startup.	MB	An unusually high usage of memory by the Java Virtual machine from the OS is a cause of concern. Further analysis is required to determine if specific applications or queries are consuming excess memory.
Pending objects	Indicates the number of objects in the heap/non-heap memory for which finalization is pending.	Number	Finalization allows an object to gracefully clean up after itself when it is being collected. When the garbage collector detects that an object is garbage, the garbage collector calls the object's Finalize method (if it exists) and then the object's memory is reclaimed - in other words, the garbage collector frees the memory allocated to the object. This measure is available only for the Heap_Memory and Nonheap_Memory descriptors of this test. A high value for this measure indicates that a chunk of heap / non-heap memory (as the case may be) in the JVM is awaiting reclamation. Object finalization facilitates efficient memory re-use, and thus ensures that the JVM memory pools do not run out of memory. If the value of this measure grows continuously, it could imply that objects are not releasing the memory resources properly. This in turn could be because the Finalize method does

Measurement	Description	Measurement Unit	Interpretation
			not exist for some objects, or there could be a bottleneck in the invocation of the method. Either way, thorough investigation can only provide pointers to the source of the problem.

3.5 Tests Disabled by Default for a Tomcat Server

The tests discussed above are enabled by default for a Tomcat server. In addition to these tests, the eG agent can be optionally configured to execute a few other tests that report critical statistics related to CPU usage, thread usage, and the uptime of the Tomcat JVM. To enable one/more of these tests, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, select *Tomcat* as the **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.

To enable one/more of these tests, open the agents – tests configuration page using the Agents -> Tests -> Configure menu sequence, select Tomcat from the Component type list, scroll down the test list that appears to view the disabled tests section, select the check box corresponding to the JVM test of interest to you, and then, click the Update button. These tests have been discussed below.

3.5.1 Java Business Transactions Test

The responsiveness of a transaction is the key determinant of user experience with that transaction; if response time increases, user experience deteriorates. To make users happy, a Java business transaction should be rapidly processed by each of the JVM nodes in its path. Processing bottlenecks on a single JVM node can slowdown/stall an entire business transaction or can cause serious transaction errors. This in turn can badly scar the experience of users. To avoid this, administrators should promptly identify slow/stalled/errored transactions, isolate the JVM node on which the slowness/error occurred, and uncover what caused the aberration on that node – is it owing to SQL queries executed by the node? Or is it because of external calls – eg., async calls, SAP JCO calls, HTTP calls, etc. - made by that node? The **Java Business Transactions** test helps with this!

This test runs on a BTM-enabled JVM in an IT infrastructure, tracks all the transaction requests received by that JVM, and groups requests based on user-configured pattern specifications. For each transaction pattern, the test then computes and reports the average time taken by that JVM

node to respond to the transaction requests of that pattern. In the process, the test identifies the slow/stalled transactions of that pattern, and reports the count of such transactions and their responsiveness. Detailed diagnostics provided by the test accurately pinpoint the exact transaction URLs that are slow/stalled, the total round-trip time of each transaction, and also indicate when such transaction requests were received by that node. The slowest transaction in the group can thus be identified.

For this test to run and report metrics on Tomcat, you first need to BTM-enable the Tomcat JVM. To know how, refer to the Installing eG Java BTM on an Apache Tomcat Server topic in the *Java Business Transaction Monitoring* document.

Then, proceed to configure this test. Refer to the Java Business Transactions Test topic in the *Java Business Transaction Monitoring* document to know how to configure this test and learn about the metrics it reports.

3.5.2 JVM Details Test

This reveals the health of the Tomcat class loaders and daemon threads, and also indicates JVM availability.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target

Parameter	Description
	Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.

Parameter	Description
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total classes loaded	Refers to the total number of classes that have been loaded since the Java virtual machine started execution.	Number	
Classes loaded	Indicates the number of classes that have been loaded in the Java virtual machine since the last measurement period.	Number	Classes are fundamental to the design of Java programming language. Like many server applications, Tomcat installs a variety of class loaders (that is, classes that implement java.lang.ClassLoader) to allow different portions of the container, and the web applications running on the container, to have access to different repositories of available classes and resources. A consistent 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 web applications, thereby severely affecting application performance.
Classes unloaded	Indicates the total number of classes that have been unloaded in the Java virtual machine since the last measurement period.	Number	
Total daemon threads	Indicates the current number of live daemon threads.	Number	The main function of the daemon threads is to provide service to other threads, running in the same process as the daemon thread. If the value of this measure is equal to that of the Live_ threads measure, then JVM would stop executing; in other words, when the

Measurement	Description	Measurement Unit	Interpretation
Live threads	Indicates the current number of live daemon and non-daemon threads.	Number	only remaining threads are the daemon threads, then JVM shuts down and so does Tomcat.
Deadlock threads	The current number of deadlock threads	Number	Ideally, the value of this measure should be 0. A non-zero value indicates the occurrence of a deadlock. When a thread attempts to acquire a resource that is already locked by another thread, a deadlock situation arises.
Server uptime	The uptime of Java virtual machine	Mins	To ensure that the JVM is up and running for a long time, you need to make sure that at least one non-daemon thread is executing at all times. This is because, without any non-daemon threads, the daemon threads have no recipients for the service it provides; it hence brings down both the JVM and Tomcat.

3.5.3 Tomcat JVM Garbage Collection 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 Tomcat'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". This test reports the performance statistics pertaining to the JVM's garbage collection.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.

Parameter	Description
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http:///{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
No of collections	Indicates the number of garbage collections that were performed by the JVM since the last measurement period.	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 reducing the performance for applications, 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.
Time taken	Indicates the time taken for GC execution, since the last measurement period.	Secs	A shorter GC execution time is desired to avoid issues in application performance and database connection

Measurement	Description	Measurement Unit	Interpretation
			bottlenecks.
Elapsed time	Indicates the percentage of time spent on GC execution.	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, you can fine-tune the GC activity.

3.5.4 JVM Memory Test

The memory system of the Java Virtual machines manages two types of memory, which are Heap and Non Heap. To define Heap, the Java virtual machine is a heap that is the runtime data area from which memory for all class instances and arrays are allocated. It is created at the Java Virtual Machine start-up and the memory for the objects is reclaimed by an automatic memory management systems known as Garbage collector. The Heap may be of fixed size or may be expanded and shrunk.

The Java virtual machine has a method area that is shared among all threads. This method area is called non heap. It stores per class structures such as runtime constant pool field and method data, and the code for methods and constructors. It is created at the Java Virtual Machine start up.

In order to ensure the uninterrupted functioning of the Tomcat server, sufficient memory resources need to be made available to the JVM memory pools. Inadequacies in memory allocations could lead to slow start up issues or intermittent application failures. This test periodically reports usage statistics of the JVM memory pools, so that memory bottlenecks are promptly detected and resolved.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

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

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.

Parameter	Description
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the

Parameter	Description
	URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Used	Indicates the amount of memory currently in use.	MB	
Free	Indicates the amount of unused memory currently available in a server.	MB	Ideally, this value should be high. An unusually low value for the available memory can indicate a memory bottleneck. Check the memory utilization of individual processes to figure out the process(es) that has (have) maximum memory consumption and look to tune their memory usages and allocations accordingly.
Initial	Indicates the initial amount of memory that Java virtual machine requests from the operating system (OS) for memory management during startup.	MB	An unusually high usage of memory by the Java Virtual machine from the OS is a cause of concern. Further analysis is required to determine if specific applications or queries are consuming excess memory.

Measurement	Description	Measurement Unit	Interpretation
Pending objects	Indicates the number of objects in the heap/non-heap memory for which finalization is pending.	Number	Finalization allows an object to gracefully clean up after itself when it is being collected. When the garbage collector detects that an object is garbage, the garbage collector calls the object's Finalize method (if it exists) and then the object's memory is reclaimed - in other words, the garbage collector frees the memory allocated to the object. This measure is available only for the Heap_Memory and Nonheap_Memory descriptors of this test. A high value for this measure indicates that a chunk of heap / non-heap memory (as the case may be) in the JVM is awaiting reclamation. Object finalization facilitates efficient memory re-use, and thus ensures that the JVM memory pools do not run out of memory. If the value of this measure grows continuously, it could imply that objects are not releasing the memory resources properly. This in turn could be because the Finalize method does not exist for some objects, or there could be a bottleneck in the invocation of the method. Either way, thorough investigation can only provide pointers to the source of the problem.

3.6 Tests Disabled by Default for a Tomcat Server

The tests discussed above are enabled by default for a Tomcat server. In addition to these tests, the eG agent can be optionally configured to execute a few other tests that report critical statistics related to CPU usage, thread usage, and the uptime of the Tomcat JVM. To enable one/more of these tests, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, select *Tomcat* as the **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.

To enable one/more of these tests, open the agents – tests configuration page using the Agents -> Tests -> Configure menu sequence, select Tomcat from the Component type list, scroll down the test list that appears to view the disabled tests section, select the check box corresponding to the JVM test of interest to you, and then, click the Update button. These tests have been discussed below.

3.6.1 Java Business Transactions Test

The responsiveness of a transaction is the key determinant of user experience with that transaction; if response time increases, user experience deteriorates. To make users happy, a Java business transaction should be rapidly processed by each of the JVM nodes in its path. Processing bottlenecks on a single JVM node can slowdown/stall an entire business transaction or can cause serious transaction errors. This in turn can badly scar the experience of users. To avoid this, administrators should promptly identify slow/stalled/errored transactions, isolate the JVM node on which the slowness/error occurred, and uncover what caused the aberration on that node – is it owing to SQL queries executed by the node? Or is it because of external calls – eg., async calls, SAP JCO calls, HTTP calls, etc. - made by that node? The **Java Business Transactions** test helps with this!

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

For this test to run and report metrics on Tomcat, you first need to BTM-enable the Tomcat JVM. To know how, refer to the Installing eG Java BTM on an Apache Tomcat Server topic in the *Java Business Transaction Monitoring* document.

Then, proceed to configure this test. Refer to the Java Business Transactions Test topic in the *Java Business Transaction Monitoring* document to know how to configure this test and learn about the metrics it reports.

3.6.2 JVM Details Test

This reveals the health of the Tomcat class loaders and daemon threads, and also indicates JVM availability.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**.

Parameter	Description
	Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http:///{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total classes loaded	Refers to the total number of classes that have been loaded since the Java virtual machine started execution.	Number	

Measurement	Description	Measurement Unit	Interpretation
Classes loaded	Indicates the number of classes that have been loaded in the Java virtual machine since the last measurement period.	Number	Classes are fundamental to the design of Java programming language. Like many server applications, Tomcat installs a variety of class loaders (that is, classes that implement java.lang.ClassLoader) to allow different portions of the container, and the web applications running on the container, to have access to different repositories of available classes and resources. A consistent 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 web applications, thereby severely affecting application performance.
Classes unloaded	Indicates the total number of classes that have been unloaded in the Java virtual machine since the last measurement period.	Number	
Total daemon threads	Indicates the current number of live daemon threads.	Number	The main function of the daemon threads is to provide service to other threads, running in the same process as the daemon thread. If the value of this measure is equal to that of the Live_ threads measure, then JVM would stop executing; in other words, when the only remaining threads are the daemon threads, then JVM shuts down and so does Tomcat.
Live threads	Indicates the current number of live daemon and non-daemon threads.	Number	
Deadlock threads	The current number of deadlock threads	Number	Ideally, the value of this measure should be 0. A non-zero value indicates the occurrence of a deadlock. When a thread attempts to acquire a resource that is already locked by another thread, a deadlock situation arises.
Server uptime	The uptime of Java virtual machine	Mins	To ensure that the JVM is up and running for a long time, you need to make sure that at least one non-

Measurement	Description	Measurement Unit	Interpretation
			daemon thread is executing at all times. This is because, without any non-daemon threads, the daemon threads have no recipients for the service it provides; it hence brings down both the JVM and Tomcat.

3.6.3 Tomcat JVM Garbage Collection 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 Tomcat'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". This test reports the performance statistics pertaining to the JVM's garbage collection.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server.

Parameter	Description
	On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from

Parameter	Description
	Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
No of collections	Indicates the number of garbage collections that were performed by the JVM since the last measurement period.	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 reducing the performance for applications, 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.
Time taken	Indicates the time taken for GC execution, since the last measurement period.	Secs	A shorter GC execution time is desired to avoid issues in application performance and database connection bottlenecks.
Elapsed time	Indicates the percentage of time spent on GC execution.	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, you can fine-tune the GC activity.

3.6.4 JVM Memory Test

The memory system of the Java Virtual machines manages two types of memory, which are Heap and Non Heap. To define Heap, the Java virtual machine is a heap that is the runtime data area from which memory for all class instances and arrays are allocated. It is created at the Java Virtual

Machine start-up and the memory for the objects is reclaimed by an automatic memory management systems known as Garbage collector. The Heap may be of fixed size or may be expanded and shrunk.

The Java virtual machine has a method area that is shared among all threads. This method area is called non heap. It stores per class structures such as runtime constant pool field and method data, and the code for methods and constructors. It is created at the Java Virtual Machine start up.

In order to ensure the uninterrupted functioning of the Tomcat server, sufficient memory resources need to be made available to the JVM memory pools. Inadequacies in memory allocations could lead to slow start up issues or intermittent application failures. This test periodically reports usage statistics of the JVM memory pools, so that memory bottlenecks are promptly detected and resolved.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement Mode is set to **JMX**. Here, specify the port at which the **JMX** listens for requests from remote hosts. Ensure that you specify

Parameter	Description
	the same port that you configured in the *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Used	Indicates the amount of memory currently in use.	MB	
Free	Indicates the amount of unused memory currently available in a server.	MB	Ideally, this value should be high. An unusually low value for the available memory can indicate a memory bottleneck. Check the memory utilization of individual processes to figure out the process(es) that has (have) maximum memory consumption and look to tune their memory usages and allocations accordingly.
Initial	Indicates the initial amount of memory that Java virtual machine requests from the operating system (OS) for memory management during startup.	MB	An unusually high usage of memory by the Java Virtual machine from the OS is a cause of concern. Further analysis is required to determine if specific applications or queries are consuming excess memory.
Pending objects	Indicates the number of objects in the heap/non-heap memory for which finalization is pending.	Number	Finalization allows an object to gracefully clean up after itself when it is being collected. When the garbage collector detects that an object is garbage, the garbage collector calls the object's Finalize method (if it exists) and then the object's memory is reclaimed - in other words, the garbage collector frees the memory allocated to the object. This measure is available only for the Heap_Memory and Nonheap_Memory descriptors of this test. A high value for this measure indicates that a chunk of heap / non-heap memory (as the case may be) in the JVM is awaiting reclamation. Object finalization

Measurement	Description	Measurement Unit	Interpretation
			facilitates efficient memory re-use, and thus ensures that the JVM memory pools do not run out of memory. If the value of this measure grows continuously, it could imply that objects are not releasing the memory resources properly. This in turn could be because the Finalize method does not exist for some objects, or there could be a bottleneck in the invocation of the method. Either way, thorough investigation can only provide pointers to the source of the problem.

3.7 The Tomcat Container Layer

The tests mapped to this layer monitor the health of the Tomcat container by reporting:

- How well the Tomcat JVM uses the memory resources available to it
- How effectively the Tomcat cache has been utilized;
- Whether/not the Tomcat thread pools have been used optimally;
- How efficiently each Tomcat connector processes the data traffic to it.

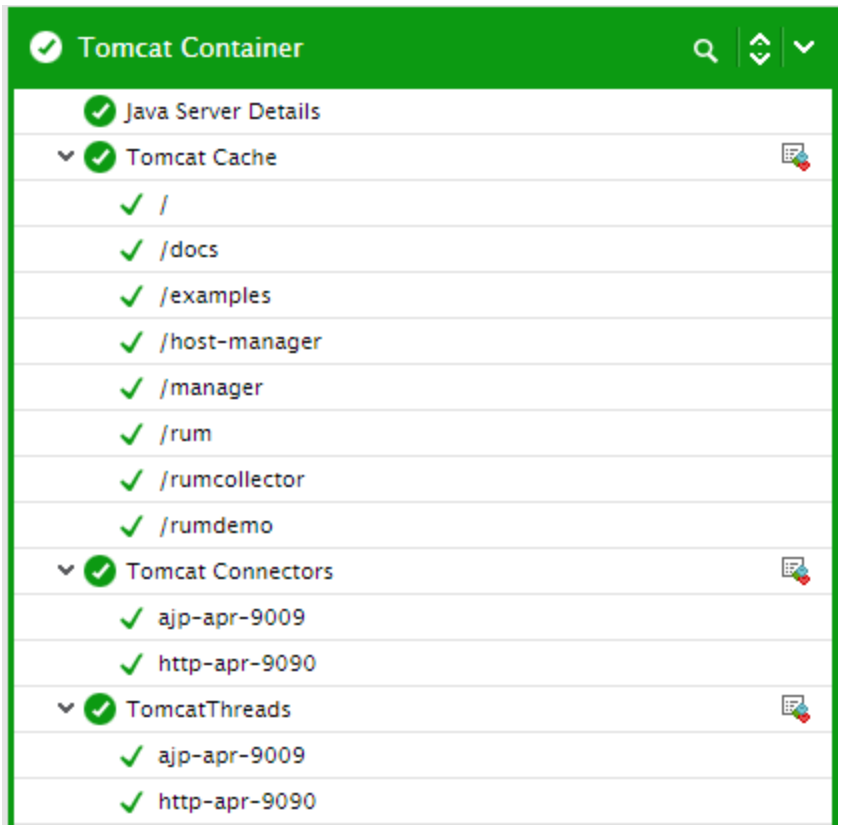

Figure 3.5: The tests mapped to the Tomcat Container layer

3.7.1 Java Server Details Test

This test reports the performance statistics pertaining to the Java Virtual Machine (JVM) running on a Tomcat server.

Target of the test : A Tomcat server

Agent deploying the test : An internal 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 host on which the Tomcat server is running.
Port	The port on which the specified Tomcat server is listening for HTTP requests.

Parameter	Description
EGURI	The **Java Server Details** test makes use of a file named **EgPerfTest.jsp** to generate measures. By default, this file is located in the <EG_INSTALL_DIR>/lib directory. To execute this test, move this file to one of the application directories of the Tomcat server. On Unix, you can execute the script /opt/egurkha/bin/setup_jserver to do this. While configuring this test, specify the location of the application directory where the **EgPerfTest.jsp** file resides, in the EGURI text box, in the following format: *http://<Ipaddress:portNo/directory name>*. For example, assume that the **EgPerfTest.jsp** is available in the **JavaTest** directory of the host 192.168.10.57:7077. Therefore, EGURI would be: *http://192.168.10.57:7077/JavaTest*.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Active threads	Indicates the number of active threads in the JVM.	Number	A high value for this measure is indicative of a high load on the JVM.
Total memory usage	Indicates the total memory available in the JVM.	MB	A high value indicates a high processing capability of the JVM. Watch for increasing memory usage over time, which could indicate a memory leak in one or more applications hosted on the application server.
Free memory	Indicates the unused memory in the JVM.	MB	A very low value of free memory is an indication of high memory utilization on the JVM.

3.7.2 Tomcat Cache Test

A well-sized cache could go a long way in reducing direct disk accesses and the resultant processing overheads. The **Tomcat Cache** test reveals whether/not the Tomcat server cache is effectively utilized.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.

Parameter	Description
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http:///{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Access count	The number of cache accesses since the last measurement period.	Number	
Hits count	Indicates the number of requests served from the cache since the last measurement period.	Number	Physical I/O takes a significant amount of time, and also increases the CPU resources required. The server configuration should therefore ensure that the required information is available on the memory. A low value of this measure indicates that physical I/O is greater.
Cache size	Indicates the current size of cache.	KB	A high value ensures higher cache hits and lower physical I/O.

3.7.3 Tomcat Threads Test

The Http connector in the Tomcat Web server represents a Connector component that supports the HTTP/1.1 protocol, which enables Catalina to function as a stand-alone web server, besides its ability to execute servlets and JSP pages. A particular instance of this component listens for connections on a specific TCP port number on the server to perform request processing and response creation. You can have one or more such Connectors configured to form a part of a single Service with each forwarding service to the associated Engine.

As soon as your server startup time is up, this Connector will create a number of request processing threads which are based on the value configured for the *minSpareThreads* attribute. Each incoming request requires a thread for the duration of that request. If more simultaneous requests are received than can be handled by the currently available request processing threads, additional threads will be created up to the configured maximum (the value of the *maxThreads* attribute). If still more simultaneous requests are received, they are stacked up inside the server socket created by the Connector, up to the configured maximum (the value of the *acceptCount* attribute). Any further simultaneous requests will receive "connection refused" errors, until resources are available to process them.

Continuous monitoring of thread pools is imperative to ensure the smooth transaction of business on the Tomcat web server. The **Tomcat Threads** test periodically observes thread pool usage to proactively determine inadequacies in the allocation of threads to the pool, and to predict future thread requirements.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each thread pool managed by the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms:

Parameter	Description
	• By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName,	These parameters appear only if the Measurement Mode is set to **War File**. In the

Parameter	Description
Password, and Confirm Password	UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Thread count	Indicates the number of threads that are currently assigned to a connector.	Number	
Threads busy	Indicates the number of threads which are currently busy in processing requests.	Number	A high value for this measure indicates that there is hectic activity at the connector level.
Max threads	Indicates the maximum number of threads that this pool can contain.	Number	In the Tomcat server, a *maxThreads* attribute can be set for every connector to indicate the maximum number of simultaneous requests that can be handled by that connector. This measure reports this *maxThreads* value. The default *maxThreads* value for a connector is 200. If the value of the *Thread count* measure grows close to the value of this measure, it indicates that the number of threads in the pool needs to be increased for effective operation of this application. Alternatively, you may also consider changing the maximum number of threads that a pool can

Measurement	Description	Measurement Unit	Interpretation
			contain. However, exercise caution when altering the maximum thread count, as a very high thread count can cause the app to slowdown from excessive memory usage. Likewise, if the maximum thread count is set too low, it will cause requests to block or timeout.
Max spare threads	Indicates the maximum number of spare threads that can exist in a thread pool.	Number	Ideally for a connector, the default value is set to 50, which will determine the maximum number of unused request processing threads that will be allowed to exist until the thread pool starts stopping the unnecessary threads.
Min spare threads	Indicates the minimum number of spare threads that are currently available for processing requests.	Number	Generally, for a connector, the default value for this parameter is set to 4, which is less than the value set in *maxThreads*. This attribute will determine the number of request processing threads that will be created when this Connector is first started. The connector will also make sure it has the specified number of idle processing threads available.

3.7.4 Tomcat Connectors Test

The Http connector in the Tomcat Web server, represents a Connector component that supports the HTTP/1.1 protocol, which enables Catalina to function as a stand-alone web server, besides its ability to execute servlets and JSP pages. A particular instance of this component listens for connections on a specific TCP port number on the server to perform request processing and response creation. You can have one or more such Connectors configured to form a part of a single Service with each forwarding service to the associated engine.

This observes the data traffic on each connector and measures the connector's processing ability.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every connector configured on the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.

Parameter	Description
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Request count	Indicates the number of requests received by the connector, since the last measurement period.	Number	
Avg processing time	Indicates the average time taken by the connector to process requests.	Secs	This measure is a clear indicator of the Connector health. Ideally, this value should be low. A very high value indicates processing bottlenecks.
Data sent	Reports the data that was sent by the connector since the last	KB	

Measurement	Description	Measurement Unit	Interpretation
	measurement period.		
Data received	Reports the data that was received by the connector since the last measurement period.	KB	Both the *Data sent* and *Data received* measures together indicate the load on the Tomcat server.
Error count	Indicates the number of errors that were reported by the connector since the last measurement period.	Number	Ideally, the value of this measure should be 0. A non-zero value warrants further investigation.

3.7.5 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 continuosly 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 Tomcat 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.

Parameter	Description
Host	The IP address of the host for which this test is to be configured.
Port	Refers to the port at which the specified host listens to.
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 Section **3.7.5.1**.
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.

3.7.5.1 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:

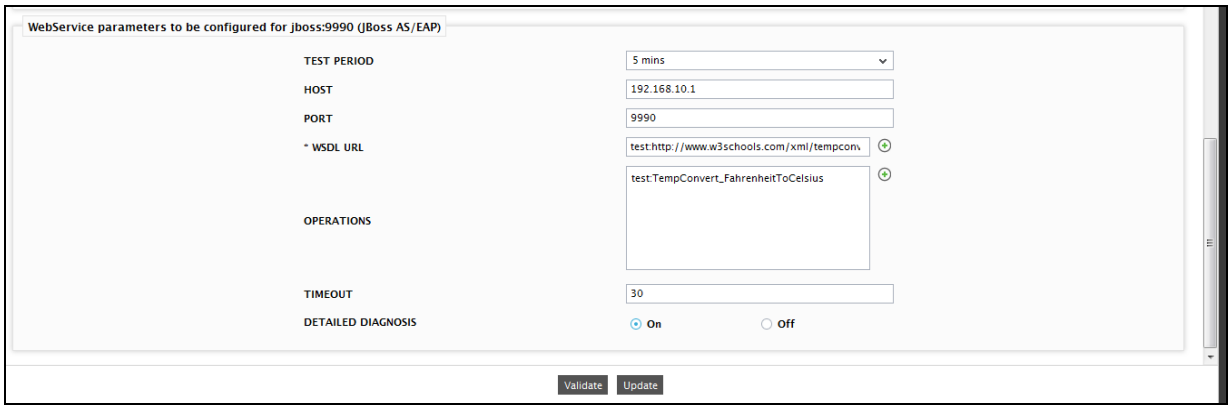

Figure 3.6: Configuring the WebService test

1. Click on the encircled '+' button alongside the WSDL URL text box. Figure 3.7 will then appear.

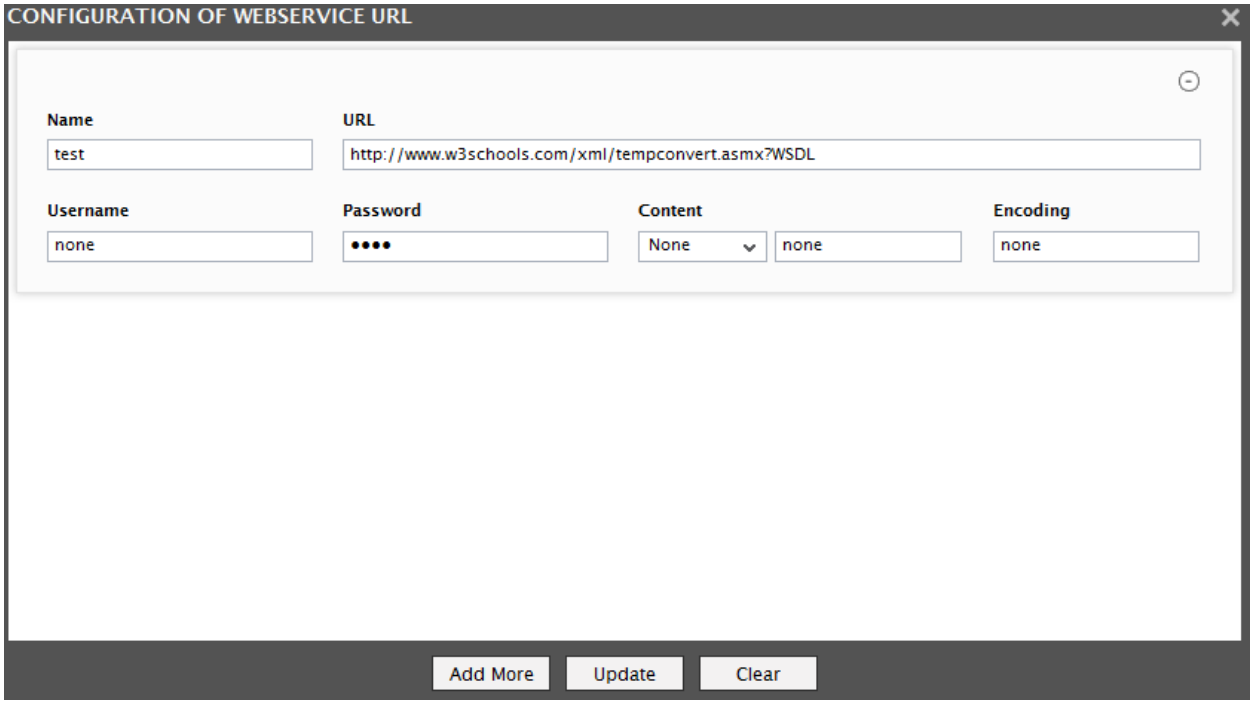

Figure 3.7: The WebService URL Configuration page

2. Specify the following in Figure 3.7:

   - **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:
     - *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 3.6. 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.

3.7.5.2 Configuring Multiple Operations for Monitoring - WebServiceTest

By default, the **WebService** 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 3.6. Figure 3.8 will then appear.

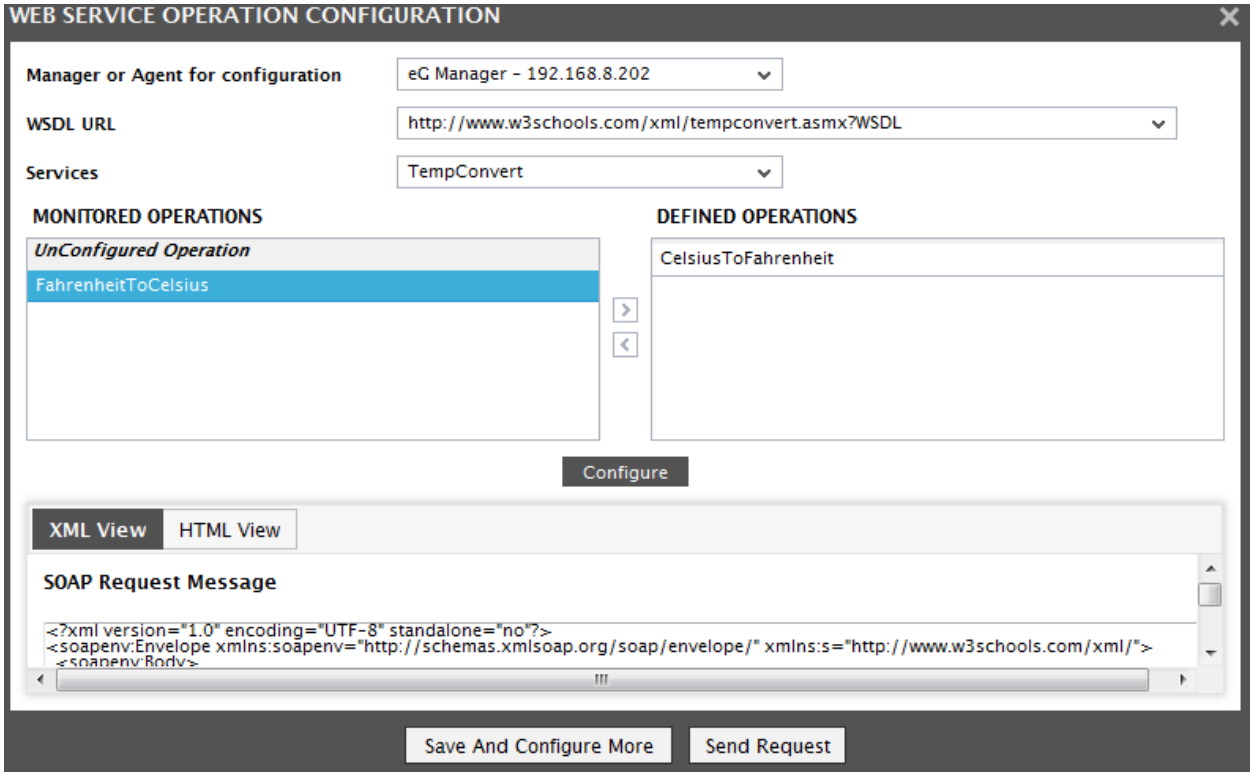

Figure 3.8: Configuring the Web Service Operation

2. Specify the following in Figure 3.8:

   - **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 3.6). 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 3.9. Each time the test is executed, this value will be provided as an input to the chosen operation.

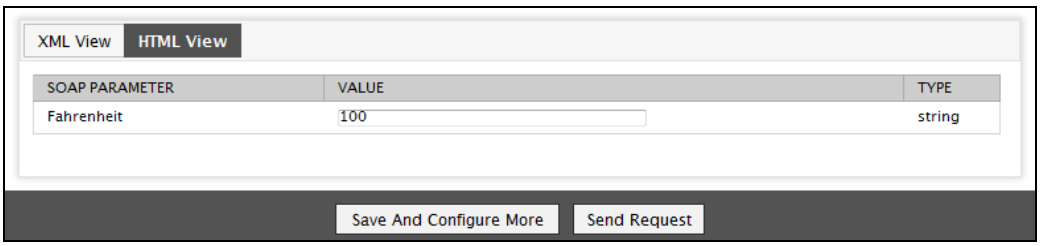

Figure 3.9: Specifying the value for the chosen operation

- Click the *Save and Cofigure 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 3.10 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>

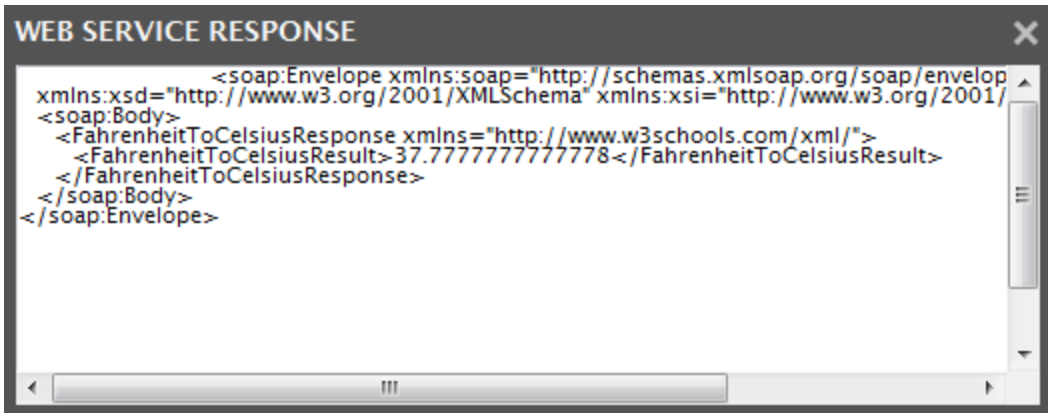

Figure 3.10: The value that appears when the operation is performed successfully

- If you have specified an invalid value, then a message as follows will be displayed in the pop up window:

  <FahrenheitToCelsiusResult>Error</FahrenheitToCelsiusResult>

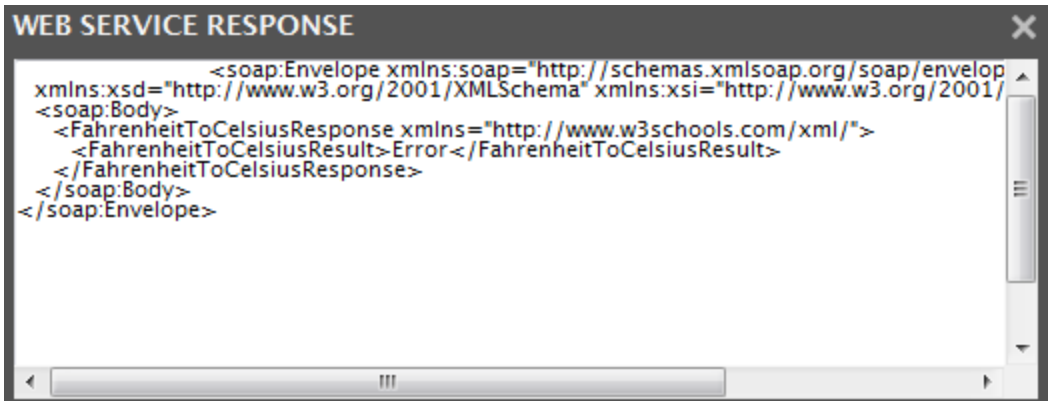

Figure 3.11: An Error appearing during value conversion

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.

3. Similarly, you can configure multiple Operations by clicking the **Configure** button in Figure 3.8. To remove an operation, select the operation from the **MONITORED OPERATION** list and click the **>** button.
4. Once **Save and Configure More** button is clicked, you will return to the test configuration page (see Figure 3.7). 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.

3.8 The Tomcat Application Layer

The tests associated with this layer report useful statistics related to the web applications deployed on the Tomcat server. From these metrics, the state of the sessions to each application can be ascertained, the request processing ability of the servlets can be measured, and the count of JSPs loaded and reloaded can be determined.

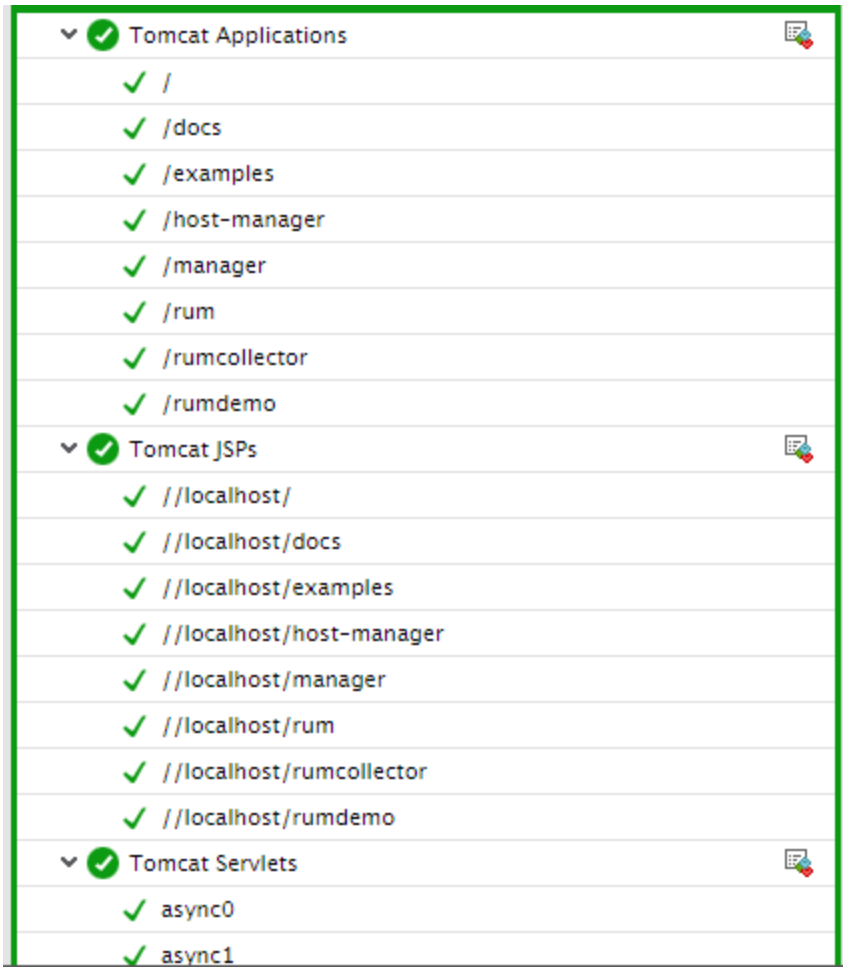

Figure 3.12: Tests associated with the Tomcat Applications layer

3.8.1 Tomcat Applications Test

The Manager element on the Tomcat server represents the session manager that will be used to create and maintain HTTP sessions as requested by the associated web application. A session state is maintained for a period, typically starting with user interactions and ending with last interaction.

Besides memory management, manager also looks into various aspects including CPU usage, network load that large sessions introduce. The scalability and performance of any web based application depends upon how well HTTP sessions are transmitted and managed over network.

An application must guarantee that a user's session state is properly maintained in order to exhibit correct behavior for that user. The **Tomcat Applications** test closely monitors the state of sessions to every application on the Tomcat server.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every application on the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).

Parameter	Description
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the MBean attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Active sessions	Indicates the sessions that	Number	A high value for this measure is

Measurement	Description	Measurement Unit	Interpretation
	are currently active.		indicative of heavy load on the Tomcat server.
Max sessions	Indicates the maximum number of active sessions that can be allowed at one time on the Tomcat server.	Number	
Expired sessions	Indicates the number of sessions that expired since the last measurement period.	Number	A large number of expired sessions could hint at the need to reset the TIMEOUT period for sessions on the Tomcat server.
Rejected sessions	Indicates the number of sessions that were rejected since the last measurement period.	Number	It is imperative to check if there is any loss of session state happening due to network congestion. Therefore, if the value of this measure is unusually high, the reasons behind the unusual occurrence would have to be investigated. You can also consider increasing the maximum active session count.

3.8.2 Tomcat Jsps Test

The TomcatJsps test monitors the performance of the JSPs used by each of the applications deployed on Tomcat.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every application deployed on the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract

Parameter	Description
	measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Load count	Indicates the number of JSPs that have been loaded into the web application since the last measurement period.	Number	
Reload count	Indicates the number of JSPs that have been reloaded since the last measurement period.	Number	As one of the rules, by default, the JSP container will automatically reload the translated class of a JSP page whenever the page is retranslated, a class called by the page is modified (presuming the class was loaded by the OC4J JSP container class loader, not the system class loader), or any page in the same application is reloaded. Using this measure therefore, you can keep track of the changes made to the JSP pages.

3.8.3 Tomcat Servlets Test

A servlet is a small Java program that runs within a web server. These servlets receive and respond to requests from web clients, usually across HTTP.

Typically, these servlets run inside a servlet container that handles recurring multiple requests. The performance and reliability of any web based application depends upon how well the requests from web clients are managed. The TomcatServlets test enables administrators to assess the performance of servlets on the basis of their request handling capabilities.

Target of the test : A Tomcat Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every servlet on the Tomcat 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	Refers to the port at which the specified host listens to.
Measurement Mode	This test can extract metrics from Tomcat using either of the following mechanisms: • By deploying the **egtomcat.war** file in the <EG_INSTALL_DIR>\lib directory of the eG agent host on the Tomcat server; • By contacting the Java runtime (JRE) of Tomcat via JMX To configure the test to use **egtomcat.war** file, first select the **War file** option. Then, refer to the Section **Chapter 2** to know how to deploy the WAR file on the target Tomcat server. On the other hand, if you want the test to use JMX instead, then first, select the **JMX** option. Then, follow the procedure detailed in the Section **Chapter 2** to configure the test to use jmx. By default, the **JMX** option is chosen here.
JMX Remote Port	This parameter appears only if the Measurement 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 *catalina.sh* (or *catalina.bat*) file in the <CATALINA_HOME_DIR>/bin folder of the target Tomcat server (refer to the Section **Chapter 2** for more details).

Parameter	Description
JMX User, JMX Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **JMX**. If JMX equires **authentication only** (but no security), then ensure that the JMX User and JMX 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 Section **Chapter 2**. Confirm the password by retyping it in the Confirm Password text box.
JNDIName	This parameter appears only if the Measurement 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 registery using a different lookup name, then you can change this default value to reflect the same.
JMX Provider	This parameter appears only if the Measurement Mode is set to **JMX**. This test uses a JMX Provider to access the *MBean* attributes of the target Java application and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to **com.sun.jmx.remote.protocol**.
SSL	This parameter appears only if the Measurement Mode is set to **War File**. Indicate **Yes** if the Tomcat server is SSL-enabled.
URL	This parameter appears only if the Measurement Mode is set to **War File**. Specify the URL of the managed Tomcat server to enable the test to connect to it and extract measures from it. The URL specification should be of the format: *http://{TomcatIP}:{TomcatPort}*.
UserName, Password, and Confirm Password	These parameters appear only if the Measurement Mode is set to **War File**. In the UserName text box, specify a name of a user who has been assigned the **Manager role** on the Tomcat server to be monitored; these users are typically allowed to control web applications deployed on the Tomcat server. Specify the Password of this user, and confirm the password by retyping it in the Confirm Password text box.
EncryptPass	This parameter appears only if the measurement modeis set to **War File**. Select **Yes** if you want to encrypt the password.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from Tomcat. If there is no response from Tomcat beyond the configured duration, the test will timeout. By default, this is set to **240** seconds.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Request count	Indicates the number of	Number	

Measurement	Description	Measurement Unit	Interpretation
	requests handled by the servlet since the last measurement period.		
Avg processing time	Indicates the average time taken by the servlet to process the requests since the last measurement period.	Secs	If the value of this measure increases consistently, it indicates a gradual deterioration in the server performance.
Max processing time	Indicates the maximum time taken by the servlet to process the requests.	Secs	If the time taken is significantly high, check for the errors or bottlenecks that interfere with the servlet's normal operation.
Min processing time	Indicates the minimum time taken by the servlet to process the current request.	Secs	Minimum time taken to process the request is indicative of error free execution of service by the servlet.
Error count	Indicates the errors encountered by the servlet, when a request is processed	Number	Ideally, the value for this measure should be 0. An error reported by this measure is a cause of concern; an analysis of the underlying reasons is hence necessitated.

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.