Monitoring IBM WebSphere Liberty Server

eG Innovations Product Documentation

www.eginnovations.com

Total Performance Visibility

Table of Contents

Table of Figures

Chapter 1: Introduction

IBM WebSphere Liberty is a fast, dynamic, easy-to-use Java EE application server. The IBM WebSphere Libety is not only ideal for developers but also ready for production, on-premise or in the cloud. IBM WebSphere Liberty is a combination of IBM technology and open source software, with fast startup times (<2 seconds), no server restarts to pick up changes, and a simple XML configuration. The IBM WebSphere Liberty supports two models of application deployment:

- Deploy an application by dropping it into the *dropins* directory.
- Deploy an application by adding it to the server configuration file.

Like any other application server, performance setbacks suffered by the IBM WebSphere Liberty server too can affect the availability of critical services offered. To avoid such an eventuality, you need to continuously monitor the performance of the IBM WebSphere Liberty server.

The eG Enterprise suite facilitates 24x7 monitoring of the IBM WebSphere Liberty server and proactive alerting of probable error conditions detected on the server.

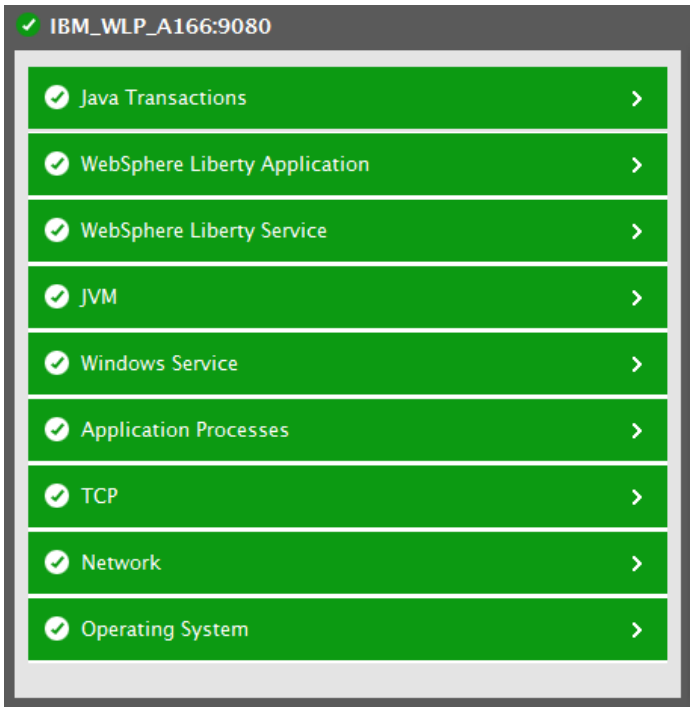


Figure 1.1: The layer model of the IBM WebSphere Liberty server

Each layer of Figure 1 above is mapped to a variety of tests that execute on the IBM WebSphere Liberty server and collect performance statistics that reveal the following:

- What is the rate at which each servlet is hit with requests?
- What is the average time taken by each servlet to respond to requests? Which servlet is taking too long to respond to requests?
- How many active sessions are currently accessing each WAR file deployed on the target IBM WebSphere Liberty server?
- How many invalidated sessions are accessing each WAR file deployed on the target IBM WebSphere Liberty server?
- How many sessions are accessing each WAR file per second?
- What is the current state of each web application deployed on the target IBM WebSphere Liberty server?
- What is the current state of each queue on the target IBM WebSphere Liberty server?
- What is the maximum configured size of each queue?
- What is the current queue depth of each queue? Which queue is latent and is unable to process I/O requests quickly?
- Are get operations allowed on each queue?
- What is the maximum configures size of each topic?
- Are get operations allowed on each topic?
- What is the current depth of each topic? Which topic is latent and is unable to process I/O requests quickly?
- What is the size of each thread pool?
- Haw many active threads are there in each thread pool?
- How well the CPU is utilized by the JVM engine?
- How well does the JVM engine manage memory?
- What is the uptime of the JVM engine?
- How many classes have been loaded/unloaded from memory of the JVM engine?
- Did garbage collection take too long to complete? If so, which memory pools spent too much time in garbage collection?
- Are too many threads in waiting state in the JVM?
- Which threads in the JVM are consuming CPU?

The chapters that follow elaborately discusses the tests, metrics, and how they are collected.

Chapter 2: How does eG Enterprise monitor the IBM WebSphere Liberty?

eG Enterprise monitors the IBM WebSphere Liberty server in an agent-based manner only. The eG agent uses the JMX architecture supported by the IBM WebSphere Liberty server to gather the required metrics. The eG agent should be installed on the same host where the target IBM WebSphere Liberty server is installed.

2.1 Pre-Requisites for monitoring the IBM WebSphere Liberty installed on Windows environments

The eG agent uses the JMX architecture to collect the required metrics. For the eG agent to collect the metrics, the following pre-requisites need to be fulfilled:

1. The IBM WebSphere Liberty server should be up and running.
2. To monitor the target IBM WebSphere Liberty server and to enable the eG agent to collect metrics from the target IBM WebSphere Liberty server, the following 'Liberty features' should be enabled. *Features* are the units of functionality by which administrators control the pieces of the runtime environment that are loaded into a particular server.

   - monitor-1.0
   - localconnector-1.0

   To enable the features, do the following:

   - Open the **server.xml** file that is available in the **<liberty_ server_ install_ dir>\ usr\servers\server_name** folder using a text editor.
   - Append the following lines within the **<featureManager>** section of the **server.xml** file:

     ```
     <feature>monitor-1.0</feature>
     <feature>localConnector-1.0</feature>
     ```

   - Finally, save the file.

   Once the local connector is enabled, a *com.ibm.ws.jmx.local.address* file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses the contents of this file to collect the required metrics from the target IBM WebSphere Liberty server. For the eG agent to use this file, it is necessary to specify the exact path to this file against the **Service URL Path** parameter while configuring the tests applicable to the IBM WebSphere Liberty server.

2.2 Pre-Requisites for monitoring the IBM WebSphere Liberty server installed on Linux environments

Apart from the pre-requisites mentioned in Section **2.1**the following pre-requisite needs to be fulfilled if an IBM WebSphere Liberty server is installed on Linux environments:

For the eG agent to collect the required metrics, the eG agent communicates with the target IBM WebSphere Liberty server with the privileges of a user who is able to connect via SSH. The user should be within the user group that is privileged to access the target IBM WebSphere Liberty server.

Chapter 3: How to Monitor IBM WebSphere Liberty Using eG Enterprise?

The broad steps for monitoring IBM WebSphere Liberty using eG Enterprise are as follows:

- Managing the IBM WebSphere Liberty server
- Configuring the tests

These steps have been discussed in this topic.

3.1 Managing the IBM WebSphere Liberty server

The IBM WebSphere Liberty server cannot be automatically discovered by eG Enterprise. This implies that you will have to manually add the server into the eG Enterprise system to manage it. Follow the steps below to achieve the same:

1. Follow the *Components - > Add/Modify* menu sequence in the *Infrastructure* tile menu of the eG admin interface.
2. Next, select *IBM WebSphere Liberty* from the **Component type** drop-down and then click the **Add New Component** button.
3. When Figure 3.1 appears, provide the **Host IP/Name** of the IBM WebSphere Liberty server that you want to manage.

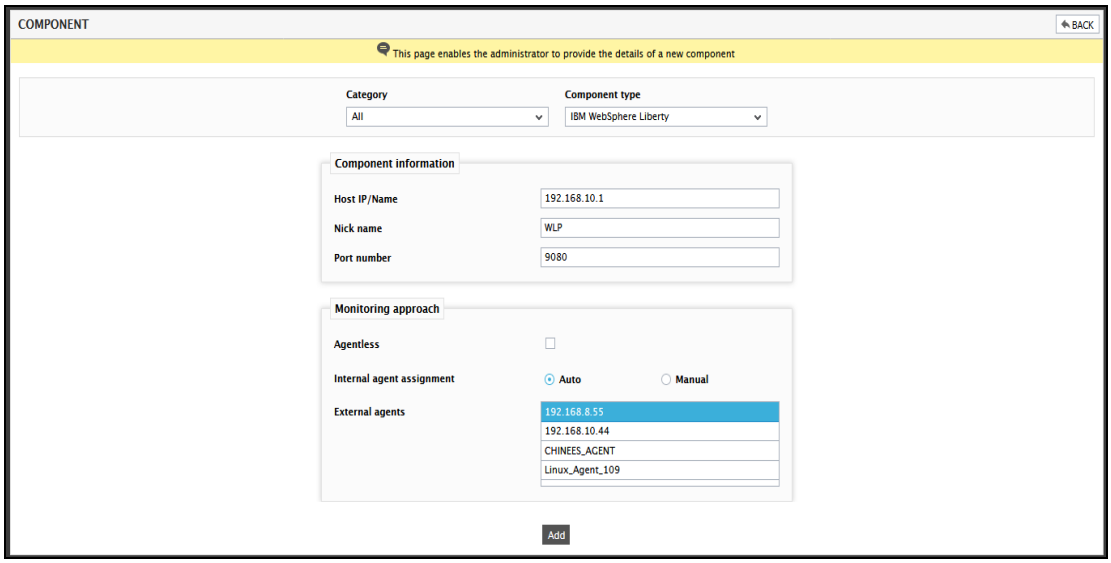


Figure 3.1: Adding an IBM WebSphere Liberty server

4. Then, provide a **Nick name** for the server.
5. The **Port** number will be set as *9080* by default. If the IBM WebSphere Liberty server is listening on a different port in your environment, then override this default setting.
6. By default, the *IBM WebSphere Liberty* server is monitored in an agent-based manner. Therefore, just pick an external agent from the **External agents** list box and click the **Add** button to add the component for monitoring.
7. Finally, click the **Signout** button at the right, top corner of the eG admin interface to sign out.

3.2 Configuring the tests

When you try to sign out of the eG admin interface, a **LIST OF UNCONFIGURED TESTS** page will appear, revealing the list of tests mapped to the IBM WebSphere Liberty server that require manual configuration:

Performance		WLP:9080
Java Classes	JMX Connection to JVM	JVM CPU Usage
JVM File Descriptors	JVM Garbage Collections	JVM Threads
JVM Memory Pool Garbage Collections	JVM Memory Usage	Windows Services
JVM Uptime	Processes	WebSphere Liberty Connection Pools
WebSphere Liberty Queues	WebSphere Liberty Servlets	WebSphere Liberty Sessions
WebSphere Liberty Thread Pools	WebSphere Liberty Topics	WebSphere Liberty Web Applications

Figure 3.2: The list of tests that are required to be configured manually

Click on the *WebSphere Liberty Connection Pools* test in Figure 3.1 to configure it. Figure 3.2 then appears.

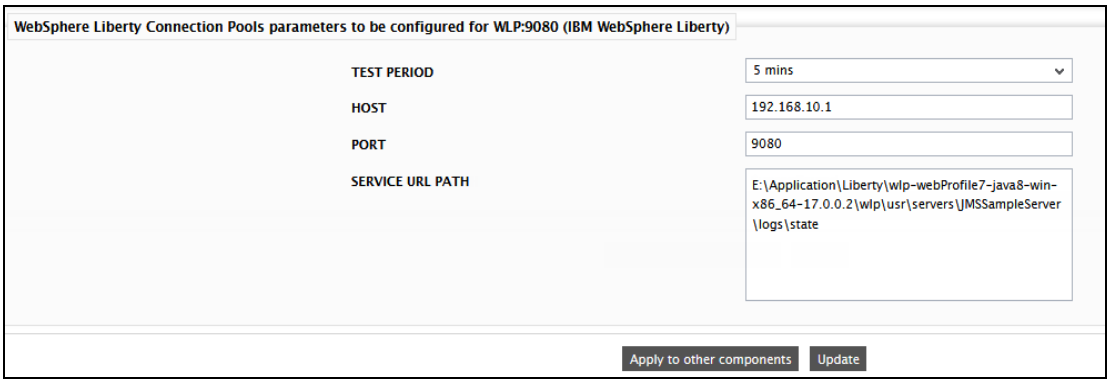


Figure 3.3: Configuring the WebSphere Liberty Connection Pools test

To know how to specify the parameters, refer to Section **4.2.1**.

When you signout of the eG admin interface, you will be prompted to configure the *Processes* and *Windows Services* test. To know how to configure the *Processes* test and the *Windows Services* test, refer to the *Monitoring Unix and Windows Servers* document.

After configuring the *Processes* and *Windows Services* test, sign out of the eG administrative interface. Then, login to the eG monitoring console to view the state of and metrics reported by the specialized monitoring model that eG Enterprise offers for the IBM WebSphere Liberty server.

Chapter 4: Monitoring the IBM WebSphere Liberty Server

This chapter takes you inside the top 4 layers of the IBM WebSphere Liberty server monitoring model, and discusses the tests and measures mapped to each layer.

4.1 The JVM Layer

The JVM layer measures the overall health of the JVM engine by reporting statistics related to the following:

- How well the CPU is utilized by the JVM engine?
- How well does the JVM engine manage memory?
- What is the uptime of the JVM engine?
- How many classes have been loaded/unloaded from memory?
- Did garbage collection take too long to complete? If so, which memory pools spent too much time in garbage collection?
- Are too many threads in waiting state in the JVM?
- Which threads are consuming CPU?

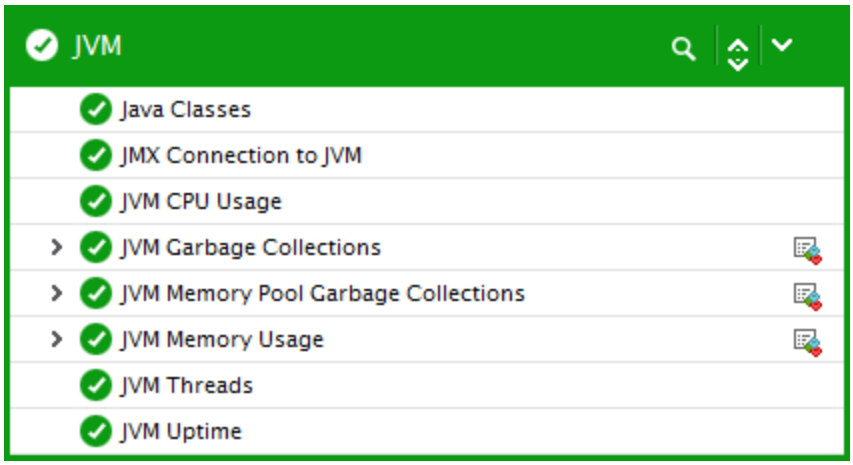


Figure 4.1: The tests associated with the JVM layer

4.1.1 Java Classes Test

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

Target of the test : An IBM WebSphere Liberty server

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 host for which the test is to be configured.
Port	The port number 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 Service URL 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.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.
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.
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 (see page 1).
SNMP Version	This parameter appears only if the Mode is set to **SNMP**. The default selection in the

Parameter	Description
	SNMP version list is v1. However, for this test to work, you have to select SNMP v2 or v3 from this list, depending upon which version of SNMP is in use in the target environment.
SNMP Community	This parameter appears only if the Mode is set to **SNMP**. Here, specify the SNMP community name that the test uses to communicate with the mail server. The default is public. This parameter is specific to SNMP v1 and v2 only. Therefore, if the SNMP version chosen is v3, then this parameter will not appear.
User name	This parameter appears only when v3 is selected as the SNMP version. 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 user name. This parameter once again appears only if the snmpversion selected is v3.
Confirm password	Confirm the Authpass by retyping it here
Authtype	This parameter too appears only if v3 is selected as the SNMPVersion. From the authtype list box, choose the authentication algorithm using which SNMP v3 converts the specified username and password into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • MD5 – Message Digest Algorithm • SHA – Secure Hash Algorithm
Encryptflag	This flag appears only when v3 is selected as the SNMPVersion. By default, the eG

Parameter	Description
	agent does not encrypt SNMP requests. Accordingly, the 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 the 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 is applicable only if mode is set to SNMP. By default, in an IT environment, all data transmission occurs over UDP. Some environments however, may be specifically configured to offload a fraction of the data traffic – for instance, certain types of data traffic or traffic pertaining to specific components – to other protocols like TCP, so as to prevent UDP overloads. In such environments, you can instruct the eG agent to conduct the SNMP data traffic related to the monitored target over TCP (and not UDP). For this, set this flag to **Yes**. By default, this flag is set to **No**.

Measurements made by the test

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

Measurement	Description	Measurement Unit	Interpretation
Classes unloaded	Indicates the number of classes currently unloaded from memory.	Number	unchecked, critical resources/classes could be rendered inaccessible to the application, thereby severely affecting its performance.
Total classes loaded	Indicates the total number of classes loaded into memory since the JVM started, including those subsequently unloaded.	Number	

4.1.2 JMX Connection to JVM

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 : An IBM WebSphere Liberty server

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 host for which the test is to be configured.
Port	The port number at which the specified host listens to.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.

Parameter	Description
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; • 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.

4.1.3 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:

This test will not report metrics if the Mode parameter of the test is set to **SNMP**.

Target of the test : An IBM WebSphere Liberty server

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 host for which the test is to be configured.
Port	The port number 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 Service URL 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.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.
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.
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 (see page 1).
SNMP Version	This parameter appears only if the Mode is set to SNMP. The default selection in the

Parameter	Description
	SNMP version list is v1. However, for this test to work, you have to select SNMP v2 or v3 from this list, depending upon which version of SNMP is in use in the target environment.
SNMP Community	This parameter appears only if the Mode is set to SNMP. Here, specify the SNMP community name that the test uses to communicate with the mail server. The default is public. This parameter is specific to SNMP v1 and v2 only. Therefore, if the SNMP version chosen is v3, then this parameter will not appear.
User name	This parameter appears only when v3 is selected as the SNMP version. 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 user name. This parameter once again appears only if the snmpversion selected is v3.
Confirm password	Confirm the Authpass by retyping it here
Authtype	This parameter too appears only if v3 is selected as the snmpversion. From the authtype list box, choose the authentication algorithm using which SNMP v3 converts the specified username and password into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options: • MD5 – Message Digest Algorithm • SHA – Secure Hash Algorithm
Encryptflag	This flag appears only when v3 is selected as the snmpversion. By default, the eG

Parameter	Description
	agent does not encrypt SNMP requests. Accordingly, the flag is set to No by default. To ensure that SNMP requests sent by the eG agent are encrypted, select the Yes option. • 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 is applicable only if 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, this 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 this 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_SIR>\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.

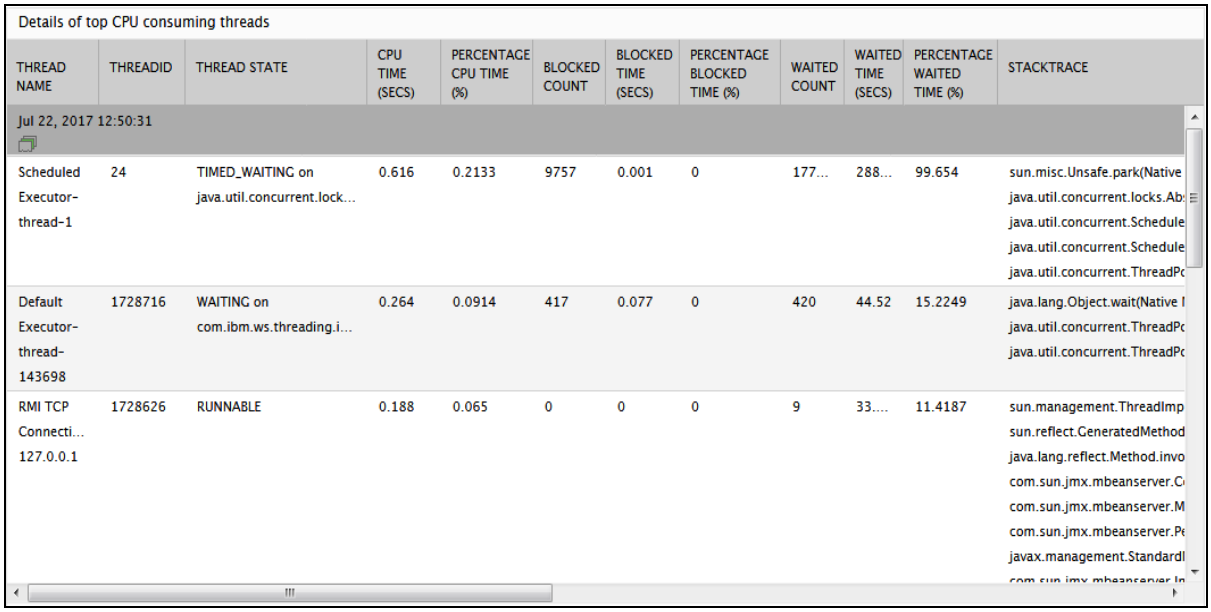
Parameter	Description
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To 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 top CPU consuming threads

THREAD NAME	THREADID	THREAD STATE	CPU TIME (SECS)	PERCENTAGE CPU TIME (%)	BLOCKED COUNT	BLOCKED TIME (SECS)	PERCENTAGE BLOCKED TIME (%)	WAITED COUNT	WAITED TIME (SECS)	PERCENTAGE WAITED TIME (%)	STACKTRACE
Jul 22, 2017 12:50:31											
Scheduled Executor-thread-1	24	TIMED_WAITING on java.util.concurrent.lock...	0.616	0.2133	9757	0.001	0	177...	288...	99.654	sun.misc.Unsafe.park(Native java.util.concurrent.locks.Ab java.util.concurrent.Schedule java.util.concurrent.Schedule java.util.concurrent.ThreadPo
Default Executor-thread-143698	1728716	WAITING on com.ibm.ws.threading.i...	0.264	0.0914	417	0.077	0	420	44.52	15.2249	java.lang.Object.wait(Native I java.util.concurrent.ThreadPo java.util.concurrent.ThreadPo
RMI TCP Connecti... 127.0.0.1	1728626	RUNNABLE	0.188	0.065	0	0	0	9	33....	11.4187	sun.management.ThreadImp sun.reflect.GeneratedMethod java.lang.reflect.Method.invo com.sun.jmx.mbeanserver.C com.sun.jmx.mbeanserver.M com.sun.jmx.mbeanserver.Pe javax.management.StandardI

Figure 4.2: The detailed diagnosis of the CPU utilization of JVM measure

4.1.4 JVM File Descriptors Test

This test reports useful statistics pertaining to file descriptors.

Note:

This test will work only if the target Java application uses the JDK/JRE offered by one of the following vendors only: Oracle, Sun, OpenJDK. IBM JDK/JRE is not supported.

Target of the test : An IBM WebSphere Liberty server

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 host for which the test is to be configured.
Port	The port number at which the specified host listens to.
Service URL path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.
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 parameter 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	

4.1.5 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 : An IBM WebSphere Liberty server

Agent deploying the test : An internal 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 host for which the test is to be configured.
Port	The port number 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 Service URL 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.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.
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.
SNMPPort	This parameter appears only if the Mode is set to SNMP. Here specify the port number

Parameter	Description
	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 (see page 1).
SNMP Version	This parameter appears only if the Mode is set to SNMP. The default selection in the SNMP version list is v1. However, for this test to work, you have to select SNMP v2 or v3 from this list, depending upon which version of SNMP is in use in the target environment.
SNMP Community	This parameter appears only if the Mode is set to SNMP. Here, specify the SNMP community name that the test uses to communicate with the mail server. The default is public. This parameter is specific to SNMP v1 and v2 only. Therefore, if the SNMP version chosen is v3, then this parameter will not appear.
User name	This parameter appears only when v3 is selected as the SNMP version. 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 user name. 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:

Parameter	Description
	• 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 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 the 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 is applicable only if 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	Secs	Ideally, the value of both these measures should be low. This is because, the garbage collection (GC)

Measurement	Description	Measurement Unit	Interpretation
	garbage collection operation.		activity tends to suspend the operations of the application until such time that GC ends. Longer the GC time, longer it would take for the application to resume its functions. To minimize the impact of GC on application performance, it is best to ensure that GC activity does not take too long to complete.
Percent of time spent by JVM for garbage collection	Indicates the percentage of time spent by this garbage collector on garbage collection during the last measurement period.	Percent	

4.1.6 JVM Leak Suspects Test

Java implements automatic garbage collection (GC); once you stop using an object, you can depend on the garbage collector to collect it. To stop using an object, you need to eliminate all references to it. However, when a program never stops using an object by keeping a permanent reference to it, memory leaks occur. For example, let's consider the piece of code below:

```
1 import java.util.ArrayList;
2 import java.util.List;
3
4 class MemoryLeakDemo {
5     private static List<Integer> memoryLeakArea = new ArrayList<Integer>();
6
7     public static void main(String [] args) {
8
9         int iteration = 0;
10
11        // This infinite loop would eventually run out of memory
12        while (true) {
13            // Add number of the current iteration to the list.
14            Integer payload = new Integer(iteration);
15            memoryLeakArea.add(payload);
16            iteration++;
17        }
18    }
19 }
```

Figure 4.3: A sample code

In the example above, we continue adding new elements to the list memoryLeakArea without ever removing them. In addition, we keep references to the memoryLeakArea, thereby preventing GC from collecting the list itself. So although there is GC available, it cannot help because we are still using memory. The more time passes the more memory we use, which in effect requires an infinite amount memory for this program to continue running. When no more memory is remaining, an

OutOfMemoryError alert will be thrown and generate an exception like this: Exception in thread "main" java.lang.OutOfMemoryError: Java heap space at MemoryLeakDemo.main (MemoryLeakDemo.java:14)

Typically, such alerts signal a potential memory leak!

A memory leak can diminish the performance of your mission-critical Java applications by reducing the amount of available memory. Eventually, in the worst case, it may cause the application to crash due to thrashing. To avert such unwarranted application failures, it is imperative that memory leaks are detected at the earliest and the objects responsible for them accurately isolated. This is where, the **JVM Leak Suspects** test helps! This test continuously monitors the JVM heap usage and promptly alerts administrators when memory usage crosses a configured limit. The detailed diagnostics of the test will then lead you to the classes that are consuming memory excessively, thereby pointing you to those classes that may have caused the leak.

Note:

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

- The test should be executed in an agent-based manner only.
- The target Java application should use the JDK/JRE offered by one of the following vendors only: Oracle, Sun, OpenJDK. IBM JDK/JRE is not supported.
- The monitored Java application should use JDK/JRE 1.6 or higher.
- For this test to run and report metrics, the eG agent install user should be the same as the Java application (or) Java web/application server install user.
- By default , this test programmatically dumps a heap dump (.hprof files) in the folder <EG_ AGENT_INSTALL_DIR>\agent\logs folder. To enable the eG agent to read/analyse such files, you need to add the eG agent install user to the Java application (or) Java web/application server install user group. If this is not done, then the dump files will be created, but will not be processed by the eG agent, thus ending up unnecessarily occupying disk space (note that .hprof files are normally 1-5 GB in size).

Warning:

This test is CPU & Memory intensive and can cause slowness to the underlying application. It is hence NOT advisable to enable this test on production environments. It is ideally suited for Development and Staging environments.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the desired **Component type**, set

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

Target of the test : An IBM WebSphere Liberty server

Agent deploying the test : An internal agent

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

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed. By default, this is set to 1 hour.
Host	The host for which the test is to be configured.
Port	The port number at which the specified host listens to
Service URL path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.
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.
PCT Heap Limit	This test counts all those classes that are consuming memory beyond the limit (in percentage) specified against PCT Heap Limit as 'memory leak suspects'. This count is reported as the value of the Leak suspect classes measure. By default, 30 (%) is the PCT Heap Limit. This implies that the test, by default, reports each class that consumes over 30% of the Allocated heap memory as a Leak suspect class. Such classes are listed as part of detailed diagnostics.
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 this parameter.

Parameter	Description
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: • The eG manager license should allow the detailed diagnosis capability • Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Allocated Heap Memory	Indicates the total amount of memory space occupied by the objects that are currently loaded on to the JVM.	MB	
Leak suspected classes	Indicates the number of classes that are memory leak suspects.	Number	Use the detailed diagnosis of this measure to know which classes are using more memory than the configured pct heap limit. Remember that all applications/classes that throw OutofMemory exceptions need not be guilty of leaking memory. Such an exception can occur even if a class requires more memory for normal functioning. To distinguish between a memory leak and an application that simply needs more memory, we need to look at the "peak load" concept. When program has just started no users have yet used it, and as a result it typically needs much less memory then when thousands

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>of users are interacting with it. Thus, measuring memory usage immediately after a program starts is not the best way to gauge how much memory it needs! To measure how much memory an application needs, memory size measurements should be taken at the time of peak load—when it is most heavily used. Therefore, it is good practice to check the memory usage of the ‘suspected classes’ at the time of peak load to determine whether they are indeed leaking memory or not.</td></tr>
<tr><td>Number of objects</td><td>Indicates the number of objects present in the JVM.</td><td>Number</td><td>Use the detailed diagnosis of this measure to view the top-20 classes in the JVM in terms of memory usage.</td></tr>
<tr><td>Number of classes</td><td>Indicates the number of classes currently present in the JVM.</td><td>Number</td><td></td></tr>
<tr><td>Number of class loaders</td><td>Indicates the number of class loaders currently present in the JVM.</td><td>Number</td><td></td></tr>
<tr><td>Number of GC roots</td><td>Indicate the number of GC roots currently present in the JVM.</td><td>Number</td><td>A garbage collection root is an object that is accessible from outside the heap. The following reasons make an object a GC root:<table><tr><th>Reason</th><th>Description</th></tr><tr><td>System Class</td><td>Class loaded by bootstrap/system class loader. For example, everything from the rt.jar like java.util.</td></tr><tr><td>JNI Local</td><td>Local variable in</td></tr></table></td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Reason</th><th>Description</th></tr>
<tr><td></td><td>native code, such as user defined JNI code or JVM internal code</td></tr>
<tr><td>JNI Global</td><td>Global variable in native code, such as user defined JNI code or JVM internal code</td></tr>
<tr><td>Thread Block</td><td>Object referred to from a currently active thread block</td></tr>
<tr><td>Thread</td><td>A started, but not stopped, thread</td></tr>
<tr><td>Busy Monitor</td><td>Everything that has called wait () or notify() or that is synchronized. For example, by calling synchronized(Object) or by entering a synchronized method. Static method means class, non-static method means object</td></tr>
<tr><td>Java Local</td><td>Local variable. For example, input parameters or locally created objects of methods that are still in the</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Reason</th><th>Description</th></tr>
<tr><td></td><td>stack of a thread.</td></tr>
<tr><td>Native Stack</td><td>In or out parameters in native code, such as user defined JNI code or JVM internal code. or reflection.</td></tr>
<tr><td>Finalizer</td><td>An object which is in a queue awaiting its finalizer to be run.</td></tr>
<tr><td>Unfinalized</td><td>An object which has a finalize method, but has not been finalized and is not yet on the finalizer queue.</td></tr>
<tr><td>Unreachable</td><td>An object which is unreachable from any other root, but has been marked as a root by MAT to retain objects which otherwise would not be included in the analysis.</td></tr>
<tr><td>Unknown</td><td>An object of unknown root type.</td></tr>
</table>
</td></tr>
<tr><td>Objects pending for finalization</td><td>Indicates the number of objects that are pending for finalization.</td><td>Number</td><td>Sometimes an object will need to perform some action when it is destroyed. For</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
			example, if an object is holding some non-java resource such as a file handle or window character font, then you might want to make sure these resources are freed before an object is destroyed. To handle such situations, Java provides a mechanism called finalization. By using finalization, you can define specific actions that will occur when an object is just about to be reclaimed by the garbage collector. A high value for this measure indicates the existence of many objects that are still occupying the JVM memory space and are unable to be reclaimed by GC. A consistent rise in this value is also a sign of a memory leak.

The detailed diagnosis of the *Leak suspected classes* measure lists the names of all classes for which the memory usage is over the configured **PCT HEAP LIMIT.** In addition, the detailed diagnosis also reveals the **PERCENTAGE RETAINED HEAP** of each class - this is the percentage of the total *Allocated heap size* that is used by every class. From this, you can easily infer which class is consuming the maximum memory, and is hence, the key memory leak suspect. By observing the memory usage of this class during times of peak load, you can corroborate eG's findings - i.e., you can know for sure whether that class is indeed leaking memory or not!

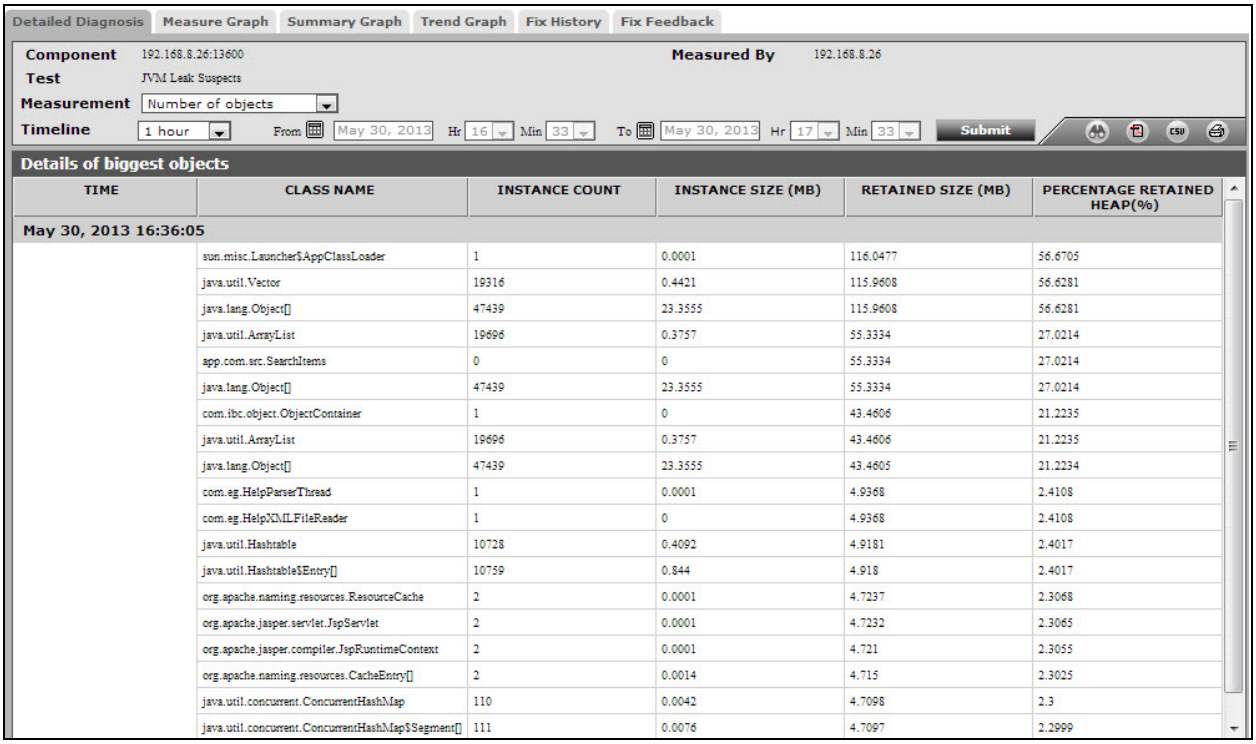
Details of leak suspects

TIME	CLASS NAME	INSTANCE COUNT	INSTANCE SIZE (MB)	RETAINED SIZE (MB)	PERCENTAGE RETAINED HEAP(%)
May 30, 2013 16:36:05					
	sun.misc.Launcher$AppClassLoader	1	0.0001	116.0477	56.6705
	java.util.Vector	19316	0.4421	115.9608	56.6281
	java.lang.Object[]	47439	23.3555	115.9608	56.6281

Figure 4.4: The detailed diagnosis of the Leak suspect classes measure

The detailed diagnosis of the Number of objects measure lists the names of the top-20 classes in the JVM, in terms of memory usage. In addition, the detailed diagnosis also reveals the percentage retained heap of each class - this is the percentage of the total Allocated heap size that is used by every class. From this, you can easily infer which class is consuming the maximum memory, and is hence, the key memory leak suspect. By observing the memory usage of this class during times of

peak load, you can corroborate eG's findings - i.e., you can know for sure whether that class is indeed leaking memory or not!



TIME	CLASS NAME	INSTANCE COUNT	INSTANCE SIZE (MB)	RETAINED SIZE (MB)	PERCENTAGE RETAINED HEAP(%)
May 30, 2013 16:36:05					
	sun.misc.Launcher$AppClassLoader	1	0.0001	116.0477	56.6705
	java.util.Vector	19316	0.4421	115.9608	56.6281
	java.lang.Object[]	47439	23.3555	115.9608	56.6281
	java.util.ArrayList	19696	0.3757	55.3334	27.0214
	app.com.src.SearchItems	0	0	55.3334	27.0214
	java.lang.Object[]	47439	23.3555	55.3334	27.0214
	com.ibc.object.ObjectContainer	1	0	43.4606	21.2235
	java.util.ArrayList	19696	0.3757	43.4606	21.2235
	java.lang.Object[]	47439	23.3555	43.4605	21.2234
	com.eg.HelpParserThread	1	0.0001	4.9368	2.4108
	com.eg.HelpXMLFileReader	1	0	4.9368	2.4108
	java.util.Hashtable	10728	0.4092	4.9181	2.4017
	java.util.Hashtable$Entry[]	10759	0.844	4.918	2.4017
	org.apache.naming.resources.ResourceCache	2	0.0001	4.7237	2.3068
	org.apache.jasper.servlet.JspServlet	2	0.0001	4.7232	2.3065
	org.apache.jasper.compiler.JspRuntimeContext	2	0.0001	4.721	2.3055
	org.apache.naming.resources.CacheEntry[]	2	0.0014	4.715	2.3025
	java.util.concurrent.ConcurrentHashMap	110	0.0042	4.7098	2.3
	java.util.concurrent.ConcurrentHashMap$Segment[]	111	0.0076	4.7097	2.2999

Figure 4.5: The detailed diagnosis of the Number of objects measure

4.1.7 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.

Note:

This test will work only if the target Java application uses the JDK/JRE offered by one of the following vendors: Oracle, Sun, OpenJDK. IBM JDK/JRE is not supported.

Target of the test : An IBM WebSphere Liberty server

Agent deploying the test : An internal 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 host for which the test is to be configured.
Port	The port number 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 Service URL of the application via JMX • Using GC logs To use JMX for metrics collections, set the Measure Mode to JMX. On the other hand, if you intend to use the GC log files for collecting the required metrics, set the Measure Mode to Log File. In this case, you would be required to enable GC logging. The procedure for this has been detailed in Section **4.1.7.1**.
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.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.
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 / Value**: 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 memory-intensive process when GC

Measurement	Description	Measurement Unit	Interpretation
			is being performed, then the Initial memory after GC may be higher than the Initial memory before GC.
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	
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.

4.1.7.1 Enabling GC Logging

If you want the **JVM Memory Pools Garbage Collections** test to use the GC log file to report metrics, then, you first need to enable GC logging. For this, follow the steps below:

1. Edit the startup script file of the Java application being monitored. Figure 4.6 depicts the startup script file of a sample application.

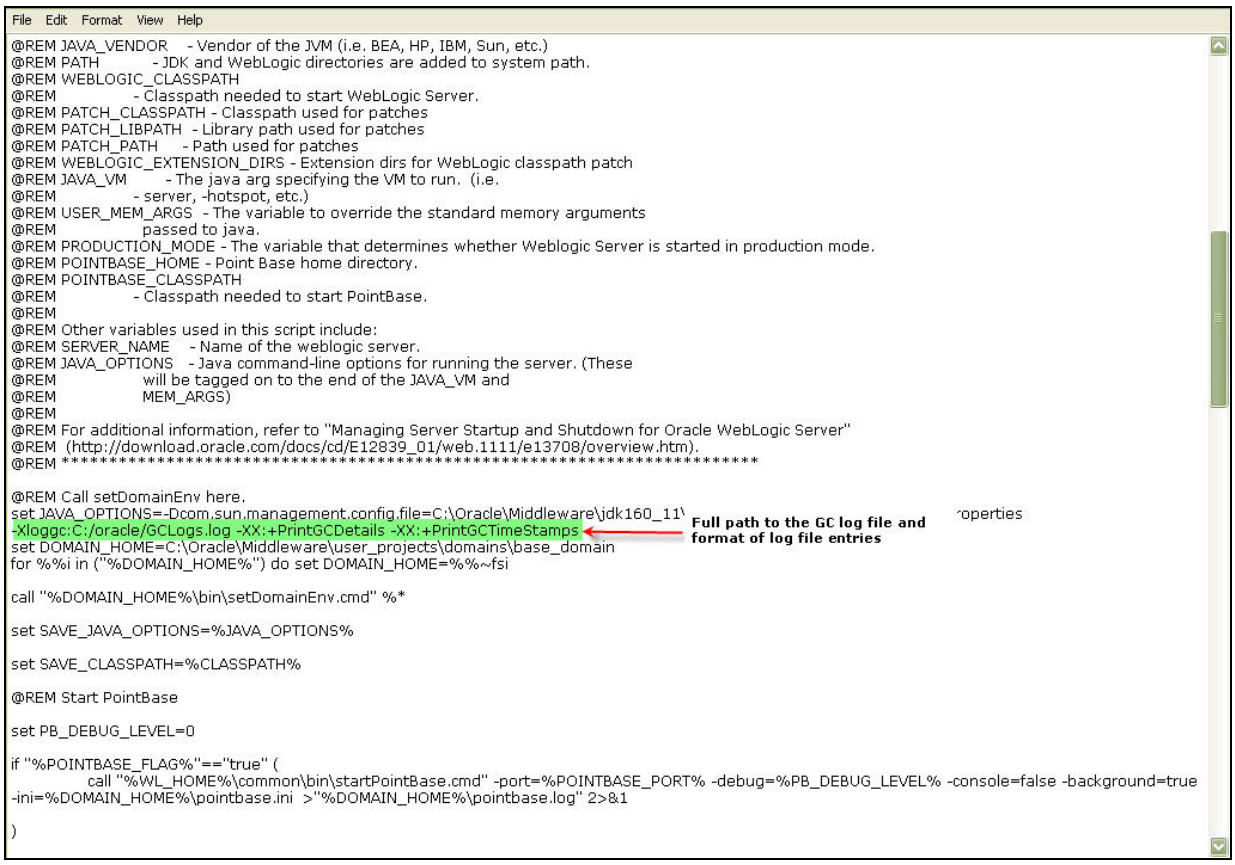


Figure 4.6: Editing the startup script file of a sample Java application

2. Add the line indicated by Figure 4.6 to the startup script file. This line should be of the following format:

```
-Xloggc:<Full path to the GC log file to which GC details are to be logged> -
XX:+PrintGCDetails -XX:+PrintGCTimeStamps
```

Here, the entry, -XX:+PrintGCDetails -XX:+PrintGCTimeStamps, refers to the format in which GC details are to be logged in the specified log file. Note that this test can monitor only those GC

log files which contain log entries of this format.

3. Finally, save the file and restart the application.

4.1.8 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:

- For this test to report detailed diagnostics, the target Java application should use the JDK/JRE offered by one of the following vendors only: Oracle, Sun, OpenJDK.
- If the target Java application is running on an AIX system using an IBM JRE/JDK, then, this test will not report detailed diagnostics. To enable the test to report DD, a MAT plugin is required. This plugin needs to be downloaded and extracted into the target AIX host. Once this is done, then the next time the eG agent runs this test, it takes the help of the plugin to read the usage statistics of object types from the heap dump file, and finally reports these metrics to the eG manager. To know how to install and configure the MAT plugin, refer to the *Installing and Configuring the MAT Plugin* section of the *Monitoring Java Applications* document.
- This test can provide detailed diagnosis information for only those monitored Java applications that use **JRE 1.6 or higher**.
- This test can run in an agent-based manner only.

Target of the test : An IBM WebSphere Liberty server

Agent deploying the test : An internal 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 host for which the test is to be configured.
Port	The port number 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 Service URL 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.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.
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.
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 (see page 1).
SNMP Version	This parameter appears only if the Mode is set to SNMP. The default selection in the SNMP version list is v1. However, for this test to work, you have to select SNMP v2 or v3 from this list, depending upon which version of SNMP is in use in the target environment.
SNMP Community	This parameter appears only if the Mode is set to SNMP. Here, specify the SNMP community name that the test uses to communicate with the mail server. The default is public. This parameter is specific to SNMP v1 and v2 only. Therefore, if the SNMP version chosen is v3, then this parameter will not appear.
User name	This parameter appears only when v3 is selected as the SNMP version. 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

Parameter	Description
	the name of such a user against this parameter.
Authpass	Specify the password that corresponds to the above-mentioned user name. 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 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 the 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 is applicable only if 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.
Use Sudo	This parameter is of significance to Unix platforms only. By default, the Use Sudo parameter is set to false. This indicates that, by default, JVM Memory Usage test will report the detailed diagnosis for the Used memory measure of each memory type being monitored by executing the /usr/bin/jmap command.

Parameter	Description
	However, in some highly secure environments, this command cannot be executed directly as the eG agent install user is different from the user privileged to access the target Java application. In such cases, do the following: • Edit the SUDOERS file on the target host and append an entry of the following format to it: <eG_agent_install_user> ALL=(ALL) NOPASSWD:<Command_with_path> • For instance, if the eG agent install user is eguser, then the entries in the SUDOERS file should be: eguser ALL=(ALL) NOPASSWD: /usr/bin/jmap • Finally, save the file. Then, when configuring the test using the eG admin interface, set the Use Sudo parameter to True. This will enable the eG agent to execute the sudo /usr/bin/jmap command and retrieve the detailed diagnosis of the Used memory measure.
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. • Heap analytics / detailed diagnostics will be provided only if the Java application being monitored supports Oracle Hotspot.
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. 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:

Parameter	Description
	• 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 this parameter.
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: • The eG manager license should allow the detailed diagnosis capability • Both the normal and abnormal frequencies configured for the detailed diagnosis

Parameter	Description
	measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
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.
Committed memory	Indicates the amount of memory guaranteed to be available for use by the JVM.	MB	The amount of Committed 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.
Free memory	Indicates the amount of memory currently available for use by the JVM.	MB	This is the difference between the Max allocated memory and Used memory measures. Ideally, the value of this measure should be high. **Note:** Sometimes, administrators may not

Measurement	Description	Measurement Unit	Interpretation
			want to cap/limit the maximum amount of memory that a JVM can use. In such cases, they may set the maximum memory to -1. If this is done, then it implies that the JVM can use any amount of memory. In this case therefore, the Maximum allocated memory will also report the value -1, but the Free memory measure will not be reported.
Max allocated 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 4.7: The detailed diagnosis of the Used memory measure

4.1.9 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.

Note:

If the Mode parameter of this test is set to **SNMP**, then stack trace will not be available. Also, detailed diagnostics will not report *CPU Time*.

Target of the test : An IBM WebSphere Liberty server

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 host for which the test is to be configured.
Port	The port number at which the specified host listens to
Mode	This test can extract metrics from the Java application using either of the following

Parameter	Description
	mechanisms: • Using SNMP-based access to the Java runtime MIB statistics; • By contacting the Service URL 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.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.
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.
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 (see page 1).
SNMP Version	This parameter appears only if the Mode is set to SNMP. The default selection in the SNMP version list is v1. However, for this test to work, you have to select SNMP v2 or v3 from this list, depending upon which version of SNMP is in use in the target environment.
SNMP Community	This parameter appears only if the Mode is set to SNMP. Here, specify the SNMP community name that the test uses to communicate with the mail server. The default is public. This parameter is specific to SNMP v1 and v2 only. Therefore, if the SNMP version chosen is v3, then this parameter will not appear.
User name	This parameter appears only when v3 is selected as the SNMP version. 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

Parameter	Description
	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 user name. 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 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 the 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.

Parameter	Description
PCT Medium CPU Util Threads	By default, this 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 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, this 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.
Max Thread Count	By default, this parameter is set to 20. This implies that the detailed diagnosis of the Runnable threads, Waiting threads, and *Timed waiting threads* measures will by default display only the top-20 JVM threads in terms of CPU consumption. To view more threads as part of detailed diagnostics, increase the value of this parameter. To view all threads that are in the said state (eg., runnable, waiting, and timed waiting), specify All or * against this parameter.
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, this 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 this 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_SIR>\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.
Data over TCP	This parameter is applicable only if 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 this parameter.
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	Number	

Measurement	Description	Measurement Unit	Interpretation
	daemon and non-daemon threads).		
Runnable threads	Indicates the current number of threads in a runnable state.	Number	The detailed diagnosis of this measure, if enabled, lists the names of the top-20 (default) runnable threads in terms of their CPU usage. The time for which the thread was in a blocked state, waiting state, etc., are provided as part of the detailed diagnostics. You can change the sort order to view threads by waiting time, blocked time, 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 all 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. By default, the top-20 waiting threads in terms of CPU usage will be listed. You can change the sort order to view

Measurement	Description	Measurement Unit	Interpretation
			threads by waiting time, blocked time, etc. While waiting, the Java application program does no productive work and its ability to complete the task-at-hand is degraded. A certain amount of waiting may be acceptable for Java application programs. However, when the amount of time spent waiting becomes excessive or if the number of times that waits occur exceeds a reasonable amount, the Java application program may not be programmed correctly to take advantage of the available resources. When this happens, the delay caused by the waiting Java application programs elongates the response time experienced by an end user. An enterprise may use Java application programs to perform various functions. Delays based on abnormal degradation consume employee time and may be costly to corporations.
Timed waiting threads	Indicates the number of threads in a *TIMED_WAITING* state.	Number	When a thread is in the TIMED_WAITING state, it implies that the thread is waiting for another thread to do something, but will give up after a specified time out period. To view the details of threads in the *TIMED_WAITING* state, use the detailed diagnosis of this measure, if enabled. By default, the top-20 timed waiting threads in terms of CPU usage will be listed. You can change the sort order to view threads by waiting time, blocked time, etc.

Measurement	Description	Measurement Unit	Interpretation
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	To know which threads are consuming low CPU, use the detailed diagnosis of this measure.
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	To know which threads are consuming medium CPU, use the detailed diagnosis of this measure.
High CPU threads	Indicates the number of threads that are currently consuming CPU that is greater than the percentage configured in the PCT High CPU Util Threads text box.	Number	Ideally, the value of this measure should be very low. A high value is indicative of a resource contention at the JVM. Under such circumstances, you might want to identify the resource-hungry threads. 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 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	Number	

Measurement	Description	Measurement Unit	Interpretation
	threads.		
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.

4.1.9.1 Accessing Stack Trace using the STACK TRACE link in the Measurements Panel

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

Note:

If the mode set for the **JVM Thread** test is SNMP, the stack trace details may not be available.

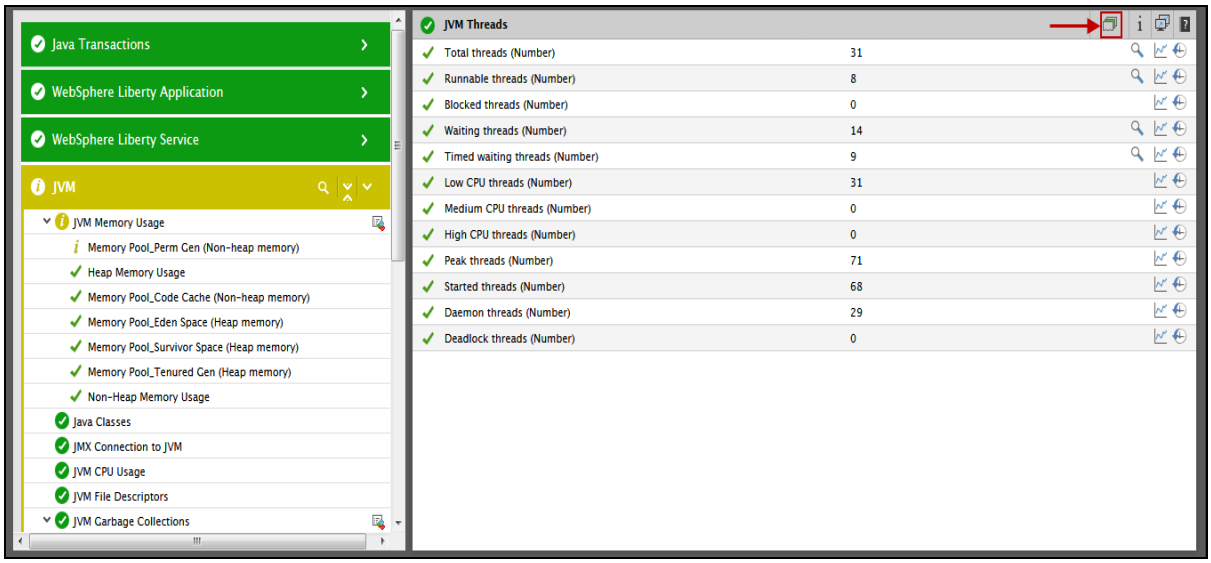


Figure 4.8: The STACK TRACE link

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

Typically, when a Java application begins exhibiting erratic resource usage patterns, it often takes administrators hours, even days to figure out what is causing this anomaly – could it be owing to one/more resource-intensive threads being executed by the application? If so, what is causing the thread to erode resources? Is it an inefficient piece of code? In which case, which line of code could be the most likely cause for the spike in resource usage? To be able to answer these questions accurately, administrators need to know the complete list of threads that the application executes, view the stack trace of each thread, analyze each stack trace in a top-down manner, and trace where the problem originated.

eG Enterprise simplifies this seemingly laborious procedure by not only alerting administrators instantly to excessive resource usage by a target application, but also by automatically identifying the problematic thread(s), and providing the administrator with quick and easy access to the stack trace information of that thread; with the help of stack trace, administrators can effortlessly drill down to the exact line of code that requires optimization.

To access the stack trace information of a thread, click on the **STACK TRACE** icon in the **Measurements** panel of Figure 4.8.

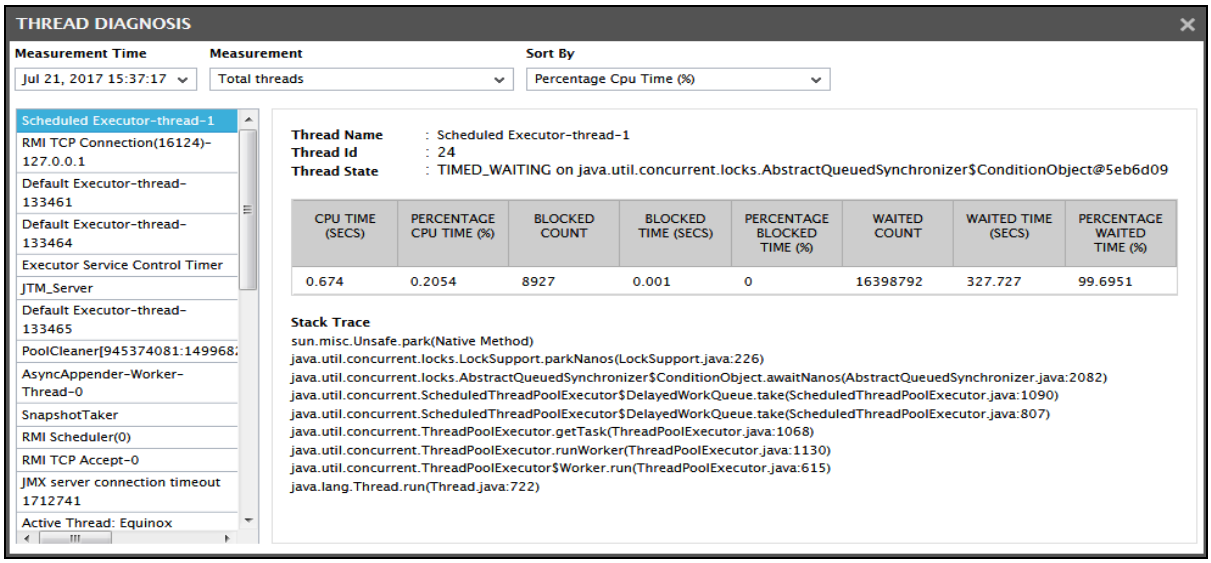


Figure 4.9: Stack trace of a resource-intensive thread

Figure 4.9 that appears comprises of two panels. The left panel, by default, lists all the threads that the target application executes, starting with the threads that are currently live. Accordingly, the **Total Threads** option is chosen by default from the **Measurement** list. If need be, you can override the default setting by choosing a different option from the **Measurement** list – in other words, instead of viewing the complete list of threads, you can choose to view threads of a particular type or which are in a particular state alone in Figure 4.9, by selecting a different **Measurement** from Figure 4.9. For instance, to ensure that the left panel displays only those threads that are currently in a runnable state, select the **Runnable threads** option from the **Measurement** list. The contents of the left panel will change as depicted by Figure 4.10.

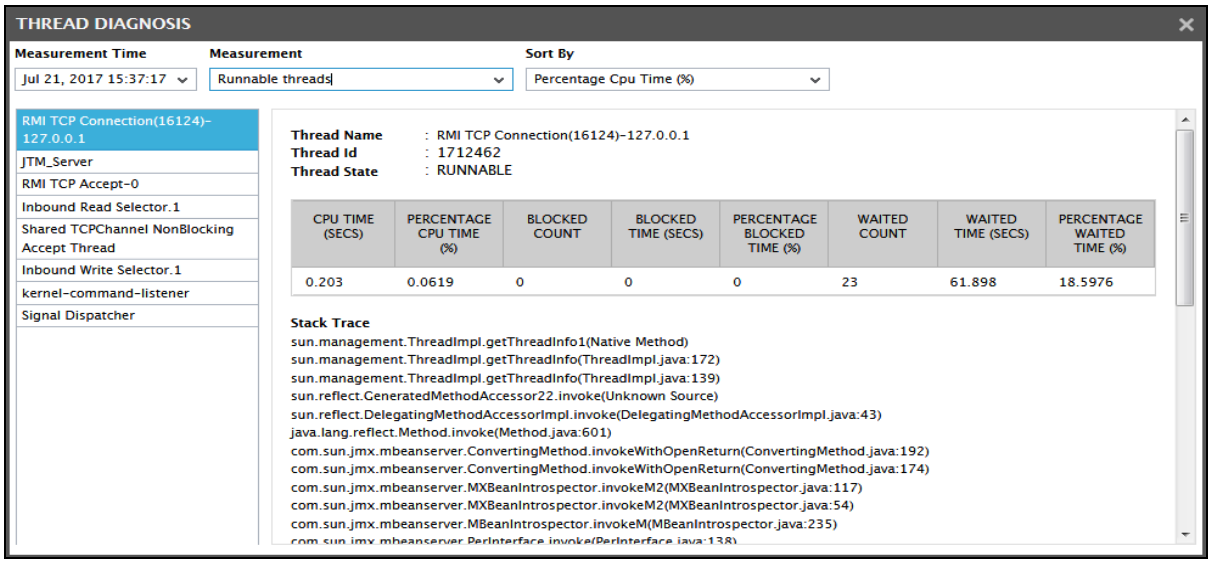


Figure 4.10: Thread diagnosis of live threads

Also, the thread list in the left panel is by default sorted in the descending order of the Percent CPU Time of the threads. This implies that, by default, the first thread in the list will be the thread that is currently active and consuming the maximum CPU. You can change the sort order by selecting a different option from the **Sort by** list in Figure 4.10.

Typically, the contents of the right panel change according to the thread chosen from the left. Since the first thread is the default selection in the left panel, and this thread by default consumes the maximum CPU, we can conclude that the right panel will by default display the details of the leading CPU consumer. Besides the name and state of the chosen thread, the right panel will provide the following information:

- **CPU Time** : The amount of CPU processing time (in seconds) consumed by the thread during the last measurement period;
- **Percentage CPU Time**: The percentage of time the thread was using the CPU during the last measurement period;
- **Blocked Count**: The number of the times during the last measurement period the thread was blocked waiting for another thread;
- **Blocked Time**: The total duration for which the thread was blocked during the last measurement period;
- **Percentage Blocked Time**: The percentage of time (in seconds) for which the thread was blocked during the last measurement period;

- **Waited Count**: The number of times during the last measurement period the thread was waiting for some event to happen (eg., wait for a thread to finish, wait for a timing event to finish, etc.);
- **Waited Time**: The total duration (in seconds) for which the thread was waiting during the last measurement period;
- **Percentage Waited Time**: The percentage of time for which the thread was waiting during the last measurement period.

In addition to the above details, the right panel provides the **Stack Trace** of the thread.

In the event of a sudden surge in the CPU usage of the target Java application, the **Thread Diagnosis** window of Figure 4.10 will lead you to the CPU-intensive thread, and will also provide you with the **Stack Trace** of that thread. By analyzing the stack trace in a top-down manner, you can figure out which method/routine called which, and thus locate the exact line of code that could have contributed to the sudden CPU spike.

If the CPU usage has been increasing over a period of time, then, you might have to analyze the stack trace for one/more prior periods, so as to perform accurate root-cause diagnosis. By default, the **Thread Diagnosis** window of Figure 4.10 provides the stack trace for the current measurement period only. If you want to view the stack trace for a previous measurement period, you will just have to select a different option from the **Measurement Time** list. By reviewing the code executed by a thread for different measurement periods, you can figure out if the same line of code is responsible for the increase in CPU usage.

4.1.10 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.

Target of the test : An IBM WebSphere Liberty server

Agent deploying the test : An internal agent

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

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed

Parameter	Description
Host	The host for which the test is to be configured.
Port	The port number 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 Service URL 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.
Service URL	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.
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.
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 (see page 1).
SNMP Version	This parameter appears only if the Mode is set to SNMP. The default selection in the SNMP version list is v1. However, for this test to work, you have to select SNMP v2 or v3 from this list, depending upon which version of SNMP is in use in the target environment.
SNMP Community	This parameter appears only if the Mode is set to SNMP. Here, specify the SNMP community name that the test uses to communicate with the mail server. The default is public. This parameter is specific to SNMP v1 and v2 only. Therefore, if the SNMP version chosen is v3, then this parameter will not appear.
User name	This parameter appears only when v3 is selected as the SNMP version. SNMP version

Parameter	Description
	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 User nameprovided 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 user name. 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 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 the 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

Parameter	Description
	• AES – Advanced Encryption Standard
Encryptpassword	Specify the encryption password here.
Confirm password	Confirm the encryption password by retyping it here.
ReportManagerTime	By default, this flag is set to **Yes**, indicating that, by default, the detailed diagnosis of this test, if enabled, will report the shutdown and reboot times of the server in the manager's time zone. If this flag is set to **No**, then the shutdown and reboot times are shown in the time zone of the system where the agent is running i.e., the system being managed for agent-based monitoring.
Data over TCP	This parameter is applicable only if 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 this parameter.
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 measurement period – the smaller the measurement period, greater the</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
			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.

4.2 The WebSphere Liberty Service Layer

Using the tests mapped to this layer, administrators can ascertain the following:

- The number of threads that are currently active in each thread pool;
- The number of threads in each thread pool;
- The rate at which connections were created in each connection pool;
- The average waiting time for a connection to be granted from each connection pool;
- The number of connections that were available for use in each connection pool;
- The current state of each queue;
- The number of outstanding read/write requests in each queue;
- The maximum configured size of each queue;

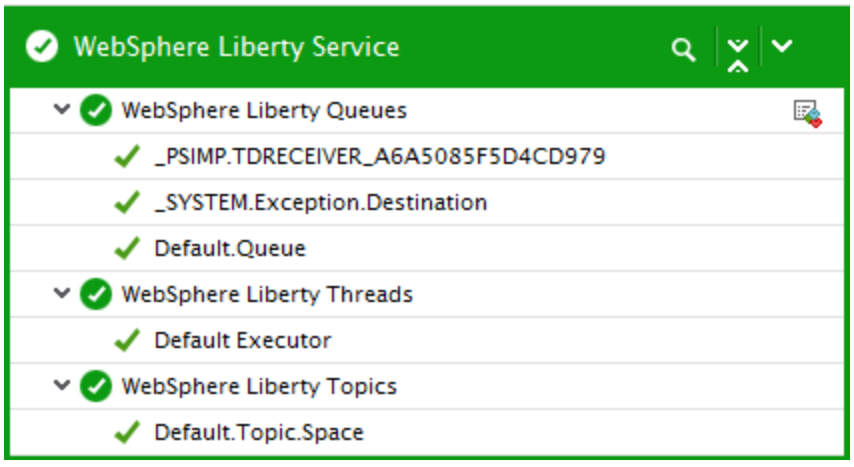


Figure 4.11: The tests mapped to the WebSphere Liberty Application layer

4.2.1 WebSphere Liberty Connection Pools Test

This test auto-discovers the connection pools on the target IBM WebSphere Liberty server and for each connection pool, this test reports the rate at which connections were created. This test also reports the rate at which connections were destroyed and the average time taken for a connection to be granted from each connection pool. Using this test, administrators can figure out the connection pool that is busy and the connection pool on which maximum number of requests were created.

Target of the test : An IBM WebSphere Liberty

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every connection pool on the target IBM WebSphere Liberty server being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port number at which the specified host listens.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Connection create rate	Indicates the rate at which connections were created in this connection pool.	Connections/sec	This measure is a good indicator of load on each connection pool. Compare the value of this measure

Measurement	Description	Measurement Unit	Interpretation
			across connection pools to figure out the connection pool on which maximum number of connections were created.
Connection destroy rate	Indicates the rate at which connections were destroyed in this connection pool.	Connections/sec	
Managed connection count	Indicates the number of managed connection objects that were in use in this connection pool.	Number	
Average wait time	Indicates the average waiting time until a connection from this connection pool was granted.	Milliseconds	A low value is desired for this measure. Compare the value of this measure across the connection pools to figure out the connection pool that takes too long to grant a connection i.e., in other words you can figure out the connection pool that is too busy to grant a connection.
Connection handled count	Indicates the number of connection objects that were handled by this connection pool.	Number	
Free connection count	Indicates the number of connections that were available for use in this connection pool.	Number	A high value is desired for this measure. A sudden/gradual decrease in the value of this measure indicates that the connection pool is busy. If too many connection requests are received continuously to this connection pool, then administrators may try increasing the number of connections in the connection pool.

4.2.2 WebSphere Liberty Queues Test

This test auto-discovers the queues on the target IBM WebSphere Liberty server and reports the current state and maximum configured size of each queue. This test also helps the administrators to figure out whether get operations are allowed on each queue and the number of outstanding I/O requests in each queue. Using this test, administrators can figure out the queue that is latent and is unable to process requests at a faster pace!

Target of the test : An IBM WebSphere Liberty

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each queue on the target IBM WebSphere Liberty server being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port number at which the specified host listens.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
State	Indicates the current state of this queue.		The values reported by this measure and its numeric equivalents are mentioned in the table below:

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Measure value</th><th>Numeric Value</th></tr>
<tr><td>Active</td><td>0</td></tr>
<tr><td>Inactive</td><td>1</td></tr>
</table>
Note:
By default, this measure reports the Measure Values listed in the table above to indicate the current state of this queue. The graph of this measure however, represents the status of a server using the numeric equivalents only - 0 or 1.
</td></tr>
<tr><td>Send allowed</td><td>Indicates whether/not get operations were allowed in this queue.</td><td></td><td>The values reported by this measure and its numeric equivalents are mentioned in the table below:
<table>
<tr><th>Measure value</th><th>Numeric Value</th></tr>
<tr><td>True</td><td>0</td></tr>
<tr><td>False</td><td>1</td></tr>
</table>
Note:
By default, this measure reports the Measure Values listed in the table above to indicate whether/not get operations were allowed in this queue. The graph of this measure however, represents the status of a server using the numeric equivalents only - 0 or 1.
</td></tr>
<tr><td>Depth</td><td>Indicates the number of outstanding read/write (I/O) requests in this queue.</td><td>Number</td><td>If the value of this measure keeps increasing steadily and significantly for each queue, it could indicate that the queue is latent, and is unable to process I/O requests quickly.
The value of this measure therefore should be low at all times.</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
Maximum queue size	Indicates the maximum configured size of this queue.	Number	

4.2.3 WebSphere Liberty Thread Pools Test

By continuously monitoring the thread pools of the IBM WebSphere Liberty server, administrators can figure out the number of active threads in each thread pool and the size of the thread pool. Using this test, administrators can determine the thread pool containing the maximum number of active threads.

Target of the test : An IBM WebSphere Liberty

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each thread pool on the target IBM WebSphere Liberty server being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port number at which the specified host listens.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Active threads	Indicates the number of threads in this thread pool that are currently active in servicing the requests.	Number	Comparing the value of this measure across the thread pools helps administrators identify the thread pool containing the maximum number of active threads. If the value of this measure is equal to the value of the Pool Size measure, then, it indicates that the thread pool is busy servicing the requests and new requests have to wait for a while till the threads are released. In such circumstances, administrators may consider increasing the size of the thread pool.
Pool size	Indicates the number of threads that are currently available in this thread pool.	Number	

4.2.4 WebSphere Liberty Topics Test

This test auto-discovers the topics of the applications deployed on the target IBM WebSphere Liberty server and reports the maximum configured size of each topic. This test also helps administrators to figure out whether get operations are allowed on each topic. Using this test, administrators can figure out the topic that is latent and is unable to process I/O requests quickly!

Target of the test : An IBM WebSphere Liberty

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each topic on the target IBM WebSphere Liberty server being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed .
Host	The host for which the test is to be configured.
Port	The port number at which the specified host listens.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Send allowed	Indicates whether/not get operations were allowed in this topic.		The values reported by this measure and its numeric equivalents are mentioned in the table below: **Measure value** / **Numeric Value** True / 0 False / 1 **Note:** By default, this measure reports the **Measure Value**s listed in the table above to indicate get operations were allowed in this topic. The graph of this measure however, represents the status of a server using the numeric equivalents only - 0 or 1.
Depth	Indicates the number of outstanding read/write	Number	If the value of this measure keeps increasing steadily and significantly for

Measurement	Description	Measurement Unit	Interpretation
	(I/O) requests to this topic.		each topic, it could indicate that the topic is latent, and is unable to process I/O requests quickly. The value of this measure therefore should be low at all times.
Maximum topic size	Indicates the maximum configured size of this topic.	Number	

4.3 The WebSphere Liberty Application Layer

Using the tests mapped to this layer, administrators can ascertain the following:

- The rate at which each servlet was hit;
- The average time taken by each servlet to respond to requests;
- The number of requests to each servlet;
- The number of sessions that accessed each WAR file per second;
- The number of sessions that were currently active and accessing each WAR file;
- The number of sessions that were invalidated and timed out while accessing each WAR file;
- The current state of each web application;

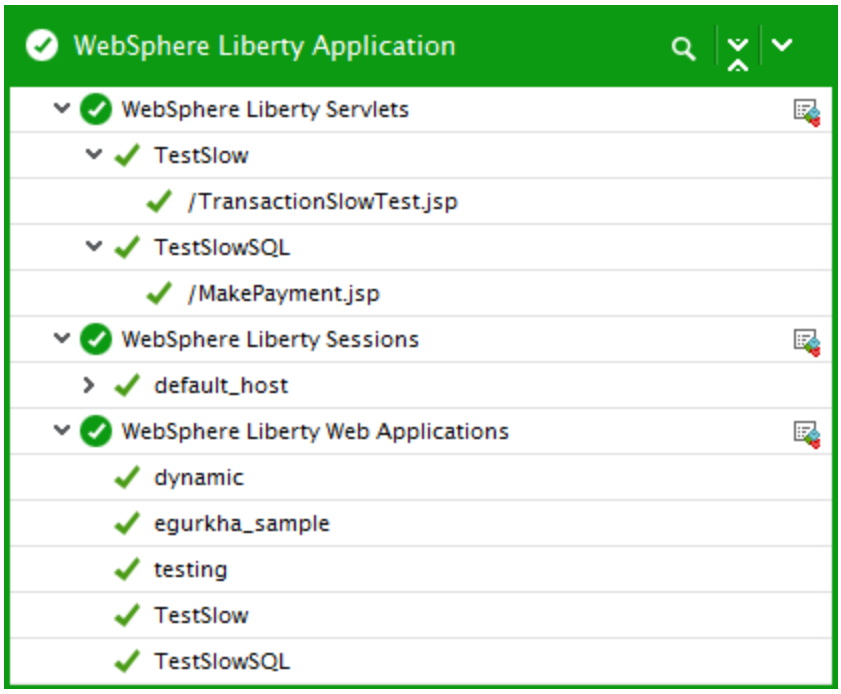


Figure 4.12: The tests associated with the WebSphere Liberty Application layer

4.3.1 WebSphere Liberty Servlets Test

For each servlet of the web application deployed on the target IBM WebSphere Liberty server, this test reports the rate at which each servlet was hit and the maximum time taken by each servlet to respond to user requests. In addition, this test reports the number of hits to each servlet. Using this test, administrators can figure out the servlet that has garnered the maximum number of hits and the servlet that took too long to respond to requests.

Target of the test : An IBM WebSphere Liberty

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each *web application:servlet* deployed on the target IBM WebSphere Liberty server being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.

Parameters	Description
Port	The port number at which the specified host listens.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Requests	Indicates the number of requests(hits) to access this servlet.	Number	Compare the value of this measure across the servlets to figure out the servlet to which maximum number of requests were made.
Request rate	Indicates the rate at which this servlet was hit.	Requests/sec	
Average response time	Indicates the average time taken by this servlet to respond to requests.	Milliseconds	Compare the value of this measure across servlets to figure out the servlet that is taking too long to respond.

4.3.2 WebSphere Liberty Sessions Test

For each WAR File deployed on the target IBM WebSphere Liberty server, this test monitors the number of sessions that are accessing the file per second. In addition, this test reports the number of currently cached sessions that were accessing the file, the number of currently active sessions that were accessing the file, the number of invalidated sessions that were accessing the file etc. Using this test, administrators can figure out the WAR file that is being accessed by the maximum number of sessions.

Target of the test : An IBM WebSphere Liberty

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each *hostname:WAR file* on the target IBM WebSphere Liberty server being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed .
Host	The host for which the test is to be configured.
Port	The port number at which the specified host listens.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Session create rate	Indicates the number of sessions that accessed this WAR file per second.	Sessions/sec	
Live sessions	Indicates the number of sessions that were accessing this WAR file and are currently cached in the memory of the sever.	Number	Compare the value of this measure across the WAR files to figure out the WAR file that is accessed using the maximum number of sessions.
Active sessions	Indicates the number of currently active sessions that were accessing this file.	Number	Compare the value of this measure across the WAR files to figure out the WAR file that is accessed by the maximum number of active sessions.
Invalidated sessions	Indicates the number of invalidated sessions that were accessing this file.	Number	

Measurement	Description	Measurement Unit	Interpretation
Timeout invalidates	Indicates the number of invalidated sessions (due to timeout) that were accessing this file.	Number	

4.3.3 WebSphere Liberty Web Applications Test

This test auto-discovers the web applications deployed on the IBM WebSphere Liberty and reports the current state of each web application. Using this test, administrators can easily identify the web applications that stopped in the recent past.

Target of the test : An IBM WebSphere Liberty

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each web application deployed on the target IBM WebSphere Liberty server being monitored

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port number at which the specified host listens.
Service URL Path	For the eG agent to collect metrics from the target IBM WebSphere Liberty server, the local connector should be enabled on the target server. Once the connector is enabled, a **com.ibm.ws.jmx.local.address** file will be created in the *${server.output.dir}/logs/state* folder. The eG agent uses this file to connect to the target server and collect the required metrics from it. Therefore, specify the exact path to this file in the Service URL text box. For example, in case of Windows environments, the Service URL Path can be *C:\wlp\usr\servers\server1\logs\state* and in case of Linux environments, the Service URL Path can be */opt/wlp/ur/servers/server1/logs/state*.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Application state	Indicates the current state of this web application.		The values reported by this measure and its numeric equivalents are mentioned in the table below: **Measure value** / **Numeric Value** Running: 0 Stopped: 1 Installed: 2 **Note:** By default, this measure reports the **Measure Value**s listed in the table above to indicate the current state of this web application. The graph of this measure however, represents the status of a server using the numeric equivalents only - 0 to 2.

4.4 The Java Transactions Layer

Using the *Java Transactions* test mapped to this layer, you will be allowed monitor configured patterns of transactions to the target Java application, and report their response times, so that slow transactions and transaction exceptions are isolated and the reasons for the same analyzed. For more details on the **Java Transactions** test, please refer to the *Monitoring Java Applications* document.

Note:

Java Transaction Monitoring (JTM) can be enabled only for those Java applications that use JDK 1.5 or higher.

Similarly, you can enable the *Java Business Transactions* test and *Java Key Business Transactions* test. To enable the tests, follow the *Agents -> Tests -> Enable/Disable* menu sequence, select **IBM WebSphere Liberty** as the **Component type**, **Performance** as the **Test type**, and then select *Java Business Transactions* and *Java Key Business Transactions* from the **DISABLED TESTS** list. Click the **Enable** button to enable the selected tests, and click the **Update** button to save the changes.

The metrics reported will then be captured by the **Java Transactions** layer. The **Java Business transactions** test and the **Java Key Business Transactions** test are discussed in detail in the *Business Transaction Monitoring* document.

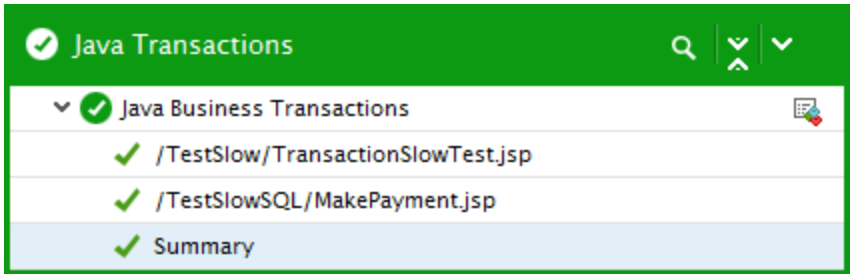


Figure 4.13: The test mapped to the Java Transactions layer

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.