Monitoring the eG Manager

eG Enterprise v6

Table of Contents

Table of Figures

Chapter 1

Monitoring the eG Manager

The purpose of an IT infrastructure monitoring solution is to observe the performance of the applications, devices, and systems that underlie their mission-critical service offerings, proactively detect potential deviations, and promptly alert administrators to the root-cause of service outages/slowdowns. To be able to provide such critical performance monitoring and fault management services in a timely, reliable, and uninterrupted manner, the monitoring solution , like the components it monitors, should perform at peak capacity at all times!

An efficient monitoring solution is one that not only monitors other applications but also analyzes its own performance at periodic intervals. This way, the solution can instantly capture snags in its operations, accurately point administrators to the source of these performance degradations, and facilitate the speedy resolution of issues. In the absence of such self-monitoring capabilities, you may end up in a situation where slowdowns experienced by the monitoring solution - say, owing to a session overload, a backend that is low on space, or data collectors that are not operating - delay problem identification and alerting in the target environment, causing issues to go unnoticed or to aggravate.

To avoid such unpleasant eventualities, the eG Enterprise Suite provides a dedicated *eG Manager* monitoring model that monitors each of the core functions of the eG manager at configured frequencies and reports abnormalities (if any).

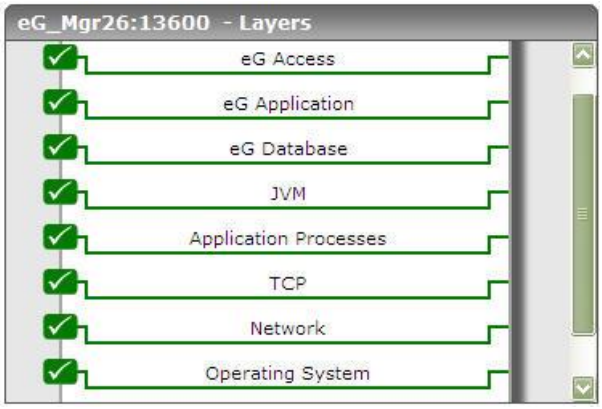


Figure 1: Layer model of the eG Manager

Each layer of Figure 1 above is mapped to tests that report a wealth of performance information related to the eG manager. To pull out these performance metrics, the eG agent can be deployed on the eG manager host itself (*agent-based*) or on any remote Windows host in the environment (*agentless*).

Regardless of where it is installed, the **eG agent should be configured to connect to the JRE used by the eG manager to pull out metrics of interest**. To enable the eG agent to connect to the JRE, by default, **JMX support has been enabled for the eG manager's JRE.** JMX enables external programs like the eG agent to connect to the JRE of an application and pull out metrics in real-time.

With the help of the metrics collected via JMX, eG administrators can find quick and accurate answers for the following queries:

- Has database cleanup run today?
- Did cleanup take too long to complete?
- Was cleanup incomplete on any table? If so,which ones?
- Did cleanup fail on any table? If so, which ones?
- Does the database connection pool have enough free connections?
- Are too many connections to the database waiting to close?
- Does the eG database have too many large sized tables and tables with many rows?
- Are any eG agents not running currently?
- Have any eG agents configured on the manager not being physically installed?
- Did any errors occur on the eG manager? If so, when - during manager installation/configuration/starting? during threshold computation? or at the time of trend computation?
- Have all object pools on the eG manager been correctly sized? Are there any object pools that are running out of free objects?
- Is any servlet responding poorly to client requests?
- How busy is the state manager process?
- How is the threshold manager performing? Is it taking too long to compute thresholds?
- On what tables did threshold computation fail and why?
- Did trend run today?
- Is the trend manager process healthy? Is it taking an unusually long time to compute trends?
- Did trend computation fail on any table? If so, which ones?
- Are cleanup/threshold computation/trend computation performed by separate Java processes or by the eG manager itself?
- How many users are registered with the eG Enterprise system and how many user subscriptions are about to expire?
- Is the eG manager overloaded with sessions? If so, what are the sessions that are currently active on the manager?
- Who are the distinct users who are logged into the manager, and when did they login?

The sections that follow will discuss the top 4 layers of Figure 1 only, as the other layers have been discussed extensively in the *Monitoring Unix and Windows Servers* document.

1.1 The JVM Layer

The tests mapped to this layer report the overall health and resource usage of the eG manager's JVM. To pull out JVM-related statistics from the eG manager, the eG agent uses either of the following mechanisms:

- Using SNMP-based access to the Java runtime MIB statistics of the eG manager;
- By contacting the Java runtime (JRE) of the eG manager via JMX
- This is why, each test mapped to the **JVM** layer provides administrators with the option to pick a monitoring **MODE** - i.e., either **JMX** or **SNMP**. The remaining test configuration depends upon the mode chosen. Section 1.1.1 of this document will explain how to enable SNMP support for the eG manager's JRE.

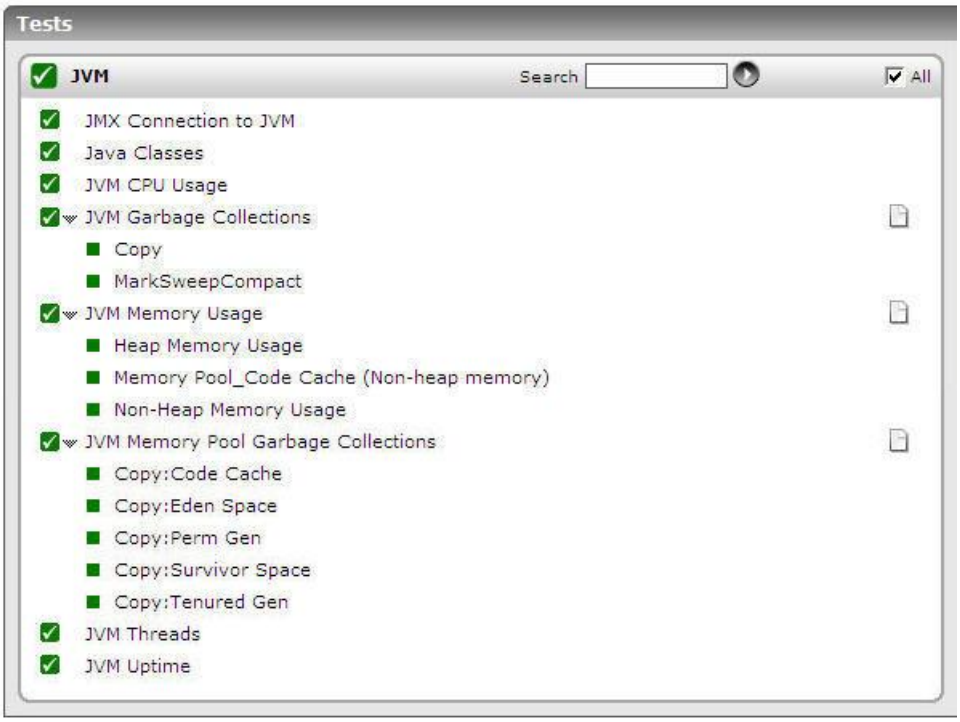


Figure 2: The tests mapped to the JVM layer

1.1.1 Enabling SNMP Support for the eG Manager's JRE

Instead of JMX, you can configure the eG agent to report JVM-related metrics using SNMP-based access to the Java runtime MIB statistics.

In some environments, SNMP access might have to be authenticated by an ACL (Access Control List), and in some other cases, it might not require an ACL.

If SNMP access **does not require ACL authentication**, then follow the steps below to enable SNMP support:

1. Login to the eG manager host.
2. Ensure that the SNMP service and the SNMP Trap Service are running on the host.

3. Next, edit the **management.properties** file in the **<EG_INSTALL_DIR>\manager** folder (on Windows; on Unix, this will be **/opt/egurkha/manager**) of the eG manager.

4. In the **management.properties** file, you will find a **'For SNMP Settings'** block that is 'commented' by default.

```
################## For SNMP Settings ####################
#
#com.sun.management.snmp.port=<port-number>
#com.sun.management.snmp.acl=false
#com.sun.management.snmp.acl.file=filepath
#com.sun.management.snmp.interface=localhost
#
############################################################
```

5. Uncomment the contents of this block as indicated below:

```
################## For SNMP Settings ####################

com.sun.management.snmp.port=<port-number>
com.sun.management.snmp.acl=false
com.sun.management.snmp.acl.file=filepath
com.sun.management.snmp.interface=localhost

############################################################
```

6. Against the **com.sun.management.snmp.port** parameter in the 'uncommented' block, enter the SNMP Port of the eG manager.

   For instance, if the SNMP port is 1166, then the first line of the above specification will be:

```
com.sun.management.snmp.port=1166
```

7. Next, set the **com.sun.management.snmp.interface** parameter to the IP address of the host from which the eG manager's JRE will accept SNMP requests. By default, it is set to *localhost*, indicating that the eG manager's JRE will accept SNMP requests from the eG manager's host only. To ensure that the JRE services only those SNMP requests that are received from the eG agent, set this parameter to the IP address of the agent host. For instance, if the eG agent that monitors the eG manager is executing on *192.168.10.152*, then the **com.sun.management.snmp.interface** specification will be:

```
com.sun.management.snmp.interface=192.168.10.152
```

8. Next, edit the start-up script of the eG manager, include the following line it, and save the script file.

```
-Dcom.sun.management.config.file=<management.properties_file_path>
```

   For instance, on a Windows host, the *<management.properties_file_path>* can be expressed as: **D:\eGurkha\manager**.

   On other hand, on a Unix/Linux/Solaris host, a sample *<management.properties_file_path>* specification will be as follows: **/opt/egurkha/manager**

On the contrary, if SNMP access requires **ACL authentication**, then follow the steps below to enable SNMP support for the JRE:

1. Login to the eG manager host. If the eG manager is executing on a Windows host, login as a local/domain administrator.
2. Ensure that the SNMP service and SNMP Trap Service are running on the host.
3. Copy the *snmp.acl.template* file in the **<JAVA_HOME>\jre\lib\management** folder to another location on the local host. Rename the *snmap.acl.template* file as *snmp.acl*, and copy the *snmp.acl* file back to the **<JAVA_HOME>\jre\lib\management** folder.
4. Next, edit the *snmp.acl* file, and set rules for SNMP access in the file.

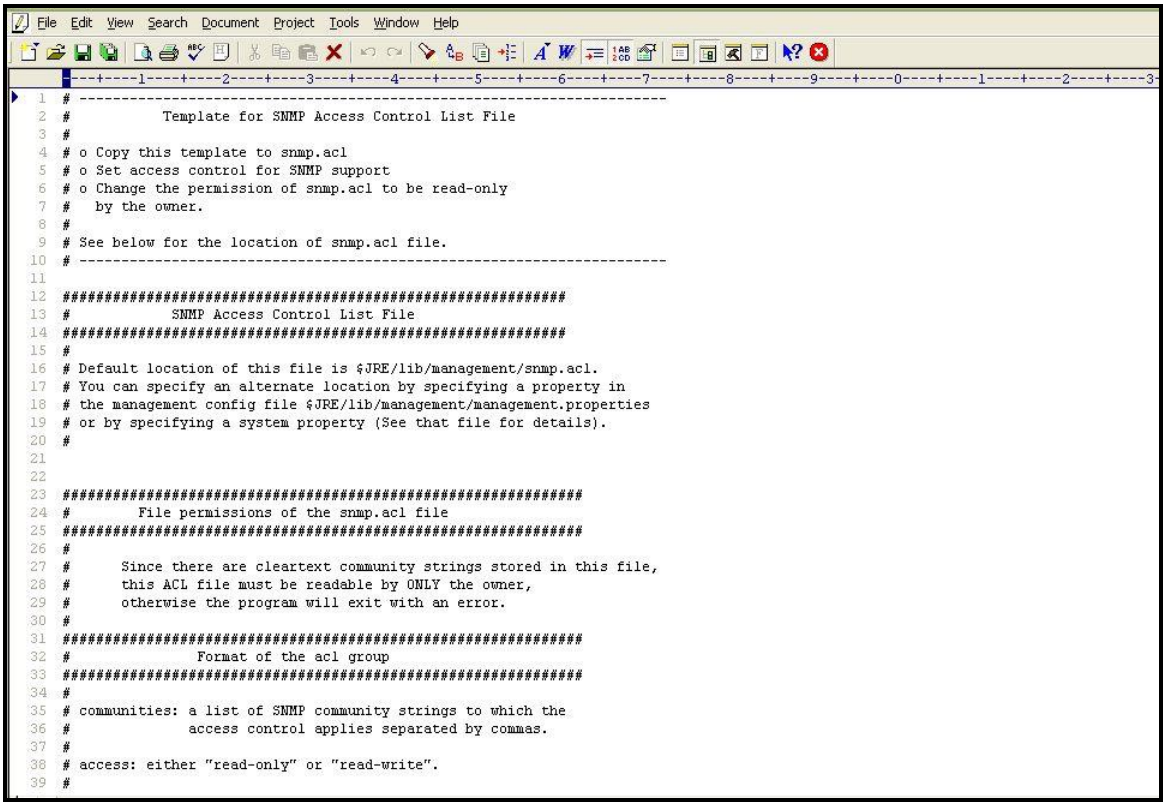
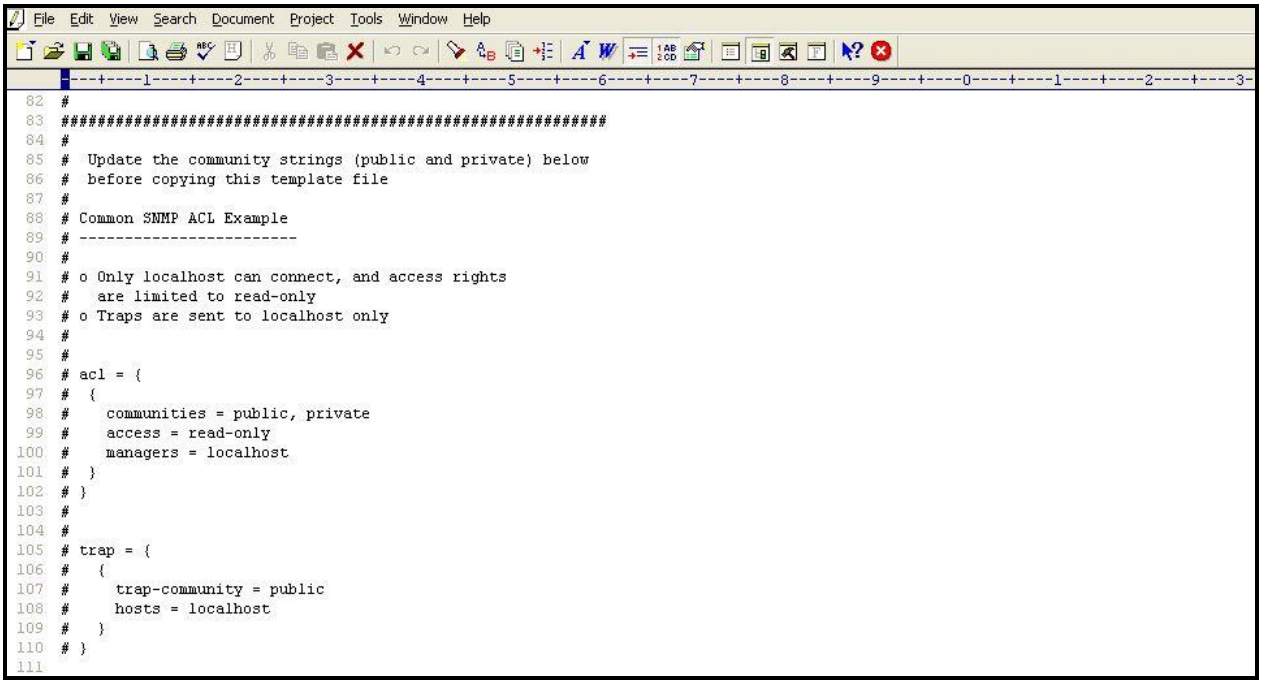


Figure 3: The snmp.acl file

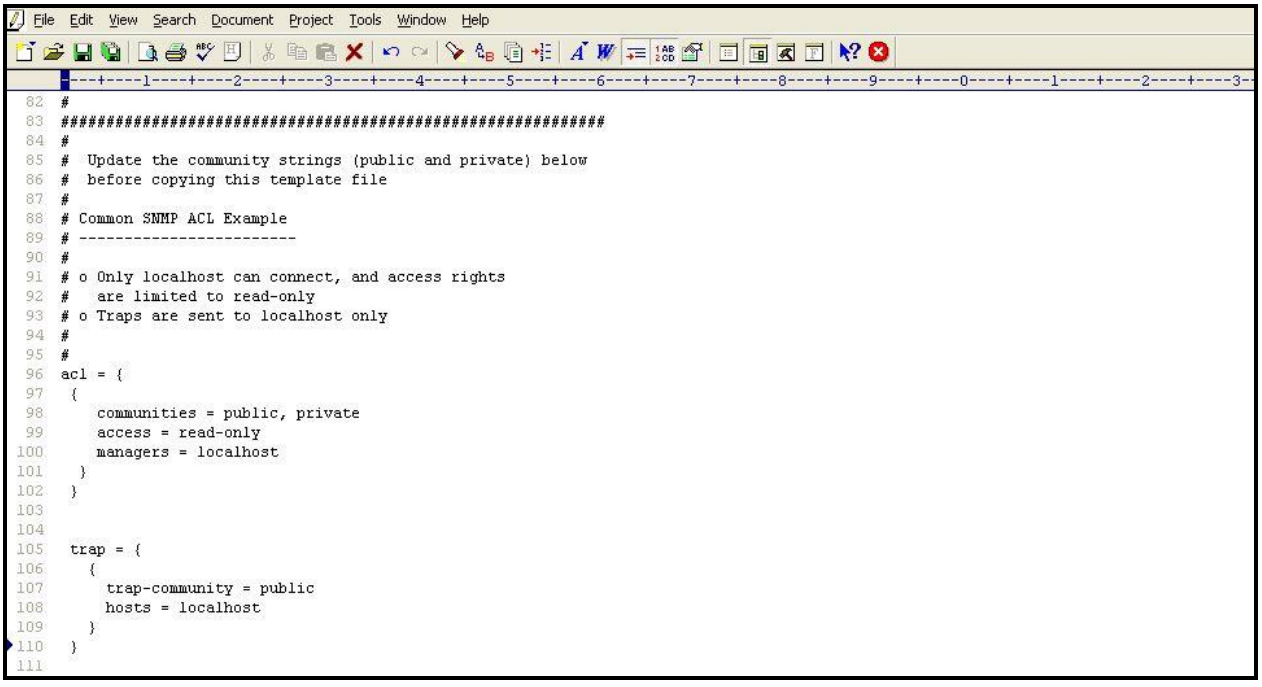
5. For that, first scroll down the file to view the sample code block revealed by Figure 4.



```
File  Edit  View  Search  Document  Project  Tools  Window  Help
82  #
83  ############################################################
84  #
85  #  Update the community strings (public and private) below
86  #  before copying this template file
87  #
88  # Common SNMP ACL Example
89  # ------------------------
90  #
91  # o Only localhost can connect, and access rights
92  #   are limited to read-only
93  # o Traps are sent to localhost only
94  #
95  #
96  # acl = {
97  #  {
98  #    communities = public, private
99  #    access = read-only
100 #    managers = localhost
101 #  }
102 # }
103 #
104 #
105 # trap = {
106 #   {
107 #     trap-community = public
108 #     hosts = localhost
109 #   }
110 # }
111
```

Figure 4: The snmp.acl file revealing the SNMP ACL example

6. Uncomment the code block by removing the # symbol preceding each line of the block as indicated by Figure 5.



```
File  Edit  View  Search  Document  Project  Tools  Window  Help
82  #
83  ############################################################
84  #
85  #  Update the community strings (public and private) below
86  #  before copying this template file
87  #
88  # Common SNMP ACL Example
89  # ------------------------
90  #
91  # o Only localhost can connect, and access rights
92  #   are limited to read-only
93  # o Traps are sent to localhost only
94  #
95  #
96  acl = {
97   {
98      communities = public, private
99      access = read-only
100     managers = localhost
101    }
102   }
103
104
105   trap = {
106     {
107       trap-community = public
108       hosts = localhost
109     }
110   }
111
```

Figure 5: Uncommenting the code block

7. Next, edit the code block to suit your environment.

8. The *acl* block expects the following parameters:

   - *communities* : Provide a comma-separated list of community strings, which an SNMP request should carry

for it to be serviced by this JRE; in the example illustrated by Figure 17, the community strings recognized by this JRE are *public* and *private*. You can add more to this list, or remove a *community string* from this list, if need be.

- *access* : Indicate the access rights that SNMP requests containing the defined *communities* will have; in Figure 5, SNMP requests containing the community string *public* or *private*, will have only *read-only* access to the MIB statistics. To grant full access, you can specify *read-write* instead.
- *managers :* Specify a comma-separated list of SNMP managers or hosts from which SNMP requests will be accepted by this JRE; in the example illustrated by Figure 5, all SNMP requests from the *localhost* will be serviced by this JRE. Typically, since the SNMP requests originate from an eG agent, the IP of the eG agent should be configured against the *managers* parameter. For instance, if the IP address of the agent host is *192.16.10.160*, then, to ensure that the JRE accepts requests from the eG agent alone, set *managers* to *192.168.10.160*, instead of *localhost*.

9. Every *acl* block in the *snmp.acl* file should have a corresponding *trap* block. This *trap* block should be configured with the following values:
   - *trap-community*: Provide a comma-separated list of community strings that can be used by SNMP traps sent by the eG manager to the *managers* specified in the *acl* block. In the example of Figure 5, all SNMP traps sent by the eG manager being monitored should use the community string *public* only.
   - *hosts*: Specify a comma-separated list of IP addresses / host names of hosts from which SNMP traps can be sent. In the case of Figure 5, traps can be sent by the *localhost* only. If a single *snmp.acl* file is being centrally used by multiple applications/devices executing on multiple hosts, then to ensure that all such applications are able to send traps to the configured SNMP *managers* (in the *acl* block), you can provide the IP address/hostname of these applications as a comma-separated list against *hosts*.

10. Figure 6 depicts how the *acl* and *trap* blocks can be slightly changed to suit the monitoring needs of an application.

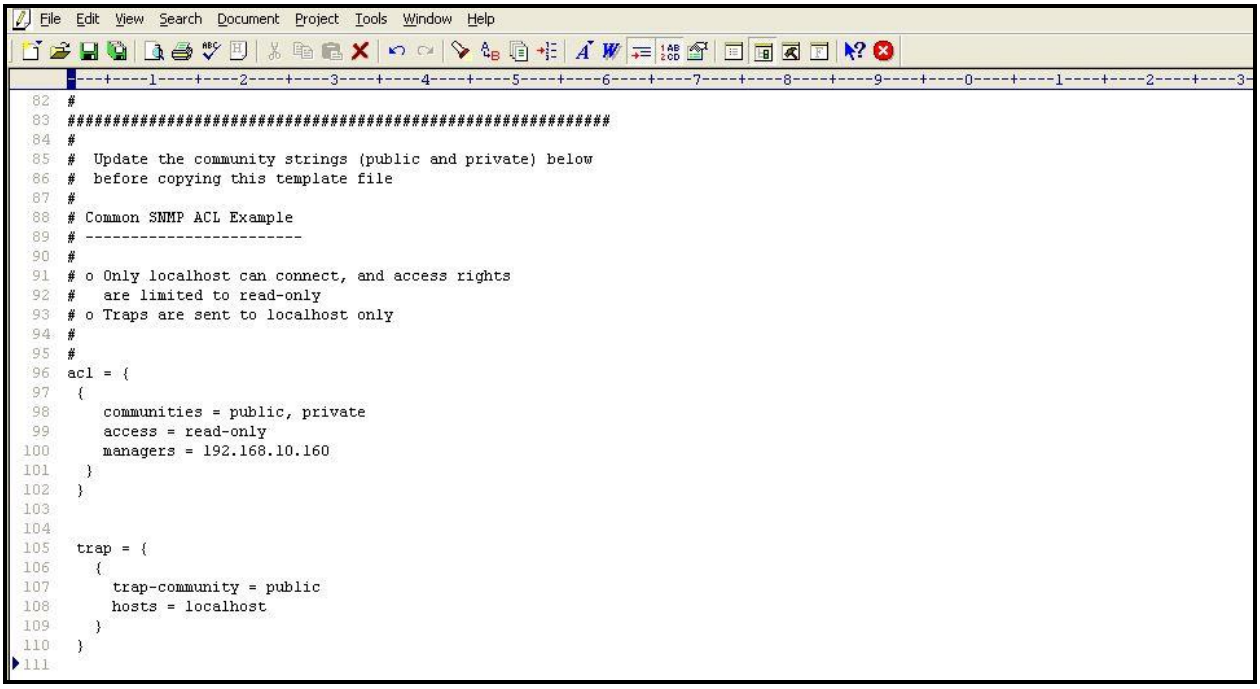


Figure 6: The edited block

11. Then, proceed to make the *snmp.acl* file secure by granting a single user "full access" to that file. For monitoring an eG manager executing on Windows in particular, only the *Owner* of the *snmp,.acl* file should have full control of that file. To know how to grant this privilege to the *Owner* of a file, refer to the *Monitoring Java Applications* docjment. This section actually details the procedure for making the *jmxremote.password* file on Windows, secure. Use the same procedure for making the *snmp.acl* file on Windows secure, but make sure that you select the *snmp.acl* file and not the *jmxremote.password* file.

    In case of an eG manager executing on Solaris / Linux hosts on the other hand, any user can be granted full access to the *snmp.acl* file, by following the steps below:

    - Login to the eG manager host as the user who is to be granted full control of the *snmp.acl* file.
    - Issue the following command:

      chmod 600 snmp.acl

    - This will automatically grant the login user full access to the *jmxremote.password* file.

12. Next, edit the **management.properties** file in the **<EG_INSTALL_DIR>\manager** folder (on Windows; on Unix, this will be: **/opt/egurkha/manager**).

13. Uncomment the **'For SNMP Settings'** block therein as indicated below:

```
################### For SNMP Settings ####################

com.sun.management.snmp.port=<port-number>
com.sun.management.snmp.acl=false
com.sun.management.snmp.acl.file=filepath
com.sun.management.snmp.interface=localhost

############################################################
```

14. Set the **com.sun.management.snmp.port** parameter in the block to the SNMP port of the eG manager. Then, set the **com.sun.management.snmp.acl** parameter to **true** (default: **false**). Next, against the **com.sun.management.snmp.acl.file** parameter, specify the full path to the **snmp.acl** file on the eG manager. Finally, provide the IP address of the host from which the eG manager's JRE should accept SNMP requests, against the **com.sun.management.snmp.interface** parameter. If this is set to, *0.0.0.0*, then it indicates that the JRE will accept SNMP requests from any host in the environment. To ensure that the JRE services only those SNMP requests that are received from the eG agent, set the **com.sun.management.snmp.interface** parameter to the IP address of the agent host.

    For example, if the eG manager being monitored listens for SNMP requests at port number 1166, the eG agent monitoring the eG manager is deployed on *192.168.10.152*, and these SNMP requests need to be authenticated using the *snmp.acl* file in the **D:\bea\jrockit_150_11\jre\lib** directory, then the above specification will read as follows:

    ```
    com.sun.management.snmp.port=1166
    com.sun.management.snmp.acl=true
    com.sun.management.snmp.acl.file=D:\\bea\\jrockit_150_11\\jre\\lib\\management\\snmp.acl
    com.sun.management.snmp.interface=192.168.10.152
    ```

    However, if the eG manager being monitored is executing on a Unix/Solaris/Linux host, and the *snmp.acl* file is in the **/usr/jdk1.5.0_05/jre/lib/management** folder of the host, then the **com.sun.management.snmp.acl.file** parameter will be set to:

    ```
    com.sun.management.snmp.acl.file =/usr/jdk1.5.0_05/jre/lib/management/snmp.acl
    ```

15. Next, edit the start-up script of the eG manager, include the following line in it, and save the script file.

    ```
    -Dcom.sun.management.config.file=<management.properties_file_path>
    ```

    For instance, on a Windows host, the *<management.properties_file_path>* can be expressed as: **D:\eGurkha\manager**.

    On other hand, on a Unix/Linux/Solaris host, the *<management.properties_file_path>* specification will be as follows: **/opt/egurkha/manager**.

1.2 The eG Database Layer

The eG manager uses an Oracle or an Microsoft SQL database to store performance metrics reported by the eG agent. To ensure the continuous availability of the eG manager, the eG database should be available, accessible, and have sufficient storage space at all times. The tests mapped to this layer focus on the health of the eG backend, and reveals the following:

- Whether the automatic database cleanup operations are functioning properly;
- Whether the database connection pool has adequate free connections;
- Whether the database has many large tables.

Figure 7: The tests mapped to the eG Database layer

1.2.1 eG Database Cleanup

To conserve space in the eG backend, administrators can schedule the automatic cleanup of tables from the eG database at specified intervals. If the cleanup process does not function as per schedule, the database may grow in size, thereby choking manager operations. You can avoid this by using this test to periodically check on the status of the cleanup activity, quickly detect cleanup failures, and also zero-in on the exact tables for which the activity failed, so that remedial measures can be swiftly initiated.

Purpose	To periodically check on the status of the cleanup activity, quickly detect cleanup failures, and also identify the tables for which the activity failed
Target of the test	The eG Manager
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST - The host for which the test is to be configured
3. PORT - The port number at which the specified HOST listens
4. JMX REMOTE PORT – Here, specify the port at which the JMX listens for requests from remote hosts. In the <EG_MANAGER_INSTALL_DIR>\manager directory (on Windows; on Unix, this will be the /opt/egurkha/manager directory) of the eG manager, you will find a management.properties file. Set the port defined against the com.sun.management.jmxremote.port parameter of the file as the JMX REMOTE PORT.
5. USER, PASSWORD, and CONFIRM PASSWORD – By default, JMX requires no authentication or security. Therefore, the user, PASSWORD , and CONFIRM PASSWORD parameters are set to none by default.
6. JNDINAME – The JNDINAME is a lookup name for connecting to the JMX connector. By default, this is jmxrmi. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
7. JMX PROVIDER – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to com.sun.jmx.remote.protocol.
8. TIMEOUT – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to 240 seconds.
9. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the eG manager being monitored</td></tr>
<tr><td>Measurements made by the</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

<table>
<tr>
<td rowspan="4">test</td>
<td>Cleanup status:
Indicates the current status of the database cleanup activity.</td>
<td></td>
<td>The values that this measure reports and the numeric values that correspond to them have been discussed in the table below:
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>Done</td><td>1</td></tr>
<tr><td>Running</td><td>0</td></tr>
<tr><td>Error</td><td>2</td></tr>
</table>
Note:
By default, this measure reports the Measure Values listed in the table above to indicate the current database cleanup status. The graph of this measure however, represents the same using the numeric equivalents only.</td>
</tr>
<tr>
<td>Time taken for cleanup:
Indicates the total time taken for the entire cleanup process.</td>
<td>Minutes</td>
<td>Ideally, the value of this measure should be low. A steady rise in this measure value is a cause for concern, as it indicates that the eG manager is taking an unusually long time to cleanup the database. This could be because of the presence of a large volume of data to be cleaned up. You may hence want to consider tuning the cleanup frequency, so that the database always has less data to cleanup.</td>
</tr>
<tr>
<td>Total successfully cleaned tables:
Indicates the number of tables that were successfully cleaned up.</td>
<td>Number</td>
<td>Use the detailed diagnosis of this measure to know which tables were successfully cleaned.</td>
</tr>
<tr>
<td>Time since recent cleanup:
Indicates the elapsed time since the last cleanup.</td>
<td>Minutes</td>
<td>With the help of this measure, you can quickly detect whether a scheduled cleanup occurred on the eG database or not.</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>Has cleanup run today?:

Indicates whether cleanup has run this day or not.</td>
<td></td>
<td>Typically, cleanup is scheduled to take place at the end of every day. If the value of this measure is Yes, it indicates that cleanup has run today. On the other hand, if this measure reports the value No, it indicates that cleanup is yet to run for that day or has not run at all.

The numeric values that correspond to the current day's cleanup status are as follows:

<table>
<tr><th>Measure Value</th><th>Numeric 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 Measure Values listed in the table above to indicate whether/not cleanup has run today. The graph of this measure however, represents the same using the numeric equivalents only.</td>
</tr>
</table>

	Is cleanup running as a separate process?: Indicates whether/not cleanup is running as a separate process.		The eG manager is a 32-bit application that runs as a Java process. The maximum heap memory that can be allocated to this process is limited to 1.5 GB. Even if the physical server on which the eG manager is installed has more memory, since it is a single Java process, the eG manager cannot exploit the additional memory available on the server. To overcome this limitation, in eG Enterprise, the critical eG manager functions such as email alert management, threshold computation, trending, and database cleanup activities can all be run as separate Java processes (i.e., in addition to the core eG manager process). Removing these key functions from the core eG manager process makes additional memory available for the core eG manager functions including data reception and analysis, alarm correlation, and web-based access and reporting. This reconfiguration of the eG manager into separate Java processes allows the eG manager to make better utilization of available server hardware resources and thereby offers enhanced scalability. In turn, this allows customers to get more leverage from their existing investment in the hardware that hosts the eG manager. If cleanup has been configured to run as a separate Java process, then the value of this measure will be *Yes*. If not, then this measure reports the value *No*. The numeric values that correspond to the measure values above are as follows: **Measure Value** / **Numeric Value** Yes / 1 No / 0

			Note: By default, this measure reports the **Measure Value**s listed in the table above to indicate whether/not cleanup runs as a separate Java process. The graph of this measure however, represents the same using the numeric equivalents only.
	Has cleanup timed out?: Indicates whether/not the last database cleanup process timed out.		If the last cleanup process timed out, then the value of this measure will be *Yes*. If not, then this measure reports the value *No*. The numeric values that correspond to the measure values above are as follows: **Measure Value** / **Numeric Value** Yes / 1 No / 0 **Note:** By default, this measure reports the **Measure Value**s listed in the table above to indicate whether/not cleanup timed out. The graph of this measure however, represents the same using the numeric equivalents only.
	Total backlogged tables: Indicates the number of tables on which cleanup was incomplete.	Number	This measure indicates the number of tables on which cleanup occurred, but could not delete all the records that were marked for deletion. This can happen if a table contains too many records to be deleted. In such circumstances, the eG manager cleans up a few records from the table each day for the next few days. Use the detailed diagnosis of this measure to know which tables have records that are yet to be cleaned by cleanup.
	Total tables that failed cleanup: Indicates the number of tables on which cleanup failed.	Number	Ideally, the value of this measure should be 0. A non-zero value is indicative of cleanup failures. Use the detailed diagnosis of this measure to isolate the tables on which cleanup failed.

The detailed diagnosis of the *Total successfully cleaned tables* measure lists the tables that were successfully cleaned and the time taken by the cleanup process to clean each table.

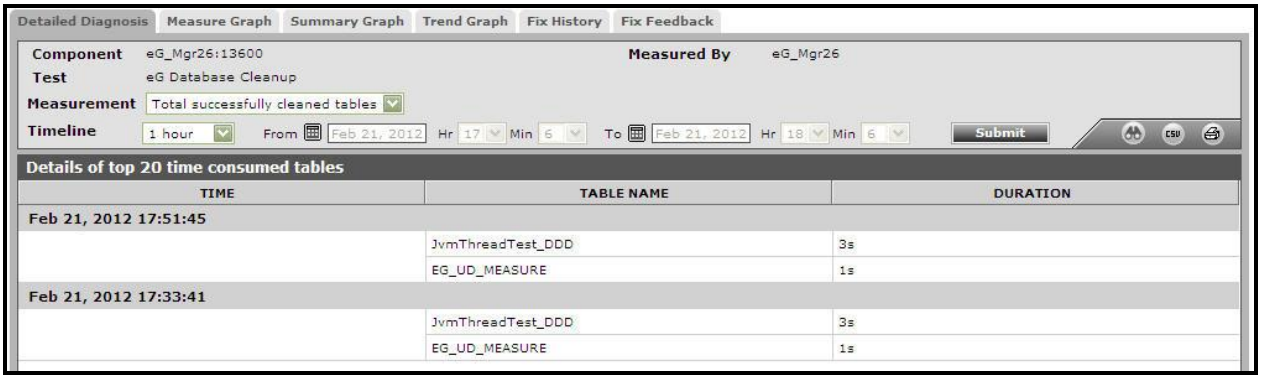


Figure 8: The detailed diagnosis of the Total successfully cleaned tables measure

The detailed diagnosis of the *Total backlogged tables* measure lists the tables on which cleanup was incomplete. For each table, the detailed diagnosis also provides the number of days of data from the table that is yet to be cleaned and number of days of data that has already been cleaned.

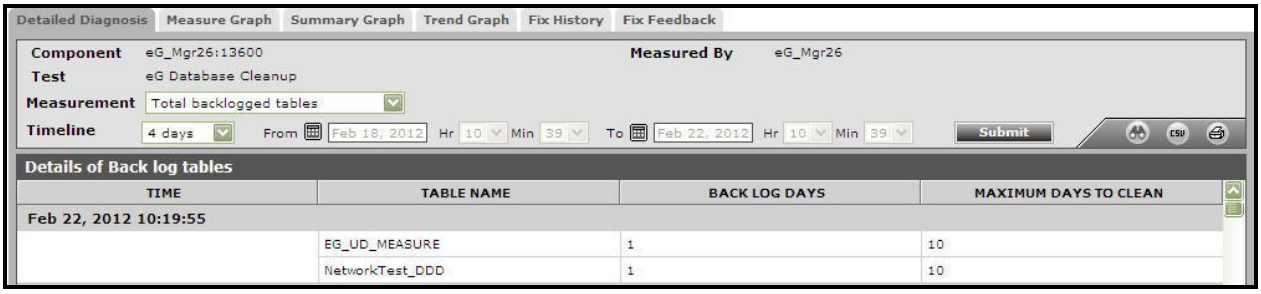


Figure 9: The detailed diagnosis of the Total backlogged tables measure

1.2.2 eG Database Connections Test

To optimize accesses to the database, the eG manager uses connection pooling. By using a pre-established set of connections and multiplexing requests over these connections, the eG manager ensures that individual connections are not established and closed for each request. Using the eG administrative interface, an administrator can configure the initial number of connections in the pool and the maximum number of connections. If the pool has not been configured with enough connections, then the eG manager may not be able to connect to the database to store the critical performance metrics reported by the eG agents. Therefore, to be able to judge the adequacy of this configuration, you need to understand how the eG manager uses the connection pool. The **eG Database Connections** test sheds light on the usage of the connection pool by the eG manager, and indicates whether/not the connection pool has been rightly sized.

Purpose	Sheds light on the usage of the connection pool by the eG manager, and indicates whether/not the connection pool has been rightly sized.
Target of the test	The eG Manager
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST -** The host for which the test is to be configured 3. **PORT -** The port number at which the specified **HOST** listens 4. **JMX REMOTE PORT** – Here, specify the port at which the **JMX** listens for requests from remote hosts. In the **<EG_MANAGER_INSTALL_DIR>\manager** directory (on Windows; on Unix, this will be the **/opt/egurkha/manager** directory) of the eG manager, you will find a **management.properties** file. Set the port defined against the **com.sun.management.jmxremote.port** parameter of the file as the **JMX REMOTE PORT**. 5. **USER**, **PASSWORD**, and **CONFIRM PASSWORD** – By default, JMX requires no authentication or security. Therefore, the user, **PASSWORD** , and **CONFIRM PASSWORD** parameters are set to *none* by default. 6. **JNDINAME** – The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is *jmxrmi*. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same. 7. **JMX PROVIDER** – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to *com.sun.jmx.remote.protocol*. 8. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to *240* seconds. 9. **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.
Outputs of the test	One set of results for the eG manager being monitored

Measurements made by the test	Measurement	Measurement Unit	Interpretation
	Initial connections: Indicates the number of initial connections allocated to the pool.	Number	Both these parameters can be configured using the **DATABASE SETTINGS** page that appears when the Configure -> Data Management -> Database Settings menu sequence is followed in the eG administrative interface. Consider increasing the number of connections in the connection pool as the number of components monitored or the number of users accessing the system increases.
	Maximum connections: Indicates the maximum number of database connections that the eG manager can use from the pool.	Number	
	Used connections: Indicates the number of connections currently used by the eG manager.	Number	If the value of this measure grows dangerously close to the value of the *Maximum connections* measure, it indicates that the pool is rapidly running out of connections. In such a case, you can use the detailed diagnosis of this measure to know which classes are currently utilizing the database connections, and how many connections have been used per class. Classes that are over-utilizing database connections can thus be identified, and the reasons for the same can be determined.
	Free connections: Indicates the time taken for connections to time out.	Number	A low value for this measure is a cause for concern, as it indicates that the connection pool does not have enough free connections. This could be owing to an increase in the number of components monitored or the number of users accessing the systems. You may hence want to consider fine-tuning the **Initial Connections** and **Maximum Connections** parameters (using the eG administrative interface), so that the pool is sufficiently sized to handle the increase in workload.
	Connection timeout: Indicates the time taken for connections to time out.	Secs	By carefully observing the value for this measure over time, you can figure out the maximum ideal time for which a connection should stay alive.
	Total connections: Indicates the total number of connections in the pool.	Number	

	Connections waiting: Indicates the number of connections that are currently waiting to be closed on the eG manager.	Number	Ideally, the value of this measure should be low. A very high value of this measure is a cause for concern as it indicates that too many connections to the database are currently open and are waiting to be closed, but are unable to close owing to certain issues. Besides unnecessarily consuming connections from the pool and causing a paucity of connections in the pool, these waiting connections, if allowed to increase, can even cause the manager to restart.

The detailed diagnosis of the *Used connections* measure lists the eG manager classes that are using connections from the database connection pool, and the number of connections each class is using. Classes that are consuming too many connections can thus be identified and the reasons for this abnormal usage can be investigated.

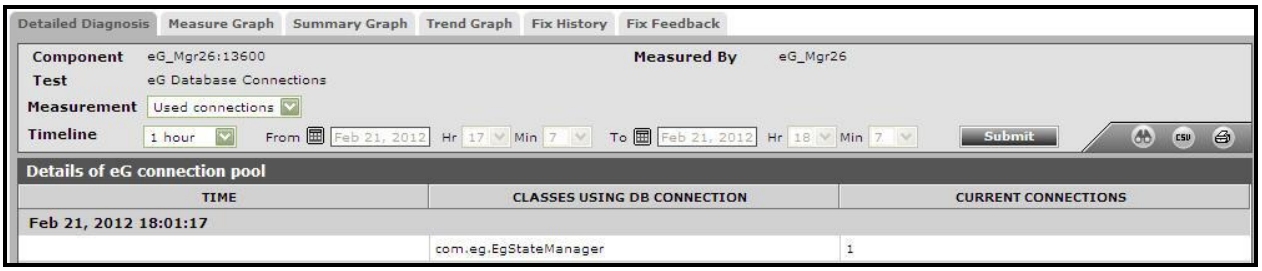


Figure 10: The detailed diagnosis of the Used connections measure

1.2.3 eG Database Tables Test

This test monitors the tables in the eG database and turns the spotlight on the following:

- Large sized tables in terms of the volume of data they store;
- Tables with too many rows of data;
- Tables that are badly structured - i.e., tables without index or tables without primary key

Purpose	Monitors the tables in the eG database and turns the spotlight on the following: • Large sized tables in terms of the volume of data they store; • Tables with too many rows of data; • Tables that are badly structured - i.e., tables without index or tables without primary key

Target of the test	The eG Manager
Agent deploying the test	An internal/remote agent
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST -** The host for which the test is to be configured 3. **PORT -** The port number at which the specified **HOST** listens 4. **JMX REMOTE PORT** – Here, specify the port at which the **JMX** listens for requests from remote hosts. In the **<EG_MANAGER_INSTALL_DIR>\manager** directory (on Windows; on Unix, this will be the **/opt/egurkha/manager** directory) of the eG manager, you will find a **management.properties** file. Set the port defined against the **com.sun.management.jmxremote.port** parameter of the file as the **JMX REMOTE PORT**. 5. **USER**, **PASSWORD**, and **CONFIRM PASSWORD** – By default, JMX requires no authentication or security. Therefore, the user, **PASSWORD** , and **CONFIRM PASSWORD** parameters are set to *none* by default. 6. **JNDINAME** – The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is *jmxrmi*. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same. 7. **JMX PROVIDER** – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to *com.sun.jmx.remote.protocol*. 8. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to *240* seconds. 9. **MAXIMUM TABLE SIZE** - Specify the maximum size limit (in KB) that a table should exceed in order to be counted as a high volume table. 10. **MAXIMUM TABLE SIZE** - Specify the minimum size limit (in KB) that a table should fall below in order to be counted as a low volume table. 11. **MAXIMUM ROW SIZE** - This test reports all tables with more number of rows than the row limit specified here as *Tables with many rows*. 12. **MINIMUM ROW SIZE** - This test counts reports tables with less number of rows than the row limit specified here as *Tables with few rows*. 13. **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.

Outputs of the test	One set of results for the eG manager being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Total tables: Indicates the total number of tables that currently exist in the eG database.	Number	
	Default tables: Indicates the number of default tables in the eG database.	Number	
	Dynamic tables: Indicates the number of dynamic tables currently present in the eG manager database.		Dynamic tables are typically a set of virtual tables used to record current database activity.
	Tables without primary key: Indicates the number of tables without a primary key.	Number	The purpose of a primary key is to uniquely identify each record in a table. An index on the other hand, is an alphabetical listing of the terms found on database records and the number of records on which the terms are found. A database table can survive without a primary key or an index, but this deters the whole purpose of a database: the easy retrieval of information. To know which tables do not have an index or a primary key, use the detailed diagnosis of this measure.
	Tables without index: Indicates the number of tables without an index.	Number	
	Newly created tables: Indicates the number of tables created in the eG database since the last measurement period.	Number	New tables will be created in the eG database as and when new components are monitored in the environment.
	Active sessions: Indicates the number of active sessions to the eG backend.	Number	This is a good indicator of the current load on the eG database. Use the detailed diagnosis of this measure to view the hosts from which the active sessions originated.

	Inactive sessions: Indicates number of currently inactive sessions to the eG database.	Number	Open, but inactive sessions are serious resource drains. Use the detailed diagnosis of this measure to view the hosts that are responsible for initiating the inactive sessions, determine the reason for the inactivity, and if found necessary, terminate the sessions in order to protect the database from abuse.
	Tables with highest volume of data: Indicates the number of tables that contain a high data volume.	Number	This measure value includes all tables that are of a size greater than the **MAXIMUM TABLE SIZE** configured for this test. Use the detailed diagnosis to identify the large sized tables.
	Tables with lowest volume of data: Indicates the number of tables that contain a low data volume.	Number	This measure value includes all tables that are of a size lesser than the **MINIMUM TABLE SIZE** configured for this test. Use the detailed diagnosis to identify the small sized tables.
	Tables with many rows: Indicates the number of tables with many rows.	Number	This measure value includes all tables with more number of rows than the **MAXIMUM ROW SIZE** configured for this test. Use the detailed diagnosis to identify the tables with too many rows.
	Tables with few rows: Indicates the number of tables with few rows.	Number	This measure value includes all tables with less number of rows than the **MINIMUM ROW SIZE** configured for this test. Use the detailed diagnosis to identify the tables with only a few rows.

The detailed diagnosis of the *Tables with highest volume of data* measure displays all the tables with high data volume, and reports the current size of each table.

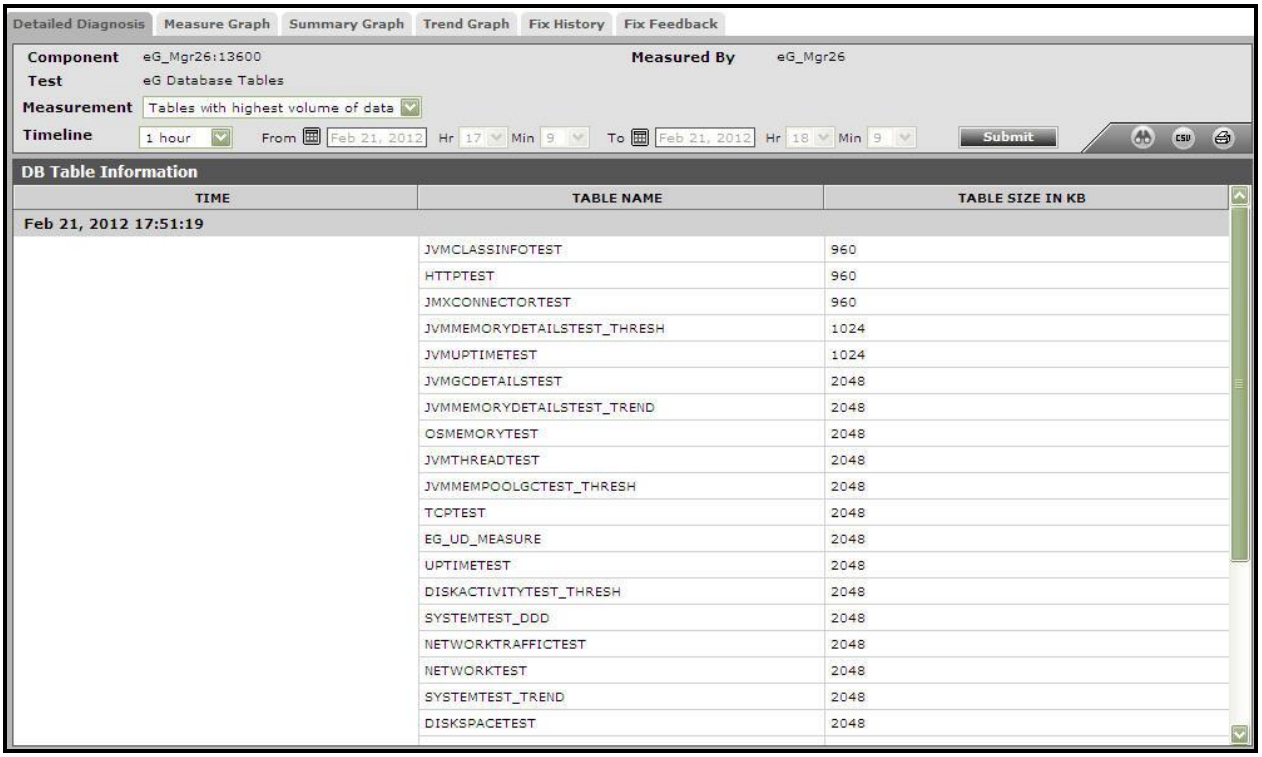


Figure 11: The detailed diagnosis of the Tables with highest volume of data measure

The detailed diagnosis of the *Tables with lowest volume of data* measure displays all the tables with low data volume, and reports the current size of each table.

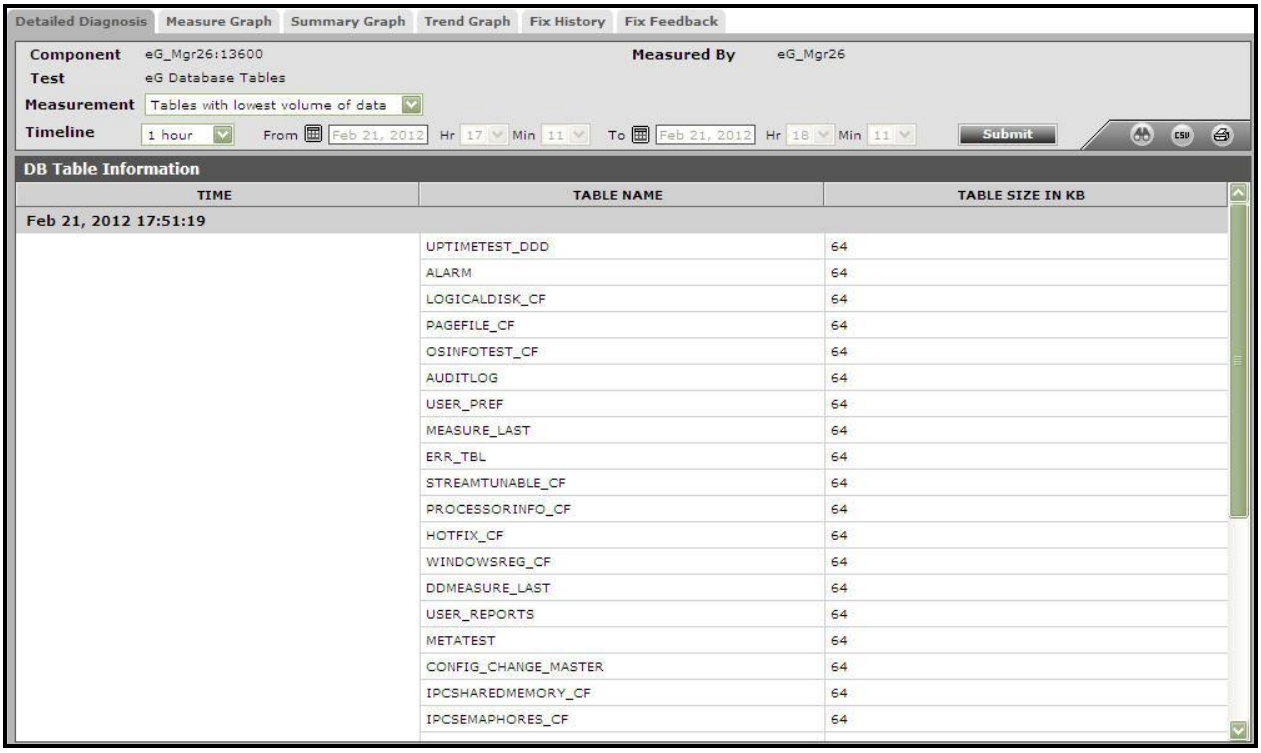


Figure 12: The detailed diagnosis of the Tables with lowest volume of data measure

The detailed diagnosis of the *Tables with many rows* measure displays all the tables with many rows, and the number of rows in each table.

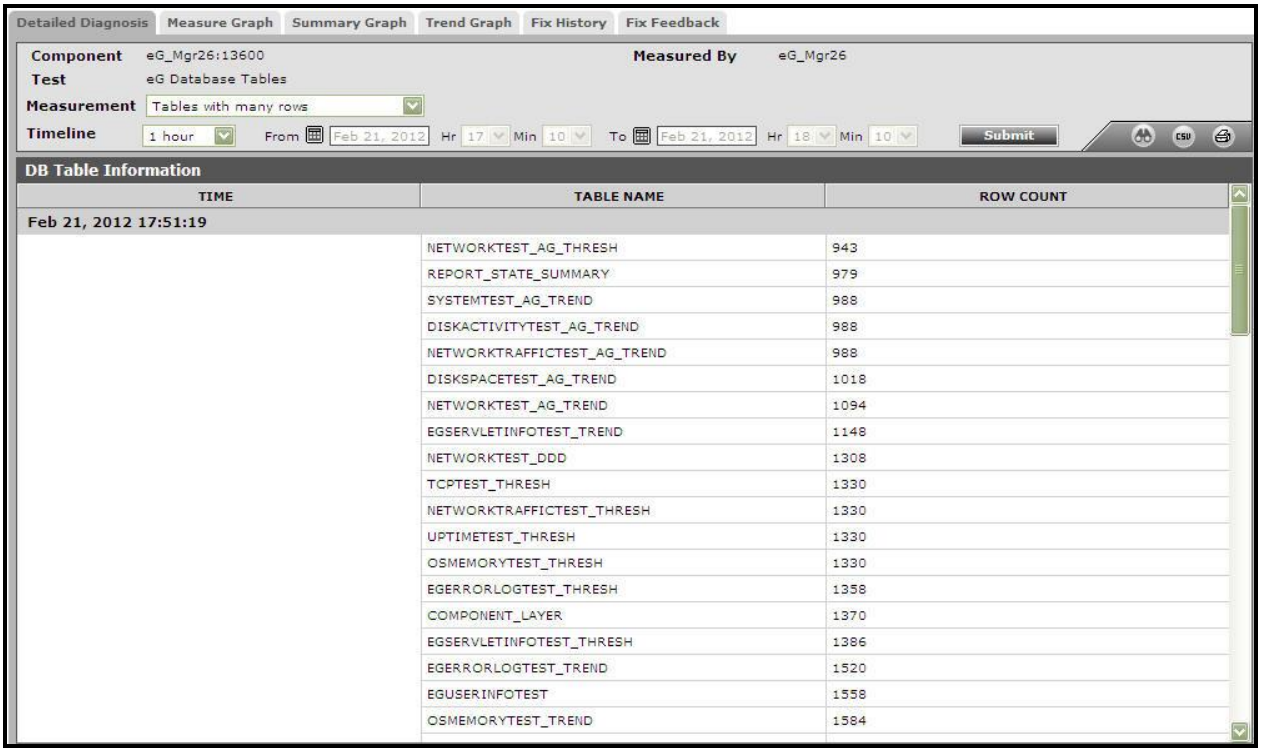


Figure 13: The detailed diagnosis of the Tables with many rows measure

The detailed diagnosis of the *Tables with few rows* measure displays all the tables with few rows, and the number of rows in each table.

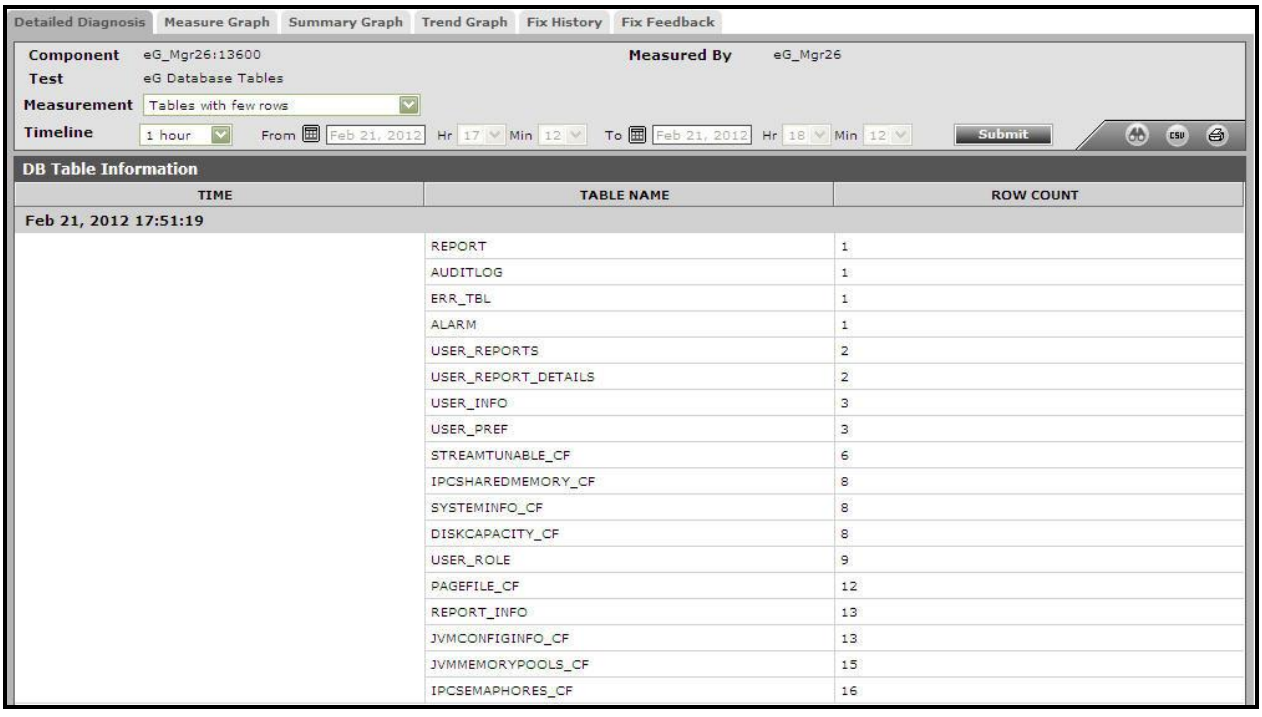


Figure 14: The detailed diagnosis of the Tables with few rows measure

The detailed diagnosis of the *Tables without primary key* measure displays all the tables that do not have any primary key.

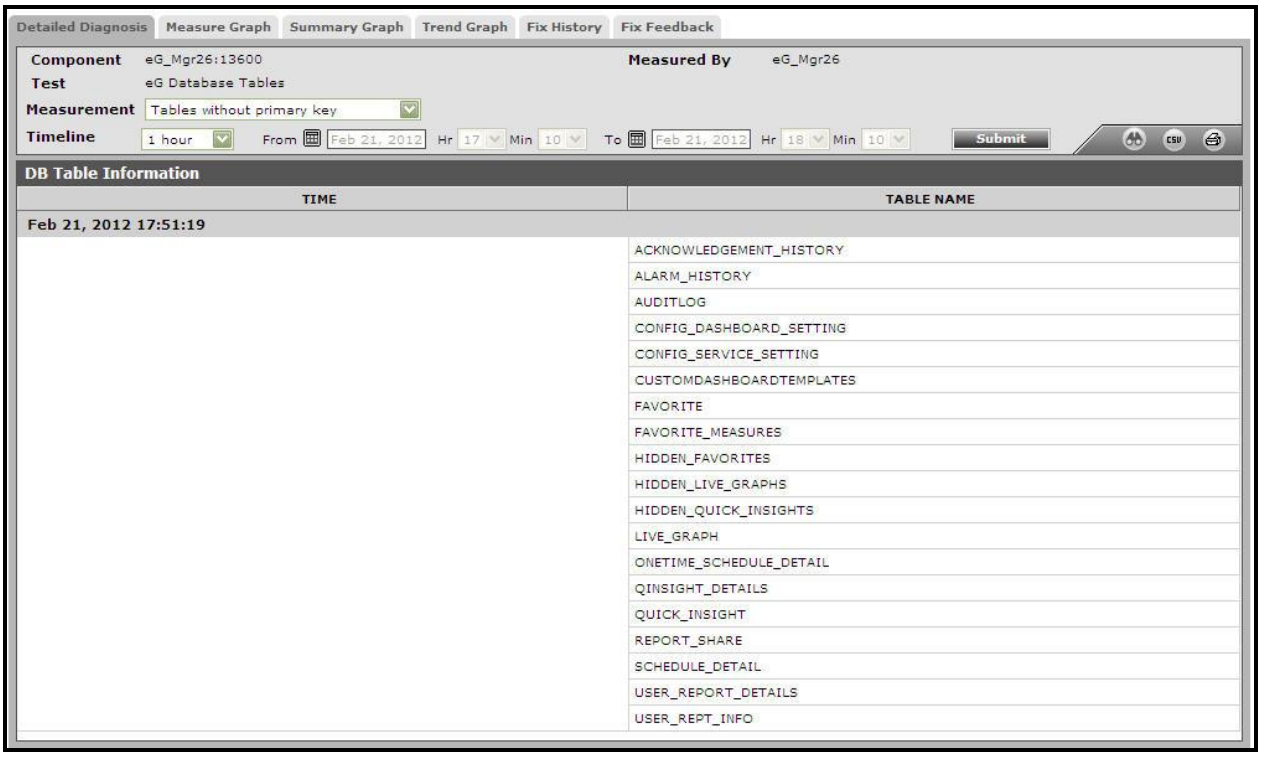


Figure 15: The detailed diagnosis of the Tables without primary key measure

The detailed diagnosis of the *Tables without index* measure displays all the tables that do not have any index.

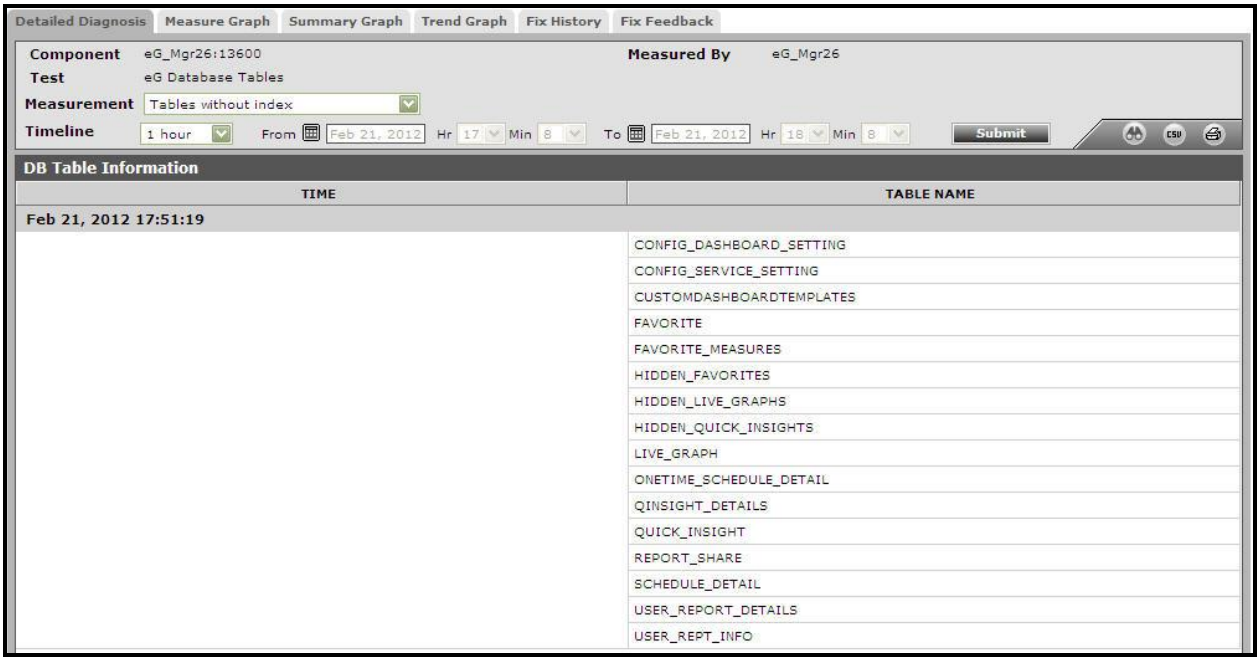


Figure 16: The detailed diagnosis of the Tables without index measure

1.2.4 eG Table Triggers Test

This test scans each table in the eG database for triggers and reports the total number of triggers, the count of triggers enabled, and those that have been dropped/are missing.

Purpose	• Scans each table in the eG database for triggers and reports the total number of triggers, the count of triggers enabled, and those that have been dropped/are missing in the eG database
Target of the test	The eG Manager
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST - The host for which the test is to be configured
3. PORT - The port number at which the specified HOST listens
4. JMX REMOTE PORT – Here, specify the port at which the JMX listens for requests from remote hosts. In the <EG_MANAGER_INSTALL_DIR>\manager directory (on Windows; on Unix, this will be the /opt/egurkha/manager directory) of the eG manager, you will find a management.properties file. Set the port defined against the com.sun.management.jmxremote.port parameter of the file as the JMX REMOTE PORT.
5. USER, PASSWORD, and CONFIRM PASSWORD – By default, JMX requires no authentication or security. Therefore, the user, PASSWORD , and CONFIRM PASSWORD parameters are set to none by default.
6. JNDINAME – The JNDINAME is a lookup name for connecting to the JMX connector. By default, this is jmxrmi. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
7. JMX PROVIDER – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to com.sun.jmx.remote.protocol.
8. TIMEOUT – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to 240 seconds.
9. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the eG manager being monitored</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Total triggers count:
Indicates the total number of triggers in the eG database.</td><td>Number</td><td rowspan="2"></td></tr>
<tr><td>Enabled triggers:
Indicates the number of triggers that are currently enabled.</td><td>Number</td></tr>
</table>

	Disabled triggers: Indicates the number of triggers that are currently disabled.	Number	
	Missing triggers: Indicates the number of triggers that have been dropped and are hence missing from the database.	Number	

1.3 The eG Application Layer

Using the tests mapped to this layer, you can quickly capture errors/warnings logged in the various eG manager logs, and instantly detect abnormalities in the functioning of the following:

- eG agents
- eG State manager process
- eG Threshold manager process
- eG Trend manager process
- eG Object Pools
- eG Servlets
- If the eG manager experiences a slowdown or a performance degradation, the tests mapped to this layer will give you effective pointers to the source of the slowdown.

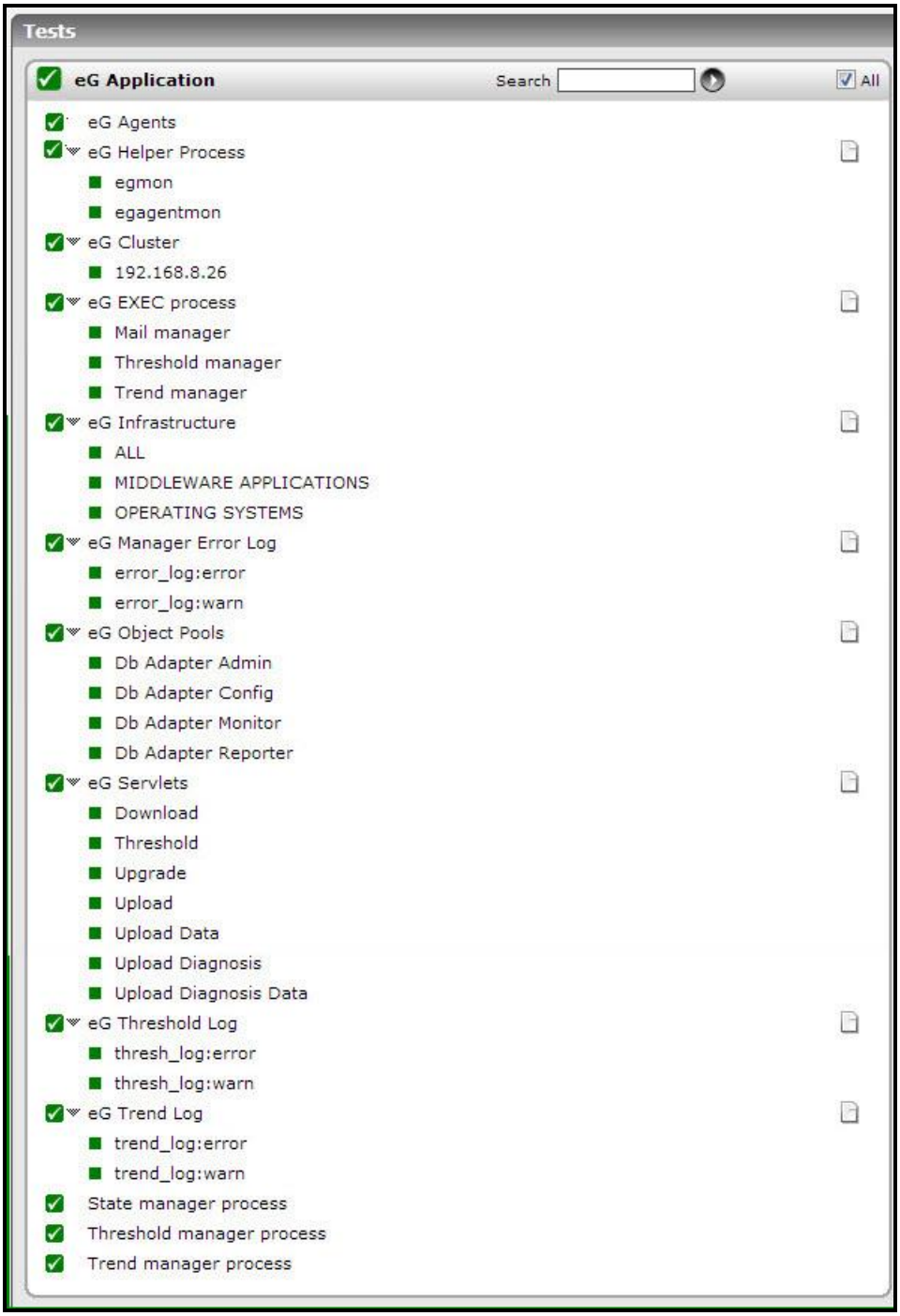


Figure 17: The tests mapped to the eG Application layer

1.3.1 eG Infrastructure Test

This test auto-discovers the applications managed by the eG Enterprise system, automatically categorizes the applications into pre-configured categories, and for every category, reports the total number of managed, unmanaged, and deleted components. The test also tracks the recent management activity on the eG Enterprise

system by reporting the number of components that were managed, unmanaged, and deleted, recently. Detailed diagnostics reveals the names of components managed, unmanaged, and deleted.

Purpose	Auto-discovers the applications managed by the eG Enterprise system, automatically categorizes the applications into pre-configured categories, and for every category, reports the total number of managed, unmanaged, and deleted components
Target of the test	The eG Manager
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST -** The host for which the test is to be configured 3. **PORT -** The port number at which the specified **HOST** listens 4. **JMX REMOTE PORT** – Here, specify the port at which the **JMX** listens for requests from remote hosts. In the **<EG_MANAGER_INSTALL_DIR>\manager** directory (on Windows; on Unix, this will be the **/opt/egurkha/manager** directory) of the eG manager, you will find a **management.properties** file. Set the port defined against the **com.sun.management.jmxremote.port** parameter of the file as the **JMX REMOTE PORT**. 5. **USER**, **PASSWORD**, and **CONFIRM PASSWORD** – By default, JMX requires no authentication or security. Therefore, the user, **PASSWORD** , and **CONFIRM PASSWORD** parameters are set to *none* by default. 6. **JNDINAME** – The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is *jmxrmi*. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same. 7. **JMX PROVIDER** – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to *com.sun.jmx.remote.protocol*. 8. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to *240* seconds. 9. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against **DD FREQUENCY**. 10. **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.		
Outputs of the test	One set of results for each category of components managed by the eG manager being monitored		
Measurements made by the	**Measurement**	**Measurement Unit**	**Interpretation**

test	**Total managed components:** Indicates the total number of components of this category that are managed by the eG manager.	Number	Use the detailed diagnosis of this measure to know how many components of which component type are being managed by the eG manager.
	Recently managed components: Indicates the number of components of this category that were managed during the last measurement period.	Number	Use the detailed diagnosis of this measure to know how many components of which component type were managed in the last measurement period.
	Total unmanaged components: Indicates the total number of components of this category that are unmanaged.	Number	Use the detailed diagnosis of this measure to know how many components of which component type are unmanaged.
	Recently unmanaged components: Indicates the number of components of this category that were unmanaged during the last measurement period.	Number	Use the detailed diagnosis of this measure to know how many components of which component type were recently unmanaged.
	Total deleted components: Indicates the total number of components of this category that have been deleted.	Number	Use the detailed diagnosis of this measure to know how many components of which component type have been deleted.
	Recently deleted components: Indicates the number of components of this category that were deleted during the last measurement period.	Number	Use the detailed diagnosis of this measure to know how many components of which component type were recently deleted.

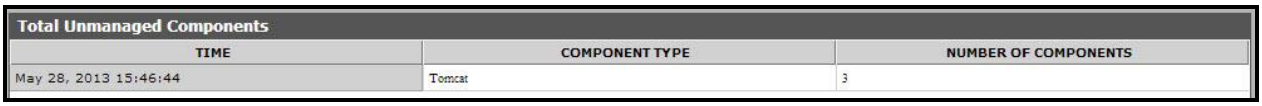
The detailed diagnosis of the *Total managed components* measure displays the number of components of each component type that are being managed.

Total Managed Components

TIME	COMPONENT TYPE	NUMBER OF COMPONENTS
May 28, 2013 15:46:44		
	eG Manager	1
	Java Application	1

Figure 18: The detailed diagnosis of the Total managed components measure

The detailed diagnosis of the *Total unmanaged components* measure displays the number of components of each component type that have been unmanaged.

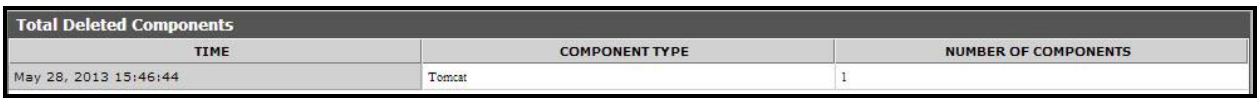


Total Unmanaged Components

TIME	COMPONENT TYPE	NUMBER OF COMPONENTS
May 28, 2013 15:46:44	Tomcat	3

Figure 19: The detailed diagnosis of the Total unmanaged components measure

The detailed diagnosis of the *Total deleted components* measure displays the number of components of each component type that have been deleted.

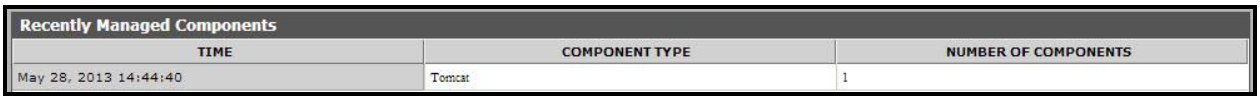


Total Deleted Components

TIME	COMPONENT TYPE	NUMBER OF COMPONENTS
May 28, 2013 15:46:44	Tomcat	1

Figure 20: The detailed diagnosis of the Total deleted components measure

The detailed diagnosis of the *Recently managed components* measure displays the number of components of each component type that were managed during the last measurement period.

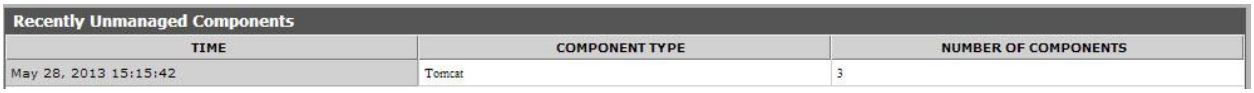


Recently Managed Components

TIME	COMPONENT TYPE	NUMBER OF COMPONENTS
May 28, 2013 14:44:40	Tomcat	1

Figure 21: The detailed diagnosis of the Recently managed components measure

The detailed diagnosis of the *Recently unmanaged components* measure displays the number of components of each component type that were unmanaged during the last measurement period.



Recently Unmanaged Components

TIME	COMPONENT TYPE	NUMBER OF COMPONENTS
May 28, 2013 15:15:42	Tomcat	3

Figure 22: The detailed diagnosis of the Recently unmanaged components measure

1.3.2 eG EXEC Process Test

Administrators can optionally configure the eG manager to run each of its critical functions - i.e., email alert management, threshold computation, trending, and database cleanup activities - as separate processes, so that additional memory is available for the core eG manager functions such as data reception and analysis, alarm correlation, and web-based access and reporting. Using this test, you can identify the manager functions that have been set to execute as separate processes. The test also sheds light on situations where more than one instance of a process may be running, so that you can conserve resources by terminating the unnecessary instances.

Purpose	Helps identify the manager functions that have been set to execute as separate processes and sheds light on situations where more than one instance of a process may be running
Target of the test	The eG Manager

Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST -** The host for which the test is to be configured 3. **PORT -** The port number at which the specified **HOST** listens 4. **JMX REMOTE PORT** – Here, specify the port at which the **JMX** listens for requests from remote hosts. In the **<EG_MANAGER_INSTALL_DIR>\manager** directory (on Windows; on Unix, this will be the **/opt/egurkha/manager** directory) of the eG manager, you will find a **management.properties** file. Set the port defined against the **com.sun.management.jmxremote.port** parameter of the file as the **JMX REMOTE PORT**. 5. **USER**, **PASSWORD**, and **CONFIRM PASSWORD** – By default, JMX requires no authentication or security. Therefore, the user, **PASSWORD** , and **CONFIRM PASSWORD** parameters are set to *none* by default. 6. **JNDINAME** – The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is *jmxrmi*. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same. 7. **JMX PROVIDER** – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to *com.sun.jmx.remote.protocol*. 8. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to *240* seconds. 9. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against **DD FREQUENCY**. 10. **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.		
Outputs of the test	One set of results for critical eG manager function		
Measurements made by the	**Measurement**	**Measurement Unit**	**Interpretation**

<table>
<tr><td>test</td><td>Process count:

Indicates the number of instances of this process that is currently running.</td><td>Number</td><td>The test will report this measure only if the value of the 'Is EXEC process?' measure is 'Yes'.

The value 1 is desired for this measure. Any value above 1 is a sign that more instances of a process are unnecessarily running and draining resources. Use the detailed diagnosis of this measure to know the process ID of the additional processes, so that you can kill them to conserve resources.</td></tr>
<tr><td></td><td>Is EXEC process ?

Indicates whether this critical manager function has been configured to execute as a separate process.</td><td></td><td>The values that this measure can report and their corresponding numeric values have been listed in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>0</td></tr></table>
It is recommended that the critical manager functions of email alert management, threshold computation, trending, and database cleanup activities are configured to run as separate processes. This is because, removing these key functions from the core eG manager process makes additional memory available for the core eG manager functions including data reception and analysis, alarm correlation, and web-based access and reporting. In addition, it also allows the eG manager to make better utilization of available server hardware resources and thereby offers enhanced scalability. In turn, this allows customers to get more leverage from their existing investment in the hardware that hosts the eG manager.

Note:

By default, the test reports the Measure Values listed in the table above to indicate whether a function has been set to run as a separate process or not. In the graph of this measure however, the same is represented using the numeric equivalents only.</td></tr>
</table>

1.3.3 eG Helper Process Test

To ensure enterprise-class monitoring, the eG manager includes the capability to monitor its various components and to recover from failure of these components. When the eG manager is started, a separate eG recovery process is started. This process is called eGmon. Likewise, when the eG agent is started, a recovery process named *eGagentmon* also starts simultaneously.

The eGmon process periodically attempts to connect to the eG manager, access the various components of the manager, including the eG database. If it detects any problems during such access, the recovery process attempts to perform further diagnosis. The specific actions performed by the recovery process are as follows:

- If the eG manager is not accessible, the recovery process attempts to restart the eG manager. If it fails to restart the eG manager thrice in succession, the recovery process generates an alert message to the eG administrator (using the **MAIL SENDER ID** specified in the **Mail Configuration** settings of the administration interface).
- If the eG manager is accessible, the recovery process tests the connections from the eG manager to the database server that it uses. In the event it detects problems, it alerts the administrator of potential problems with the database server access. By connecting directly to the database server (i.e., without using any other eG manager components), the recovery process further determines whether the database access problem is being caused either because of a database failure or because the eG manager's pool of database connections is not sufficient to handle the current load on the manager.

When the eG manager is stopped manually, the eG recovery process is also shutdown.

In the same way, the eGagentmon process attempts to connect to the eG agent, and upon detection of accessibility issues, restarts the agent. However, note that if the eG agent is stopped manually, the agent recovery process is also shutdown.

This test reports the health of the *eGmon* and *eGagentmon* processes. Using this test, you can determine whether these helper processes are running or not, and if running, whether/not they are performing the checks that they are programmed to perform at pre-configured intervals. This way, you can be proactively alerted to the inadvertent termination of these critical help processes and errors in their operations.

Purpose	Reports the health of the *eGmon* and *eGagentmon* processes
Target of the test	The eG Manager
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST -** The host for which the test is to be configured 3. **PORT -** The port number at which the specified **HOST** listens 4. **JMX REMOTE PORT** – Here, specify the port at which the **JMX** listens for requests from remote hosts. In the **<EG_MANAGER_INSTALL_DIR>\manager** directory (on Windows; on Unix, this will be the **/opt/egurkha/manager** directory) of the eG manager, you will find a **management.properties** file. Set the port defined against the **com.sun.management.jmxremote.port** parameter of the file as the **JMX REMOTE PORT**. 5. **USER**, **PASSWORD**, and **CONFIRM PASSWORD** – By default, JMX requires no authentication or security. Therefore, the user, **PASSWORD** , and **CONFIRM PASSWORD** parameters are set to *none* by default. 6. **JNDINAME** – The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is *jmxrmi*. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same. 7. **JMX PROVIDER** – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to *com.sun.jmx.remote.protocol*. 8. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to *240* seconds. 9. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against **DD FREQUENCY**. 10. **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.		
Outputs of the test	One set of results for the eGmon and eGagentmon processes		
Measurements made by the	**Measurement**	**Measurement Unit**	**Interpretation**

<table>
<tr><td>test</td><td>Process count:
Indicates the number of instances of this process that is currently running.</td><td>Number</td><td>The value 1 is desired for this measure. Any value above 1 is a sign that more instances of a process are unnecessarily running and draining resources. Use the detailed diagnosis of this measure to know the process ID of the additional processes, so that you can kill them to conserve resources.</td></tr>
<tr><td></td><td>Time since last check :
Indicates the time (in minutes) that has elapsed since this process last checked the connection to the eG manager or agent (as the case may be)</td><td>Mins</td><td>Typically, this should be the same as or close to the frequency configured for the check in the eG manager or agent's (as the case may be) configuration files. If not, it could indicate that the processes are not functioning as per the configure schedule, and could be a cause for concern.</td></tr>
<tr><td></td><td>Has process restarted ?
Indicates whether/not this process has restarted.</td><td></td><td>The values that this measure can report and their corresponding numeric values have been listed in the table below:<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>0</td></tr></table>Note:
By default, the test reports the Measure Values listed in the table above to indicate whether a function has been set to run as a separate process or not. In the graph of this measure however, the same is represented using the numeric equivalents only.</td></tr>
</table>

1.3.4 eG Agents Test

To receive precise, real-time updates on the performance and overall health of the components being monitored, administrators need to ensure that the eG agents, which perform the actual monitoring functions, are installed in the environment and are running at all times. This test periodically monitors the status of agents to report whether they are installed or not, and if installed, whether/not they are running.

Purpose	Periodically monitors the status of agents to report whether they are installed or not, and if installed, whether/not they are running
Target of the test	The eG Manager
Agent	An internal/remote agent

<table>
<tr><td>deploying the test</td><td colspan="3"></td></tr>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST - The host for which the test is to be configured
3. PORT - The port number at which the specified HOST listens
4. JMX REMOTE PORT – Here, specify the port at which the JMX listens for requests from remote hosts. In the <EG_MANAGER_INSTALL_DIR>\manager directory (on Windows; on Unix, this will be the /opt/egurkha/manager directory) of the eG manager, you will find a management.properties file. Set the port defined against the com.sun.management.jmxremote.port parameter of the file as the JMX REMOTE PORT.
5. USER, PASSWORD, and CONFIRM PASSWORD – By default, JMX requires no authentication or security. Therefore, the user, PASSWORD , and CONFIRM PASSWORD parameters are set to none by default.
6. JNDINAME – The JNDINAME is a lookup name for connecting to the JMX connector. By default, this is jmxrmi. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
7. JMX PROVIDER – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to com.sun.jmx.remote.protocol.
8. TIMEOUT – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to 240 seconds.
9. DD FREQUENCY - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is 1:1. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying none against DD FREQUENCY.
10. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the eG manager being monitored</td></tr>
<tr><td>Measurements made by the</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

test	**Total agents:** Indicates the total number of agents configured in the eG manager.	Number	
	Not running agents: Indicates the number of agents that are not running currently.	Number	An agent that is not running will not be able to collect metrics from the components assigned to it for monitoring. Without metrics, the eG manager will have no basis for determining the state of the related components, and will hence be forced to attach an **Unknown** state to all components monitored by that agent. Use the detailed diagnosis of this measure to identify the agents that are not currently running.
	Not installed agents: Indicates the number of agents that have not been installed.	Number	While actively using the eG Enterprise Suite, you may come across situations where an agent is configured in the eG manager for monitoring one/more components, but the agent software is not physically deployed on a host. The value of this measure is indicative of how many such configured but not installed agents are available in the environment. Such agents again will not be able to extract performance statistics from components. These components will hence assume the **Unknown** state, until the agent is physically deployed and started. Use the detailed diagnosis of this measure to know which agents are yet to be installed.

The detailed diagnosis of the *Not running agents* measure lists the IP address/nick name of the eG agents that are not running currently.

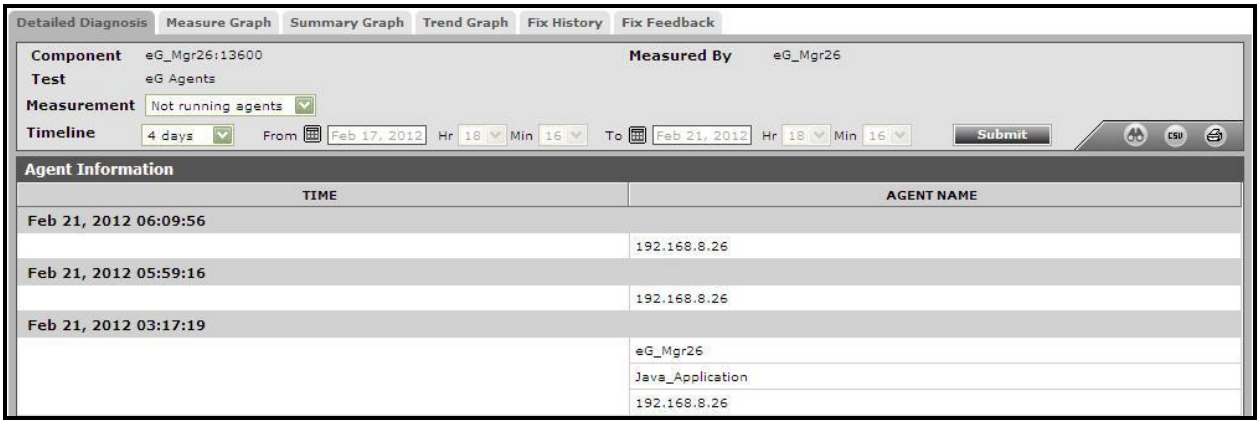


Figure 23: The detailed diagnosis of the Not running agents

The detailed diagnosis of the *Not installed agents* measure lists the IP address/nick name of the eG agents that are not installed currently.

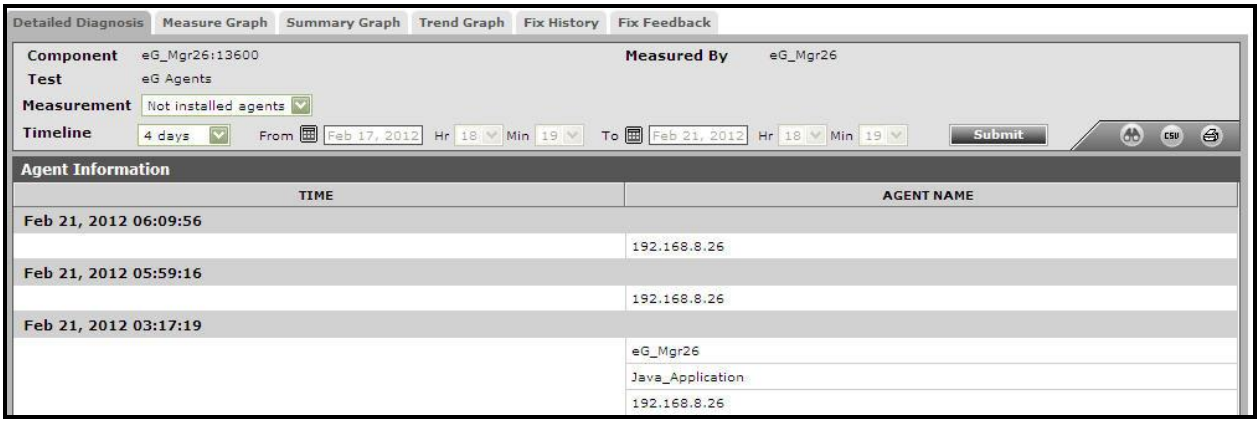


Figure 24: The detailed diagnosis of the Not installed agents

1.3.5 eG Manager Error Log Test

The eG manager, upon installation, automatically creates an **error_log** in the **<EG_INSTALL_DIR>\manager\logs** directory to which errors/warnings captured during installation, configuration and control (starting/stopping) of the eG manager, and during critical manager operations such as trending, thresholding, cleanup, and alerting are recorded. By periodically parsing this **error_log** and retrieving the details of error/warning events logged therein, you can understand which activities failed and why. This test helps you do just that.

Purpose	Parses the **error_log** file and reports the details of error/warning events logged therein, so as to enable administrators to efficiently troubleshoot issures related to the eG manager activities
Target of the test	The eG manager
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured. 3. **PORT** – The port at which the server listens 4. **ALERTFILE** - Specify the full path to the log file to be monitored. Since the **error_log** is by default found in the **<EG_INSTALL_DIR>\manager\logs** directory, the same is displayed here by default. Also, instead of a specific log file path, the path to the directory containing the **error_log** files can be provided - eg., */opt/egurkha/manager/logs*. This ensures that eG Enterprise monitors the most recent log files in the specified directory. Specific log file name patterns can also be specified. For example, to monitor the latest log files with names containing the string *error_log*, the parameter specification can be, */opt/egurkha/manager/*error_log**. Here, '*' indicates leading/trailing characters (as the case may be). In this case, the eG agent first enumerates all the log files in the specified path that match the given pattern, and then picks only the latest log file from the result set for monitoring. Every time this test is executed, the eG agent verifies the following: • Whether any changes have occurred in the size and/or timestamp of the log files that were monitoring during the last measurement period; • Whether any new log files (that match the **ALERTFILE** specification) have been newly added since the last measurement period; • If a few lines have been added to a log file that was monitored previously, then the eG agent monitors the additions to that log file, and then proceeds to monitor newer log files (if any). If an older log file has been overwritten, then, the eG agent monitors this log file completely, and then proceeds to monitor the newer log files (if any). 11. **SEARCHPATTERN** - Enter the specific patterns of alerts to be monitored. The pattern should be in the following format: *<PatternName>:<Pattern>*, where *<PatternName>* is the pattern name that will be displayed in the monitor interface and *<Pattern>* is an expression of the form - *expr* or expr or *expr or expr*, etc. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters. Multiple search patterns can be specified as a comma-separated list. If you want all the messages in a log file to be monitored, then your specification would be: *<PatternName>:**. By default, this test monitors *error* and *warning* events logged in the **error_log** file. Therefore, the default **SEARCHPATTERN** is set as: *error:*ERROR*,warn:*WARN**. This indicates that by default, the test will monitor only those lines in the **error_log** which embed the strings *ERROR* and *WARN*.

5. **LINES** - Specify two numbers in the format x:y. This means that when a line in the alert file matches a particular pattern, then x lines before the matched line and y lines after the matched line will be reported in the detail diagnosis output (in addition to the matched line). The default value here is 0:0. Multiple entries can be provided as a comma-separated list.

   If you give 1:1 as the value for **LINES**, then this value will be applied to all the patterns specified in the **SEARCHPATTERN** field. If you give 0:0,1:1,2:1 as the value for **LINES** and if the corresponding value in the **SEARCHPATTERN** filed is like ORA:ORA-*,offline:*offline*,online:*online then:

   0:0 will be applied to ORA:ORA-* pattern

   1:1 will be applied to offline:*offline* pattern

   2:1 will be applied to online:*online pattern

6. **EXCLUDEPATTERN** - Provide a comma-separated list of patterns to be excluded from monitoring in the **EXCLUDEPATTERN** text box. For example **critical*, *exception**. By default, this parameter is set to 'none'.

7. **UNIQUEMATCH** - By default, the **UNIQUEMATCH** parameter is set to **FALSE**, indicating that, by default, the test checks every line in the log file for the existence of each of the configured **SEARCHPATTERNS**. By setting this parameter to **TRUE**, you can instruct the test to ignore a line and move to the next as soon as a match for one of the configured patterns is found in that line. For example, assume that *Pattern1:*fatal*,Pattern2:*error** is the **SEARCHPATTERN** that has been configured. If **UNIQUEMATCH** is set to **FALSE**, then the test will read every line in the log file completely to check for the existence of messages embedding the strings 'fatal' and 'error'. If both the patterns are detected in the same line, then the number of matches will be incremented by 2. On the other hand, if **UNIQUEMATCH** is set to **TRUE**, then the test will read a line only until a match for one of the configured patterns is found and not both. This means that even if the strings 'fatal' and 'error' follow one another in the same line, the test will consider only the first match and not the next. The match count in this case will therefore be incremented by only 1.

8. **ROTATINGFILE** - This flag governs the display of descriptors for this test in the eG monitoring console. This flag is set to **false** by default. In this case, the descriptors of the test will be of the following format: *<Alert_file_name>:<SearchPattern>*.

   If this flag is set to **true** and the **ALERTFILE** text box contains the full path to a specific (log/text) file, then, the descriptors of this test will be displayed in the following format: *Directory_containing_monitored_file:<SearchPattern>*. Since the **ALERTFILE** parameter is set to */manager/logs/error_log* by default in the case of this test, setting the **ROTATINGFILE** to **true** will display descriptors in the following format in the eG monitoring console: */manager/logs/error_log:<SearchPattern>*.

On the other hand, if this flag is set to **true** and the **ALERTFILE** parameter is set to the directory containing log files (say, */manager/logs*), then, the descriptors of this test will be displayed in the format: *Configured_directory_path:<SearchPattern>* - i.e., */manager/logs:<SearchPattern>*. On the other hand, if the **ROTATINGFILE** parameter had been set to **false**, then the descriptors will be of the following format: *Configured_directory:<SearchPattern>* - i.e., *logs:<SearchPattern>* in the case of the example above.

If this flag is set to true and the **ALERTFILE** parameter is set to a specific file pattern, then, the descriptors of this test will be of the following format: *<FilePattern>:<SearchPattern>*. */opt/egurkha/manager/*error_log**, and **ROTATINGFILE** is set to **true**, then, your descriptor will be: **error_log*:<SearchPattern>*. In this case, the descriptor format will not change even if the **ROTATINGFILE** flag status is changed .

12. **CASESENSITIVE** - This flag is set to **No** by default. This indicates that the test functions in a 'case-insensitive' manner by default. This implies that, by default, the test ignores the case of your **ALERTFILE** and **SEARCHPATTERN** specifications. If this flag is set to **Yes** on the other hand, then the test will function in a 'case-sensitive' manner. In this case therefore, for the test to work, even the case of your **ALERTFILE** and **SEARCHPATTERN** specifications should match with the actual.

13. **OVERWRITTENFILE** - By default, this flag is set to **false**. Set this flag to **true** if you want the test to support the 'roll over' capability of the specified **ALERTFILE**. A roll over typically occurs when the timestamp of a file changes or when the log file size crosses a pre-determined threshold. When a log file rolls over, the errors/warnings that pre-exist in that file will be automatically copied to a new file, and all errors/warnings that are captured subsequently will be logged in the original/old file. For instance, say, errors and warnings were originally logged to a file named *error_log*. When a roll over occurs, the content of the file *error_log* will be copied to a file named *error_log.1*, and all new errors/warnings will be logged in *error_log*. In such a scenario, since the **OVERWRITTENFILE** flag is set to **false** by default, the test by default scans only *error_log.1* for new log entries and ignores *error_log*. On the other hand, if the flag is set to **true**, then the test will scan both *error_log* and *error_log.1* for new entries.

    If you want this test to support the 'roll over' capability described above, the following conditions need to be fulfilled:

    - The **ALERTFILE** parameter has to be configured only with the name and/or path of one/more alert files. File patterns or directory specifications should not be specified in the **ALERTFILE** text box.
    - The roll over file name should be of the format: "<**ALERTFILE**>.1", and this file must be in the same directory as the **ALERTFILE**.

14. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against **DD FREQUENCY**.

	9. **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	Measurement Unit	Interpretation
	Recent messages: Indicates the number of messages matching the specified **SEARCHPATTERN** that were recently logged in the specified **ALERTFILE**.	Number	The detailed diagnosis of this measure sheds light on the nature of errors/warnings that occurred when installing, configuring, or starting the eG manager.

The detailed diagnosis of the *Recent messages* measure provides detailed descriptions related to the error/warning events that occurred on the eG manager when installing/configuring/starting it. This way, you can easily troubleshoot issues.

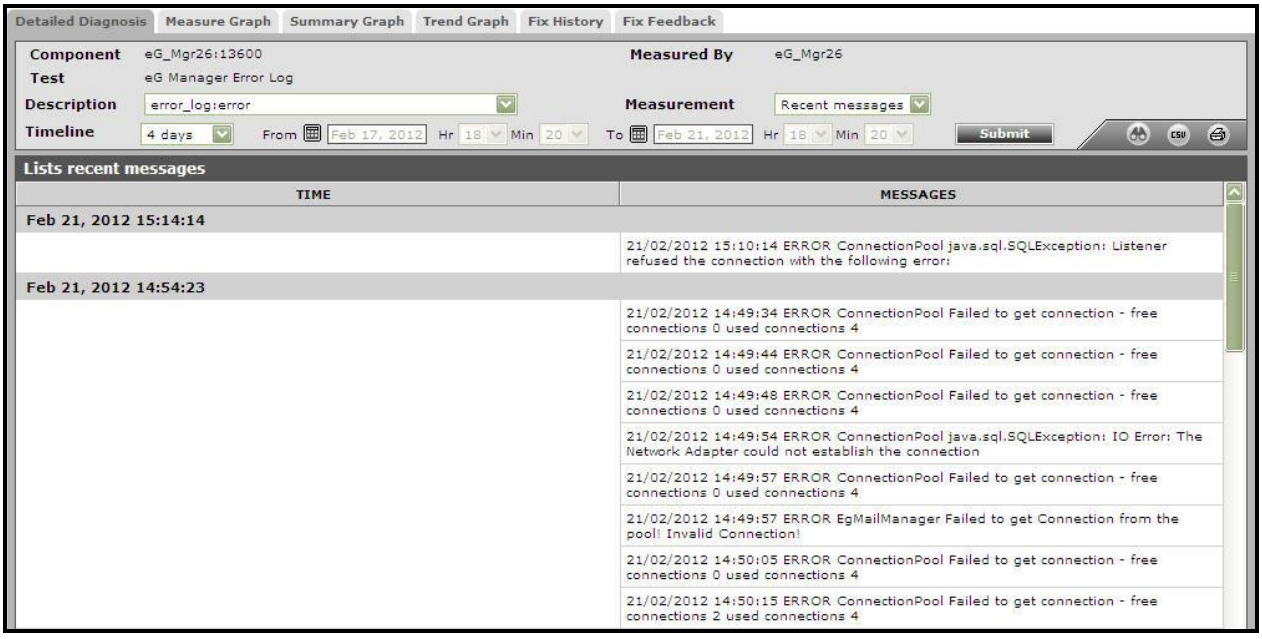


Figure 25: The detailed diagnosis of the Recent messages measure

1.3.6 eG Object Pools Test

The eG manager uses four object pools to manage database connections of each of the modules it supports. These object pools are as follows - **DB Adapter Admin**, **DB Adapter Monitor**, **DB Adapter Config**, and **DB Adapter Reporter**. If any of these pools run out of objects, then the eG manager will not be able to establish database connections and will eventually crash! To make sure that all the object pools are adequately sized, use this test and track the usage of each object pool, isolate the pools that do not have sufficient objects, and resize them.

Purpose	Tracks the usage of each object pool and helps isolate the pools that do not have sufficient objects, so that you can resize them		
Target of the test	The eG Manager		
Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST -** The host for which the test is to be configured 3. **PORT -** The port number at which the specified **HOST** listens 4. **JMX REMOTE PORT** – Here, specify the port at which the **JMX** listens for requests from remote hosts. In the **<EG_MANAGER_INSTALL_DIR>\manager** directory (on Windows; on Unix, this will be the **/opt/egurkha/manager** directory) of the eG manager, you will find a **management.properties** file. Set the port defined against the **com.sun.management.jmxremote.port** parameter of the file as the **JMX REMOTE PORT**. 5. **USER**, **PASSWORD**, and **CONFIRM PASSWORD** – By default, JMX requires no authentication or security. Therefore, the user, **PASSWORD** , and **CONFIRM PASSWORD** parameters are set to *none* by default. 6. **JNDINAME** – The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is *jmxrmi*. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same. 7. **JMX PROVIDER** – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to *com.sun.jmx.remote.protocol*. 8. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to *240* seconds.		
Outputs of the test	One set of results for each object pool of the eG manager		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Initial objects: Indicates the number of objects initially allocated to this pool.	Number	

	Maximum objects: Indicates the maximum number of objects this pool can contain.	Number	
	Used objects: Indicates the objects that are currently used in this pool.	Number	If the value of this measure grows dangerously close to the value of the *Maximum objects* measure, it indicates that the pool is rapidly running out of objects.
	Free objects: Indicates the number of unused objects in this pool.	Number	A low value for this measure is a cause for concern, as it indicates that the object pool does not have enough free objects. This could be owing to an increase in the number of components monitored or the number of users accessing the systems. You may hence want to consider resizing the pool, so that the availability of the eG manager is not affected by the lack of adequate objects.
	Object timeout: Indicates the time taken by the object to timeout.	Secs	By carefully observing the value for this measure over time, you can figure out the maximum ideal time for which an object should stay alive.
	Total objects: Indicates the total number of objects in this pool.	Number	

1.3.7 eG Servlets Test

Java servlets are the building blocks of the eG Enterprise system. These servlets are used by the eG manager to receive and respond to Web clients across HTTP. The poor responsiveness of one/more servlets can hence significantly impact the performance of the eG manager. Therefore, if administrators notice a slowdown with the eG manager, they can use this test to know if any servlet has contributed to the delay and if so, which one.

Purpose	Monitors the requests received by each servlet and measures its responsiveness
Target of the test	The eG Manager
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST - The host for which the test is to be configured
3. PORT - The port number at which the specified HOST listens
4. JMX REMOTE PORT – Here, specify the port at which the JMX listens for requests from remote hosts. In the <EG_MANAGER_INSTALL_DIR>\manager directory (on Windows; on Unix, this will be the /opt/egurkha/manager directory) of the eG manager, you will find a management.properties file. Set the port defined against the com.sun.management.jmxremote.port parameter of the file as the JMX REMOTE PORT.
5. USER, PASSWORD, and CONFIRM PASSWORD – By default, JMX requires no authentication or security. Therefore, the user, PASSWORD , and CONFIRM PASSWORD parameters are set to none by default.
6. JNDINAME – The JNDINAME is a lookup name for connecting to the JMX connector. By default, this is jmxrmi. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
7. JMX PROVIDER – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to com.sun.jmx.remote.protocol.
8. TIMEOUT – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to 240 seconds.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for each servlet on the eG manager</td></tr>
<tr><td rowspan="4">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Requests:
Indicates the number of requests handled by this servlet.</td><td>Number</td><td>This is a good indicator of the load on the servlet.</td></tr>
<tr><td>Average response time:
Indicates the time taken by this servlet to respond to requests.</td><td>Secs</td><td>A high value is indicative of the poor responsiveness of a servlet. Compare the value of this measure across servlets to identify the least responsive servlet.</td></tr>
<tr><td>Insert queries:
Indicates the number of insert queries that were executed by this servlet during the last measurement period.</td><td>Number</td><td></td></tr>
<tr><td></td><td>Average insert time:
Indicates the time taken by this servlet to execute these insert queries during the last measurement period.</td><td>Secs</td><td>A high value could be cause for concern.</td></tr>
</table>

1.3.8 State manager process

The **State manager process** is responsible for computing the current state of layers, tests, measures, and infrastructure elements (such as components, services, segments, zones, etc.), based on the metrics reported by the eG agent. Also, if agents are not able to run tests or report metrics, it is the responsibility of the **State manager process** to change the state of the 'entity' to **Unknown** and send Unknown mail alerts, if configured. This test indicates whether/not the state manager process is overloaded by reporting how busy it is and its current queue size.

Purpose	Indicates whether/not the state manager process is overloaded by reporting how busy it is and its current queue size		
Target of the test	The eG Manager		
Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	9. **TEST PERIOD** - How often should the test be executed 10. **HOST -** The host for which the test is to be configured 11. **PORT -** The port number at which the specified **HOST** listens 12. **JMX REMOTE PORT** – Here, specify the port at which the **JMX** listens for requests from remote hosts. In the **<EG_MANAGER_INSTALL_DIR>\manager** directory (on Windows; on Unix, this will be the **/opt/egurkha/manager** directory) of the eG manager, you will find a **management.properties** file. Set the port defined against the **com.sun.management.jmxremote.port** parameter of the file as the **JMX REMOTE PORT**. 13. **USER**, **PASSWORD**, and **CONFIRM PASSWORD** – By default, JMX requires no authentication or security. Therefore, the user, **PASSWORD** , and **CONFIRM PASSWORD** parameters are set to *none* by default. 14. **JNDINAME** – The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is *jmxrmi*. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same. 15. **JMX PROVIDER** – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to *com.sun.jmx.remote.protocol*. 16. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to *240* seconds.		
Outputs of the test	One set of results for the eG manager		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Statemanager queue: Indicates the current queue size of the state manager process.	Number	A high value indicates that too many requests to the state manager are pending processing. A gradual, but consistent increase in the value of this measure could indicate a processing bottleneck.

	Statemanager busy percent: Indicates how busy the state manager is currently.	Percent	A value close to 100% indicates that the state manager is too busy now, and may not have the bandwidth to attend to subsequent state requests. If the value of this measure remains high or consistently climbs up, it indicates that the state manager process is being overloaded.

1.3.9 eG Threshold Log

You can closely track the status of the threshold computation process by enabling logging for that process using the **MANAGER SETTINGS** page (Configure -> Manager Settings menu sequence) in the eG administrative interface. This will result in the creation of the **thresh_log** file (in the **<EG_INSTALL_DIR>\manager\logs** directory), to which the status and errors pertaining to the threshold-related activities will be logged as and when they occur.

This test monitors the **thresh_log** for errors and provides the complete details of these errors to simplify troubleshooting.

Purpose	Monitors the **thresh_log** for errors and provides the complete details of these errors to simplify troubleshooting
Target of the test	The eG manager
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured. 3. **PORT** – The port at which the server listens 4. **ALERTFILE** - Specify the full path to the log file to be monitored. Since the **thresh_log** is by default found in the **<EG_INSTALL_DIR>\manager\logs** directory, the same is displayed here by default. Also, instead of a specific log file path, the path to the directory containing the **thresh_log** files can be provided - eg., */opt/egurkha/manager/logs*. This ensures that eG Enterprise monitors the most recent log files in the specified directory. Specific log file name patterns can also be specified. For example, to monitor the latest log files with names containing the string *thresh_log*, the parameter specification can be, */opt/egurkha/manager/*thresh_log**. Here, '*' indicates leading/trailing characters (as the case may be). In this case, the eG agent first enumerates all the log files in the specified path that match the given pattern, and then picks only the latest log file from the result set for monitoring. Every time this test is executed, the eG agent verifies the following: • Whether any changes have occurred in the size and/or timestamp of the log files that were monitoring during the last measurement period; • Whether any new log files (that match the **ALERTFILE** specification) have been newly added since the last measurement period; • If a few lines have been added to a log file that was monitored previously, then the eG agent monitors the additions to that log file, and then proceeds to monitor newer log files (if any). If an older log file has been overwritten, then, the eG agent monitors this log file completely, and then proceeds to monitor the newer log files (if any). 5. **SEARCHPATTERN** - Enter the specific patterns of alerts to be monitored. The pattern should be in the following format: *<PatternName>:<Pattern>*, where *<PatternName>* is the pattern name that will be displayed in the monitor interface and *<Pattern>* is an expression of the form - *expr* or expr or *expr or expr*, etc. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters. Multiple search patterns can be specified as a comma-separated list. If you want all the messages in a log file to be monitored, then your specification would be: *<PatternName>:*. By default, this test monitors *error* events logged in the **thresh_log** file. Therefore, the default **SEARCHPATTERN** is set as: *error:*ERROR**. This indicates that by default, the test will monitor only those lines in the **thresh_log** which embed the string *ERROR*.

6. **LINES** - Specify two numbers in the format x:y. This means that when a line in the alert file matches a particular pattern, then x lines before the matched line and y lines after the matched line will be reported in the detail diagnosis output (in addition to the matched line). The default value here is 0:0. Multiple entries can be provided as a comma-separated list.

   If you give 1:1 as the value for **LINES**, then this value will be applied to all the patterns specified in the **SEARCHPATTERN** field. If you give 0:0,1:1,2:1 as the value for **LINES** and if the corresponding value in the **SEARCHPATTERN** field is like ORA:ORA-*,offline:*offline*,online:*online then:

   0:0 will be applied to ORA:ORA-* pattern

   1:1 will be applied to offline:*offline* pattern

   2:1 will be applied to online:*online pattern

7. **EXCLUDEPATTERN** - Provide a comma-separated list of patterns to be excluded from monitoring in the **EXCLUDEPATTERN** text box. For example **critical*, *exception**. By default, this parameter is set to 'none'.

8. **UNIQUEMATCH** - By default, the **UNIQUEMATCH** parameter is set to **FALSE**, indicating that, by default, the test checks every line in the log file for the existence of each of the configured **SEARCHPATTERNS**. By setting this parameter to **TRUE**, you can instruct the test to ignore a line and move to the next as soon as a match for one of the configured patterns is found in that line. For example, assume that *Pattern1:*fatal*,Pattern2:*error** is the **SEARCHPATTERN** that has been configured. If **UNIQUEMATCH** is set to **FALSE**, then the test will read every line in the log file completely to check for the existence of messages embedding the strings 'fatal' and 'error'. If both the patterns are detected in the same line, then the number of matches will be incremented by 2. On the other hand, if **UNIQUEMATCH** is set to **TRUE**, then the test will read a line only until a match for one of the configured patterns is found and not both. This means that even if the strings 'fatal' and 'error' follow one another in the same line, the test will consider only the first match and not the next. The match count in this case will therefore be incremented by only 1.

9. **ROTATINGFILE** - This flag governs the display of descriptors for this test in the eG monitoring console. This flag is set to **false** by default. In this case, the descriptors of the test will be of the following format: *<Alert_file_name>:<SearchPattern>*.

   If this flag is set to **true** and the **ALERTFILE** text box contains the full path to a specific (log/text) file, then, the descriptors of this test will be displayed in the following format: *Directory_containing_monitored_file:<SearchPattern>*. Since the **ALERTFILE** parameter is set to */manager/logs/thresh_log* by default in the case of this test, setting the **ROTATINGFILE** to **true** will display descriptors in the following format in the eG monitoring console: */manager/logs/thresh_log:<SearchPattern>*.

On the other hand, if this flag is set to **true** and the **ALERTFILE** parameter is set to the directory containing log files (say, */manager/logs*), then, the descriptors of this test will be displayed in the format: *Configured_directory_path:<SearchPattern>* - i.e., */manager/logs:<SearchPattern>*. On the other hand, if the **ROTATINGFILE** parameter had been set to **false**, then the descriptors will be of the following format: *Configured_directory:<SearchPattern>* - i.e., *logs:<SearchPattern>* in the case of the example above.

If this flag is set to true and the **ALERTFILE** parameter is set to a specific file pattern, then, the descriptors of this test will be of the following format: *<FilePattern>:<SearchPattern>*. For instance, if the **ALERTFILE** specification is */opt/egurkha/manager/*thresh_log** and **ROTATINGFILE** is set to **true**, then, your descriptor will be: **thresh_log*:<SearchPattern>*. In this case, the descriptor format will not change even if the **ROTATINGFILE** flag status is changed.

10. **CASESENSITIVE -** This flag is set to **No** by default. This indicates that the test functions in a 'case-insensitive' manner by default. This implies that, by default, the test ignores the case of your **ALERTFILE** and **SEARCHPATTERN** specifications. If this flag is set to **Yes** on the other hand, then the test will function in a 'case-sensitive' manner. In this case therefore, for the test to work, even the case of your alertfile and searchpattern specifications should match with the actuals.

11. **OVERWRITTENFILE** - By default, this flag is set to **false**. Set this flag to **true** if you want the test to support the 'roll over' capability of the specified **ALERTFILE**. A roll over typically occurs when the timestamp of a file changes or when the log file size crosses a pre-determined threshold. When a log file rolls over, the errors/warnings that pre-exist in that file will be automatically copied to a new file, and all errors/warnings that are captured subsequently will be logged in the original/old file. For instance, say, errors and warnings were originally logged to a file named *thresh_log*. When a roll over occurs, the content of the file *error_log* will be copied to a file named *thresh_log.1*, and all new errors/warnings will be logged in *thresh_log*. In such a scenario, since the **OVERWRITTENFILE** flag is set to **false** by default, the test by default scans only *error_log.1* for new log entries and ignores *thresh_log*. On the other hand, if the flag is set to **true**, then the test will scan both *thresh_log* and *thresh_log.1* for new entries.

    If you want this test to support the 'roll over' capability described above, the following conditions need to be fulfilled:

    - The **ALERTFILE** parameter has to be configured only with the name and/or path of one/more alert files. File patterns or directory specifications should not be specified in the **ALERTFILE** text box.
    - The roll over file name should be of the format: "<**ALERTFILE**>.1", and this file must be in the same directory as the **ALERTFILE**.

12. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against **DD FREQUENCY**.

	13. **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**	**Measurement Unit**	**Interpretation**
	Recent messages: Indicates the number of messages matching the specified **SEARCHPATTERN** that were recently logged in the specified **ALERTFILE**.	Number	The detailed diagnosis of this measure sheds light on the nature of errors that occurred when thresholds were computed.

The detailed diagnosis of the *Recent messages* measure provides detailed descriptions related to the error events that occurred on the eG manager when threshols were computed. This way, you can easily troubleshoot thresholding-related issues.

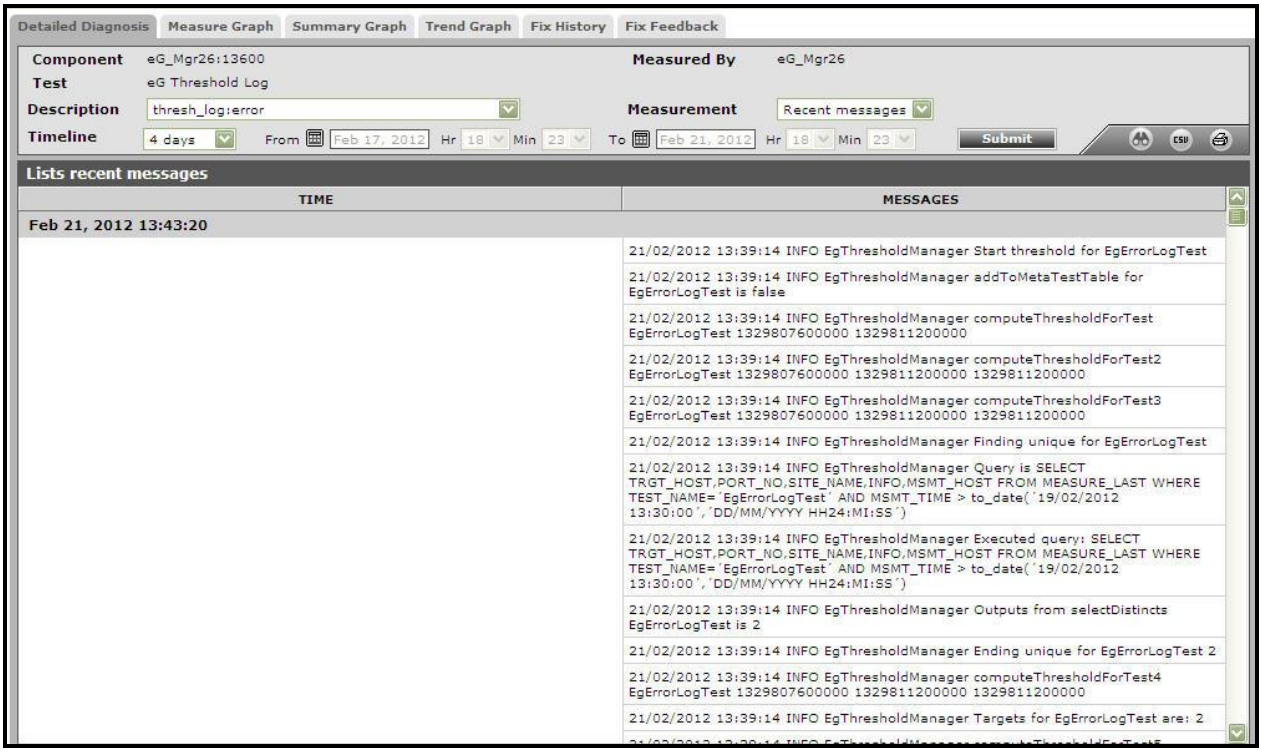


Figure 26: The detailed diagnosis of the Recent messages measure reported by the eG Threshold Log Test

1.3.10 Threshold manager process Test

The **Threshold manager process** computes the thresholds - i.e., the upper/limits of performance - for every measure collected by the eG agent and stores the thresholds so computed in the eG database. Since thresholds govern the state of a measure, if the threshold manager process fails to compute thresholds or computes them slowly or fails to even run, it can grossly impair the eG monitoring solution's ability to promptly detect problem areas. Using this test, you can understand how efficient the threshold manager process is. The test reports the current status of this process, points you to threshold computation failures and where they occurred, and reveals slowdowns in threshold computation (if any).

Purpose	Reports the current status of the threshold manager process, points you to threshold computation failures and where they occurred, and reveals slowdowns in threshold computation (if any)
Target of the test	The eG Manager
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST -** The host for which the test is to be configured 3. **PORT -** The port number at which the specified **HOST** listens 4. **JMX REMOTE PORT** – Here, specify the port at which the **JMX** listens for requests from remote hosts. In the **<EG_MANAGER_INSTALL_DIR>\manager** directory (on Windows; on Unix, this will be the **/opt/egurkha/manager** directory) of the eG manager, you will find a **management.properties** file. Set the port defined against the **com.sun.management.jmxremote.port** parameter of the file as the **JMX REMOTE PORT**. 5. **USER**, **PASSWORD**, and **CONFIRM PASSWORD** – By default, JMX requires no authentication or security. Therefore, the user, **PASSWORD** , and **CONFIRM PASSWORD** parameters are set to *none* by default. 6. **JNDINAME** – The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is *jmxrmi*. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same. 7. **JMX PROVIDER** – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to *com.sun.jmx.remote.protocol*. 8. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to *240* seconds. 9. **THRESHOLD DURATION OF TEST** - This test reports a *Successful threshold tests* measure, which indicates the number of tests for which the threshold manager successfully computed thresholds. The detailed diagnosis of this measure, if enabled, will by default list only the top-10 successful threshold tests, arranged in the descending order of the time taken by the threshold manager to compute thresholds on them. To arrive at this top-10 list, the test considers only those successful tests for which the threshold manager took more than 1 minute (by default) for threshold computation. This is why, the **THRESHOLD DURATION OF TEST** parameter is set to **1** (minute) by default. This default setting can be overridden by specifying a duration (in minutes) of your choice in the **THRESHOLD DURATION OF TEST** text box. For instance, if you specify *5* here, then, the detailed diagnosis will list the top-10 (by default) successful threshold tests for which the threshold manager took more than **5** minutes for threshold computation. 10. **TOP TIME TAKEN TEST** - As already mentioned, the detailed diagnosis of the *Successful threshold tests* measure, by default, lists the top-10 successful threshold tests, arranged in the descending order of the time taken by the threshold manager to compute thresholds on them. This is why, the **TOP TIME TAKEN TEST** is set to **10** by default. To view more or a less number of successful threshold tests in the detailed diagnosis, specify a different value in the **TOP TIME TAKEN TEST** text box. For instance, if **20** is specified here, then the detailed diagnosis will list the **top-20** successful threshold tests.

<table>
<tr><td></td><td colspan="3">11. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the eG manager</td></tr>
<tr><td rowspan="2">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Thresholding status:

Indicates the current status of the threshold manager process.</td><td></td><td>The values that this measure reports and the numeric values that correspond to them have been discussed in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Done</td><td>1</td></tr><tr><td>Running</td><td>0</td></tr><tr><td>Error</td><td>2</td></tr></table>
Note:

By default, this measure reports the Measure Values listed in the table above to indicate the current status of the threshold manager process. The graph of this measure however, represents the same using the numeric equivalents only.</td></tr>
<tr><td></td><td>Time taken for thresholding:

Indicates the total time taken by the threshold manager to compute thresholds.</td><td>Minutes</td><td>Ideally, the value of this measure should be low. A steady rise in this measure value is a cause for concern, as it indicates that the threshold manager is taking too long to compute thresholds. This can happen if the threshold manager needs to compute thresholds for too many tests, measures, and descriptors.</td></tr>
</table>

	Successful threshold tests: Indicates the number of tests for which thresholds were computed successfully.	Number	You can use the detailed diagnosis of this measure to know the tests for which thresholds have been successfully computed.
	Failed threshold tests: Indicates the number of tests for which threshold computation failed.	Number	The value 0 is desired for this measure. Any non-zero value is indicative of a thresholding failure. In this case, you can use the detailed diagnosis of this measure to identify those tests for which threshold computation failed and investigate the reason why. Without thresholds, the monitoring solution cannot detect problem conditions; nor can it compute state.
	Time since last completion: Indicates the elapsed time since the last threshold computation.	Minutes	Typically, thresholding is scheduled to take place at the end of every day. By carefully observing the values reported by this measure, you can easily find out when a scheduled threshold computation cycle was missed.

	Is threshold running as a separate process? Indicates whether/not the threshold manager is running as a separate process.		The eG manager is a 32-bit application that runs as a Java process. The maximum heap memory that can be allocated to this process is limited to 1.5 GB. Even if the physical server on which the eG manager is installed has more memory, since it is a single Java process, the eG manager cannot exploit the additional memory available on the server. To overcome this limitation, in eG Enterprise, the critical eG manager functions such as email alert management, threshold computation, trending, and database cleanup activities can all be run as separate Java processes (i.e., in addition to the core eG manager process). Removing these key functions from the core eG manager process makes additional memory available for the core eG manager functions including data reception and analysis, alarm correlation, and web-based access and reporting. This reconfiguration of the eG manager into separate Java processes allows the eG manager to make better utilization of available server hardware resources and thereby offers enhanced scalability. In turn, this allows customers to get more leverage from their existing investment in the hardware that hosts the eG manager. If cleanup has been configured to run as a separate Java process, then the value of this measure will be *Yes*. If not, then this measure reports the value *No*. The numeric values that correspond to the measure values above are as follows: **Measure Value** / **Numeric Value** Yes / 1 No / 0

			Note: By default, this measure reports the **Measure Value**s listed in the table above to indicate whether/not cleanup runs as a separate Java process. The graph of this measure however, represents the same using the numeric equivalents only.
	Slow threshold tests: Indicates the number of tests for which threshold computation was slow.	Number	Use the detailed diagnosis of this measure to know for which tests threshold computation was slow.

The detailed diagnosis of the *Failed threshold tests* measure lists all the tests on which threshold computation failed and briefly describes the reason for the failure.

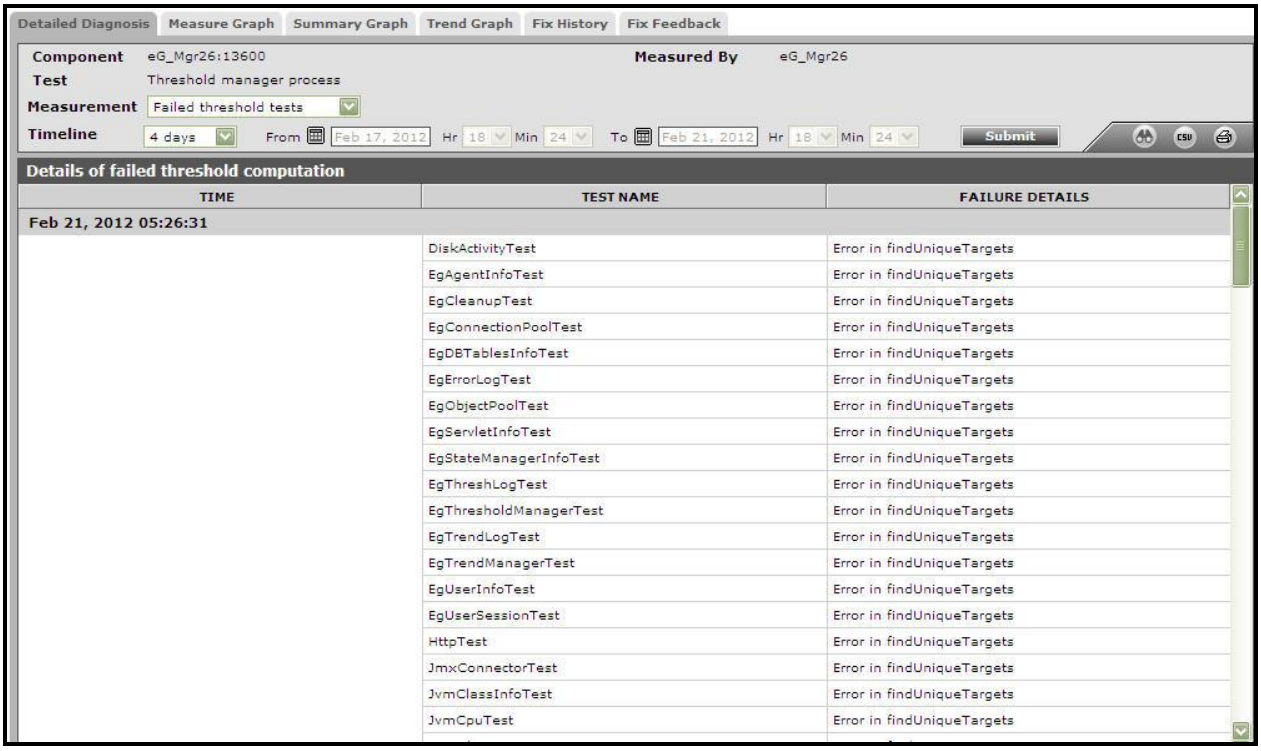


Figure 27: The detailed diagnosis of the Failed threshold tests measure

1.3.11 eG Trend Log Test

You can closely track the status of the trend computation process by enabling logging for that process using the **MANAGER SETTINGS** page (Configure -> Manager Settings menu sequence) in the eG administrative interface. This

will result in the creation of the **trend_log** file (in the **<EG_INSTALL_DIR>\manager\logs** directory), to which the status and errors pertaining to the trend-related activities will be logged as and when they occur.

This test monitors the **trend_log** for errors and provides the complete details of these errors to simplify troubleshooting.

Purpose	Monitors the **trend_log** for errors and provides the complete details of these errors to simplify troubleshooting
Target of the test	The eG manager
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST** - The host for which the test is to be configured. 3. **PORT** – The port at which the server listens 4. **ALERTFILE** - Specify the full path to the log file to be monitored. Since the **trend_log** is by default found in the **<EG_INSTALL_DIR>\manager\logs** directory, the same is displayed here by default. Also, instead of a specific log file path, the path to the directory containing the **trend_log** files can be provided - eg., */opt/egurkha/manager/logs*. This ensures that eG Enterprise monitors the most recent log files in the specified directory. Specific log file name patterns can also be specified. For example, to monitor the latest log files with names containing the string *trend_log*, the parameter specification can be, */opt/egurkha/manager/*trend_log**. Here, '*' indicates leading/trailing characters (as the case may be). In this case, the eG agent first enumerates all the log files in the specified path that match the given pattern, and then picks only the latest log file from the result set for monitoring. Every time this test is executed, the eG agent verifies the following: • Whether any changes have occurred in the size and/or timestamp of the log files that were monitoring during the last measurement period; • Whether any new log files (that match the **ALERTFILE** specification) have been newly added since the last measurement period; • If a few lines have been added to a log file that was monitored previously, then the eG agent monitors the additions to that log file, and then proceeds to monitor newer log files (if any). If an older log file has been overwritten, then, the eG agent monitors this log file completely, and then proceeds to monitor the newer log files (if any). 5. **SEARCHPATTERN** - Enter the specific patterns of alerts to be monitored. The pattern should be in the following format: *<PatternName>:<Pattern>*, where *<PatternName>* is the pattern name that will be displayed in the monitor interface and *<Pattern>* is an expression of the form - *expr* or expr or *expr or expr*, etc. A leading '*' signifies any number of leading characters, while a trailing '*' signifies any number of trailing characters. Multiple search patterns can be specified as a comma-separated list. If you want all the messages in a log file to be monitored, then your specification would be: *<PatternName>:**. By default, this test monitors *error* events logged in the **thresh_log** file. Therefore, the default **SEARCHPATTERN** is set as: *error:*ERROR**. This indicates that by default, the test will monitor only those lines in the **thresh_log** which embed the string *ERROR*.

6. **LINES** - Specify two numbers in the format x:y. This means that when a line in the alert file matches a particular pattern, then x lines before the matched line and y lines after the matched line will be reported in the detail diagnosis output (in addition to the matched line). The default value here is 0:0. Multiple entries can be provided as a comma-separated list.

   If you give 1:1 as the value for **LINES**, then this value will be applied to all the patterns specified in the **SEARCHPATTERN** field. If you give 0:0,1:1,2:1 as the value for **LINES** and if the corresponding value in the **SEARCHPATTERN** field is like ORA:ORA-*,offline:*offline*,online:*online then:

   0:0 will be applied to ORA:ORA-* pattern

   1:1 will be applied to offline:*offline* pattern

   2:1 will be applied to online:*online pattern

7. **EXCLUDEPATTERN** - Provide a comma-separated list of patterns to be excluded from monitoring in the **EXCLUDEPATTERN** text box. For example **critical*, *exception**. By default, this parameter is set to 'none'.

8. **UNIQUEMATCH** - By default, the **UNIQUEMATCH** parameter is set to **FALSE**, indicating that, by default, the test checks every line in the log file for the existence of each of the configured **SEARCHPATTERNS**. By setting this parameter to **TRUE**, you can instruct the test to ignore a line and move to the next as soon as a match for one of the configured patterns is found in that line. For example, assume that *Pattern1:*fatal*,Pattern2:*error** is the **SEARCHPATTERN** that has been configured. If **UNIQUEMATCH** is set to **FALSE**, then the test will read every line in the log file completely to check for the existence of messages embedding the strings 'fatal' and 'error'. If both the patterns are detected in the same line, then the number of matches will be incremented by 2. On the other hand, if **UNIQUEMATCH** is set to **TRUE**, then the test will read a line only until a match for one of the configured patterns is found and not both. This means that even if the strings 'fatal' and 'error' follow one another in the same line, the test will consider only the first match and not the next. The match count in this case will therefore be incremented by only 1.

9. **ROTATINGFILE** - This flag governs the display of descriptors for this test in the eG monitoring console. This flag is set to **false** by default. In this case, the descriptors of the test will be of the following format: *<Alert_file_name>:<SearchPattern>*.

   If this flag is set to **true** and the **ALERTFILE** text box contains the full path to a specific (log/text) file, then, the descriptors of this test will be displayed in the following format: *Directory_containing_monitored_file:<SearchPattern>*. Since the **ALERTFILE** parameter is set to */manager/logs/trend_log* by default in the case of this test, setting the **ROTATINGFILE** to **true** will display descriptors in the following format in the eG monitoring console: */manager/logs/trend_log:<SearchPattern>*.

On the other hand, if this flag is set to **true** and the **ALERTFILE** parameter is set to the directory containing log files (say, */manager/logs*), then, the descriptors of this test will be displayed in the format: *Configured_directory_path:<SearchPattern>* - i.e., */manager/logs:<SearchPattern>*. On the other hand, if the **ROTATINGFILE** parameter had been set to **false**, then the descriptors will be of the following format: *Configured_directory:<SearchPattern>* - i.e., *logs:<SearchPattern>* in the case of the example above.

If this flag is set to true and the **ALERTFILE** parameter is set to a specific file pattern, then, the descriptors of this test will be of the following format: *<FilePattern>:<SearchPattern>*. For instance, if the **ALERTFILE** specification is */opt/egurkha/manager/*trend_log** and **ROTATINGFILE** is set to **true**, then, your descriptor will be: **trend_log*:<SearchPattern>*. In this case, the descriptor format will not change even if the **ROTATINGFILE** flag status is changed.

10. **CASESENSITIVE -** This flag is set to **No** by default. This indicates that the test functions in a 'case-insensitive' manner by default. This implies that, by default, the test ignores the case of your **ALERTFILE** and **SEARCHPATTERN** specifications. If this flag is set to **Yes** on the other hand, then the test will function in a 'case-sensitive' manner. In this case therefore, for the test to work, even the case of your alertfile and searchpattern specifications should match with the actuals.

11. **OVERWRITTENFILE** - By default, this flag is set to **false**. Set this flag to **true** if you want the test to support the 'roll over' capability of the specified **ALERTFILE**. A roll over typically occurs when the timestamp of a file changes or when the log file size crosses a pre-determined threshold. When a log file rolls over, the errors/warnings that pre-exist in that file will be automatically copied to a new file, and all errors/warnings that are captured subsequently will be logged in the original/old file. For instance, say, errors and warnings were originally logged to a file named *trend_log*. When a roll over occurs, the content of the file *error_log* will be copied to a file named *trend_log.1*, and all new errors/warnings will be logged in *thresh_log*. In such a scenario, since the **OVERWRITTENFILE** flag is set to **false** by default, the test by default scans only *error_log.1* for new log entries and ignores *trend_log*. On the other hand, if the flag is set to **true**, then the test will scan both *trend_log* and *trend_log.1* for new entries.

    If you want this test to support the 'roll over' capability described above, the following conditions need to be fulfilled:

    - The **ALERTFILE** parameter has to be configured only with the name and/or path of one/more alert files. File patterns or directory specifications should not be specified in the **ALERTFILE** text box.
    - The roll over file name should be of the format: "<**ALERTFILE**>.1", and this file must be in the same directory as the **ALERTFILE**.

12. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against **DD FREQUENCY**.

	13. **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**	**Measurement Unit**	**Interpretation**
	Recent messages: Indicates the number of messages matching the specified **SEARCHPATTERN** that were recently logged in the specified **ALERTFILE**.	Number	The detailed diagnosis of this measure sheds light on the nature of errors that occurred when trends were computed.

The detailed diagnosis of the *Recent messages* measure provides the complete details of the errors logged in the **trend_log** file.

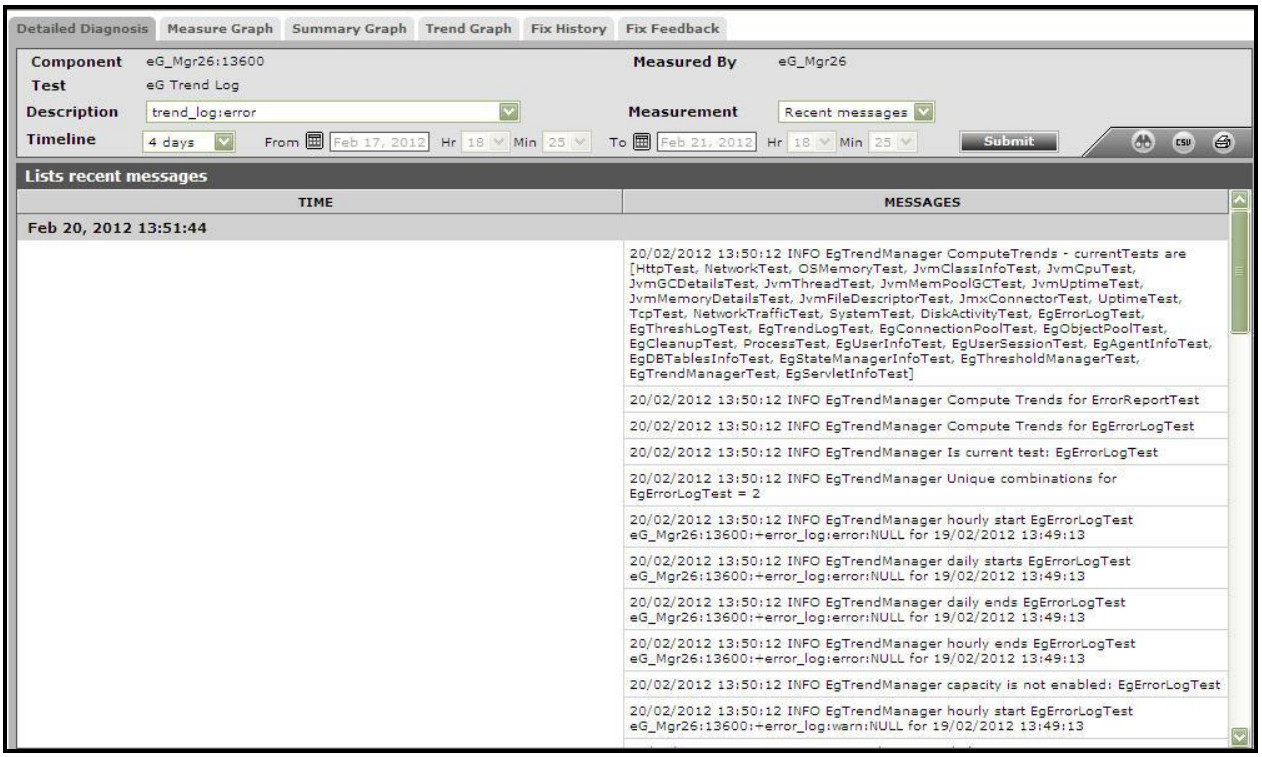


Figure 28: The detailed diagnosis of the Recent messages measure of the eG Trend Log test

1.3.12 Trend manager process Test

The **eG Trend manager process** uses the raw measurement data in the eG database to compute hourly, daily, weekly, and monthly summaries and performance averages, and stores these computations in trend tables. Users to the eG monitoring and reporting consoles can query the data in the trend tables to generate trend graphs and reports. These graphs/reports enable administrators to clearly understand the historical trends in performance, on the basis of which they can accurately predict future performance and effectively plan the future capacity of the environment. Using this test, you can understand how efficient the trend manager process is. The test reports the current status of this process, points you to trend computation failures and where they occurred, and reveals slowdowns in trend computation (if any).

Purpose	Reports the current status of this process, points you to trend computation failures and where they occurred, and reveals slowdowns in trend computation (if any)
Target of the test	The eG Manager
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST -** The host for which the test is to be configured 3. **PORT -** The port number at which the specified **HOST** listens 4. **JMX REMOTE PORT** – Here, specify the port at which the **JMX** listens for requests from remote hosts. In the **<EG_MANAGER_INSTALL_DIR>\manager** directory (on Windows; on Unix, this will be the **/opt/egurkha/manager** directory) of the eG manager, you will find a **management.properties** file. Set the port defined against the **com.sun.management.jmxremote.port** parameter of the file as the **JMX REMOTE PORT**. 5. **USER**, **PASSWORD**, and **CONFIRM PASSWORD** – By default, JMX requires no authentication or security. Therefore, the user, **PASSWORD** , and **CONFIRM PASSWORD** parameters are set to *none* by default. 6. **JNDINAME** – The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is *jmxrmi*. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same. 7. **JMX PROVIDER** – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to *com.sun.jmx.remote.protocol*. 8. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to *240* seconds. 9. **THRESHOLD DURATION OF TEST** - This test reports a *Successful trend tests* measure, which indicates the number of tests for which the trend manager successfully computed hourly, daily, weekly, and monthly trends. The detailed diagnosis of this measure, if enabled, will by default list only the top-10 successful trend tests, arranged in the descending order of the time taken by the trend manager to compute trends on them. To arrive at this top-10 list, the test considers only those successful tests for which the trend manager took more than 1 minute (by default) for trend computation. This is why, the **THRESHOLD DURATION OF TEST** parameter is set to **1** (minute) by default. This default setting can be overridden by specifying a duration (in minutes) of your choice in the **THRESHOLD DURATION OF TEST** text box. For instance, if you specify *5* here, then, the detailed diagnosis will list the top-10 (by default) successful trend tests for which the trend manager took more than **5** minutes for trend computation. 10. **TOP TIME TAKEN TEST** - As already mentioned, the detailed diagnosis of the *Successful trend tests* measure, by default, lists the top-10 successful trend tests, arranged in the descending order of the time taken by the trend manager to compute thresholds on them. This is why, the **TOP TIME TAKEN TEST** is set to **10** by default. To view more or a less number of successful trend tests in the detailed diagnosis, specify a different value in the **TOP TIME TAKEN TEST** text box. For instance, if **20** is specified here, then the detailed diagnosis will list the **top-20** successful trend tests. 11. **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.

<table>
<tr><td></td><td colspan="3">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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the eG manager</td></tr>
<tr><td rowspan="3">Measurements made by the test</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
<tr><td>Trending status:
Indicates the current status of the trend manager process.</td><td></td><td>The values that this measure reports and the numeric values that correspond to them have been discussed in the table below:<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Done</td><td>1</td></tr><tr><td>Running</td><td>0</td></tr><tr><td>Error</td><td>2</td></tr></table>Note:
By default, this measure reports the Measure Values listed in the table above to indicate the current status of the trend manager process. The graph of this measure however, represents the same using the numeric equivalents only.</td></tr>
<tr><td></td><td>Time taken for trending:
Indicates the total time taken by the trend manager process to perform trend computations.</td><td>Minutes</td><td>Ideally, the value of this measure should be low. A steady rise in this measure value is a cause for concern, as it indicates that the trend manager is taking too long to compute trends. This can happen if there are too many components, tests, and measurements for which trends need to be computed.</td></tr>
<tr><td></td><td>Successful trend tests:
Indicates the number of tests for which trends were computed successfully.</td><td>Number</td><td>You can use the detailed diagnosis of this measure to know the tests for which the trend data has been successfully computed.</td></tr>
</table>

	Failed trend tests: Indicates the number of tests for which trend computation failed.	Number	The value 0 is desired for this measure. Any non-zero value is indicative of a trending failure. In this case, you can use the detailed diagnosis of this measure to identify those tests for which trend computation failed and investigate the reason why. Without trend data, users may not be able to generate useful trend graphs and reports; in the absence of trend reports, capacity plans cannot be drawn accurately.
	Time since last completion: Indicates the elapsed time since the last trend computation.	Minutes	Typically, trending is scheduled to take place at the end of every day. By carefully observing the values reported by this measure, you can easily find out when a scheduled trend computation cycle was missed.

	Is trend running as a separate process? Indicates whether/not the trend manager is running as a separate process.		The eG manager is a 32-bit application that runs as a Java process. The maximum heap memory that can be allocated to this process is limited to 1.5 GB. Even if the physical server on which the eG manager is installed has more memory, since it is a single Java process, the eG manager cannot exploit the additional memory available on the server. To overcome this limitation, in eG Enterprise, the critical eG manager functions such as email alert management, threshold computation, trending, and database cleanup activities can all be run as separate Java processes (i.e., in addition to the core eG manager process). Removing these key functions from the core eG manager process makes additional memory available for the core eG manager functions including data reception and analysis, alarm correlation, and web-based access and reporting. This reconfiguration of the eG manager into separate Java processes allows the eG manager to make better utilization of available server hardware resources and thereby offers enhanced scalability. In turn, this allows customers to get more leverage from their existing investment in the hardware that hosts the eG manager. If cleanup has been configured to run as a separate Java process, then the value of this measure will be *Yes*. If not, then this measure reports the value *No*. The numeric values that correspond to the measure values above are as follows: **Measure Value** / **Numeric Value** Yes / 1 No / 0

<table>
<tr><td></td><td></td><td></td><td>Note:

By default, this measure reports the Measure Values listed in the table above to indicate whether/not the trend manager runs as a separate Java process. The graph of this measure however, represents the same using the numeric equivalents only.</td></tr>
<tr><td></td><td>Has trend run today:

Indicates whether/not trend has run on this day.</td><td></td><td>Typically, trending is scheduled to take place at the end of every day. If the value of this measure is Yes, it indicates that trending has run today. On the other hand, if this measure reports the value No, it indicates that trending is yet to run for that day or has not run at all.

The numeric values that correspond to the current day's trending status are as follows:

<table><tr><th>Measure Value</th><th>Numeric 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 Measure Values listed in the table above to indicate whether/not trending has run today. The graph of this measure however, represents the same using the numeric equivalents only.</td></tr>
<tr><td></td><td>Slow trend tests:

Indicates the number of tests for which trend computation was slow.</td><td>Number</td><td>Use the detailed diagnosis of this measure to know for which tests trend computation was slow.</td></tr>
</table>

1.3.13 eG Cluster Test

To ensure high availability of the eG monitoring solution, eG Enterprise offers a licensed **Redundant Manager** option. If the eG license enables this capability, then two managers can be setup to operate in an *Active-Active* or an *Active-passive* manager cluster – i.e., a secondary manager can act as an active or passive standby for the primary manager. In the event of the failure of the primary, the secondary will automatically assume the primary's role and perform all the functions of the primary – this includes receiving performance data from all eG agents, correlating the metrics, performing state computations, sending out email/SMS alerts (if configured) to users, and providing real-time performance and problem updates via the eG management console. Since this fail over occurs seamlessly, eG administrators have no way of figuring out if the eG manager being used is indeed operating in a redundant cluster,

and if so, whether it is the primary or the secondary manager of the cluster. Moreover, during the period of unavailability of the primary, the secondary stores the performance metrics it collects to a local *data* folder; when the primary comes back up, the secondary automatically replicates this data to the primary. The maximum capacity of this *data* folder is configurable. To avoid data loss, administrators should periodically check whether/not the max size setting of the *data* folder is sufficient; for this, they need to closely track the growth in size of the *data* folder. All this is possible using the **eG Cluster** test.

This test periodically checks whether the eG manager is operating in a cluster, and if so, reports what type of cluster it is. In addition, the test also reveals whether the eG manager being monitored is the primary or secondary manager in the cluster. Regardless of manager type, the test reports the number of agents that are explicitly assigned to the manager and the number of agents that are actually reporting to the manager; this way, the test points administrators to those agents that are mapped to the manager but are not actively reporting metrics and helps them initiate investigations in this regard. The test also enables administrators to track the usage of the *data* folder and figure out if the maximum amount of data that can be stored in that folder needs to be increased to avoid data loss during fail-over.

Purpose	Periodically checks whether the eG manager is operating in a cluster, and if so, reports what type of cluster it is. In addition, the test also reveals whether the eG manager being monitored is the primary or secondary manager in the cluster. Regardless of manager type, the test reports the number of agents that are explicitly assigned to the manager and the number of agents that are actually reporting to the manager; this way, the test points administrators to those agents that are mapped to the manager but are not actively reporting metrics and helps them initiate investigations in this regard. The test also enables administrators to track the usage of the *data* folder and figure out if the maximum amount of data that can be stored in that folder needs to be increased to avoid data loss during fail-over
Target of the test	The eG Manager
Agent deploying the test	An internal/remote agent

Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST -** The host for which the test is to be configured 3. **PORT -** The port number at which the specified **HOST** listens 4. **JMX REMOTE PORT** – Here, specify the port at which the **JMX** listens for requests from remote hosts. In the **<EG_MANAGER_INSTALL_DIR>\manager** directory (on Windows; on Unix, this will be the **/opt/egurkha/manager** directory) of the eG manager, you will find a **management.properties** file. Set the port defined against the **com.sun.management.jmxremote.port** parameter of the file as the **JMX REMOTE PORT**. 5. **USER**, **PASSWORD**, and **CONFIRM PASSWORD** – By default, JMX requires no authentication or security. Therefore, the user, **PASSWORD** , and **CONFIRM PASSWORD** parameters are set to *none* by default. 6. **JNDINAME** – The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is *jmxrmi*. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same. 7. **JMX PROVIDER** – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to *com.sun.jmx.remote.protocol*. 8. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to *240* seconds. 9. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against **DD FREQUENCY**. 10. **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.		
Outputs of the test	One set of results for the eG manager being monitored		
Measurements made by the	**Measurement**	**Measurement Unit**	**Interpretation**

<table>
<tr>
<td>test</td>
<td>Cluster type of the manager :

Indicates whether/not the eG manager is operating within a redundant cluster, and if it is, then the type of cluster it is.</td>
<td></td>
<td>The values that this measure reports and the numeric values that correspond to them have been discussed in the table below:

<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>Not supported</td><td>0</td></tr>
<tr><td>Active-Active</td><td>1</td></tr>
<tr><td>Active-Passive</td><td>2</td></tr>
</table>

Note:

By default, this measure reports the Measure Values listed in the table above to indicate the current cluster type of the manager. In the graph of this measure however, the same will be represented using the numeric equivalents only.</td>
</tr>
<tr>
<td></td>
<td>Is this primary manager ?

Indicates whether/not this eG manager is the primary manager in the cluster.</td>
<td></td>
<td>This measure will not be reported if the 'Cluster type of the manager' is 'Not Supported'.

The values that this measure reports and the numeric values that correspond to them have been discussed in the table below:

<table>
<tr><th>Measure Value</th><th>Numeric 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 Measure Values listed in the table above to indicate whether/not the manager is the primary manager. In the graph of this measure however, the same will be represented using the numeric equivalents only.</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>Is this manager running ?

Indicates whether/not this manager is currently running.</td>
<td></td>
<td>The values that this measure reports and the numeric values that correspond to them have been discussed in the table below:

<table><tr><th>Measure Value</th><th>Numeric 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 Measure Values listed in the table above to indicate whether/not the manager is running. In the graph of this measure however, the same will be represented using the numeric equivalents only.</td>
</tr>
<tr>
<td></td>
<td>Is other manager in the cluster running ?

Indicates whether/not the other manager in the cluster is currently running or not.</td>
<td></td>
<td>The values that this measure reports and the numeric values that correspond to them have been discussed in the table below:

<table><tr><th>Measure Value</th><th>Numeric 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 Measure Values listed in the table above to indicate whether/not the other manager in the cluster is running. In the graph of this measure however, the same will be represented using the numeric equivalents only.</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>Data storage on this manager :

Indicates whether/not data is currently stored in this manager for transmission to the other manager in the cluster.</td>
<td></td>
<td>The values that this measure reports and the numeric values that correspond to them have been discussed in the table below:

<table><tr><th>Measure Value</th><th>Numeric 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 Measure Values listed in the table above to indicate whether/not data is stored in the manager. In the graph of this measure however, the same will be represented using the numeric equivalents only.</td>
</tr>
<tr>
<td></td>
<td>Files stored for the other manager :

Indicates the number of files that are currently waiting to be sent by this manager to the other manager.</td>
<td>Number</td>
<td>• The amount of data that can be stored by a manager for transmission to other managers is controlled by two configuration settings -</td>
</tr>
</table>

	Data stored for the other manager : Indicates the amount of data currently waiting to be sent by this manager to the other manager.	MB	**maxStoragePerFile** and **filesPerManager** - that are present in the file **eg_managers.ini** located in the **<EG_INSTALL_DIR>\manager\config** directory. The setting **maxStoragePerFile** defines the amount of data (in MB) that can be stored in each temporary file that is used to store data temporarily for transmission to a manager. An eG manager can store data in multiple files for transmission to another manager. Multiple files are used for storage (rather than a single file) to minimize data read/write operations to memory for transmission to the other manager. The **filesPerManager** setting defines the maximum number of data files per manager that are used for temporary storage of data. By default, the **maxStoragePerFile** value is 0, and the **filesPerManager** is 0. This implies that a manager does not save data it receives from agents directly for transmission to another manager. If the **maxStoragePerFile** is 10 and the **filesPerManager** is 20, then 200MB of data can be saved for transmission to another manager. If the value of these two measures are consistently close to the **maxStoragePerFile** and **filesPerManager** settings of the monitored eG manager, it is a clear indication that a large volume of data is being generated and readied for transmission by that manager, but its temporary storage is not been tuned adequately to handle this load. If these settings are not changed accordingly, it may result in significant data loss in the event of a manager failure.
	Agents assigned to the manager : Indicates the number of agents that have been explicitly assigned to this manager.	Number	

	Agents reporting to this manager : Indicates the number of agents that are currently reporting metrics to this manager.	Number	

1.4 The eG Access Layer

Besides reporting the number of users registered with the eG Enterprise system and their subscription status, the tests associated with this layer also monitors the session load on the eG manager and the unique users who are logged into the manager. In addition, the layer also runs an **HTTP** test that periodically checks whether the Tomcat web server that supports the eG manager application is available, and if so, how quickly it services requests for web pages. While you can find details of the **HTTP** test in the *Monitoring Web Servers* document, the other tests mapped to this layer have been dealt with in the sections that follow.

Figure 29: The tests mapped to the eG Access Layer

1.4.1 eG Registered Users Test

This test reports the number and type (local or domain) of users who are currently registered with the eG Enterprise system. In the process, the test highlights the following:

- The number and names of users who are newly registered;
- The number and names of users whose subscriptions are about to expire or have expired;
- The AD domains with which the eG manager integrates for managing domain user logins;
-

Purpose	Reports the number and type (local or domain) of users who are currently registered with the eG Enterprise system
Target of the test	The eG Manager

Agent deploying the test	An internal/remote agent		
Configurable parameters for the test	1. **TEST PERIOD** - How often should the test be executed 2. **HOST -** The host for which the test is to be configured 3. **PORT -** The port number at which the specified **HOST** listens 4. **JMX REMOTE PORT** – Here, specify the port at which the **JMX** listens for requests from remote hosts. In the **<EG_MANAGER_INSTALL_DIR>\manager** directory (on Windows; on Unix, this will be the **/opt/egurkha/manager** directory) of the eG manager, you will find a **management.properties** file. Set the port defined against the **com.sun.management.jmxremote.port** parameter of the file as the **JMX REMOTE PORT**. 5. **USER**, **PASSWORD**, and **CONFIRM PASSWORD** – By default, JMX requires no authentication or security. Therefore, the user, **PASSWORD** , and **CONFIRM PASSWORD** parameters are set to *none* by default. 6. **JNDINAME** – The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is *jmxrmi*. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same. 7. **JMX PROVIDER** – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to *com.sun.jmx.remote.protocol*. 8. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to *240* seconds. 9. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against **DD FREQUENCY**. 10. **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.		
Outputs of the test	One set of results for the eG manager being monitored		
Measurements made by the	**Measurement**	**Measurement Unit**	**Interpretation**

test	**Total users:** Indicates the total number of users who are currently registered with the eG Enterprise system.	Number	
	New users: Indicates the number of users who registered with the eG Enterprise system during the last measurement period.	Number	Use the detailed diagnosis to know which users newly registered with the eG Enterprise system.
	Expired users: Indicates the number of user subscriptions that have expired.	Number	Use the detailed diagnosis to know which user subscriptions have expired.
	Nearing expiry users: Indicates the number of user subscriptions that are expected to expire within the next 7 days.	Number	Use the detailed diagnosis to know which user subscriptions that are about to expire within the next seven days.
	Active users: Indicates the number of user subscriptions that have not expired.	Number	
	Local users: Indicates the number of local users who have registered with the eG Enterprise system.	Number	Local users are those users whose login credentials are stored in the eG database and are hence validated by the eG database while logging in.
	Domain users: Indicates the number of domain users who have registered with the eG Enterprise system.	Number	The credentials of domain users are typically maintained by the AD server with which the eG manager integrates. Logins by such users will hence be validated by the AD server only.
	Configured domains: Indicates the number of AD domains that have been configured on the eG manager.	Number	Use the detailed diagnosis to know which domains have been configured.

-

1.4.2 eG User Sessions Test

In environments where the eG management console is accessed by multiple concurrent users, administrators might want to know how many distinct users are logged in currently, and the total session load on the eG manager. This information would enable administrators to audit user accesses to the eG manager and instantly identify unauthorized accesses (if any) that are currently active on the manager. This test provides administrators with these session-centric insights.

Purpose	Reports the session load on the eG manager and failed user sessions
Target of the test	The eG Manager
Agent deploying the test	An internal/remote agent

<table>
<tr><td>Configurable parameters for the test</td><td colspan="3">1. TEST PERIOD - How often should the test be executed
2. HOST - The host for which the test is to be configured
3. PORT - The port number at which the specified HOST listens
4. JMX REMOTE PORT – Here, specify the port at which the JMX listens for requests from remote hosts. In the <EG_MANAGER_INSTALL_DIR>\manager directory (on Windows; on Unix, this will be the /opt/egurkha/manager directory) of the eG manager, you will find a management.properties file. Set the port defined against the com.sun.management.jmxremote.port parameter of the file as the JMX REMOTE PORT.
5. USER, PASSWORD, and CONFIRM PASSWORD – By default, JMX requires no authentication or security. Therefore, the user, PASSWORD , and CONFIRM PASSWORD parameters are set to none by default.
6. JNDINAME – The JNDINAME is a lookup name for connecting to the JMX connector. By default, this is jmxrmi. If you have registered the JMX connector in the RMI registry using a different lookup name, then you can change this default value to reflect the same.
7. JMX PROVIDER – This test uses a JMX Provider to access the MBean attributes of the eG manager and collect metrics. Specify the package name of this JMX Provider here. By default, this is set to com.sun.jmx.remote.protocol.
8. TIMEOUT – Specify the duration (in seconds) for which this test should wait for a response from the eG manager. If there is no response from the eG manager beyond the configured duration, the test will timeout. By default, this is set to 240 seconds.
9. DD FREQUENCY - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is 1:1. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying none against DD FREQUENCY.
10. 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.</td></tr>
<tr><td>Outputs of the test</td><td colspan="3">One set of results for the eG manager being monitored</td></tr>
<tr><td>Measurements made by the</td><td>Measurement</td><td>Measurement Unit</td><td>Interpretation</td></tr>
</table>

test	**Active sessions:** Indicates the number of sessions that are currently active on the eG manager.	Number	This is a good indicator of the current load on the eG manager.
	Failed sessions: Indicates the number of sessions that failed during the last measurement period.	Number	Use the detailed diagnosis of this measure to view the details of the sessions that failed. These details include the user who initiated the session, the IP/host name of the host from which the session originated, when the session began, and when the session last accessed the eG manager.
	Unique user sessions: Indicates the number of distinct users who are currently logged into the eG manager.	Number	Use the detailed diagnosis to know which unique users are logged in.

The detailed diagnosis of the *Active sessions* measure reports the host from which the active sessions originated and the session start time.

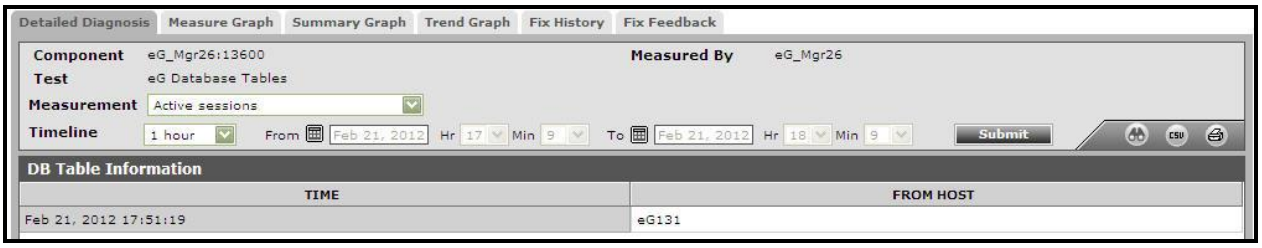


Figure 30: The detailed diagnosis of the Active sessions measure

Chapter 2

Conclusion

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

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