Enabling Service Excellence

Using the eG Integration Console

eG Enterprise v6

Table of Contents

Table of Figures

Introduction

eG Enterprise includes extensive built-in monitoring capabilities for a majority of off-the-shelf applications. However, in any realistic environment, one may encounter applications that are not supported by the eG products. Moreover, administrators may prefer to extend eG's built-in application models to suit their needs and preferences (e.g., to add specific tests from the model). This chapter is intended for users who want to customize and extend eG's monitoring capabilities for their target environment.

To support these capabilities, eG Enterprise includes the **Integration Console**. This is a GUI-based component of the eG manager that allows users to add new servers for monitoring, include new layers for diagnosing specific components, and enhance eG's measurement capability to expose additional information relating to the managed components.

The key features of the Integration Console are:

- ***Customized monitoring capabilities:*** Monitor new and custom applications or network elements. Modify eG's built-in models to suit the specific requirements of the target infrastructure.

- ***Integrated, end-to-end monitoring:*** Monitor the entire target infrastructure; A single integrated interface from where out -of-the-box and custom applications can be monitored.

- ***Seamless integration:*** Custom applications or network elements are integrated into the eG Enterprise system in the same manner as out-of-the-box applications/network elements. Custom measurements can be made, thresholds computed, and the status of the custom applications can be displayed using eG's web-based monitoring interface.

- ***Complete flexibility in the integration***: Integrate a custom application/network element into the eG Enterprise system by choosing any approach that is suitable for the target application/network element. Simple, easy to use templates allow users to develop new monitoring capabilities by reading statistics logged into a file, using SNMP, by invoking application-specific APIs, using OS-level scripting, by accessing custom databases, etc.

- ***Auto-upgrade of the agents:*** No need to bother about manually updating all of the agents. Register the new application/network element to be monitored with the eG manager and have the agents automatically discover and implement the new monitoring capabilities, without needing any manual intervention!

The following sections go into details of the usage of the Integration Console.

1.1 The Integration Console Architecture

Figure 1.1 depicts the architecture of the Integration Console (IC). A user interacts with the Integration Console to enhance eG's models of applications. This process may involve enhancements to existing models of applications or adding new models. In either case, the eG database where measurement results and state information regarding the various servers monitored is stored has to be updated. In addition to updating the database schema when required, the Integration Console is responsible for updating the configuration information that is maintained by the eG manager.

A ***test*** is an object that is invoked by an eG agent periodically to report measures pertaining to a particular component (e.g., web server, network router, database, etc.). These tests can report performance metrics or the configuration information pertaining to the monitored targets. New tests can be added and configured for use by the eG agents using the Integration Console. To enable an agent to implement the new test(s), the Integration Console includes a mechanism through which the agents can automatically discover the addition of new tests to the eG Enterprise system and can enhance their capabilities to start performing the new tests.

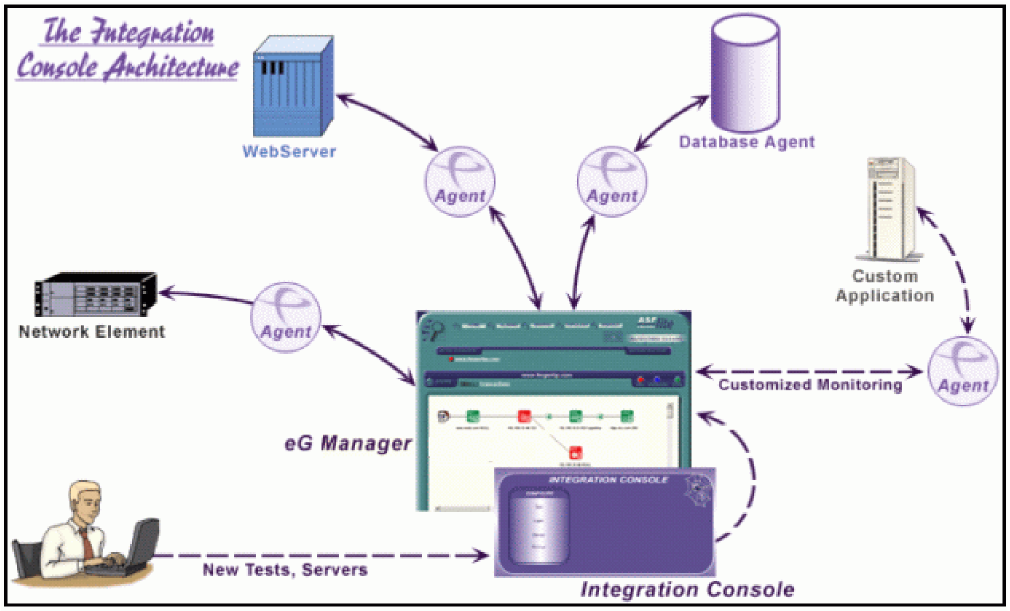


Figure 1.1: Architecture of the eG Integration Console

1.2 System Requirements

The eG Integration Console is a manager-side component, and is included as part of eG Enterprise. The system requirements for the Integration Console are the same as the requirements for the eG manager (i.e., the Integration Console does not have any special requirements).

1.3 Licensing

The eG Integration Console is enabled as an optional component of the eG manager. To see if this capability is enabled for your target environment, login to the eG administration console. If you find an **Integration Console** tile, in the **Admin** tile menu, it is a clear indication that the manager's license has the Integration Console option enabled.

This document elaborately discusses how the Integration Console plugin can be used to:

- Add new performance/configuration tests to the eG Enterprise system;

- Modify performance/configuration tests so added;
- Create new layers and associate new/existing tests with those layers;
- Configure a new component type and build a new layer model for that component type using new/existing layers;

2

Adding/Modifying Tests Using the Integration Console

Before attempting to add a new test using the Integration Console plugin, consider the following:

a) What is the purpose of the test? – should it monitor the performance of the target component or pull out configuration information from it?

b) Does the test take any input arguments for execution? If so, what are they?

c) What are the key measurements/configuration information (as the case may be) that you want the test to pull out from the target component?

d) How should the test collect measurements/configuration information from the target? By running a custom Java-based program? by executing a script/batch file? by executing a SQL query? using Perfmon counters? by polling the SNMP MIB? or by using JMX-based interfaces?

The answer to question d) above determines the type of test that you want to build. The eG Enterprise system supports six types of tests, namely, **Custom**, **Script/Batch File**, **SQL Query**, **Perfmon**, **Snmp**, and **Jmx**.

1. **Custom** - Tests of this type offer complete flexibility to users in developing and integrating new monitoring functionality into the eG Enterprise system. In order to develop a **Custom** test, a user must use eG's Java-based programming interface (which will be described later in this document).

2. **Script / Batch File** – In some cases, it may be easier to build new monitoring capabilities using simple shell scripts or batch files or VB scripts or powershell scripts. To support this capability, the Integration Console supports a **Script/Batch File** test. When choosing this type, the user provides the script/file to be used and the Integration Console takes care of integrating the script into the eG framework.

3. **SQL Query** – Many a times critical statistics or an application are stored in a database. To make it possible to extract such statistics from the database without having to write elaborate programs, the Integration Console includes an **SQL Query** test type.

4. **Perfmon** – **Perfmon** tests can operate only on Windows environments. This option provides a quick and easy way of building tests that interface with the Windows Perfmon capability to collect various metrics of interest.

5. **Snmp –** Tests of this type are typically executed on network devices such as routers, hubs, switches etc., so that performance statistics pertaining to the same can be obtained using the SNMP protocol.

6. **Jmx - Java Management eXtensions (JMX)** offers a standard way by which applications can expose custom metrics for monitoring tools. The eG Integration Console can now be configured to collect and report on applications that offer JMX-based

interfaces.

As the procedure for adding and configuring tests is different for each type of test, the sections to follow will discuss each of these test types independently using separate examples.

2.1 Adding a Custom Test

As stated earlier, a **Custom** test type offers you the complete flexibility in introducing new monitoring functionality. To implement a **Custom** test, a test class is generated using a Test Generator API to perform the specific functions expected of the test. A detailed description of the Test Generator API is available in Section 2.4.4.

Using a **Custom** test type, you can build both performance and configuration tests. This section takes the help of illustrated examples to explain the process of building a **Custom** performance test and a **Custom** configuration test.

2.1.1 Adding a Custom Performance Test

To illustrate the procedure for adding and configuring a **Custom** performance test, this section will be considering two examples. The test that will be added in the first example is the **MsFileTest**. This test should report statistics pertaining to the number of files locked on a server, and the unique count of users with open files on that server. In the first example, this test will not be configured with the detailed diagnosis capability. In the second example, the locked file count measure of the **MsFileTest** will be configured to report detailed diagnostics revealing the details of the files that are locked.

2.1.1.1 Adding a Custom Performance Test Without Detailed Diagnosis

The first step towards building the **MsFileTest** in our example is to access the **INTEGRATION CONSOLE – TEST** page. For this, first login to the eG admin interface as a user with **Admin** rights, invoke the **Admin** tile menu, and pick the **Test** option from the **Integration Console** tile (see Figure 2.1).

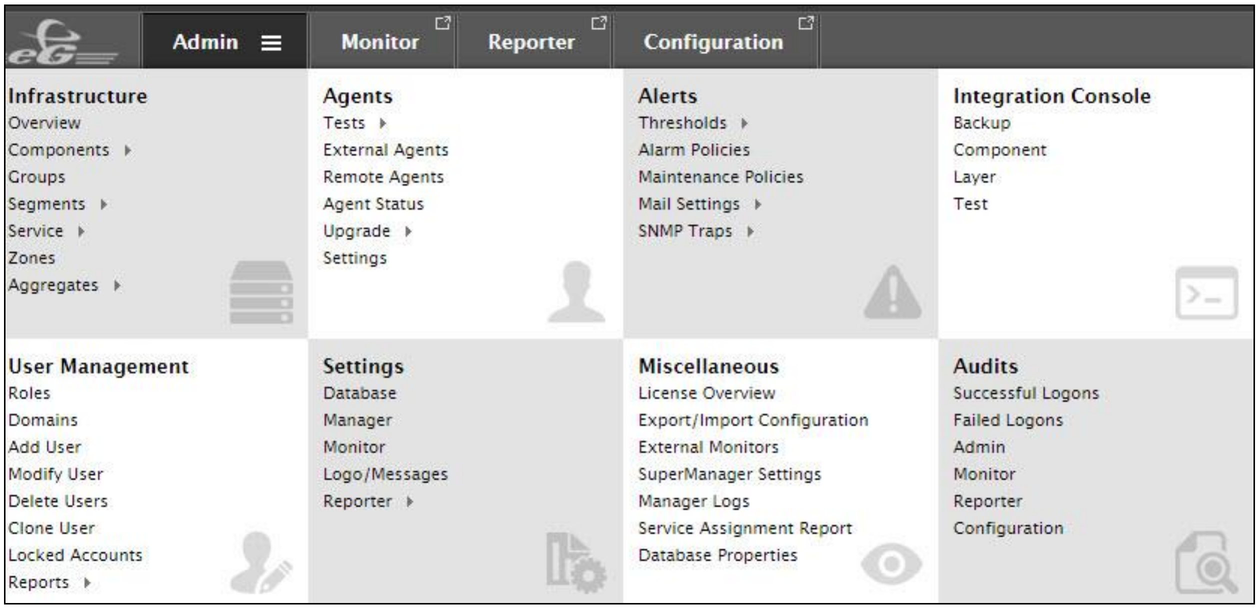
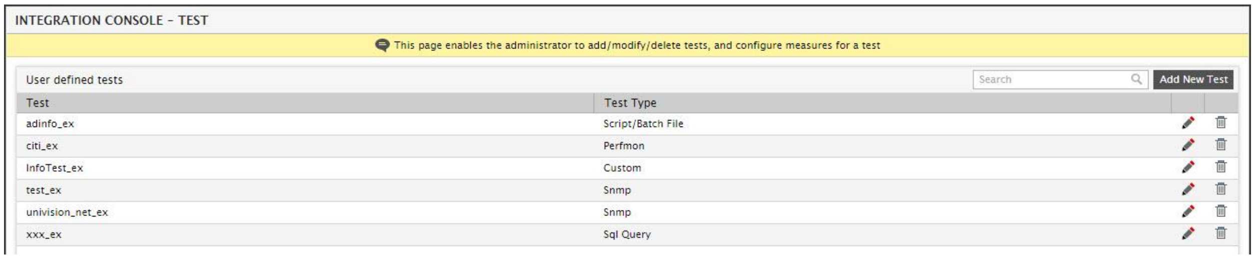


Figure 2.1: Selecting the Test option from the Integration Console tile

To add a new test, click on the **Test** option in Figure 2.1. A set of user-defined tesbs that have been previously added to the eG Enterprise system (if any) will be displayed (see Figure 2.2). Click the **Add New Test** button in Figure 2.2 to add a new test to the eG Enterprise system.


INTEGRATION CONSOLE - TEST

This page enables the administrator to add/modify/delete tests, and configure measures for a test

User defined tests

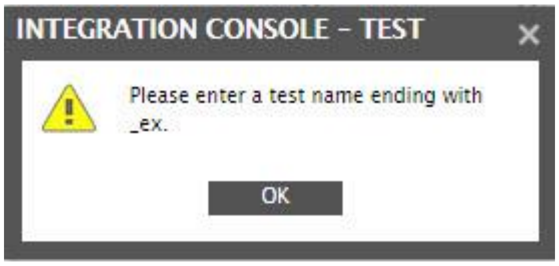
Test	Test Type
adinfo_ex	Script/Batch File
citi_ex	Perfmon
InfoTest_ex	Custom
test_ex	Snmp
univision_net_ex	Snmp
xxx_ex	Sql Query

Figure 2.2: List of user-defined tests that pre-exist

Figure 2.4 shows the inputs that have to be specified to add a new test. First, specify the **Test name** as shown in Figure 2.4.

Note

While adding a new test using the Integration Console, ensure that the **Test name** always ends with **_ex**. If not, an error message (see Figure 2.3) will appear upon clicking the **Add** button in Figure 2.2.



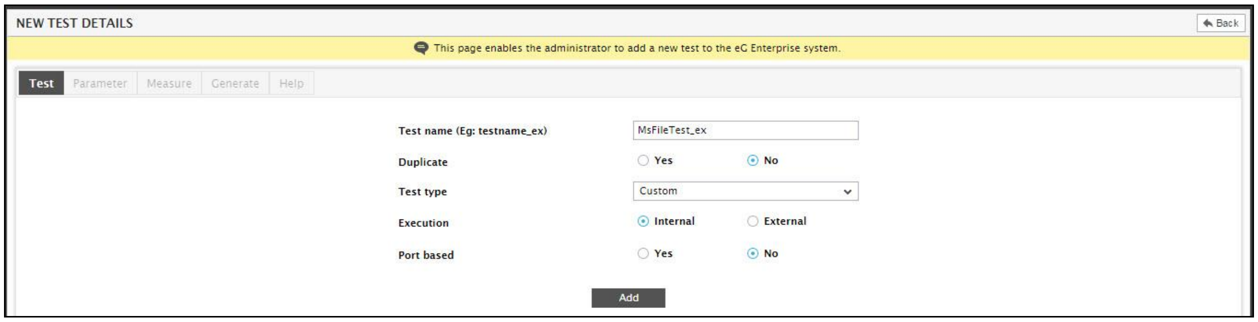
INTEGRATION CONSOLE – TEST

Please enter a test name ending with _ex.

Figure 2.3: A message box stating that the test name should end with "_ex"

Click the **OK** button in the message box to close it, and proceed to provide the **Test name** in the prescribed format.

Once the test name is specified, you wil have to indicate whether/not the test being added is a duplicate of an existing test. eG Enterprise allows users to duplicate an existing IC-based test, so that the complete configuration of the existing test (i.e., the test type, the test parameters, its measures, and all other test specifications **except the test name**) is automatically copied to the new test. This duplication is particularly useful when you want to create two/more tests with the same/similar functionality, and assign them to different layers or components.



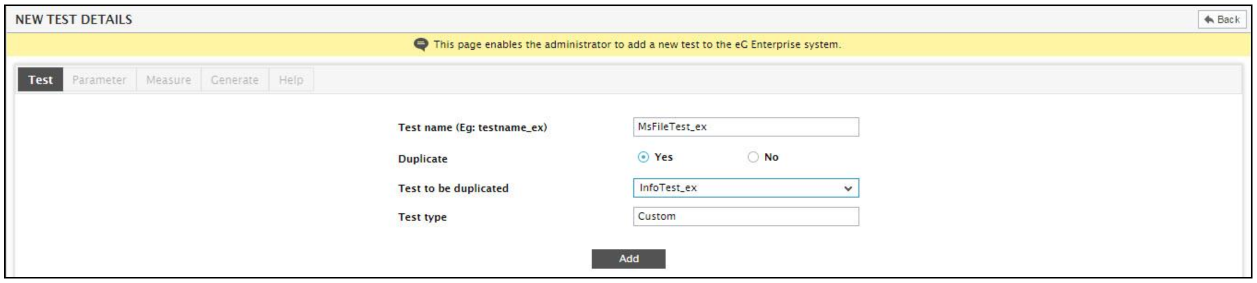
NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Test name (Eg: testname_ex)	MsFileTest_ex
Duplicate	Yes [✓ No]
Test type	Custom
Execution	[✓ Internal] External
Port based	Yes [✓ No]

Figure 2.4: Specifying the inputs corresponding to a new Custom test

If the test being added is a duplicate of an existing test, then set the **Duplicate** flag in Figure 2.4 to **Yes**. From the **Test to be duplicated** list that then appears (see Figure 2.5), pick the existing test that is to be duplicated. Upon selection of the test, the **Test type** of the chosen test will automatically apply to the new test. Now, click the **Add** button to add the new test.



NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Test name (Eg: testname_ex)	MsFileTest_ex
Duplicate	[✓ Yes] No
Test to be duplicated	InfoTest_ex
Test type	Custom

Figure 2.5: Duplicating a test

On the other hand, if you set the **Duplicate** flag to **No**, then you would have to explicitly provide the new test's details. The **MsFileTest_ex** in our example is not a duplicate of any other test. Therefore, set the **Duplicate** flag to **No** and proceed to specify the mode of **Execution** of the test (see Figure 2.4). The **Execution** field governs whether the test is to be executed by an internal agent or by an external agent. As a rule of thumb, if the test relies on application counters, log files, etc., it must be executed by an internal agent. On the other hand, if the test uses the Simple Network Management Protocol (SNMP), HTTP, or emulates user requests it can be executed by an external agent.

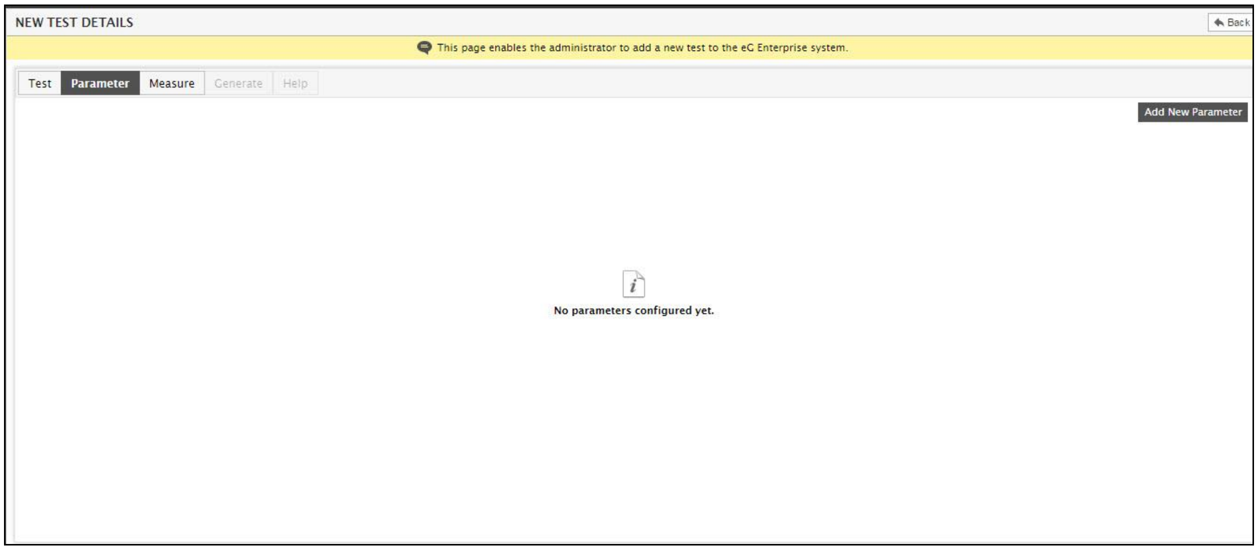
Note

Using the Integration Console plugin, you can add an internal or an external test, but you cannot add tests that need to be run by a remote agent – i.e., tests that need to be executed in an agentless manner.

When adding a test, you also have to specify whether the test is **Port based** or not. In other words, you have to indicate whether the test corresponds to a server that is listening on a specific TCP/UDP port or not.

Apart from the above, the **Test type** has to be selected. For the purpose of our example, select **Custom** as the **Test type**. Then, click the **Add** button to add the new test to the eG Enterprise system.

When the test is successfully added, control will automatically switch to the **Parameter** tab page of the **NEW TEST DETAILS** page (as shown by Figure 2.6). Parameters refer to the arguments that a test needs to be configured with in order to run and collect performance statistics from a target.



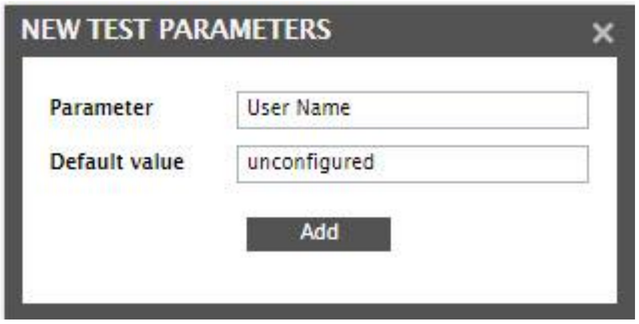
This page enables the administrator to add a new test to the eG Enterprise system.

No parameters configured yet.

Figure 2.6: The Parameter tab page

By default, any test you add using the Integration Console, will take **TEST PERIOD** and **HOST** as its parameters. While the **TEST PERIOD** represents the frequency with which the test is to be run, the **HOST** indicates the host on which the test should run. If the test is a port-based test, then, it will additionally take the **PORT** number as its parameter. You will not be able to view any of these default parameters, and hence will not be able to modify or delete them.

Some tests however, may support more parameters than the default **TEST PERIOD**, **HOST**, and **PORT**. Such parameters can be user privileges that a test may require for connecting to targets, filter conditions, pattern specifications etc. You can create these additional parameters for a new test using the **Parameter** tab page. For this, you will have to click the **Add New Parameter** button in Figure 2.6. This will invoke Figure 2.7, where you can specify the **Parameter** name and the **Default value** of the parameter.



Parameter: User Name

Default value: unconfigured

Figure 2.7: Adding a new test parameter

Once a **Default value** is set here, then every time the test runs, it will automatically use this default value for all servers on which it runs. Therefore, when setting a **Default value**, make sure that the value you provide does not have to be changed at run-time according to the target server/environment.

If the value of the parameter may vary with the target server/environment, then its best to leave the **Default value** as *unconfigured*. If this is done, then, once this test is mapped to a server and that server is managed in eG, the eG Enterprise system will explicitly prompt you to configure this parameter with a valid value before attempting to monitor that server. This way, you will be able to configure this parameter with the appropriate value at run-time and ensure that the test executes smoothly.

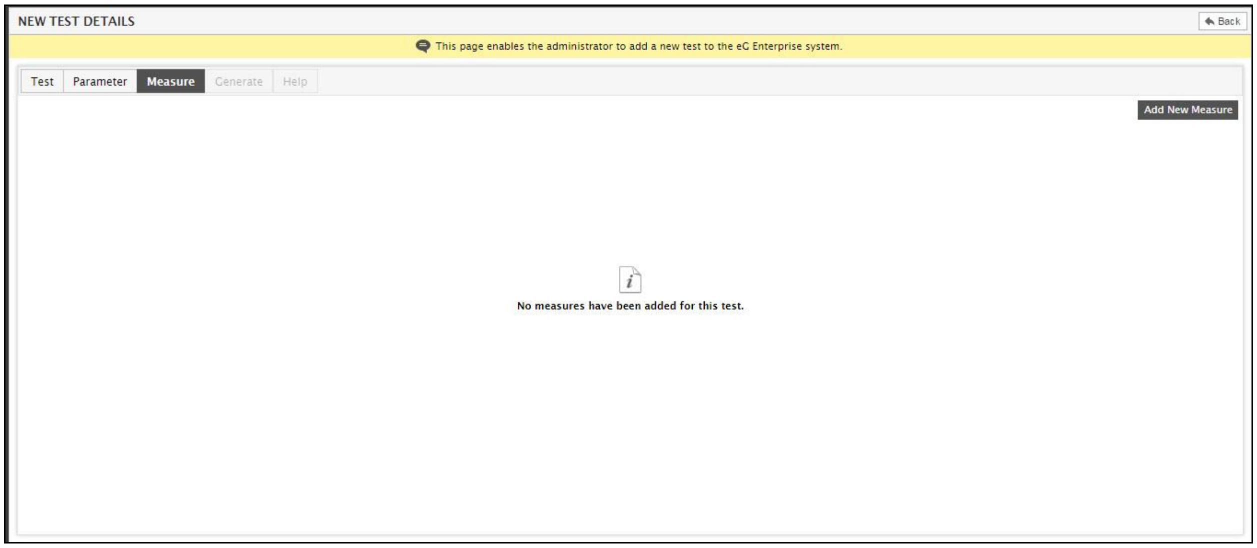
In addition to the default parameters bundled with every new test, eG Enterprise provides three ready-to-use parameters that can be, if required, configured for any new test added using the Integration Console. These parameters are, namely, **executionTime**, **useMgrTime, and executeAtFixedTime**. Typically, the **executionTime** parameter can be set for tests that need to execute at a specific time every day, or at the beginning of every hour. For instance, say, you want to add a new test that should trigger a 'database cleanup' process at 11 PM every day and alert administrators if the process fails to start at the set time. Such a test can take **executionTime** as one of its parameters, so that you can specify the exact time of the day at which the clean up process needs to begin. Similarly, if you want a test to execute at the beginning of every hour and report metrics, once again, you might want to use the **executionTime** parameter. While setting the **TESTPERIOD** as 1 hour can ensure that the test runs once every hour, it cannot guarantee that the test will run at the beginning of every hour. To add this parameter to a new test, specify **executionTime** in the **Parameter** text box of Figure 2.7. You can leave the **Default value** as **unconfigured**, as this parameter would typically expect user input.

If the **executionTime** parameter has been added to any new IC test, then, depending on need, you may want to use the **useMgrTime** parameter along with it. By default, as soon as the eG agent starts, it synchronizes time with the eG manager, and proceeds to report measures at the manager's time. In MSP environments particularly, a single, central manager will be receiving performance data from managed customer environments across the globe - i.e., from agents operating in different time zones. By default, all these agents will run tests and report measurements at the manager's time. This means, if an **executionTime** has been specified for one/more tests executed by these agents, and key maintenance processes are being triggered by these tests, such processes will run at the manager's time and not the agent's, thereby causing confusion. Therefore, whenever there is a time difference between the agent and manager zones, you will have to make sure that the test that supports the **executionTime** parameter runs according to the agent's time and not the manager's. To ensure this, you will have to additionally configure a **useMgrTime** parameter for the test. To add this parameter to an IC-based test, specify **useMgrTime** in the **Parameter** text box, and set the **Default value** to **no**. Also, note that the **executionTime** parameter can be added as a stand-alone parameter to any new IC-based test, but the **useMgrTime** parameter can only be used alongside the **executionTime** parameter.

Note

The **executeAtFixedTime** parameter can be used if a test needs to run at a fixed interval (in hours), starting from 'midnight' every day. For instance, you may have written a test to check whether/not a virus scanner runs at 12 AM and every 3 hours thereafter, every day. To make sure that the test runs at this exact frequency, first integrate the test into the eG Enterprise system via IC, and when doing so, specify **executeAtFixedTime** as a **Parameter** of the test; then, set **Yes** as its **Default value**. Later, when configuring this test for a component, make sure that you set the **TEST PERIOD** as 3 hours. Typically, eG Enterprise computes a test's frequency based on when the eG agent executing that test was started. In the case of this test however, since the **executeAtFixedTime** parameter is set to **Yes**, the eG agent uses 12 AM (midnight) as the base for computing the test frequency, regardless of when it (i.e., the eG agent) was started. This ensures that the test runs every 3 hours from midnight – i.e., at 3 AM, 6 AM, 9 AM, 12 PM, 3 PM, 6 PM, 9 PM, 12 AM, and so on! Moreover, in this case, even if the eG agent is restarted in-between – say, at 2.30 PM – it will run the test at 3 PM as originally scheduled, and not at 5.30 PM (which is 3 hours from the time the eG agent was started). When using the **executeAtFixedTime** parameter however, make sure that the **TEST PERIOD** is only set in hours and is set to a number that is a factor of 24 – i.e., it can only be 1, 2, 3, 4, 6, 8, 12, or 24 hours. This is because, this parameter computes the test frequency using the '24-hour' time format.

In our example, the **MsFileTest_ex** does not require any parameters to obtain the statistics of interest. Therefore, proceed to configure the measures to be reported by the test, by clicking on the **Measure** tab page in Figure 2.6. Since measures are yet to be configured for the **MsFileTest_ex**, you will find a message to that effect in Figure 2.8 that appears.


NEW TEST DETAILS

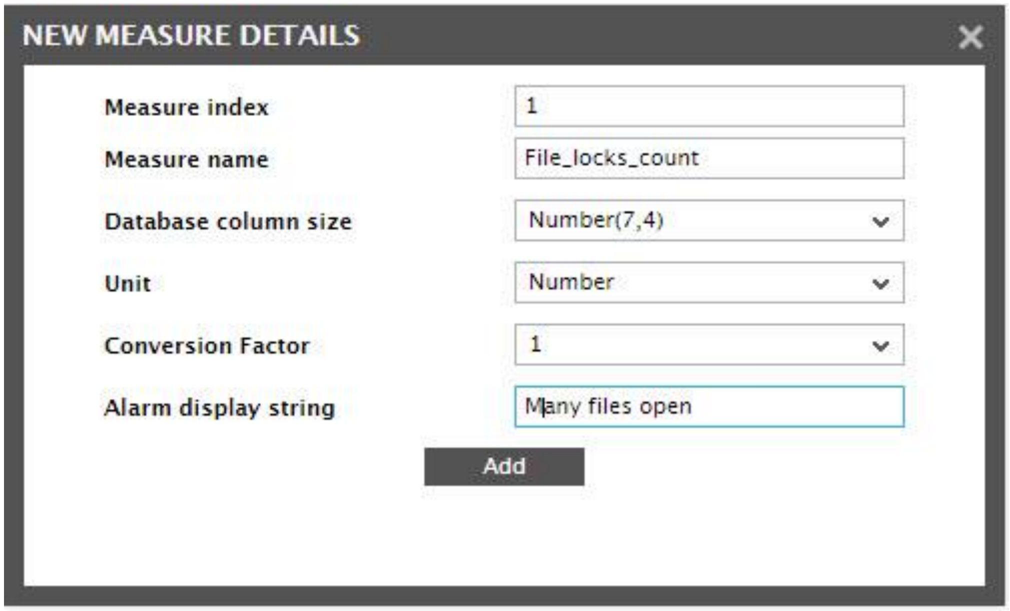
Back

This page enables the administrator to add a new test to the eG Enterprise system.

No measures have been added for this test.

Figure 2.8: The Measure tab page indicating that measures are yet to be configured for the MsFileTest_ex

Click the **Add New Measure** button in Figure 2.8 to add a new measure. Figure 2.9 will then pop-up, using which you can create a new measure.


NEW MEASURE DETAILS

Measure index	1
Measure name	File_locks_count
Database column size	Number(7,4)
Unit	Number
Conversion Factor	1
Alarm display string	Many files open

Figure 2.9: Adding a new measure

The **Measure name** field indicates the name of the measurement (this will be displayed in eG's monitor interface), while the **Unit** (whether %, seconds, requests/sec, etc.) specifies the unit in which the measurement's value is reported. The **Database column size** field indicates the size of each database record corresponding to a measurement value. For example, Number(7,4) indicates that the output of the measurement will be a number in the range 0 to

100, with the fractional value being limited to four decimal places. The **Measure index** determines the order in which the measures are to be displayed in the monitor console. For example, if **1** is the **Measure index**, then the corresponding measure will appear first in the list of measures displayed in the monitor console. This index is automatically generated by the eG Enterprise system, and cannot be edited by the user. In our example, the **Measure index** of the measure **File_locks_count** is **1**.

Sometimes, administrators may want to convert the unit of measurement of a performance metric before displaying the same in the eG monitoring console. For instance, a duration value originally available in 'milliseconds' may have to be changed to 'Secs' before it is displayed in the console. If you want the unit of the measure being added to be so converted at test run time, then select a **Conversion Factor** from the list. For example, if a value in 'milliseconds' needs to be converted to 'seconds', then select */1000* as the conversion factor. By default, *1* is chosen as the **Conversion Factor**; this implies that, by default, unit conversion does not take place for a new measure at test run time.

Note

The **Conversion Factor** list comes with a default set of conversion factors. You can however, override this list by adding more conversion factors. For instance, the default **Conversion Factor** list does not provide the option to convert Bytes to MB at run time. To include this option in the list, follow the steps below:

a. Edit the **eg_ui.ini** file in the **<EG_INSTALL_DIR>\manager\config** directory.

b. To include a new conversion factor, you will have to append an entry of the following format to the **[CONVERSION_FACTORS]** section of the file:

c. *DisplayName=Value*

d. For instance, to support 'Bytes to MB' conversion, append the following entry to the **[CONVERSION_FACTORS}** section:

   /1048576 (Bytes to MB)=0.00000095367431640625

e. In this case, the *DisplayName,* /**1048576 (Bytes to MB)**, will be displayed as an option in the **Conversion Factor** drop-down list. If this option is chosen, then, at test run time, the conversion value of **0.00000095367431640625** will be multiplied with the actual measure value that is reported in Bytes to convert it into MB. **Care should be taken while specifying the conversion value, as incorrect values will result in wrong measures being reported by the test.**

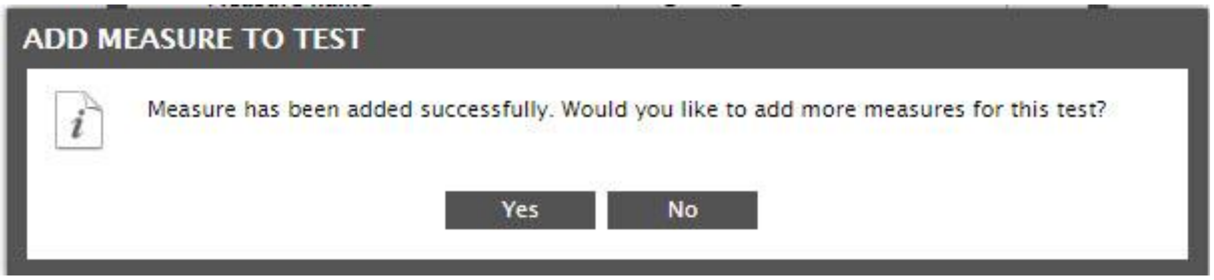
f. Once the new entry is appended to the **[CONVERSION_FACTORS]** section, save the file.

g. Finally, restart the manager.

Once this is done, you will find the string **/1048576 (Bytes to MB)** appear as an option in the **Conversion Factor** list.

When configuring a measurement for a custom test, also specify the text string that must be displayed in the eG alarm window when the corresponding measurement violates its threshold. This is only an optional field. For our example, however, let us specify an appropriate **Alarm display string** (see Figure 2.9)

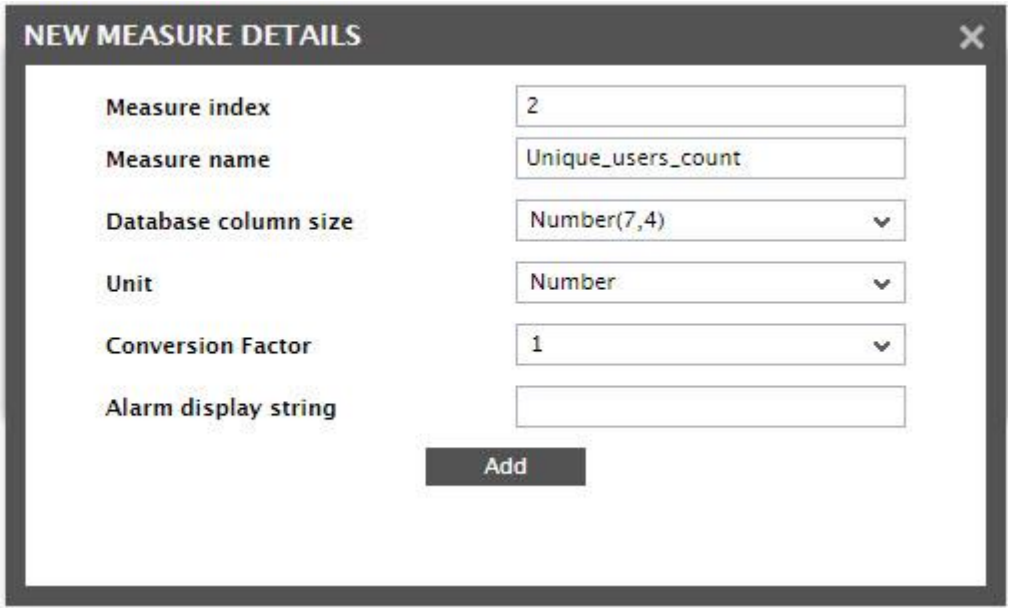
By clicking on the **Add** button in Figure 2.9, the first measurement of the **MsFileTest_ex** can be added. To add more measures, click the **Yes** button in the message box (see Figure 2.10) that appears subsequently.


ADD MEASURE TO TEST

Measure has been added successfully. Would you like to add more measures for this test?

Figure 2.10: A message box requesting you to confirm whether/not you want to add more measures for the MsFileTest_ex

This will once again open the **NEW MEASURE DETAILS** pop-up, using which you can configure the second measure – i.e., the *Unique_users_count* measure - of the **MsFileTest_ex** (see Figure 2.11)


NEW MEASURE DETAILS

Measure index	2
Measure name	Unique_users_count
Database column size	Number(7,4)
Unit	Number
Conversion Factor	1
Alarm display string	

Figure 2.11: Adding the second measure of the MsFileTest_ex

If you click on the **Add** button in Figure 2.11, the message box of Figure 2.10 will re-appear. Siince no more measures need be added for the **MsFileTest_ex**, this time, click the **No** button in the message box to end measure configuration.

Doing so will instantly lead you to the **Generate** tab page, where you will have to specify the implementation of the test. All **Custom** tests that are newly added to eG have to be implemented in Java. The later sections of this chapter will describe how the eG test generator API can be used to implement new tests.

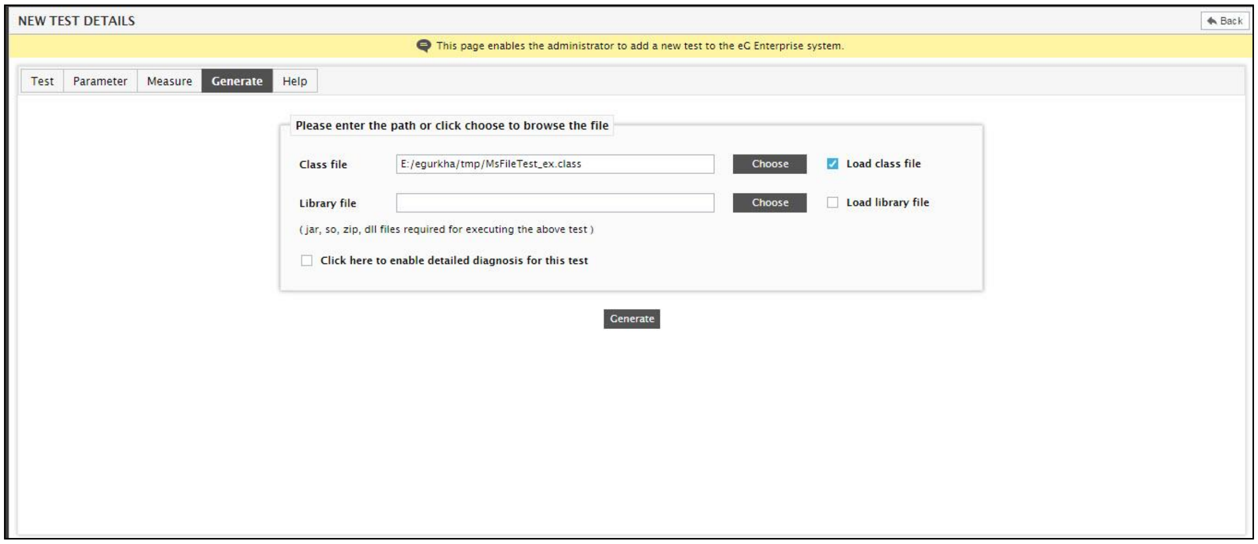

NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Please enter the path or click choose to browse the file
Class file
E:/egurkha/tmp/MsFileTest_ex.class
Choose
Load class file
Library file
Choose
Load library file
(jar, so, zip, dll files required for executing the above test)
Click here to enable detailed diagnosis for this test
Generate

Figure 2.12: Specifying a test's implementation

Figure 2.12 shows how a new test can be added to the eG Enterprise system after it has been configured. After all the measurements of the test have been specified, the directory in which the test's implementation exists should be specified in the input box corresponding to the **Class file** specification. The test implementation must exist in a Java class file with the same name as the test name – e.g., the implementation of **MsFileTest_ex** must exist in a class file named `MsFileTest_ex.class`.

If the class file is present in a remote location, then you can upload it to the eG manager, by clicking on the **Choose** button adjacent to the **Class file** text box. This will invoke a pop-up window using which you can **Browse** for the class file and specify its location (see Figure 2.13).

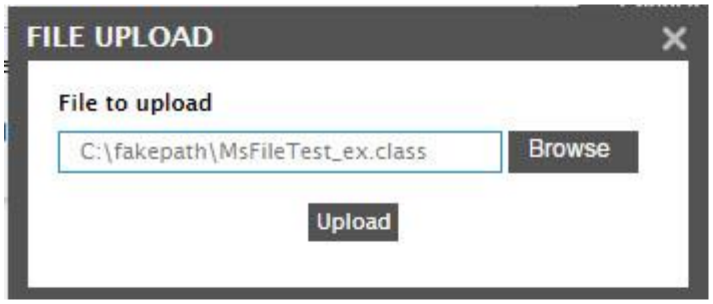
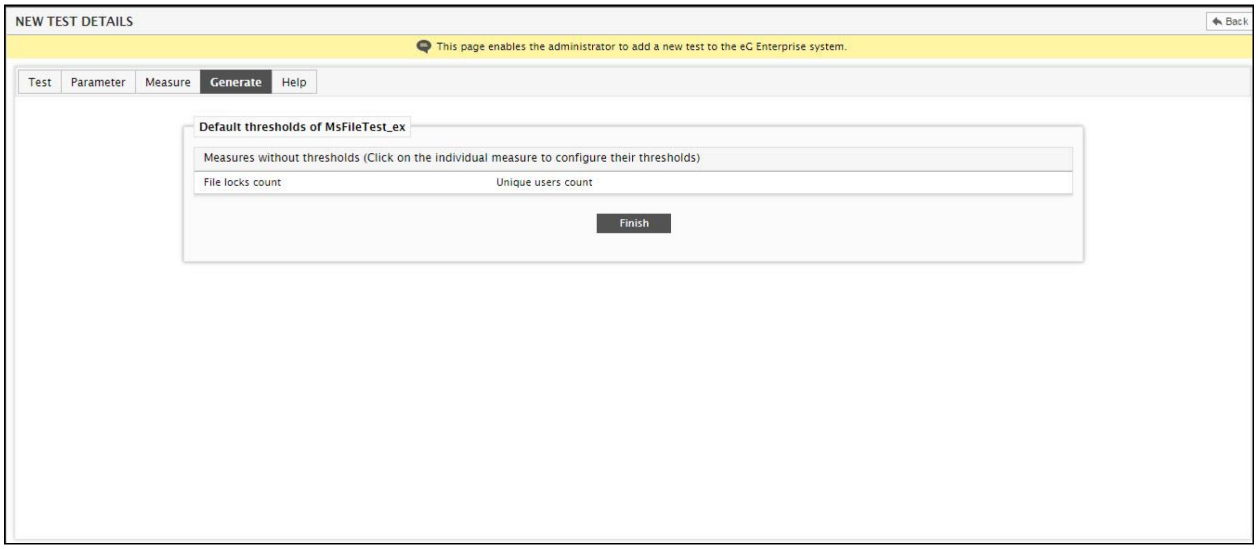


Figure 2.13: Uploading the class file to the eG manager

Finally, click on the **Upload** button in the pop-up window (see Figure 2.13), to upload the class file in the remote location to the eG manager. If the class file has already available on the eG manager system, just specify the location of the file against the **Class file** text box. Some class files may require certain library files (eg., ".jar" or ".so" (shared object files) files) for their execution. The name of the file along with the directory in which these files exist has to be specified in the **Library file** specification. Please note that the size of these files should not exceed 0.5 MB in order to prevent excessive load while uploading. If the library file is present in a remote location, then you can upload it to the eG manager, by clicking on the **Choose** button adjacent to the **Library file** text box. Note that the **Load class file** checkbox should be selected if the class file has to be loaded every time the test is implemented. The **Load library file** checkbox should be selected if the library file has to be loaded every time the test is implemented.

Since in this example, the **MsFileTest_ex** need not be configured with any detailed diagnosis capability, leave the **Click here to enable detailed diagnosis for this test** check box in Figure 2.12 unchecked. Once the class and library files are specified, click the **Generate** button to generate the test.

Clicking on **Generate** invokes Figure 2.14, where you can view the measures that have been configured for the **MsFileTest_ex**. You can set default thresholds for a measure by clicking on that measure in Figure 2.14.


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

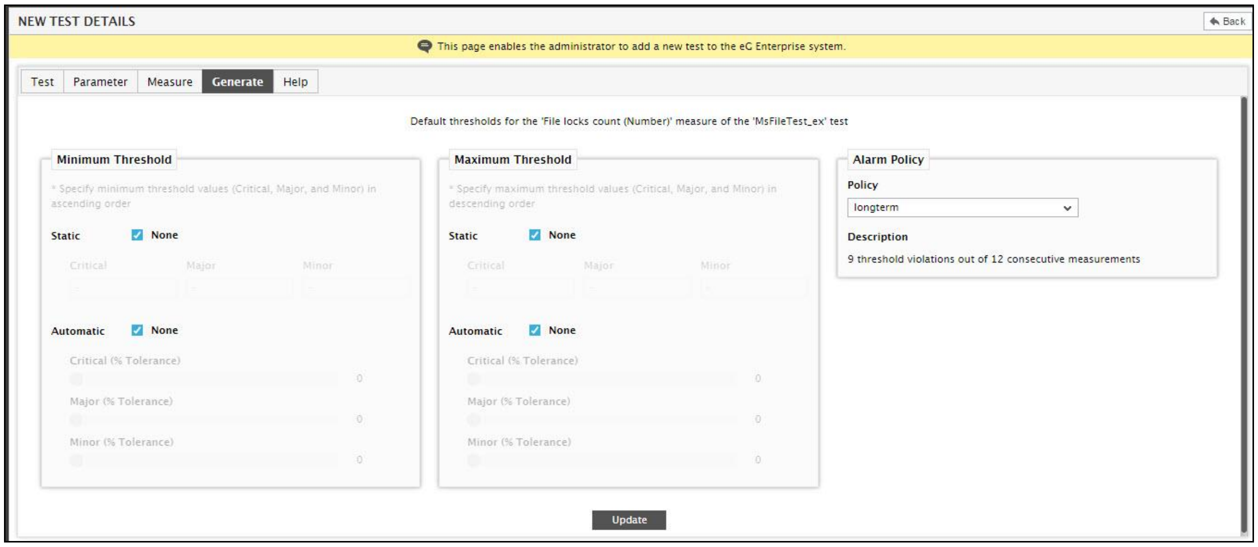
Default thresholds of MsFileTest_ex

Measures without thresholds (Click on the individual measure to configure their thresholds)

File locks count	Unique users count

Figure 2.14: Viewing the measures of the MsFileTest_ex

Figure 2.15 will then appear, which will help you configure the thresholds for the chosen measure.


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Default thresholds for the 'File locks count (Number)' measure of the 'MsFileTest_ex' test

Minimum Threshold: * Specify minimum threshold values (Critical, Major, and Minor) in ascending order. Static: None (checked). Automatic: None (checked). Critical (% Tolerance) 0, Major (% Tolerance) 0, Minor (% Tolerance) 0.

Maximum Threshold: * Specify maximum threshold values (Critical, Major, and Minor) in descending order. Static: None (checked). Automatic: None (checked). Critical (% Tolerance) 0, Major (% Tolerance) 0, Minor (% Tolerance) 0.

Alarm Policy: Policy: longterm. Description: 9 threshold violations out of 12 consecutive measurements

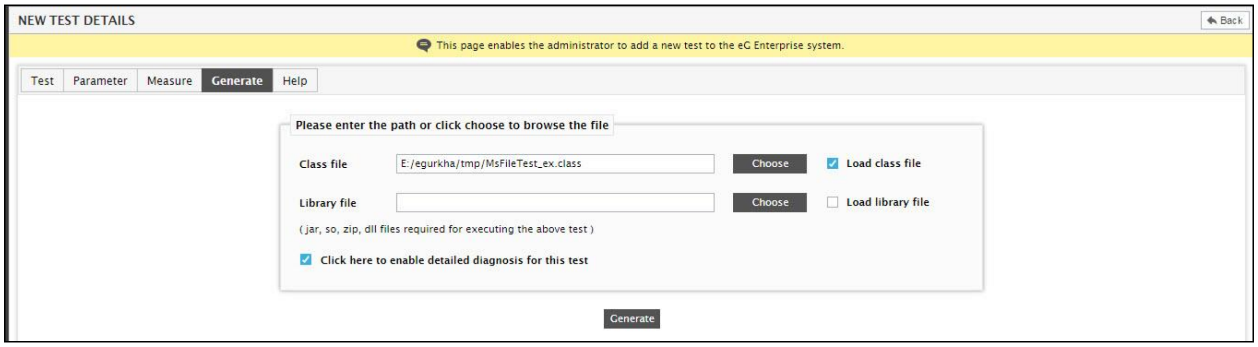
Figure 2.15: Configuring the thresholds for a measure of the MsFileTest_ex

2.1.1.2 Configuring Detailed Diagnosis for a Custom Performance Test

To make diagnosis more efficient and accurate, eG Enterprise 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. Alternatively, these tests can also be run periodically for proactive monitoring purposes.

If required, you can also configure a custom test to report detailed diagnostics. To lend this capability to the

MsFileTest_ex in our example, you will have to select the **Click here to enable detailed diagnosis for this test** check box in the **Generate** tab page, when configuring the test's implementation (see Figure 2.16).


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Please enter the path or click choose to browse the file

Class file: E:/egurkha/tmp/MsFileTest_ex.class [Choose] [✓] Load class file

Library file: [Choose] [ ] Load library file

(jar, so, zip, dll files required for executing the above test)

[✓] Click here to enable detailed diagnosis for this test

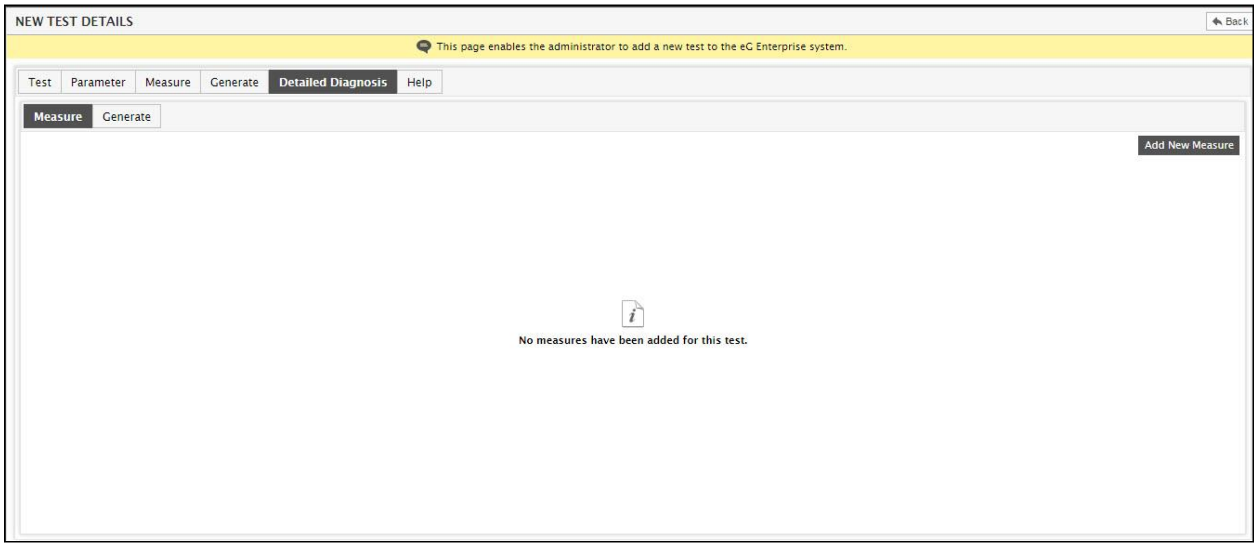
Figure 2.16: Configuring detailed diagnosis for the MsFileTest_ex

Note

The option to enable the detailed diagnosis capability for an IC test will be available only if the following conditions are fulfilled:

- The eG license should enable the detailed diagnosis capability
- Both the normal and abnormal detailed diagnosis frequencies should not be 0. For more information on configuring these frequencies, refer to the *eG User Manual*.

Finally, click the **Generate** button in Figure 2.16. This will introduce a new **Detailed Diagnosis** tab page in the **NEW TEST DETAILS** page (see Figure 2.17).


NEW TEST DETAILS

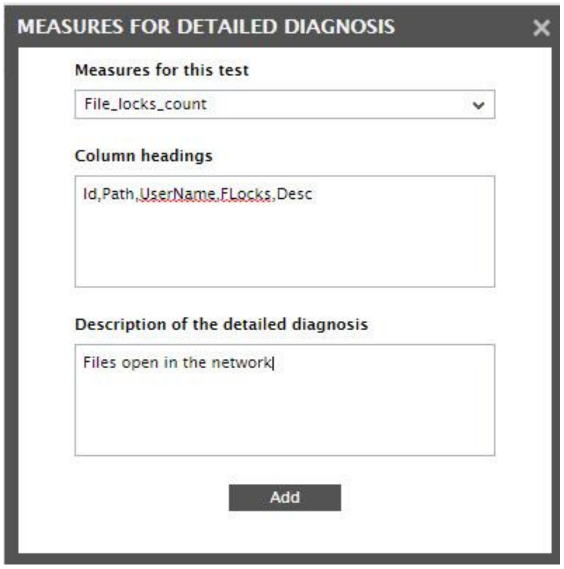
This page enables the administrator to add a new test to the eG Enterprise system.

No measures have been added for this test.

Figure 2.17: The Detailed Diagnosis tab page

The **Detailed Diagnosis** tab page provides two sub-tabs, namely, **Measure** and **Generate**. Since the detailed diagnosis capability has not yet been configured for any measure of the **MsFileTest_ex**, the **Measure** tab page will only display a message to that effect. To indicate for which measure of the **MsFileTest_ex** detailed diagnostics should be reported and what type of information should be collected as part of the detailed diagnostics, click the **Add New Measure** button in Figure 2.17.

When Figure 2.18 appears, select the measure for which detailed diagnostics are required from the **Measures for this test** drop-down list. For our example, let us configure the **File_locks_count** measure of the **MsFileTest_ex** to provide a detailed diagnosis that will reveal the list of open files in the network. Therefore, select **File_locks_count** from the list. Next, provide a comma-separated list of **Column headings** under which the detailed diagnosis information will be displayed in the eG monitor interface (see Figure 2.18). Then, provide an appropriate **Description of the detailed diagnosis** that will be displayed in the **DETAILED DIAGNOSIS** page of the eG monitor interface. Finally, click on the **Add** button in Figure 2.18 to apply the selection.



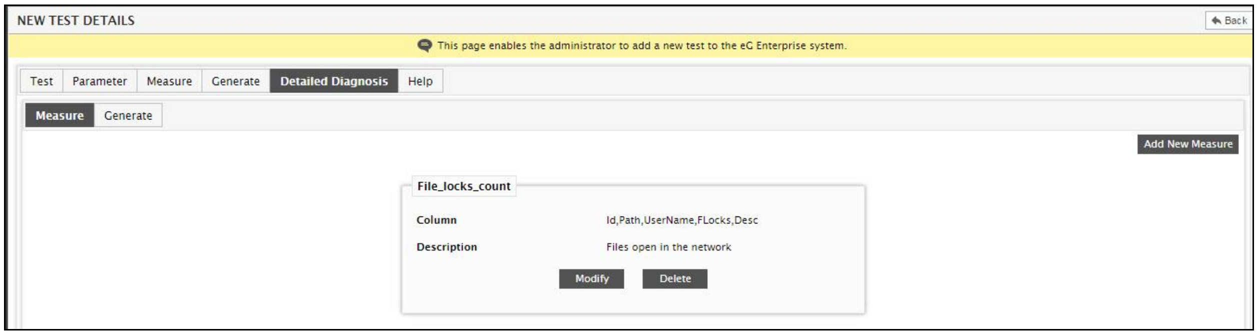
Measures for this test: File_locks_count

Column headings: Id,Path,UserName,FLocks,Desc

Description of the detailed diagnosis: Files open in the network

Figure 2.18: Configuring detailed diagnosis for the File_locks_count measure of the MsFileTest_ex

A summary of the specification will then be displayed (see Figure 2.19). You can either **Modify** the specification or **Delete** it using the buttons provided therein.



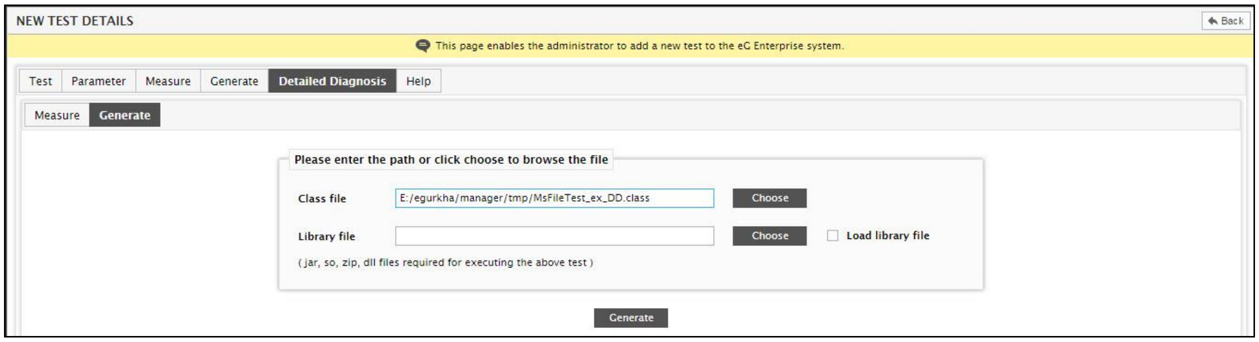
This page enables the administrator to add a new test to the eG Enterprise system.

File_locks_count

Column	Id,Path,UserName,FLocks,Desc
Description	Files open in the network

Figure 2.19: A summary of the detailed diagnosis specification of the MsFileTest_ex

If you want to proceed with the displayed specifications, simply click the **Generate** sub-tab in Figure 2.19 to specify the implementation of the detailed diagnosis.


NEW TEST DETAILS

Back

This page enables the administrator to add a new test to the eG Enterprise system.

Test | Parameter | Measure | Generate | Detailed Diagnosis | Help

Measure | Generate

Please enter the path or click choose to browse the file

Class file: E:/egurkha/manager/tmp/MsFileTest_ex_DD.class Choose

Library file: Choose Load library file

(jar, so, zip, dll files required for executing the above test)

Generate

Figure 2.20: Specifying the detailed diagnosis implementation

The detailed diagnosis implementation must exist in a Java class file whose name is the test name suffixed with *"_DD"* - e.g., the implementation of detailed diagnosis for the **MsFileTest_ex** must exist in a class file named **MsFileTest_ex_DD.class**. The path to this class file has to be mentioned in the **Class file for detailed diagnosis** text box in Figure 2.20. If the file exists in a remote location, then click on the **Choose** button adjacent to the text box to upload the file to the eG manager.

As before, specify any library files using the **Library file** specification (see Figure 2.20).

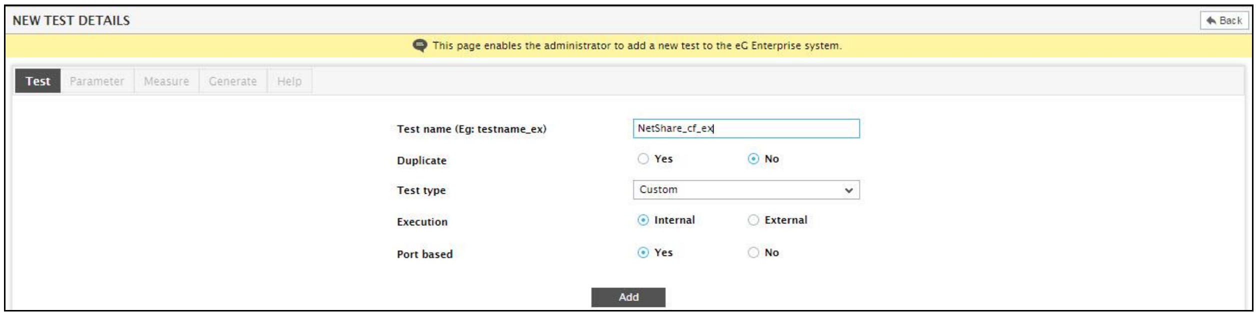
Finally, click the **Generate** button in Figure 2.20 to generate the test.

2.1.2 Adding a Custom Configuration Test

To take you step-by-step through the procedure for building a new configuration test, this section takes the help of two examples. In the first example, a descriptor-based test named **NetShare** will be added, which will report the names of shared folders on a target system, the full path to each of the shared folders, and the user-defined remarks associated with each shared folder. In the second example, a non-descriptor-based test named **NetShareCount** will be added, which will report the number of shared folders on a target system. This test too will later be associated with the **NetShare** component-type.

2.1.2.1 Adding a Descriptor-based Configuration Test

To add a descriptor-based configuration test, first, click the **Add New Test** button in the **INTEGRATION CONSOLE - TEST** page (see Figure 2.2). In the **NEW TEST DETAILS** page (see) that appears next, specify the **Test name** (see Figure 2.21). For the purpose of our example, set **Duplicate** flag to **No** (as the test being added is not a duplicate of any existing IC-based test), set **Execution** mode to **Internal**, **Port based** to **Yes**, and **Test type** to **Custom**.


NEW TEST DETAILS

Back

This page enables the administrator to add a new test to the eG Enterprise system.

Test | Parameter | Measure | Generate | Help

Test name (Eg: testname_ex): NetShare_cf_ex

Duplicate: Yes [✓ No]

Test type: Custom

Execution: [✓ Internal] External

Port based: [✓ Yes] No

Add

Figure 2.21: Adding a descriptor-based configuration test

Then, click the **Add** button in Figure 2.21 to add the new test. Figure 2.22 will then appear.

Note

- While adding a configuration test, make sure the **Test name** ends with **_cf_ex**. Therefore, the name of the test in our example will be **NetShare_cf_ex**.
- Using the Integration Console plugin, you can add an internal or an external test, but you cannot add tests that need to be run by a remote agent – i.e., tests that need to be executed in an agentless manner.

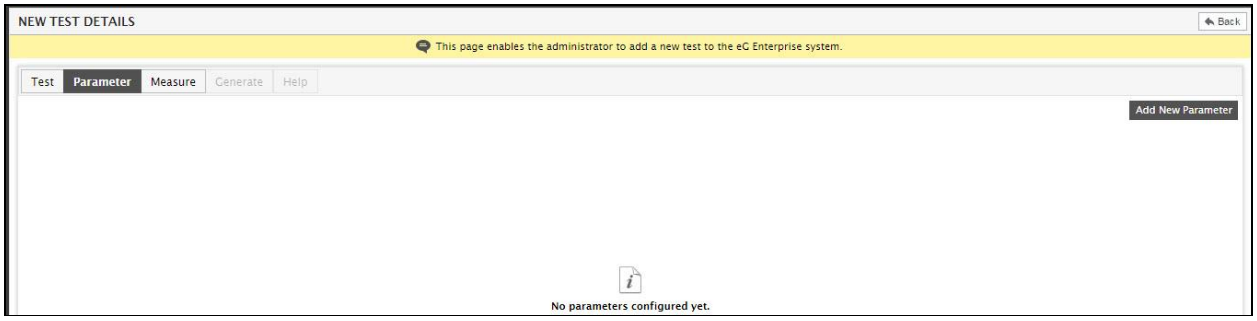
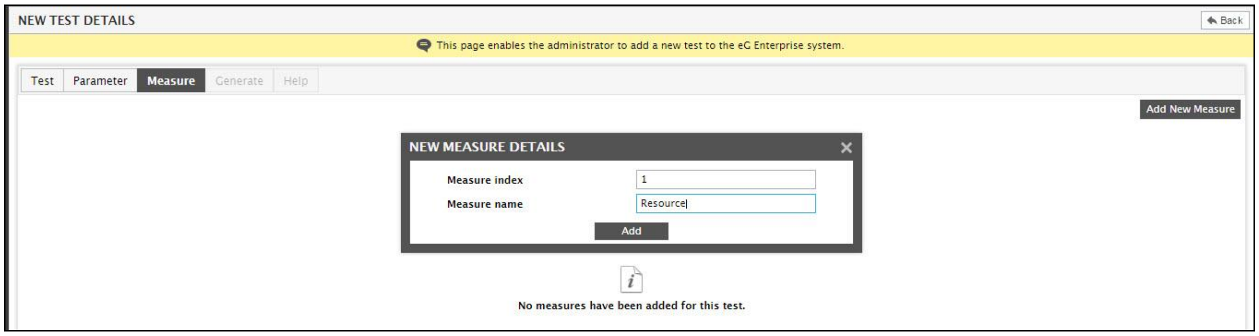

NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Add New Parameter
No parameters configured yet.

Figure 2.22: Specifying the parameters of the new test

Since the **NetShare_cf_ex** test does not take any input parameters, proceed to configure the measures for the test by clicking the **Measure** tab page in Figure 2.22.

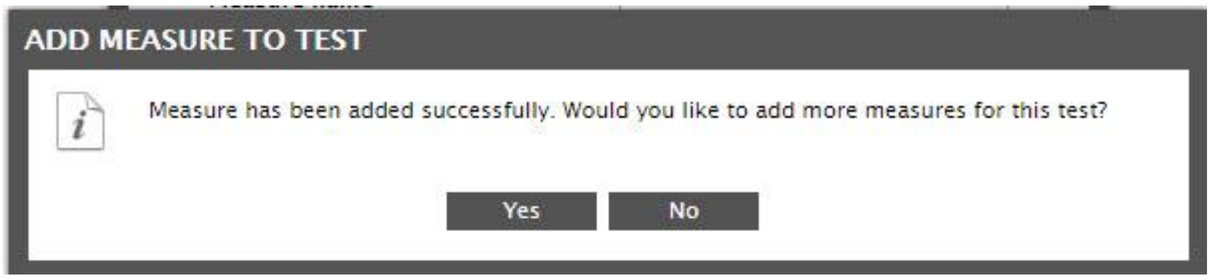
Figure 2.23 will then appear. Click the **Add New Measure** button in the **Measure** tab page of Figure 2.23. In the **NEW MEASURE DETAILS** window that pops up, specify **Resource** as the name of the first measure in the **Measure name** text box. This measure will report the full path to the shared folders on a target.


NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Add New Measure
NEW MEASURE DETAILS
Add
No measures have been added for this test.

Measure index: 1
Measure name: Resource

Figure 2.23: Configuring the first measure for the configuration test

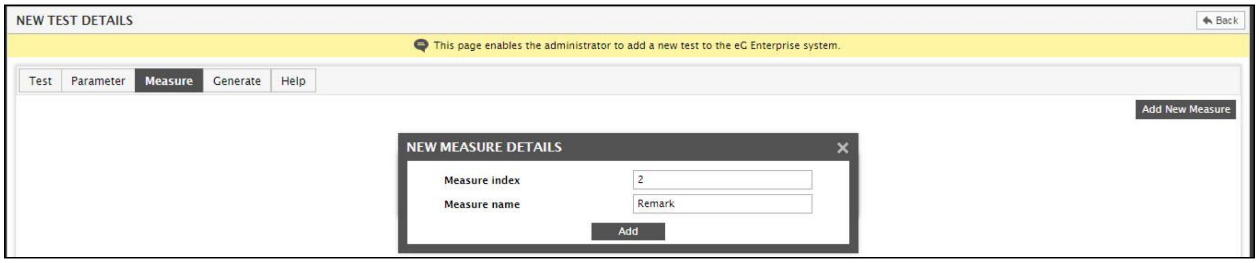
Then, click the **Add** button in Figure 2.23. When you are prompted to add more measures for the test (see Figure 2.24), click **Yes** to continue adding measures.


ADD MEASURE TO TEST

Measure has been added successfully. Would you like to add more measures for this test?

Figure 2.24: A message box requesting your confirmation to continue adding measures for the NetShare_cf_ex test

This will invoke Figure 2.25, wherein you can add the second measure – **Remark**.


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

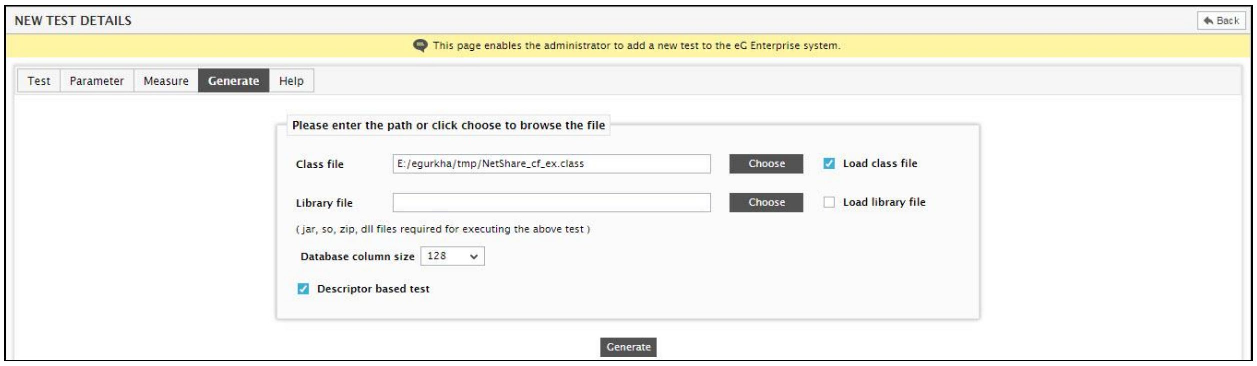
NEW MEASURE DETAILS

Measure index: 2

Measure name: Remark

Figure 2.25: Adding the second measure of the NetShare_cf_ex test

Click the **Add** button in Figure 2.25 to add the second measure. Now, proceed to generate the test. For this, once the **Generate** tab page appears, specify the directory in which the test's implementation exists in the input box corresponding to the **Class file** specification. The test implementation must exist in a Java class file with the same name as the test name – e.g., the implementation of **NetShare_cf_ex** test in our example must exist in a class file named *NetShare_cf_ex.class* (see Figure 2.26).


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Please enter the path or click choose to browse the file

Class file: E:/egurkha/tmp/NetShare_cf_ex.class

[✓] Load class file

Library file:

☐ Load library file

(jar, so, zip, dll files required for executing the above test)

Database column size: 128

[✓] Descriptor based test

Figure 2.26: Generating a test

If the class file is present in a remote location, then you can upload it to the eG manager, by clicking on the **Choose** button adjacent to the **Class file** text box. This will invoke a pop-up window using which you can **Browse** for the class file and specify its location, and click on the **Upload** button to upload the class file to the eG manager. If the class file has already available on the eG manager system, just specify the location of the file against the **Class file** text box. Some class files may require certain library files (eg., ".jar" or ".so" (shared object files) files) for their execution. The name of the file along with the directory in which these files exist has to be specified in the **Library file** specification. Please note that the size of these files should not exceed 0.5 MB in order to prevent excessive load while uploading. If the library file is present in a remote location, then you can upload it to the eG manager, by clicking on the **Choose** button adjacent to the **Library file** text box. Note that the **Load class file** checkbox should be selected if the class file has to be loaded every time the test is implemented. The **Load library file** checkbox should be selected if the library file has to be loaded every time the test is implemented.

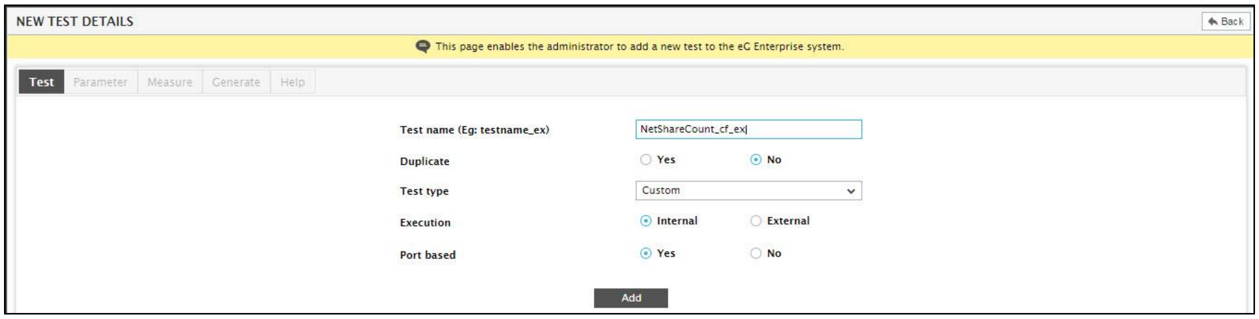
Next, indicate the **Database column size** of both the configured measures by picking an option from the drop-down list with the same name.

Since the **NetShare_cf_ex** test in our example is a descriptor-based test, select the **Descriptor based test** check box in Figure 2.26. Finally, click the **Generate** button. In the case of descriptor-based configuration tests, the eG Enterprise system will automatically append a measure named **Installed** to the list of measures that pre-exist for the test. This is done to enable the test to keep track of the installed status of the descriptors - this way, if a descriptor is removed, then the configuration monitoring module will be able to automatically capture this change and update the change tracker with it.

With that, the new descriptor-based configuration test is added.

2.1.2.2 Adding a Non-Descriptor-based Configuration Test

To add a non-descriptor-based test, first click on the **Add New Test** button in **INTEGRATION CONSOLE – TEST** page (see Figure 2.2). When the **NEW TEST DETAILS** page appears (see Figure 2.27), type **NetShareCount_cf_ex** against **Test name**, for the purpose of our example. Then, set **Duplicate** flag to **No** as the test being added is not a duplicate of any existing IC-based test. Next, set **Internal** as the **Execution** mode, set the **Port based** flag to **Yes**, and **Test type** to **Custom**. Finally, click on the **Add** button to add the new test.

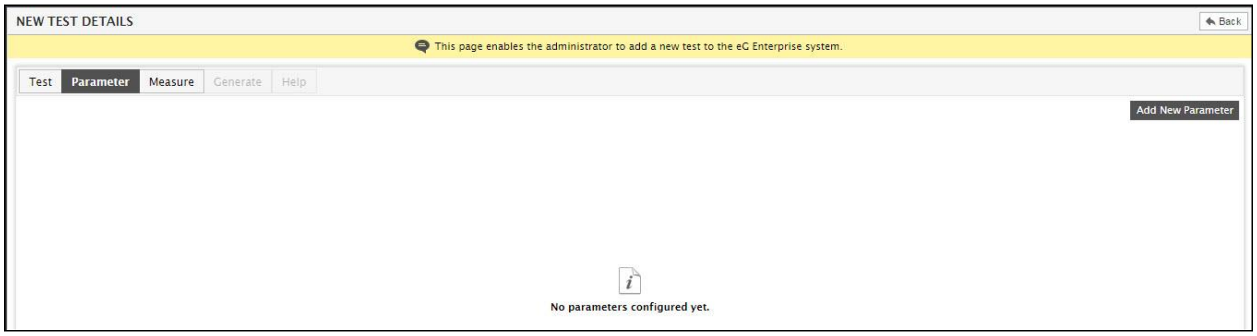


This page enables the administrator to add a new test to the eG Enterprise system.

Test name (Eg: testname_ex)	NetShareCount_cf_ex
Duplicate	Yes [✓ No]
Test type	Custom
Execution	[✓ Internal] External
Port based	[✓ Yes] No

Figure 2.27: Adding a new non-descriptor-based test

Figure 2.28 will then appear. To configure measures for the **NetShareCount_cf_ex** test, click the **Measure** tab page in Figure 2.28.



This page enables the administrator to add a new test to the eG Enterprise system.

No parameters configured yet.

Figure 2.28: Viewing the details of the non-descriptor-based test

When Figure 2.29 appears, click the **Add New Measure** button therein to add a measure for the **NetShareCount_cf_ex** test.

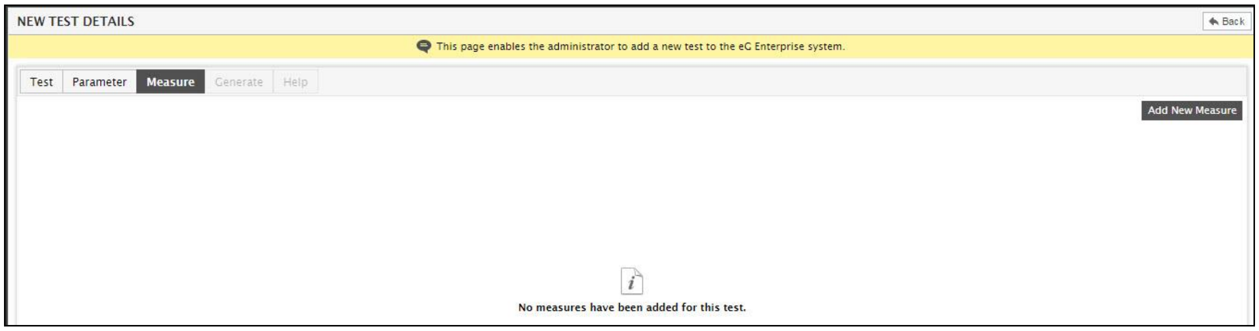
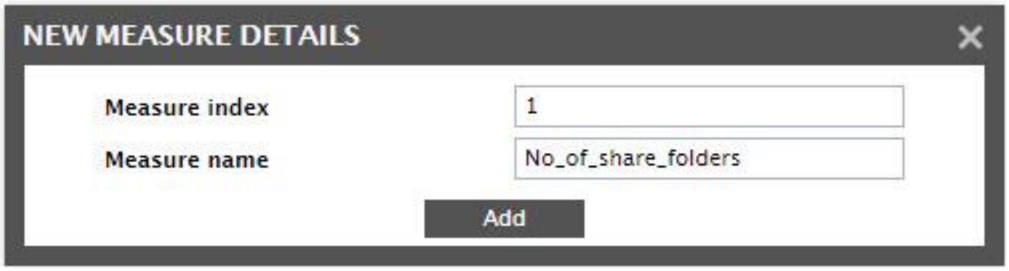

NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Add New Measure
No measures have been added for this test.

Figure 2.29: Clicking the Add New Measure button to add a new measure for the NetShareCount_cf_ex test

Figure 2.30 then appears. The **NetShareCount_cf_ex** test reports the count of shared folders on the target system. Therefore, type **No_of_share_folders** as the **Measure name** in Figure 2.30 and click the **Add** button.

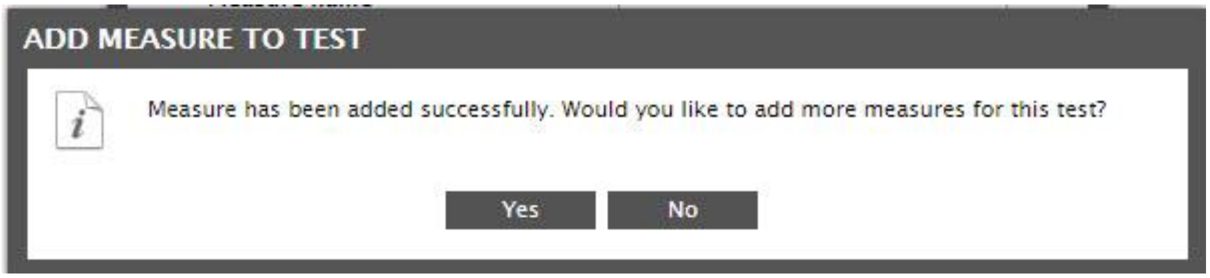

NEW MEASURE DETAILS

Measure index: 1

Measure name: No_of_share_folders

Figure 2.30: Configuring the No_of_share_folders measure of the NetShareCount_cf_ex test

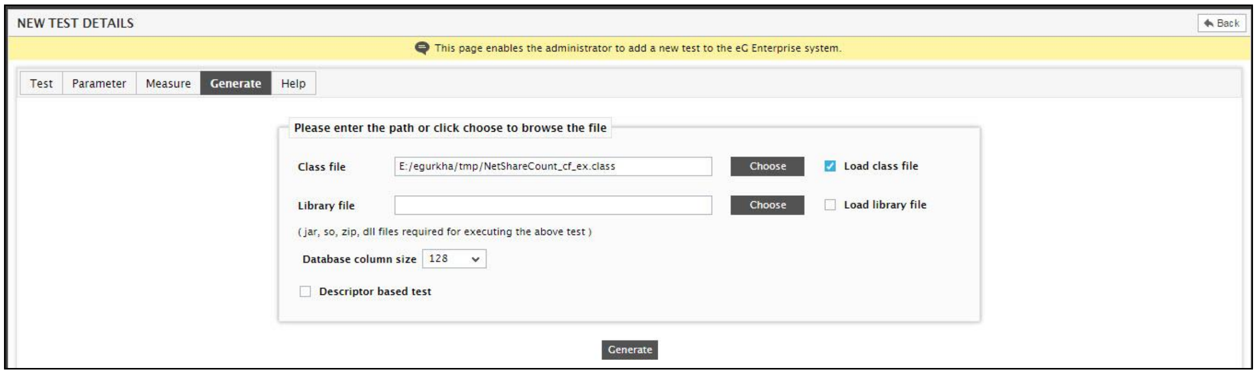
When you are prompted to add more measures for the test (see Figure 2.31), click **No** to indicate that you have finished adding measures for the test.


ADD MEASURE TO TEST

Measure has been added successfully. Would you like to add more measures for this test?

Figure 2.31: A message box requesting your confirmation to continue adding measures for the NetShareCount_cf_ex test

This will automatically take you to the **Generate** tab page. To generate the test, by specifying the full path to the **Class file** that holds the test's implementation logic.


NEW TEST DETAILS

Back

This page enables the administrator to add a new test to the eG Enterprise system.

Test | Parameter | Measure | Generate | Help

Please enter the path or click choose to browse the file

Class file: E:/egurkha/tmp/NetShareCount_cf_ex.class | Choose | [✓] Load class file

Library file: | Choose | [ ] Load library file

(jar, so, zip, dll files required for executing the above test)

Database column size: 128

[ ] Descriptor based test

Generate

Figure 2.32: Generating the non-descriptor-based test

The test implementation must exist in a Java class file with the same name as the test name – e.g., the implementation of **NetShareCount_cf_ex** test in our example must exist in a class file named `NetShareCount_cf_ex.class` (see Figure 2.32).

If the class file is in a remote location, then you can upload it to the eG manager, by clicking on the **Choose** button adjacent to the **Class file** text box. This will invoke a pop-up window using which you can **Browse** for the class file and specify its location, and click on the **Upload** button to upload the class file to the eG manager. If the class file has already available on the eG manager system, just specify the location of the file against the **Class file** text box. Some class files may require certain library files (eg., ".jar" or ".so" (shared object files) files) for their execution. The name of the file along with the directory in which these files exist has to be specified in the **Library file** specification. Please note that the size of these files should not exceed 0.5 MB in order to prevent excessive load while uploading. If the library file is present in a remote location, then you can upload it to the eG manager, by clicking on the **Choose** button adjacent to the **Library file** text box. Note that the **Load class file** checkbox should be selected if the class file has to be loaded every time the test is implemented. The **Load library file** checkbox should be selected if the library file has to be loaded every time the test is implemented.

Next, indicate the **Database column size** of the configured measure.

Since the **NetShareCount_cf_ex** test in our example is a non-descriptor-based test, just click the **Generate** button to generate the test.

2.1.3 Test Generator API

So far, we have reviewed how a new **Custom** test can be integrated into the eG Enterprise system. The one aspect that was not covered in the previous sections is how a test class can be generated to perform the specific functions expected of the **Custom** test. This section explains the test generator API that is provided with the eG Enterprise system to enable users to design and implement new testing capability.

The test generator API consists of a **GenericTest** abstract class. In order to extend eG's monitoring functionality, a user has to develop new tests that extend the **GenericTest** and implement new monitoring capabilities.

Figure 2.33 shows the architecture of the eG test generator API. The API is the module that links user-defined tests to other eG agent components.

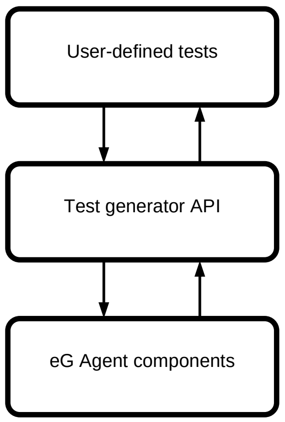


Figure 2.33: Architecture of eG's test generator API

2.1.3.1 System Requirements

The test generator API is included as part of the eG agent package. In order to use the API, the CLASSPATH environment setting for the user who is crafting new tests must be set so as to include the Java archive file **`<EG_HOME_DIR>\lib\eg_agent.jar`** and **`<EG_HOME_DIR>\lib\eg_plus.jar`**, where **`EG_HOME_DIR`** is the installation directory of the eG manager and agents (eg. **`/opt/egurkha`** on Unix, **`C:\Program Files\eGurkha`** on Windows). Consequently, all new test developments must be performed on a system on which the eG agent has been installed. Once the test is developed and compiled to produce the class file, this file must be made available on the eG manager system, so that the test can be integrated into the eG Enterprise system using the Integration Console.

2.1.3.2 Component Classes

The Test generator API consists of a single abstract class called **GenericTest** whose functionality has to be implemented in order to develop a test that would monitor a component type of user's choice.

2.1.3.3 Summary of Methods

The following methods of the Test generator API must be used for developing new tests.

Method Signature	Description
public void computeMeasures(Hashtable paramList)	This method must be overridden to implement specific test execution functionality. This method should also make necessary calls to other methods in the API as explained above. Apart from that the method may call other user-specific methods or include within itself the functionality for executing the test and assigning values to measures. The result of the execution decides the further calls in the method. **Note:** It is mandatory to override this method.
final void setMeasureCount(int measureCount)	This method must be called in the constructor to enable the eG Agent component to register the number of measures this test would report. **Note:** It is mandatory to call this method from the constructor. The count of measures should be the same as the number of measurements

	configured via the eG Integration Console interface. **Description of arguments:** `measureCount`: This argument indicates the number of measures the test would report.
final void addNewMeasure(ArrayList measureList)	This method is called whenever a set of measures are to be reported. This is to be used *only* by a non-descriptor based Test. **Note:** The ArrayList passed should always contain a group of Double objects and the list should always follow a same order. **Description of arguments:** `measureList:` This argument represents an *ArrayList* collection of all double values encapsulated into *Double* objects.
final void addNewMeasure(String descriptor, ArrayList measureList)	This method is called whenever a set of measures are to be reported. This is to be used *only* by a descriptor based Test. **Note**: The ArrayList passed should always contain a group of Double objects and the list should always follow a same order. **Description of arguments:** `descriptor`: This argument represents a *String* that describes the name with which the corresponding set of measures are associated. `measureList`: This argument represents an *ArrayList* collection of all double values encapsulated into *Double* objects.
final void setErrorMessage (String errorMsg)	This method can be called whenever the developer wants to log an error or unexpected output for a test. **Note**: The error can be read from the `agent/logs/error_log` file that is available in the agent install directory. **Description of arguments:** `errorMsg`: This argument represents a *string* which describes the error that has occurred.

Note

Tests that are to be run by an external agent can determine the target server and port number that they are monitoring using the variables - targetHost (italics) and portNo (italics) that are defined in the base class - i.e., the GenericTest class.

2.1.3.4 Writing Tests using the Test Generator API

This section outlines how user-defined tests that extend eG's functionality can be written using the test generator API. A user defined test class must extend the GenericTest abstract class and override the only abstract method to report measures to the eG agent system.

Given below is a sample test that reports the availability of a server.

```
import java.util.*;
```

```
public class AvailabilityTest extends GenericTest

{

// declare necessary variables

double availability = 0;      // default value of server availability

public AvailabilityTest ( String [] args )

   {

/*** Calling super initializes several parameters for the test

namely

•         targetHost          -         Target Host for the test

•         portNo              -         Target Port for the test value is "NULL" if its not a
port-based server

•         other parameters configured using the Integration Console

interface

          ***/


super(args);

          /***      Call setMeasureCount to initialize the number of

                    measures for the test

          ***/


setMeasureCount(1); // One measure namely - availability

   }




/*** Call computeMeasures to report measures ***/

public void computeMeasures ( Hashtable paramList )

   {

          /***      We may call new methods to perform different tasks

                    Assume we use a method called getAvailability which

                    Performs the availability check and returns the value.

           ***/


availability = getAvailability();

ArrayList al = new ArrayList();

al.add(new Double(availability));
```

```
          /***      Measures have to be reported by populating an arraylist

                    and passing it as argument to the addNewMeasure() method

          ***/


addNewMeasure(al);

   }


// User defined methods

private double getAvailability()

   {

          /***      This is a user-defined implementation. The user can choose one of several
                    approaches to determine a server's availability:


          Eg. 1     Can Establish socket connection to the targetHost and

                    portNo and find out the availability


          Eg. 2     Can ping the targetHost and find out the availability


          Eg. 3     Can establish a HTTP connection to the targetHost and

                    portNo and find out the availability


          Note :    The values of target host and port can be accessed by

                    using the public variables namely "targetHost" and

                    "portNo" respectively. These variables are defined
                in the GenericTest class itself.


          ***/

boolean status = connectToServer (); // user defined

if (status) // connection succeeded

return (100);

else

return (0);

   }

}
```

The above test is an example of a non-descriptor test in the sense that the results of the tests are not specific to a descriptor.

To illustrate how a descriptor based test works, suppose we are monitoring an application that has several instances of servers executing. In this case, it is essential for us to measure and report the availability of each of the server instances. In this case, the test is required to discover the server instances first and then check the availability of each of these instances. For each server instance, the test reports the availability of the instance.

public class AvailabilityTest extends GenericTest

```
{

// declare necessary variables

double availability = 0;

boolean serverInstancesDiscovered = false;

// internal variable that is used to indicate if we

// have already discovered the server instances

String[] servers = null; // used to maintain a list of

// serverInstances


public AvailabilityTest (String [] args)

   {

/*** Calling super initializes several parameters for the test

namely

•         targetHost          -         Target Host for the test

•         portNo              -         Target Port for the test value is "NULL" if its not a
port-based server

•         other parameters configured using the Integration
                *  Console interface

          ***/


super(args);

          /***      Call setMeasureCount to initialize the number of

                    measures for the test

          ***/


setMeasureCount(1); // One measure namely - availability

   }


/*** Call computeMeasures to report measures ***/

public void computeMeasures ( Hashtable paramList )

   {
      if (serverInstancesDiscovered == false)
      discoverServerInstances(paramList); //user defined function

if (servers == null || servers.length == 0)
      return; // unable to discover any servers

for(int i=0; i<servers.length; i++)

         {

                    /***      We may call new methods to perform different

                              tasks. Assume we use a method called

                              getAvailability which performs the availability

                              check and returns the value.

                    ***/

           availability = getAvailability(servers[i]);

           ArrayList al = new ArrayList();

           al.add(new Double(availability));


                    /***      Measures have to be reported by populating an

                              arraylist and passing it as an argument to the

                              addNewMeasure() method. In descriptor based

                              tests and additional parameter defining the

                              descriptor has to be passed to the method to

                              enable the agent report appropriate availability

                              values for different server instances of the

                              application.

                    ***/


      addNewMeasure(servers[i], al);
      // indicate that this is the measure for servers[i]

        }

     }


      // User defined methods
      private double getAvailability (String serverId)

        {

          /***      This is a user-defined implementation. The user can choose one of several
                    approaches to determine a server's availability:


          Eg. 1     Can Establish socket connection to the targetHost and

                    portNo and find out the availability
          Eg. 2     Can ping the targetHost and find out the availability


          Eg. 3     Can establish a HTTP connection to the targetHost and

                    portNo and find out the availability


          Note :    The values of target host and port can be accessed by

                    using the public variables namely "targetHost" and

                    "portNo" respectively. These variables are defined
                in the GenericTest class itself.


          ***/
boolean status = connectToServer (serverId); // user defined

if (status) // connection succeeded

return (100);

else

return (0);

       }



    private void discoverServerInstances (Hashtable paramList)

        {

        /***    Lets assume the discovery is performed by accessing a URL,

                which in turn returns the details of the existing
              server instances. In such a case the test would require a
      parameter ( "URL" in this case ). Such parameters can be
      configured using the Integration Console interface. These
      parameters are accessible from the test at runtime. The
      "paramList" variable which is a java.util.Hashtable object
      provides access to all such variables configured by the
      user.
        ***/
      String param1 = paramList.get("URL");

/***  We may call new methods to perform different tasks. Assume

we use a method called discoverServers which downloads the

               file available at "URL" and parses the response to find

               out the list of currently running server instances.
      ***/


      servers = discoverServers(param1);
      // set the global variable to indicate the list of
      // server instances
      serverInstancesDiscovered = true;
      // indicate that we have discovered the instances
      return;
        }



      public String [] discoverServers(String url)
        {
          /***     User defined Implementation


                   In this example we assume that we discover the server
                   instances by making a url connection to the url
                   configured for this purpose. connection to the
                   targetHost and portNo and find out the availability
          ***/
        }
}
```

A custom test can use various mechanisms to obtain measurements, eg., Processing log files, using sockets, SNMP etc. The following example depicts a test that uses SNMP to monitor a target. Any test that uses SNMP must extend the EgSnmpGenericTest class.

```
            import java.util.*;
                /*     Extend EgSnmpGenericTest instead of GenericTest to
                       implement common functionality of Snmp based tests
                */



                public class SnmpAvailabilityTest extends EgSnmpGenericTest
                {
                       private String OID  = ".1.3.6.1.2.1.4.8.1.1.2";
              private double availability = 0.0;
              public SnmpAvailabilityTest (String [] args)
                       {
```

```
                super(args);

                setMeasureCount(1);


           /* Arguments – snmpPort and snmpCommunity, are required for the test and are
        processed in the super class */


                    }


    public void computeMeasures (Hashtable params)

                    {

                            /*      Call the following method to walk

                                    the specified MIB OIDs. The output of

                                    the snmpwalk command is assigned to the

                                    String arrays lhs and rhs

                                    lhs -> stores left hand side of the output

                                    rhs -> stores right hand side of the output

                            */


                            runSnmpCmdForOid(OID);


                            if(lhs != null && lhs.length > 0)

                            /*      some output is stored ..

                                    the device is available

                            */

            availability = 100.0;

             else

                            {

            /*      No output stored ..

            the device is not available

                                    */

            availability = 0.0;

            ArrayList al = new ArrayList(); al.add(new Double(availability));

            addNewMeasure(al);

                            }

                    }

             }
```

A custom test can also be developed to collect configuration metrics from target components. Given below is a sample custom test script that reports the names of shared folders on a target, the full path to the folders, and the user-defined remarks for each folder.

```
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.StringTokenizer;
import com.egurkha.util.EgUtilities;

/**
•       Measures:
•       1. Resource
•       2. Remark
  */

public class NetShare_cf_ex extends GenericTest
{
public NetShare_cf_ex(String[] args)
        {
super(args);
setMeasureCount(3);
setConfigInfoTestFlag(true); // true for info based
        }

public void computeMeasures(Hashtable ht)
        {
try
                {
EgUtilities egUtil = EgUtilities.createInstance();
String netShareCmd = "net share";
ArrayList data = (ArrayList) egUtil.getExecOutputLines(netShareCmd);
if (data == null || data.size() < 2)
                        {
configError = true; // info based only
return;
                        }

data = (ArrayList) data.get(0);
if (data == null || data.size()  == 0)
                        {
return;
                        }

StringTokenizer st = null;
```

```
ArrayList measureList = null;


int size = data.size();
for (int g = 0; g < size; g++)
                      {
String line = (String) data.get(g);
//System.out.println(line);
if (line == null)
                              {
continue;
                              }
line = line.trim();
if   (line.length()   ==   0   ||   line.startsWith("Share")   ||   line.startsWith("------")   ||
line.startsWith("The command comp"))
                              {
continue;
                              }
st = new StringTokenizer(line);
int count = st.countTokens();
String shareName = "";
String resource = "-";
String remark = "-";
if (count >= 2)
                              {
shareName = st.nextToken().trim();
if (shareName.equalsIgnoreCase("IPC$")) // Remote IPC
                                      {
remark = st.nextToken().trim() + " " + st.nextToken().trim();
                                      }
else
                                      {
resource = st.nextToken().trim();
if (count >= 3)
                                              {
remark = st.nextToken().trim() + " " + st.nextToken().trim();
                                              }
                                      }
                              }
measureList = new ArrayList();
measureList.add(resource);
measureList.add(remark);
addNewMeasure(shareName,measureList);
                      }
```

```
               }
catch (Exception e)
               {
e.printStackTrace();
               }
       }

public static void main(String[] args)
       {
NetShare_cf_ex net = new NetShare_cf_ex(args);
net.computeMeasures(new Hashtable());
       }
}
```

The sample script above is for a descriptor-based configuration test. Non-descriptor-based configuration tests can also be developed. Given below is a sample script for the same. This script simply reports the number of shared folders on a target.

```
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.StringTokenizer;
import com.egurkha.util.EgUtilities;

/**
•       Measure:
•       1. No of share folders
  */

public class NetShareCount_cf_ex extends GenericTest
{
public NetShareCount_cf_ex(String[] args)
       {
super(args);
setMeasureCount(1);
       }

public void computeMeasures(Hashtable ht)
       {
try
               {
EgUtilities egUtil = EgUtilities.createInstance();
String netShareCmd = "net share";
ArrayList data = (ArrayList) egUtil.getExecOutputLines(netShareCmd);
if (data == null || data.size() < 2)
```

```
                       {
return;
                       }

data = (ArrayList) data.get(0);
if (data == null || data.size() == 0)
                       {
return;
                       }

StringTokenizer st = null;
ArrayList measureList = new ArrayList();
int k = 0;
int size = data.size();
for (int g = 0; g < size; g++)
                       {
String line = (String) data.get(g);
if (line == null)
                              {
continue;
                              }
line = line.trim();
if  (line.length()  ==  0  ||  line.startsWith("Share")  ||  line.startsWith("------")  ||
line.startsWith("The command comp"))
                              {
continue;
                              }
++k;
                       }
measureList.add(k+"");
addNewMeasure(measureList);
               }
catch (Exception e)
               {
e.printStackTrace();
               }
       }
public static void main(String[] args)
       {
NetShareCount_cf_ex net = new NetShareCount_cf_ex(args);
net.computeMeasures(new Hashtable());
       }
}
```

2.1.3.5 Writing Detailed Diagnosis Tests

This section outlines how to write detailed diagnosis tests for the user-defined classes that extend eG's functionality.

The Java class that implements detailed diagnosis for a test must be named after the test and be suffixed by an **_ex_DD**. Such tests extend **EgTest_DD** and must include a method for every measure for which detailed diagnosis is desired. The name of the class is case-sensitive.

For the **MsFileTest_ex** in our example, the Java class that implements detailed diagnosis should be called **MsFileTest_ex_DD**. Since detailed diagnosis has been configured for the measure **File_locks_count**, a method with the name **File_locks_count_dd** has to be implemented. Like the class name, the method name is also case-sensitive.

Given below is a sample test that gives a detailed diagnosis on the files opened for **MsFileTest_ex**:

```
/*

        Detailed diagnosis for MS_FILE_SERVER_ex component type is enabled for the measure
            File_locks_count

 */

import java.util.*;

/*

        Tests for Detailed Diagnosis must by suffixed by _ex_DD.

 */

public class MsFileTest_ex_DD extends EgTest_DD

{

        /*

        Declare the necessary variables

         */

private boolean uploadStatus = false;

public MsFileTest_ex_DD(String entity)

        {
      super(entity);

        }

        /*

        For every measure that requires detailed diagnosis,

        add a method, the name of which is of the type <measureName>_dd

         */

public void File_locks_count_dd(

            String methodName,

            String measureName,

            String targetHost,

            String reportingName,
            String portNo,
            String siteName,
            String info,
            String msmtHost,
            String msmtTime,
            String state,
            EgTest test)
        {
      String ddData;


                /*
        The data that is to be displayed as part of detailed diagnosis is collected using the
             * addLine() method.
                 */
this.addLine(ddData);
                /*
        Once all the data for detailed diagnosis has been added,
        send the results to the manager using the uploadResults() method
                 */
uploadStatus = this.uploadResults(
            methodName,
            measureName,
            reportingName,
            portNo,
            siteName,
            info,
            msmtHost,
            msmtTime,
            state);
                }
        }
}
```

Note

Though any number of lines of data can be added using the addLine() method, it is recommended that the implementation of the detailed diagnosis does not send large amounts of data to the manager everytime.

2.1.3.6 Troubleshooting

To troubleshoot problems when a user-defined test is integrated into the eG Enterprise system, check the entries in the agent error log, which is available in the file `<EG_HOME_DIR >\agent\logs\error_log.` For entries specific to a user-defined test to be available in the error log, it is necessary for the test to log errors explicitly.

2.2 Adding a Script/Batch File-based Test

Many system administrators would not prefer to write **Custom** tests, by programming in Java. To allow administrators to easily extend eG's capabilities, the Integration Console includes a **Script/Batch File** test. This test allows an administrator to simply write a shell script / batch file / VB script / powershell script, which they can incorporate into the Integration Console to provide custom monitoring capability. This section will help you gain a lucid understanding of how to add and configure a new test of the **Script/Batch File** type.

To illustrate how to add a **Script/Batch File** test, let us take an example of a **DiskSpaceTest**, which tracks the disk utilization of a server. In order to create the test, first, click the **Add New Test** button in the **INTEGRATION CONSOLE - TEST** page (see Figure 2.2). In the **NEW TEST DETAILS** page (see Figure 2.34) that appears next, specify the **Test name**.

Note

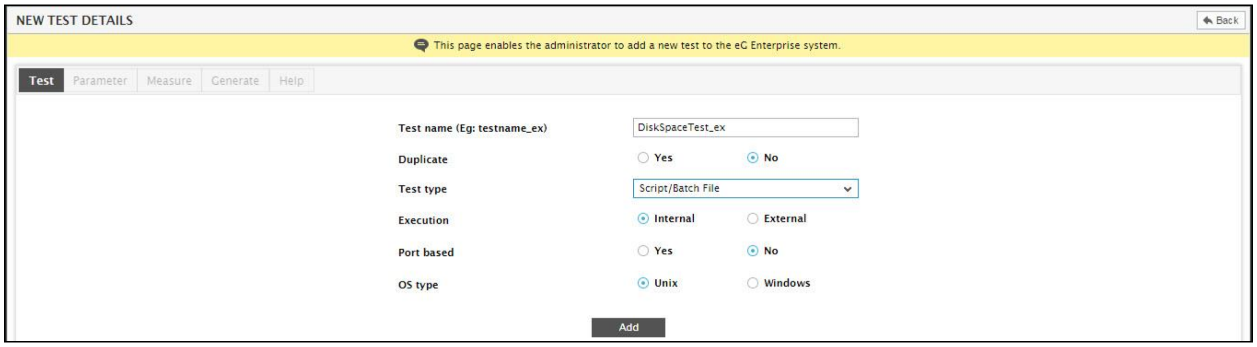
While adding a new test using the Integration Console, ensure that the **Test name** always ends with **_ex**. If not, an error message (see Figure 2.3) will appear upon clicking the **Add** button in Figure 2.34.

Since the new test is not a duplicate of any existing test, set the **Duplicate** flag to **No**. Then, select **Script/Batch File** as the **Test type**. Next, mention the **Execution** mode. The choice of an **Execution** mode depends upon whether the test is to be executed by an internal agent or an external agent. Since the **DiskSpaceTest_ex** is to be run only by an internal agent, select the **Internal** option against the **Execution** field.

Note

Using the Integration Console plugin, you can add an internal or an external test, but you cannot add tests that need to be run by a remote agent – i.e., tests that need to be executed in an agentless manner.

Next, as the **DiskSpaceTest_ex** is to be run at the system level, select **No** against **Port based** to enable it to be associated with any of the component types.



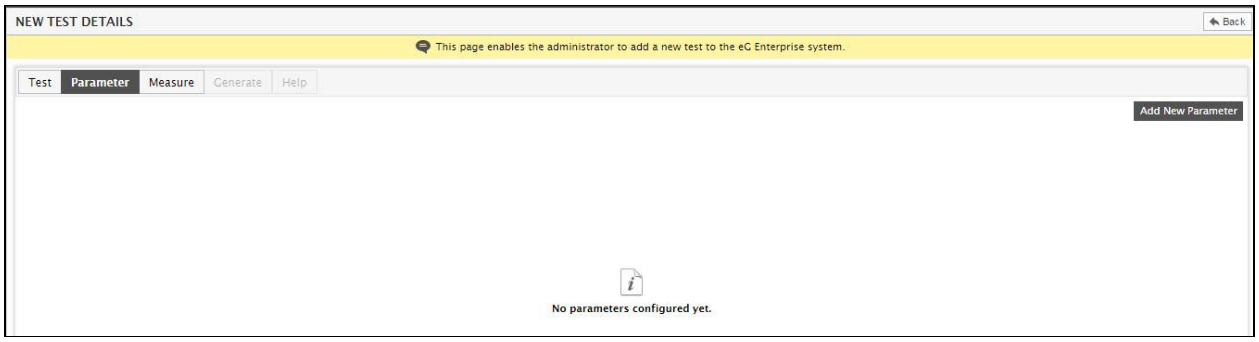
This page enables the administrator to add a new test to the eG Enterprise system.

Test name (Eg: testname_ex)	DiskSpaceTest_ex
Duplicate	Yes [✓ No]
Test type	Script/Batch File
Execution	[✓ Internal] External
Port based	Yes [✓ No]
OS type	[✓ Unix] Windows

Figure 2.34: Providing the details of the new test of type Script/Batch File

If the test is to be associated with a shell script file, choose **Unix** as the **OS type**. If the test is to be associated with a batch file/VB/powershell script, choose the **Windows** option. Our **DiskSpaceTest_ex** is associated with a script, and hence, the **Unix** option needs to be chosen. Then, click the **Add** button.

The Parameter tab page will automatically open (see Figure 2.35). To add a new parameter, click the **Add New Parameter** button. However, as the **DiskSpaceTest_ex** does not take any parameters, click the **Measure** tab page in Figure 2.35.

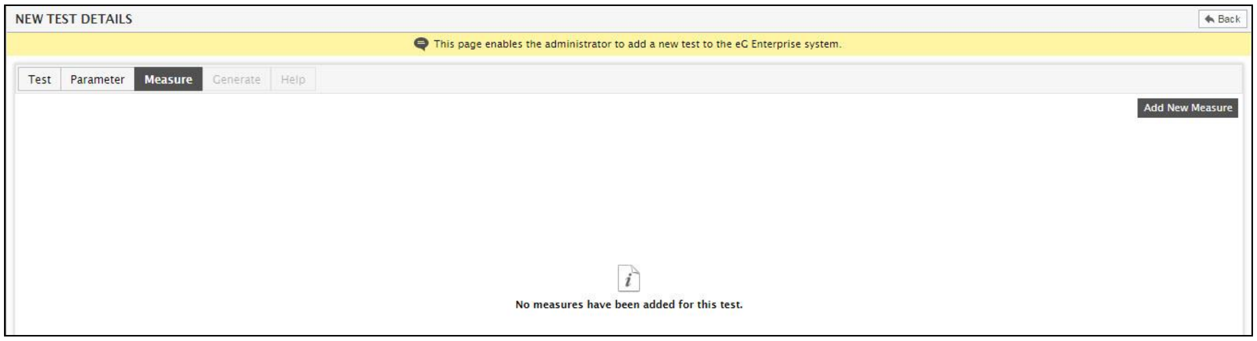


This page enables the administrator to add a new test to the eG Enterprise system.

No parameters configured yet.

Figure 2.35: The Parameter tab page of the DiskSpaceTest_ex

Click the **Add New Measure** button in the **Measure** tab page of Figure 2.36 to add a measure for the **DiskSpaceTest_ex**.

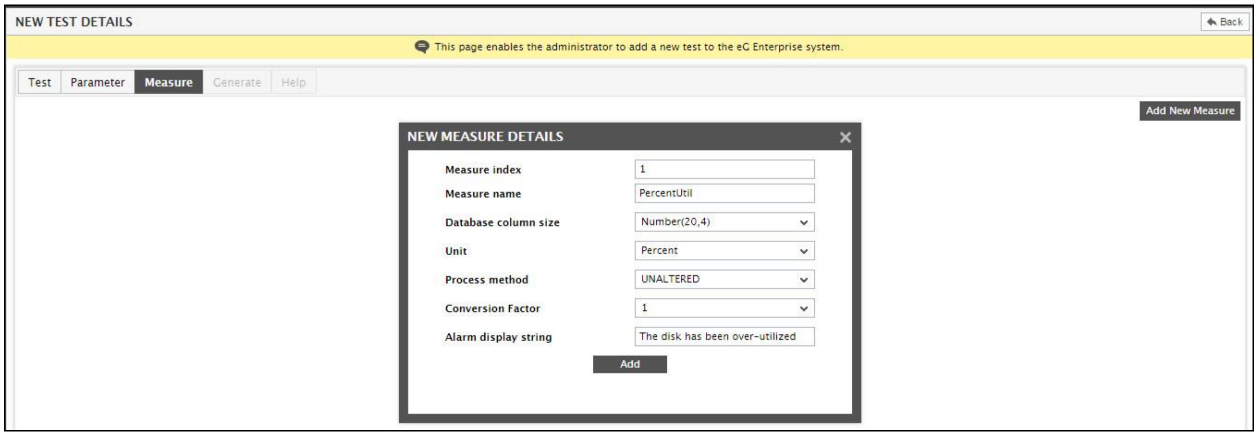


This page enables the administrator to add a new test to the eG Enterprise system.

No measures have been added for this test.

Figure 2.36: The Measure tab page of the DiskSpaceTest_ex

Figure 2.37 will then appear, using which you can configure the new measure for the **DiskSpaceTest_ex**. As the test should indicate the percentage of disk space utilized, add **PercentUtil** as the only measure of this test. Therefore, specify **PercentUtil** as the **Measure name**. Then, specify the **Database column size**, the **Unit**, **Conversion Factor**, and the **Alarm display string** in the same manner as discussed in the previous section.



Measure index	1
Measure name	PercentUtil
Database column size	Number(20,4)
Unit	Percent
Process method	UNALTERED
Conversion Factor	1
Alarm display string	The disk has been over-utilized

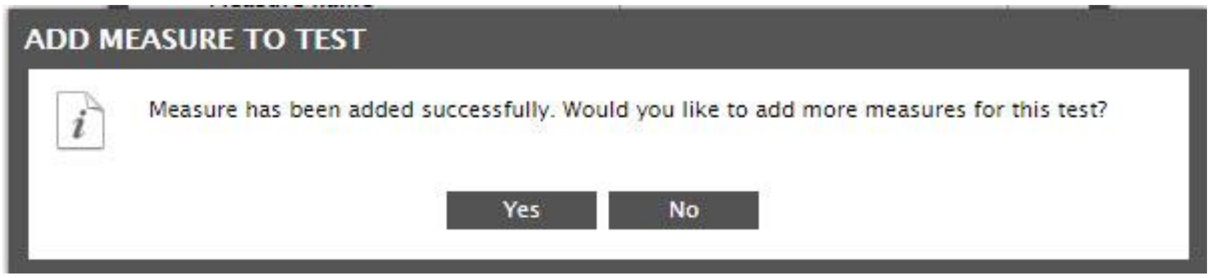
Figure 2.37: Specification of a measure (PercentUtil) of the DiskSpaceTest_ex

In addition to the above, this test type requests the specification of a **Process method**. By selecting an option from the **Process method** list box, you can indicate the processing that must be performed on the script / batch file's output before passing the results to the eG agent. The options offered by this list box are:

Option	Information
UNALTERED	Selecting this option will ensure that no additional information is displayed along with the specified measure in the monitor console. In other words the **Current Value** of the measure is displayed.
PERCENT_INCREASE	Selecting this option will display the percentage by which the current value of the measure exceeds its previous value (i.e. the value of the measure during the previous test execution). While an increase displays a positive value, a decrease will display a negative value. The formula used is: **[(Current Value – Previous Value) / Previous Value] * 100**
PERCENT_DECREASE	This option will display the percentage by which the current value of the measure falls below its previous value. While a decrease displays a positive value, an increase will display a negative value. **[(Previous Value – Current Value) / Previous Value] * 100**
PERCENT_CHANGE	This displays the percentage change between the current value and the previous value. **ABS[(Current Value – Previous Value) / Previous Value] * 100**
RATIO	This displays the ratio of the current value of the measure over its previous value. **(Current Value / Previous Value)**
RATE	This displays the result of the following formula: **(Current Value - Previous Value) / Time since the last measurement**
DIFFERENCE	This displays the absolute value of the difference between the current value and the previous value. **ABS(Current Value – Previous Value)**

By default, **UNALTERED** is selected as the **Process method**. In the **DiskSpaceTest_ex** example, no further processing is required. Hence, the default selection is left as is. Finally, click the **Add** button to add the measure.

When you are prompted to add more measures for the test (see Figure 2.38), click **No** to stop adding any more measures; this is because, the **DiskSpaceTest_ex** in our example reports only one measure.

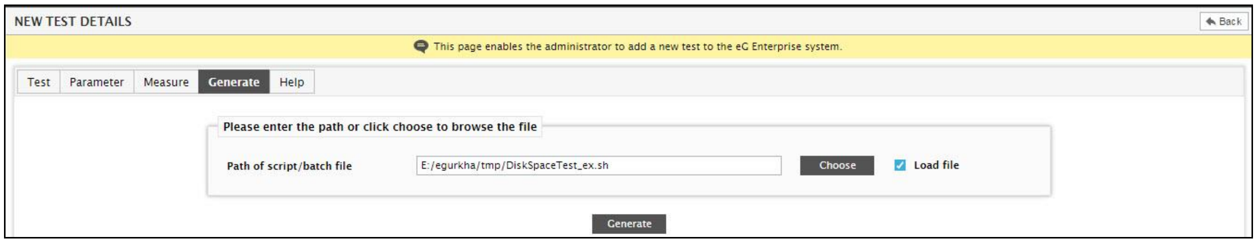


ADD MEASURE TO TEST

Measure has been added successfully. Would you like to add more measures for this test?

Figure 2.38: A message box requesting your confirmation to continue adding measures for the NetShare_cf_ex test

Doing so will instantly open the **Generate** tab page (see Figure 2.39). Here, proceed to specify the **Path of script/batch File** (see Figure 2.39) associated with the test. Multiple script / batch file paths can also be specified, but remember to separate each entry using a comma.



NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Please enter the path or click choose to browse the file

Path of script/batch file: E:/egurkha/tmp/DiskSpaceTest_ex.sh

Load file ✓

Figure 2.39: Generating a test of type Script/Batch File

Note

While associating a test with multiple script / batch files, make sure that the "main" script that needs to be executed is specified last. Otherwise, no measures will be reported by the eG Enterprise system.

Any script used for a **Script/Batch File** test should provide one or more lines of output. If a test is descriptor-based, it can have multiple lines of output, with the first entry of each line being the descriptor. In case of scripts other than VB/powershell scripts, the descriptor and its corresponding measures are separated by whitespaces (space or tab), as shown below:

Output of a (non-VB/non-powershell) script for a descriptor-based test

```
DESC 1          Value 1            Value 2       .      Value N

DESC 2          Value 1            Value 2       .      Value N

    .

    .

DESC N          Value 1            Value 2       .      Value N
```

In case of VB/powershell scripts on the other hand, the descriptor and its corresponding measures are separated by ":", as shown below:

Output of a VB/powershell script for a descriptor-based test

```
DESC 1:Value 1:Value 2:…:Value N

DESC 2:Value 1:Value 2: …:Value N

    .

    .

DESC N:Value 1:Value 2:…:Value N
```

The **DiskSpaceTest_ex** we have configured is a descriptor-based test and hence, its output would be of the following format (if the script associated with this test is not a VB/powershell script):

```
/tmp      30

/boot     22

/usr      12
```

In the above output, **/tmp**, **/boot** and **/usr** are some of the descriptors for the **DiskSpaceTest_ex** and 30, 22 and 12 are the values of the **PercentUtil** measure of each of these descriptors. For example, for the **/tmp** descriptor, the disk space utilized is 30%.

If a test is not descriptor-based, then the script should report only one line of output. The first entry of this line should be "NONE".

Output of a (non-VB/non-powershell) script for a non-descriptor-based test

```
NONE                      Value 1               Value 2       .      Value N
```

Output of a VB/powershell script for a non-descriptor-based test

```
ine ine ntat

NONE:Value 1:Value 2:…: Value N
```

If the **DiskSpaceTest_ex** we have configured is a non-descriptor-based test, its output would be of the following format (provided, a VB/powershell script is not associated with this test):

```
NONE   30
```

A script on Linux would look like this:

```
#!/bin/sh

df -k | grep "/" | awk '{print $6 "      " $5 -1}'
```

A sample powershell script has also been provided below:

```
$services = get-service
foreach($service in $services)
{
$name = $service.displayname
$status = $service.status
if($status -eq "Running")
       {
$value = 100
```

```
       }
else
       {
$value = 0
       }
Write-Host($name,$value)  -Separator “:”

}
```

The above script implements a descriptor-based test. This script reports the status of each of the services available on a Windows host.

Find below a sample non-descriptor-based powershell script:

```
$x = Get-Random -minimum 50 -maximum 101
$y = Get-Random -minimum 25 -maximum 50
$z = $x + $y
$w = $x - $y
Write-Host($x,$y,$z,$w)  -Separator “:”
```

The above script reports random I/O-related measurements pertaining to a target Windows host.

Note

Powershell scripts can be executed on only those targets that have Powershell SDK v1/v2 installed.

All the scripts sampled above (the Linux script and the powershell scripts) did not take any arguments. To offer more flexibility in script execution, the Integration Console allows a user to specify multiple arguments for a script/batch file test. When the script/batch file is executed each time, the test’s arguments are passed to the script/batch file. Note that the arguments are typically passed to a script with a hyphen (i.e., ‘-‘) preceding them. Each argument is expected to be followed by its value (e.g., -argument1 <argument1Val> -argument2 <argument2Val>). The script/batch file has to parse the arguments that are passed to it at the time of invocation and perform the appropriate functions. The following example provides an illustration of how a Linux shell script can parse the arguments provided to it:

```
#!/bin/sh
# This is an example of a simple script that processes its arguments.

# This script takes two arguments and outputs the values of the arguments.

out1=””;
out2=””;
#out1 and out2 are output variables

while [ $# -ge 1 ]

do

case $1 in

•      argument1) shift; out1=$1;;  # if the current argument is argument1

•      argument2) shift; out2=$1;;  # if the current argument is argument2

esac

shift;
```

```
done
echo "NONE $out1 $out2"
```

If the script files are present in a remote location, then you can upload them to the eG manager, by clicking on the **Choose** button adjacent to the **Path of script/batch File** text box in Figure 2.39. This will invoke a pop-up window using which you can **Browse** for the files and specify their location. Finally, click on the **Upload** button in the pop-up window, to upload the script/batch files in the remote location to the eG manager. However, if the files have already been uploaded to the eG manager, then this procedure can be dispensed with. Instead, just specify the location of the files against the **Path of script/batch File** text box.

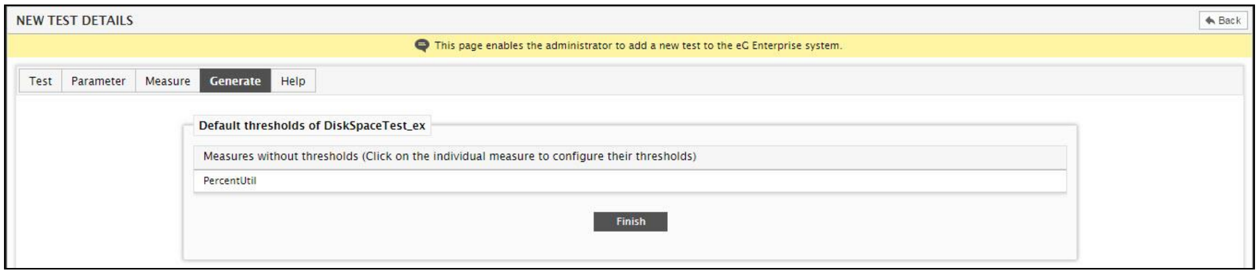
Select the **Load file** (see Figure 2.39) check box if:

- The specified script/batch file has been modified, or
- The measures are being associated with the script/batch file for the very first time

If the script/batch file has changed, then selecting the **Load file** check box will ensure that when the test runs, the agent downloads the revised version of the file from the manager, and executes the same. This in turn ensures that the changes take effect. While associating the script/batch file with the test for the first time, it is mandatory to select this check box, as doing so enables the agent to download the file along with the measurements during test execution.

Finally, click the **Generate** button in Figure 2.39 to generate the test.

When a test's measurements are successfully configured and the associated script/batch file has been added to the eG Enterprise system, the eG Enterprise system prompts you to specify the default threshold settings for each of the measurements made by the newly added test (see Figure 2.40). Click on the measure name in Figure 2.40 to configure its thresholds.



This page enables the administrator to add a new test to the eG Enterprise system.

Default thresholds of DiskSpaceTest_ex

Measures without thresholds (Click on the individual measure to configure their thresholds)

PercentUtil

Figure 2.40: Specifying the threshold values of the measures of the DiskSpaceTest_ex

2.3 Adding an SQL Query/Stored Procedure-based Test

Many applications store critical statistics in a database. To simplify the writing of tests, the Integration Console includes an **SQL Query** test type that allows a user to include a new test that retrieves measures by simply executing a sql query / stored procedure on the database, instead of writing elaborate java code.

2.3.1 Using a SQL Query

To understand this concept better, consider an example. In this example, a new test named **SqlTest** of type **Sql Query** will be created, which will be configured to measure the number of current users to a custom application. This application is executing on a host with IP address 192.168.10.8, and uses an Oracle database executing on the same system (192.168.10.8) as its back end. This example will demonstrate how application-specific metrics stored in the database can be retrieved using a SQL query and integrated into the eG Enterprise system.

Adding/Modifying Tests using the Integration Console

To add this test, first, click the **Add New Test** button in the **INTEGRATION CONSOLE - TEST** page (see Figure 2.2). In the **NEW TEST DETAILS** page (see Figure 2.41) that appears next, specify the following:

- **Test name**: SqlTest_ex

Note

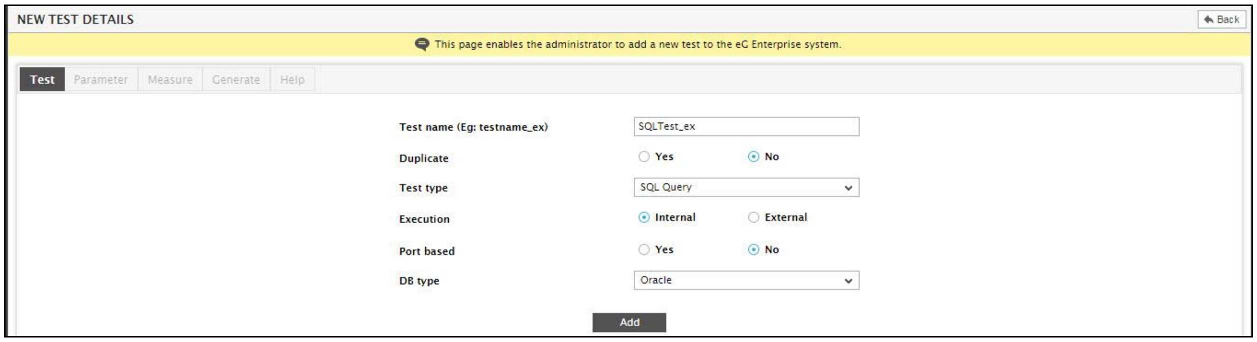
While adding a new test using the Integration Console, ensure that the **Test name** always ends with **_ex**. If not, an error message (see Figure 2.3) will appear upon clicking the **Add** button in Figure 2.41.

- **Duplicate**: Since the new test is not a duplicate of any existing test, set the **Duplicate** flag to **No**.
- **Execution**: Internal, as the test is to be executed by an internal agent

Note

Using the Integration Console plugin, you can add an internal or an external test, but you cannot add tests that need to be run by a remote agent – i.e., tests that need to be executed in an agentless manner.

- **Port based**: Since the **Webmall** application is not listening on a specific TCP port, select **No** against the **Port based** input selection.
- **Test type**: Sql Query
- **DB type**: If the query is to be executed on an Oracle database, select **Oracle** from this list box. On the other hand, if the query is to be executed on an MS SQL database, select **MsSql** from the list box. Similarly, if the query is to be retrieve data from a Sybase, MySql, DB2, or a PostgreSQL database, then, choose **Sybase**, **MySql**, **DB2**, or **PostGres** from the list box. For our example, select **Oracle**. Then, click the **Add** button.

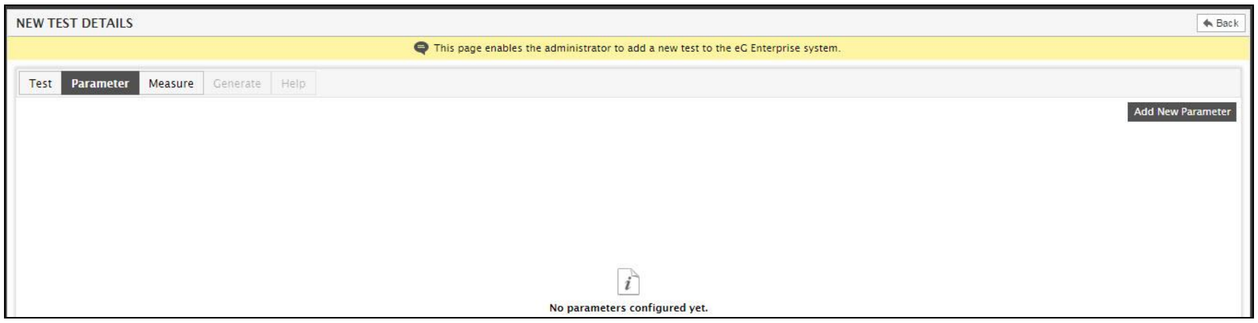


This page enables the administrator to add a new test to the eG Enterprise system.

Test name (Eg: testname_ex)	SQLTest_ex
Duplicate	Yes [✓ No]
Test type	SQL Query
Execution	[✓ Internal] External
Port based	Yes [✓ No]
DB type	Oracle

Figure 2.41: Providing the new test details

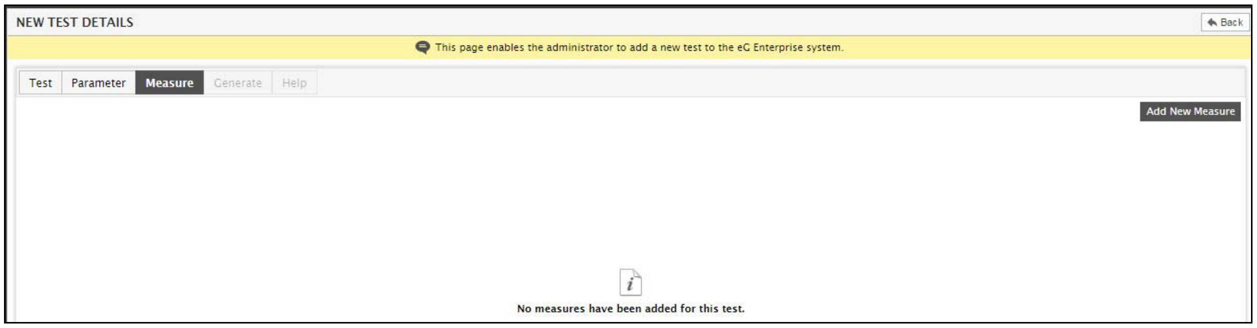
Next, using Figure 2.42, the parameters to the test need to be specified. To add the new parameter, click the **Add New Parameter** button in Figure 2.42.


NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Add New Parameter

No parameters configured yet.

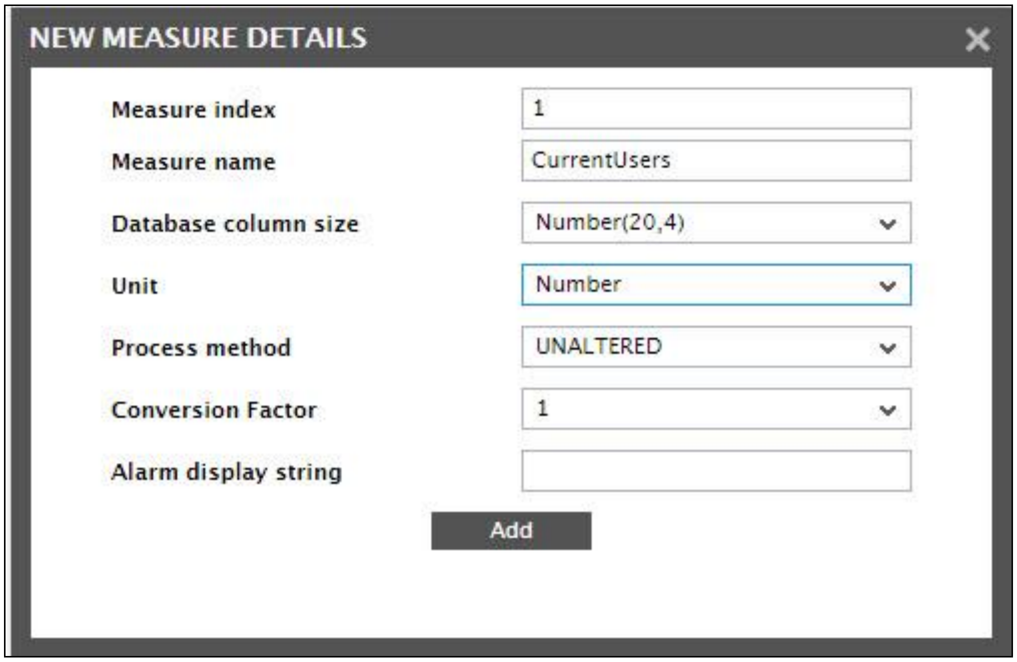
Figure 2.42: Modifying the details of the SqlTest_ex

As the **SqlTest_ex** will not be taking any parameters, simply proceed to configure the measures for this test by clicking the **Measure** tab page in Figure 2.42. When Figure 2.43 appears, click the **Add New Measure** in Figure 2.43 to open the **NEW MEASURE DETAILS** pop-up (see Figure 2.44).


NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Add New Measure

No measures have been added for this test.

Figure 2.43: The Measure tab page reporting that no measures have been configured for the SQL query-based test



Measure index	1
Measure name	CurrentUsers
Database column size	Number(20,4)
Unit	Number
Process method	UNALTERED
Conversion Factor	1
Alarm display string	

Figure 2.44: Adding the CurrentUsers measure for the SQL query-based test

Figure 2.44 shows how a measurement of the **Sql Query** test type is specified using the **NEW MEASURE DETAILS** pop-up. To generate measures pertaining to the number of users, create a measure named **CurrentUsers**. Accordingly, specify the **Measure name**, the **Database column size**, the **Unit**, **Conversion Factor**, and the **Process method** (see Figure 2.44).

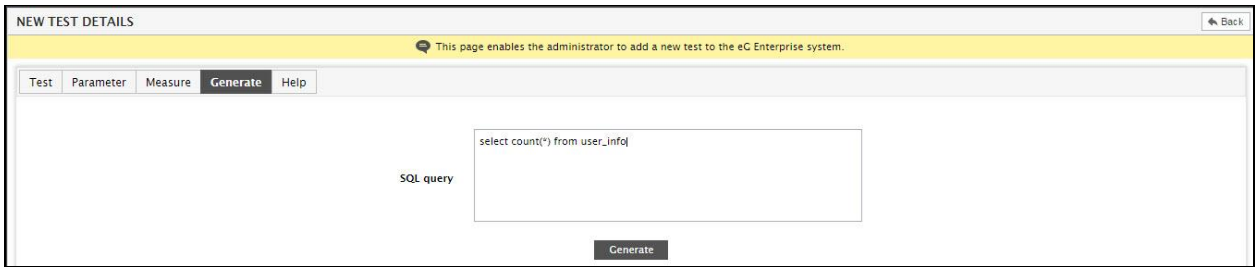
Refer

To know more about the **Process method**, refer to Page 41 of this document.

To know more about the **Conversion factor**, refer to Page 12 of this document.

Since we do not want to associate an alarm description with this measure, leave the **Alarm display string** field blank (see Figure 2.44).

Finally, click on the **Add** button in Figure 2.44 to add the measure. When prompted to add more measures for the **SQLTest_ex**, click **No** to stop configuring any more measures (see Figure Figure 2.10). Clicking **No** will instantly lead you to the **Generate** tab page (see Figure 2.45).


NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
SQL query
Generate

```
select count(*) from user_info
```

Figure 2.45: Specifying the Sql query associated with the SqlTest_ex

In Figure 2.45, specify the **SQL query** that will, on execution, fetch the number of current users. Finally, click the **Generate** button to integrate the test's implementation into the eG Enterprise system. You can even click on the **Add Help** button therein to create and upload **Admin** and **Monitor** help pages for the new test. To know how, refer to Section 2.1.2 of this document.

An **Sql Query** test can be both descriptor-based and non-descriptor based. For example, the following query returns a descriptor-based ouput. The query retrieves from a table named *metatest* the number of records that carry the same value in a field named *info*.

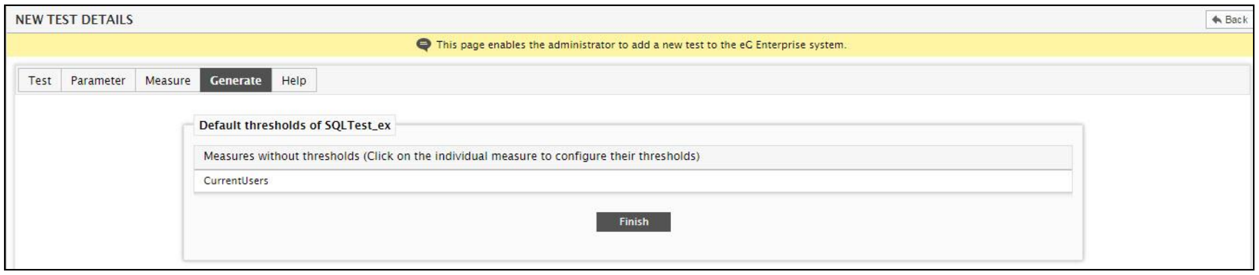
```
select substr(info, 2), count(*) from metatest where info<>'+' group by info
```

For a descriptor-based test, the first value of the results of the query must be a string. The other results should be integer or double values. If the first value of a query's result is not a string, the test is not descriptor-based and only the first row of the result set will be used.

An Sql query can return multiple outputs. For example:

```
select read_rate, write_rate from disktest where msmt_time=(select max(msmt_time) from
disktest) and info='+/'
```

When a test's measurements are successfully configured, the eG Enterprise system prompts the user to specify the default threshold settings for each of the measurements made by the newly added test (see Figure 2.46).


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Default thresholds of SQLTest_ex

Measures without thresholds (Click on the individual measure to configure their thresholds)
CurrentUsers

Figure 2.46: Specifying the threshold values of the CurrentUsers measure

2.3.2 Using Stored Procedure

Typically, large or complex processing that might require the execution of several SQL statements is moved into stored procedures. You might choose a stored procedure over a SQL query, if:

- you want to execute multiple SQL queries, simultaneously
- you not only want to query metrics from the database, but also intend to perform mathematical computations on the result set and display the net output in the eG monitor interface.

To help you clearly understand how a stored procedure can be used to build a test's functionality, let us take another example. In this example, we would be attempting to create an 'info-based' test, which will execute a stored procedure on an MS SQL server database; this stored procedure will take a **HOST IP** from the user, calculate the average CPU utilization of every processor on the given **HOST**, and report the computations to the eG manager.

To add the new test, first click the **Test** option in the **Integration Console** tile. Then, click on the **Add New Test** button in the **INTEGRATION CONSOLE - TEST** page that appears (see Figure Figure 2.2) that opens next. In the **NEW TEST DETAILS** page (see Figure 2.47), specify the following:

- **Test name**: AvgCpuUtilTest_ex

Note

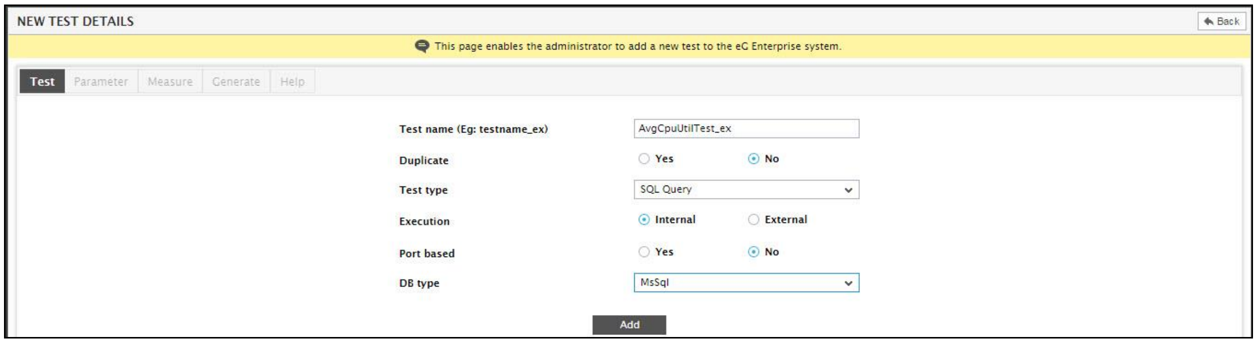
While adding a new test using the Integration Console, ensure that the **Test name** always ends with **_ex.** If not, an error message (see Figure 2.3) will appear upon clicking the **Add** button in Figure 2.47.

- **Duplicate:** Since the new test is not a duplicate of any existing test, set the **Duplicate** flag to **No**.
- **Execution**: Internal, as the test is to be executed by an internal agent

Note

Using the Integration Console plugin, you can add an internal or an external test, but you cannot add tests that need to be run by a remote agent – i.e., tests that need to be executed in an agentless manner.

- **Port based:** Select **No** against the **Port based** input selection.
- **Test type:** Sql Query
- **DB type:** For our example, select **MsSQL** from the list box. The other options are **Oracle**, **Sybase**, **MySql**, **DB2**, and **PostGres**.

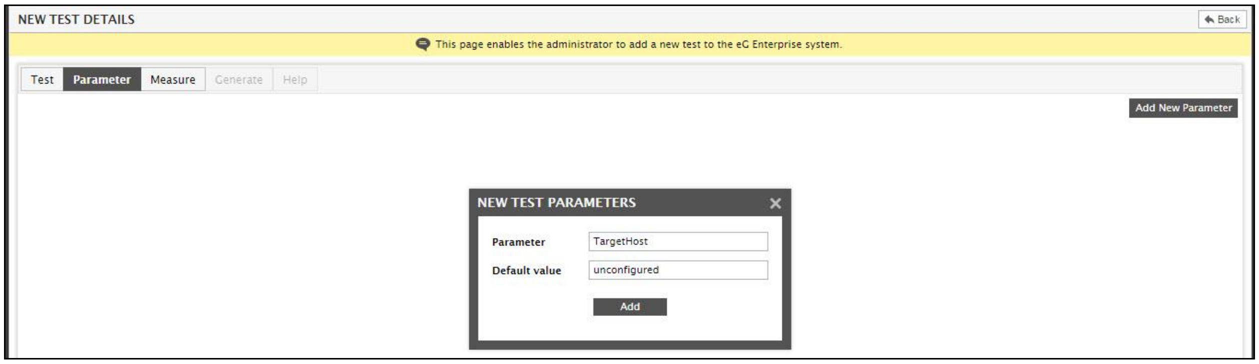

NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Test name (Eg: testname_ex)	AvgCpuUtilTest_ex
Duplicate	Yes [✓ No]
Test type	SQL Query
Execution	[✓ Internal] External
Port based	Yes [✓ No]
DB type	MsSql

Figure 2.47: Providing the details of the SQL stored procedure-based test

Once the test details are specified in the **TEST** tab page of Figure 2.47, click the **Add** button to add the new test. This will automatically take you to the **Parameter** tab page, where the parameters to the test need to be specified. To add new test parameters, click the **Add New Parameter** button in Figure 2.48. This will invoke the **NEW TEST PARAMETERS** pop-up (see Figure 2.48).


NEW TEST DETAILS

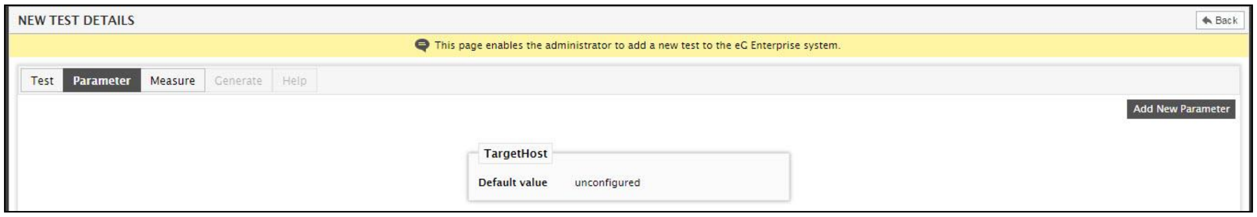
This page enables the administrator to add a new test to the eG Enterprise system.

NEW TEST PARAMETERS

Parameter	TargetHost
Default value	unconfigured

Figure 2.48: Adding a new test parameter for the SQL stored procedure-based test

Since the **AvgCpuUtilTest_ex** takes the IP address of a monitored host as its parameter, specify **TargetHost** against **Parameter**, and click the **Add** button in the **NEW TEST PARAMETERS** pop-up. eG will now request you to confirm whether/not you want to add more parameters to the test. As the **AvgCpuUtilTest_ex** in our example does not take any more parameters, click **No** in the message box to stop adding parameters. You will now return to the **Parameter** tab page, where you can quickly review your parameter settings (see Figure 2.49).


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

TargetHost

Default value	unconfigured

Figure 2.49: Reviewing the parameter specification of the SQL stored procedure-based test

Then, click the **Measure** tab page to configure the measures of the test. Figure 2.50 will then appear.

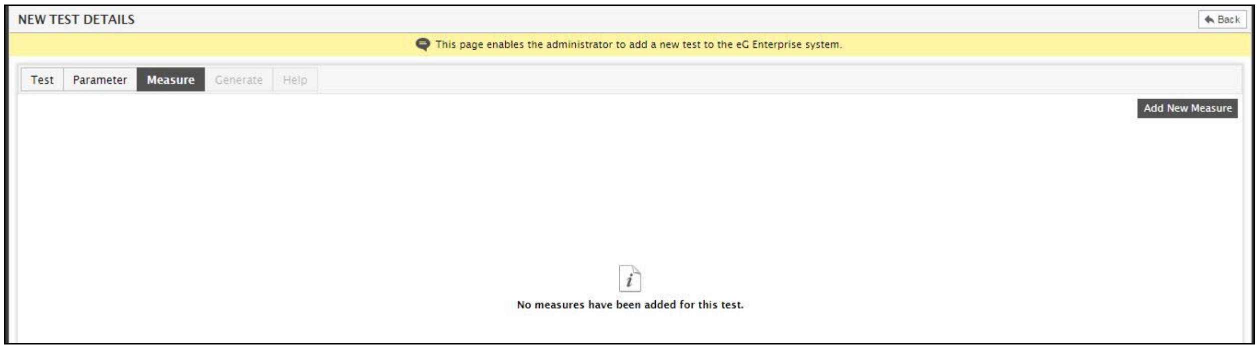

NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Add New Measure
No measures have been added for this test.

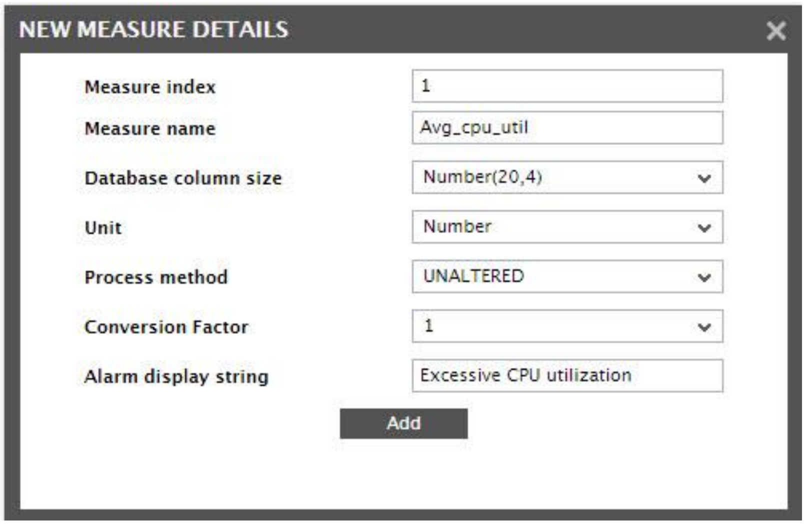
Figure 2.50: The Measure tab page indicating that no measures have been configured yet for the SQL stored procedure-based test

Click the **Add New Measure** button to add a new measure for the test. The **NEW MEASURE DETAILS** (see Figure 2.51) will then pop up. To generate a measure that indicates the average CPU utilization of a processor, create a measure with the **Measure name**, **Avg_cpu_util**. Also provide the **Database column size**, **Unit**, **Conversion Factor** and the **Process method** specifications as indicated by Figure 2.51.

Refer

To know more about the **Process method**, refer to Page 41 of this document.

To know more about the **Conversion factor**, refer to Page 12 of this document.

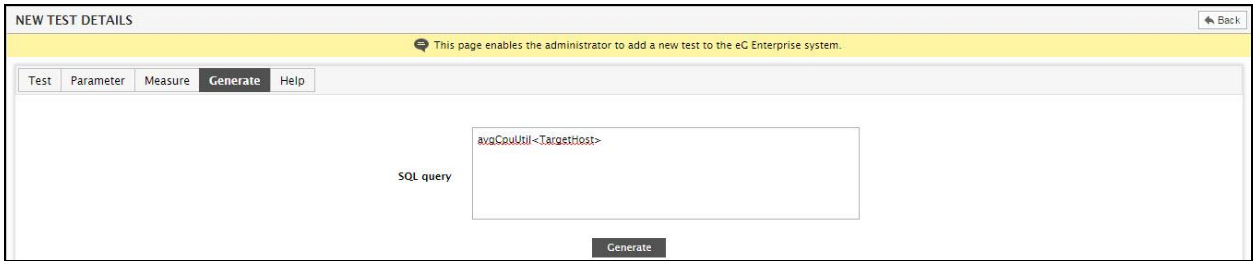

NEW MEASURE DETAILS

Measure index	1
Measure name	Avg_cpu_util
Database column size	Number(20,4)
Unit	Number
Process method	UNALTERED
Conversion Factor	1
Alarm display string	Excessive CPU utilization

Figure 2.51: Adding the Avg_cpu_util measure of the SQL stored procedure-based test

Likewise, specify an **Alarm display string** similar to the one provided in Figure 2.51.

After specifying all the required details, click on the **Add** button in Figure 2.51 to add the measure. You will then be prompted to indicate whether/not you want to continue adding measures for the test. Since the **AvgCpuUtilTest_ex** in our example does not report any more measures, click **No** against the prompt to stop adding more measures. This will lead you to Figure 2.52.


NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Generate

SQL query: avgCpuUtil<TargetHost>

Figure 2.52: Specifying the stored procedure associated with the Ag_cpu_util measure

Against the **SQL query** field in Figure 2.52, issue the command for invoking a stored procedure named **avgCpuUtil**, and click the **Generate** button (see Figure 2.52).

The **avgCpuUtil** stored procedure has been specifically created for the purpose of our example, and performs the following tasks:

- Retrieves the CPU utilization metrics for every processor that a specified host supports, from the **systemtest** table in the MS SQL database
- Computes the average of the CPU utilization metrics per processor

Typically, the syntax for the command to be issued to execute a stored procedure is: *StoredProcedurename*. In our example however, the stored procedure accepts a **Host IP** from the user and retrieves the CPU usage statistics that correspond to the given IP address. To execute a stored procedure that supports input parameters/arguments (such as the one in our example), you should use the command: *StoredProcedureName <<Argument>>*. In the case of our example therefore, the command would be: **avgCpuUtil <<TargetHost>>**, where **avgCpuUtil** is the name of the stored procedure, and **TargetHost** is the name of the parameter that the procedure supports.

Note

The arguments/parameters that are passed to a stored proocedure are case-sensitive, and should always be enclosed within angular brackets (<<>>). This implies that the *Argument* provided in the command should be of the same case as the parameter configured for the **AvgCpuUtilTest_ex** in Figure 2.48. Therefore, the parameter **TargetHost** should be expressed as **<<TargetHost>>** in the command.

A stored procedure that is executed on an MS SQL database can take any number of arguments, and returns a result set. A result set with multiple columns, where the first column contains character values, is said to be 'info-based'. On the other hand, if a result set consists of multiple columns, and all columns support only numeric values, then such a result set is said to be 'non-info-based'.

A Stored Procedure on MS SQL that returns an 'Info-based' result set:

An info based test will typically return multiple rows of output, with each row representing the metrics for a particular info. A non-info based test, on the contrary, will always have a single row of output.

In case of a non-info-based test therefore:

The total number of measures for the test = The total number of columns returned by the query

In case of an info-based test:

The total number of measures for the test = (The total number of columns in the query output) - 1.

The first column of an info-based result set represents the name of the info.

Since the stored procedure in our example needs to return one set of measures for every processor supported by a given **TargetHost**, it should return an info-based result set. Given below is the stored procedure, **avgCpuUtil**, which has been created on the MS SQL server database for the purpose of our example:

```
CREATE PROCEDURE avgCpuUtil @host varchar(30)
   as
    SELECT 'Processor_'+info, avg(cpu_util) Avg_cpu_util
        FROM systemtest
        WHERE trgt_host=@host
        GROUP BY info
        ORDER BY info
```

Note that the stored procedure takes the argument, **@host**. You can see that the value for this argument is matched with the value of the **trgt_host** column in the **systemtest** table. **trgt_host** is a column in the **systemtest** table, which holds the IP address of the monitored hosts. While configuring the **AvgCpuUtilTest_ex** using the eG administrative interface, you will be required to pass a value to the **TARGETHOST** parameter of the test. When the stored procedure executes, it assigns the **TARGETHOST** value to the **@host** argument, and then compares the **@host** value with the values in the **trgt_host** column. Once a match is found, the procedure retrieves the processor names and CPU usage statistics that correspond to that **trgt_host** from the **systemtest** table, computes the average CPU usage for every processor, and returns the resultant value to the alias, **Avg_cpu_util** - this is nothing but the measure that we had configured in Figure 2.51.

A Stored Procedure on MS SQL that returns a 'Non-info-based' result set:

Let us also see how a non-info-based stored procedure is to be constructed. When a stored procedure returns a result set comprising of multiple columns, all of which contain only numeric values, then this is a 'non-info-based' stored procedure. For example, assume that you need to create a stored procedure that computes the average CPU utilization of a host across processors (and not per processor). Such a stored procedure is 'non-info-based', and can be coded as follows:

```
CREATE PROCEDURE avgCpuUtil @host varchar(30)
   as
    SELECT avg(cpu_util) Avg_cpu_util
        FROM systemtest
        WHERE trgt_host=@host
```

A Stored Procedure on Oracle:

The broad steps that you should follow for creating a stored procedure on Oracle are as follows:

1. First, you have to create a package of type CURSOR in the Oracle database from which the metrics are to be extracted.
2. Next, you should write a function that returns a cursor of that type.

For instance, to write a stored procedure that should return the average CPU usage of every processor on a specific host, you should follow the steps given below:

1. Create a package named, say, **cpuUtilAvg_pack** of type **cpuUtilAvg_cursor** in the Oracle database from which the metrics are to be extracted

```
CREATE OR REPLACE PACKAGE cpuUtilAvg_pack
AS
```

```
TYPE cpuUtilAvg_cursor IS REF CURSOR;
END cpuUtilAvg_pack;
```

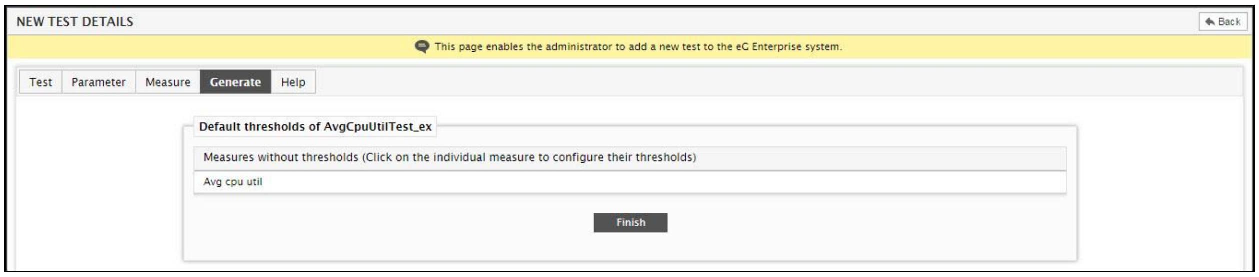
2. Then, write a function that returns a cursor of type **cpuUtilAvg_cursor**

```
CREATE OR REPLACE PROCEDURE cpuUtilAvg_procedure (host IN
systemtest.trgt_host%TYPE, resultCursor OUT cpuUtilAvg_pack.cpuUtilAvg_cursor)
AS
BEGIN
  OPEN resultCursor
  FOR
    SELECT decode(info,'+','DEFAULT',info) as info, avg(cpu_util) from systemtest
WHERE trgt_host = host GROUP BY info ORDER BY info;
END cpuUtilAvg_procedure;
```

The above lines of code create a stored procedure named **cpuUtilAvg_procedure**, which performs the following functions:

- Queries the average CPU utilization of a given host
- Groups the CPU usage value by *processor*
- Returns the results to the cursor, **<packagename>.<packagetype>** - i.e., **cpuUtilAvg_pack.cpuUtilAvg_cursor**.

If there is no error in the generation of the stored procedure, the eG Enterprise system prompts the user to specify the default threshold settings for the measurement made by the newly added test (see Figure 2.53).



This page enables the administrator to add a new test to the eG Enterprise system.

Default thresholds of AvgCpuUtilTest_ex

Measures without thresholds (Click on the individual measure to configure their thresholds)

Avg cpu util

Figure 2.53: Specifying the threshold values of the Avg_cpu_util measure

You can even click on the **Add Help** button therein to create and upload **Admin** and **Monitor** help pages for the new test. To know how, refer to Section 2.1.2 of this document.

2.4 Adding a Perfmon-based Test

A test of the **Perfmon** type can be executed only on Windows environments. Various applications and services on Windows environments expose critical performance statistics via perfmon counters. To allow an administrator to monitor any perfmon counter without having to write elaborate programs, the Integration Console includes a **Perfmon** test type.

Adding/Modifying Tests using the Integration Console

This section facilitates the easy understanding and effective implementation of the **Perfmon** test type, by taking the help of an illustrated example.

In this example, a **ProcessorTest_ex** of type **Perfmon** will be created and its measures will be configured.

As before, begin adding a new test by selecting the **Test** option from the **Integration Console** tile. In the **INTEGRATION CONSOLE - TEST** page that appears next, click the **Add New Test** button. The **NEW TEST DETAILS** page (see Figure 2.54) appears, wherein the following details need to be provided for our example:

- **Test name** – ProcessorTest_ex

Note

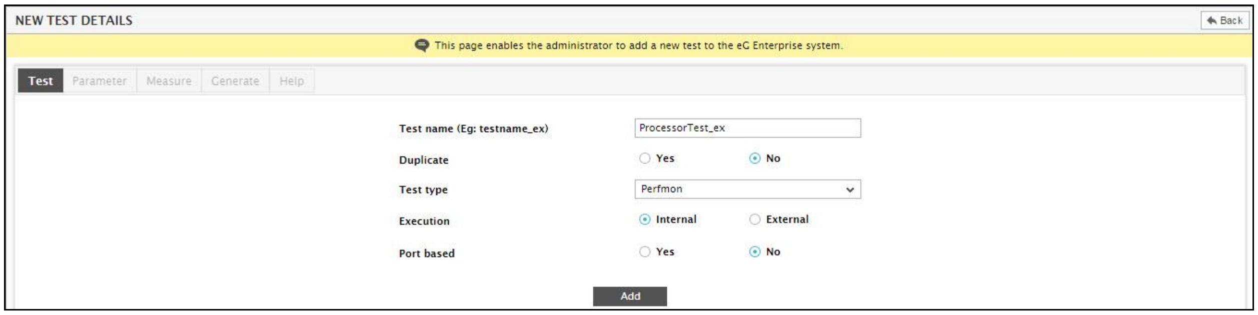
While adding a new test using the Integration Console, ensure that the **Test name** always ends with **_ex**. If not, an error message (see Figure 2.3) will appear upon clicking the **Add** button in Figure 2.54.

- **Duplicate** - Since the new test is not a duplicate of any existing test, set the **Duplicate** flag to **No.**
- **Execution** – Internal, as an internal agent will be executing the **ProcessorTest_ex**

Note

Using the Integration Console plugin, you can add an internal or an external test, but you cannot add tests that need to be run by a remote agent – i.e., tests that need to be executed in an agentless manner.

- **Port based** – As the **ProcessorTest_ex** is to be run at the system level, select **No** against **Port Based** to enable it to be associated with any of the server types.
- **Test type** - Perfmon

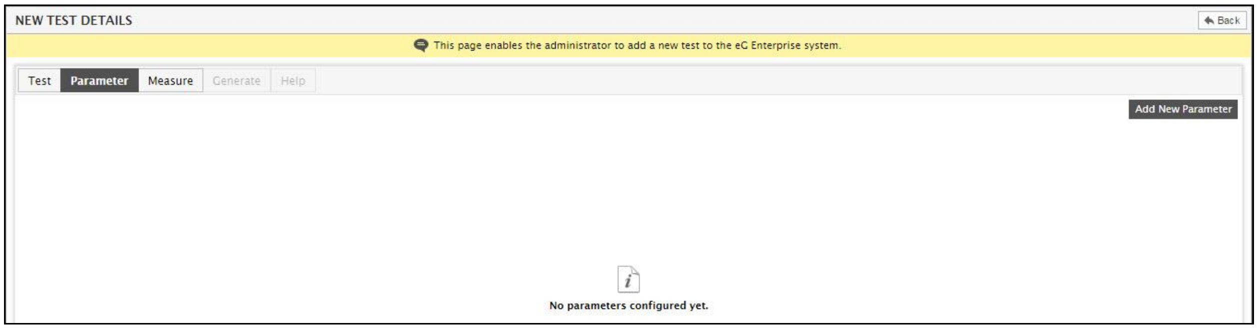


This page enables the administrator to add a new test to the eG Enterprise system.

Test name (Eg: testname_ex)	ProcessorTest_ex
Duplicate	Yes [✓ No]
Test type	Perfmon
Execution	[✓ Internal] External
Port based	Yes [✓ No]

Figure 2.54: Adding a new test of type Perfmon

Then, click the **Add** button to add the new test to the eG Enterprise system. The **Parameter** tab page then opens (see Figure 2.55). Since the **ProcessorTest_ex** in our example does not take any additional parameters, click the **Measure** tab in Figure 2.55 to configure measures for the test.

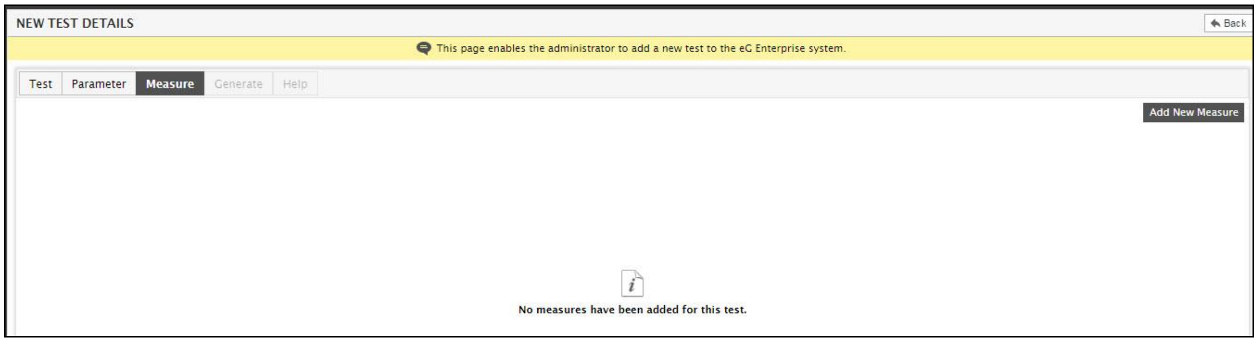


This page enables the administrator to add a new test to the eG Enterprise system.

No parameters configured yet.

Figure 2.55: The Parameter tab page that appears when configuring a test of type Perfmon

Figure 2.56 will then appear. Click the **Add New Measure** button in Figure 2.56 to add a new measure for the test.



This page enables the administrator to add a new test to the eG Enterprise system.

No measures have been added for this test.

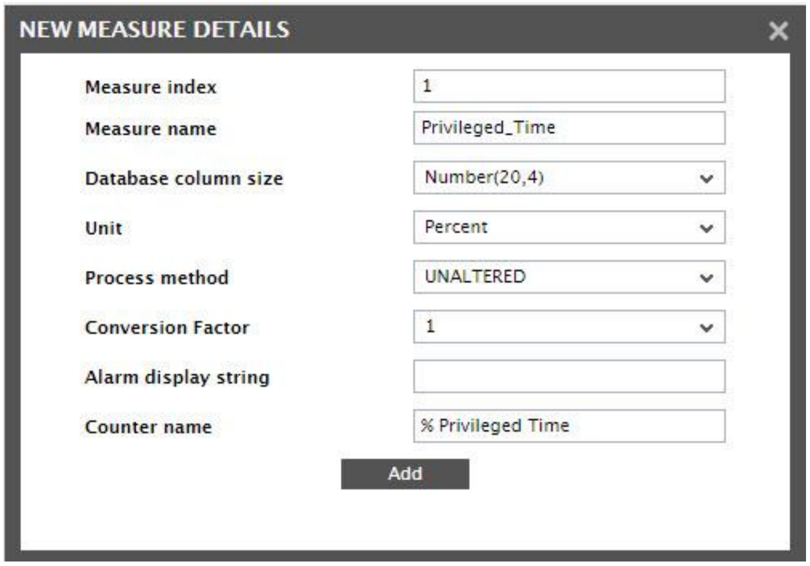
Figure 2.56: The Measure tab page indicating that no measures have been configured yet for the Perfmon Test

Figure 2.57 shows how a measurement of the **Perfmon** test type is specified. Here, specify **Privileged_Time** as the **Measure name**. Then, mention the **Database column size**, the **Unit**, the **Conversion Factor**, and the **Process method** in the same manner as discussed in the previous section. Here again, leave the **Alarm display string** blank.

Refer

To know more about the **Process method**, refer to Page 41 of this document.

To know more about the **Conversion factor**, refer to Page 12 of this document.

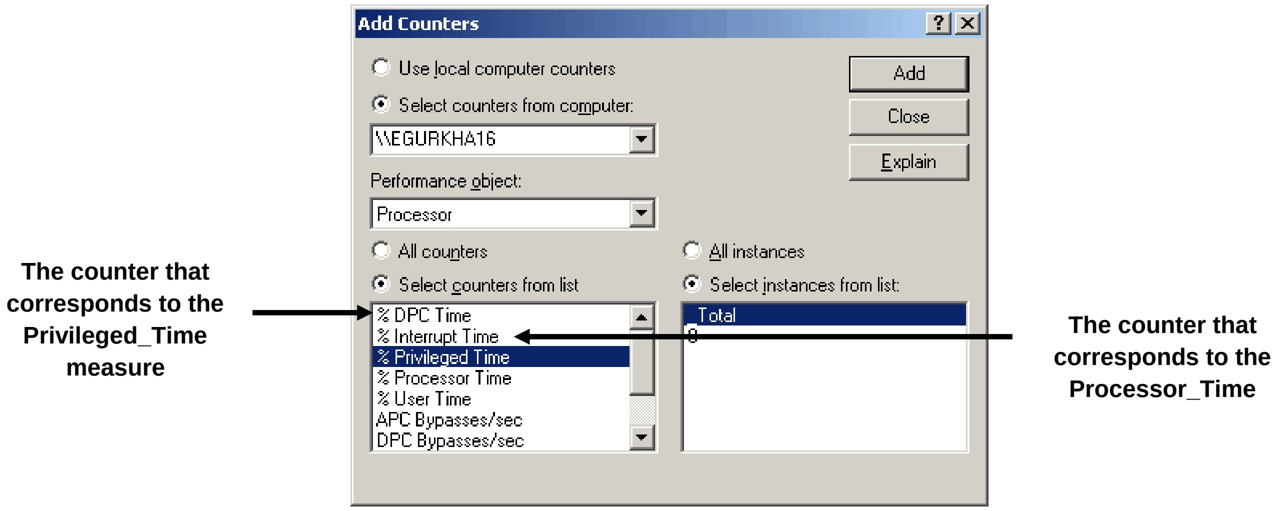


NEW MEASURE DETAILS

Measure index	1
Measure name	Privileged_Time
Database column size	Number(20,4)
Unit	Percent
Process method	UNALTERED
Conversion Factor	1
Alarm display string	
Counter name	% Privileged Time

Figure 2.57: Specification of the first output (Privileged_Time) of the ProcessorTest_ex of type Perfmon

In addition to the above, a **Counter name** text box exists (see Figure 2.57). Against this text box, enter the name of the perfmon counter associated with the specified measure. The name of the perfmon counter should be the same as that which appears in the **Add Counters** dialog box of the **Performance** console in a Microsoft Windows server (see Figure 2.58).



Add Counters

Use local computer counters [✓ Select counters from computer:] \\EGURKHA16

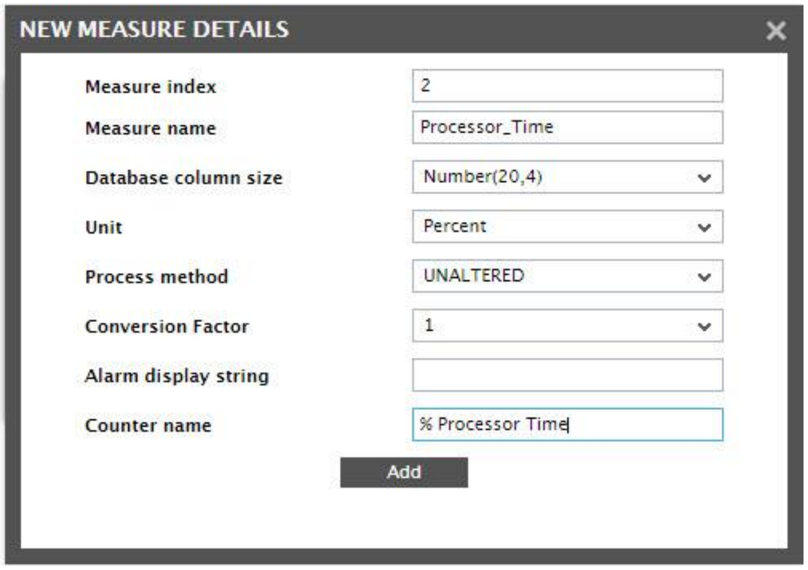
Performance object: Processor

All counters [✓ Select counters from list]

All instances [✓ Select instances from list:]

Figure 2.58: The Add Counters dialog box

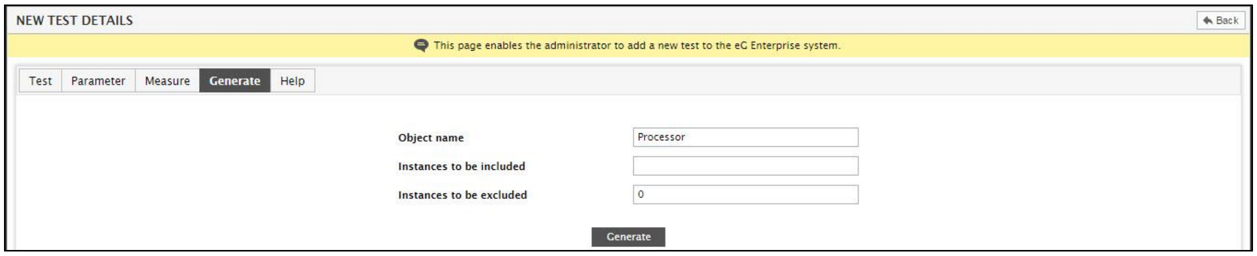
After specifying all the required details, click on the **Add** button in Figure 2.57 to add the first measure of the eG Enterprise system. You will then be prompted to indicate whether you want to add more measures for the test. To add another measure, click the **Yes** button at the prompt. The **NEW MEASURE DETAILS** window will pop up once again (see Figure 2.59). Provide the details of the **Processor_Time** measure in Figure 2.59.



Measure index	2
Measure name	Processor_Time
Database column size	Number(20,4)
Unit	Percent
Process method	UNALTERED
Conversion Factor	1
Alarm display string	
Counter name	% Processor Time

Figure 2.59: Specifying the object and instance names associated with the measures

Once the second measure is added, you will once again be prompted to indicate whether/not you want to add more measures for the test. This time, click **No** at the prompt. This will automatically lead you to the **Generate** tab page (see Figure 2.60).



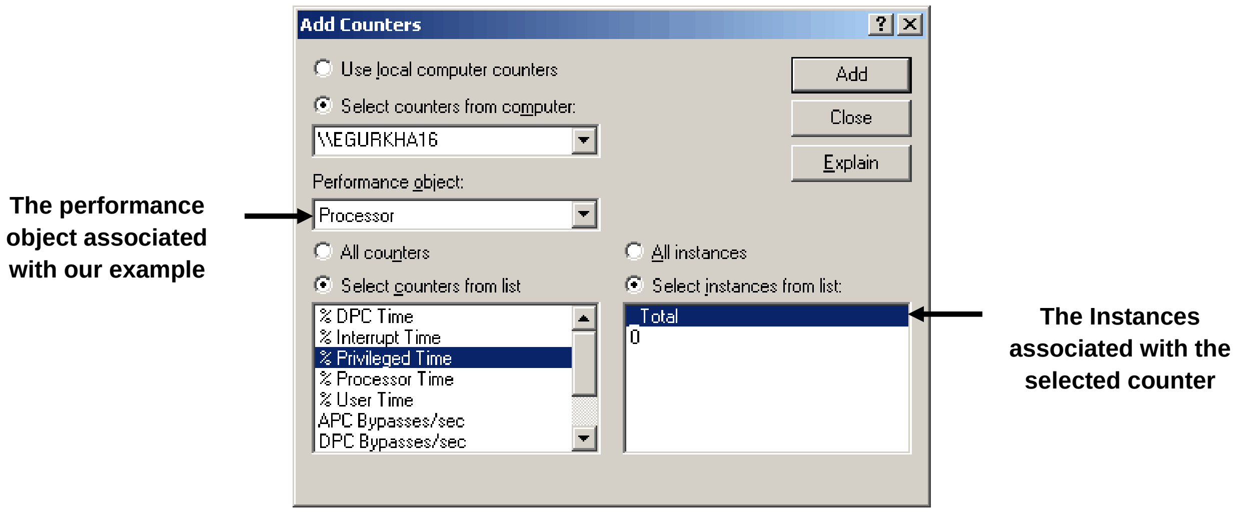
This page enables the administrator to add a new test to the eG Enterprise system.

Object name	Processor
Instances to be included	
Instances to be excluded	0

Figure 2.60: Configuring the implementation of the Perfmon test

As shown in Figure 2.60, proceed to specify the following details:

- **Object name** – Enter the name of the performance object with which the specified measures are associated. This information can be obtained from the **Add Counters** dialog box of the Performance console in a Microsoft Windows server (see Figure 2.61).


The performance object associated with our example
Add Counters
Add
Close
Explain
The Instances associated with the selected counter

Use local computer counters
[✓ Select counters from computer:] \\EGURKHA16

Performance object: Processor

All counters
[✓ Select counters from list]

% DPC Time
% Interrupt Time
[✓ % Privileged Time]
% Processor Time
% User Time
APC Bypasses/sec
DPC Bypasses/sec

All instances
[✓ Select instances from list:]

[✓ _Total]
0

Figure 2.61: The Add Counters dialog box with a Performance object selected

- **Instances to be included** –Mention the specific instances of the measure that need to be monitored. Separate the multiple instances by commas (,). The list of instances associated with a counter can be obtained from the **Add Counters** dialog box of the Performance console in a Microsoft Windows server (see Figure 2.61).

- **Instances to be excluded** – If the **Instances to be included** are large in number, specify the **Instances to be excluded** instead. The eG Enterprise system can thus be instructed to consider all instances except the ones specified against this field, during monitoring. The list of instances associated with a counter can be obtained from the **Add Counters** dialog box of the Performance console in a Microsoft Windows server. In our example, "0" has been specified against the **Instances to be excluded** text box (see Figure 2.61). From Figure 2.61, we can infer that _Total and 0 are the two instances associated with the **% Privileged Time** counter of the **Processor** performance object. The same instances are also available for the **% Processor Time** counter. Our example however, requires the measures pertaining to instance _Total only. Therefore, in order to exclude the instance "0", the same has been specified in the **Instances to be excluded** text box.

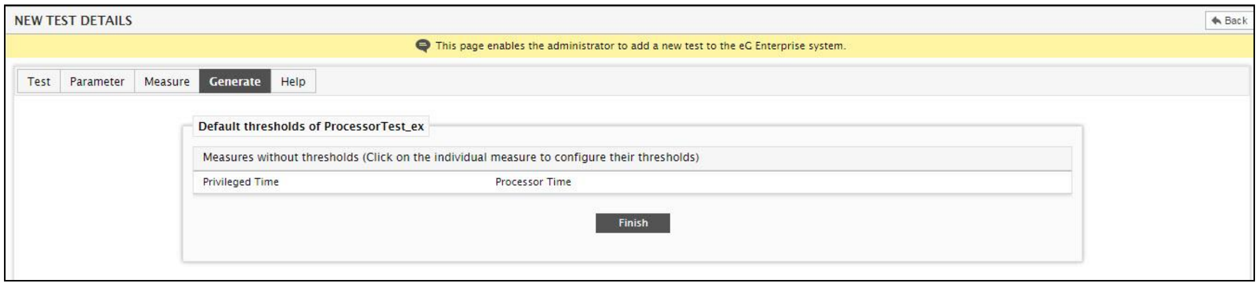
Note

The **Object names** and names of **Instances** should be exactly the same as that which appear in the **Add Counters** dialog box of the Windows Perfmon console. Even the case and spaces should match. Otherwise, measures will not be reported.

For more details on performance objects and instances, refer to the *Microsoft Windows documentation*.

Finally, click the **Generate** button to integrate the test's implementation into the eG Enterprise system.

When a test's measurements are successfully configured, the eG Enterprise system prompts the user to specify the default threshold settings for each of the measurements made by the newly added test (see Figure 2.62).


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Default thresholds of ProcessorTest_ex

Measures without thresholds (Click on the individual measure to configure their thresholds)

Privileged Time Processor Time

Figure 2.62: Specifying the threshold values for the measures of the ProcessorTest_ex

2.5 Adding an SNMP-based Test

Most network devices and some applications support the Simple Network Management Protocol (SNMP). To make it simple for administrators to monitor network devices and applications that are not supported out-of-the-box by eG products, the Integration Console offers another programming-free test-type called the **Snmp** test type. Before adding an **Snmp** test you should decide what objects from the Management Information Base (MIB) are to be monitored.

To illustrate the **Snmp** test, we will be considering two examples. While the first example will help you configure a non-descriptor based **Snmp** test, the second one will help you configure a descriptor-based **Snmp** test.

2.5.1 Adding a Non-Descriptor-Based SNMP Test

First, let us take the example of a Nortel (Bay Networks) switch that is available in your environment. This example involves creating a **BaySwitchTest**, which will use the Nortel (Bay Networks) switch's SNMP MIB to report the number of services running on it.

Begin adding a new test by selecting the **Test** option from the **Integration Console** tile (see Figure 2.1). In the **INTEGRATION CONSOLE - TEST** page that appears next, click the **Add New Test** button. The **NEW TEST DETAILS** page (see Figure 2.63) appears, wherein the following details need to be provided for our example:

- **Test name** – BaySwitchTest_ex

Note

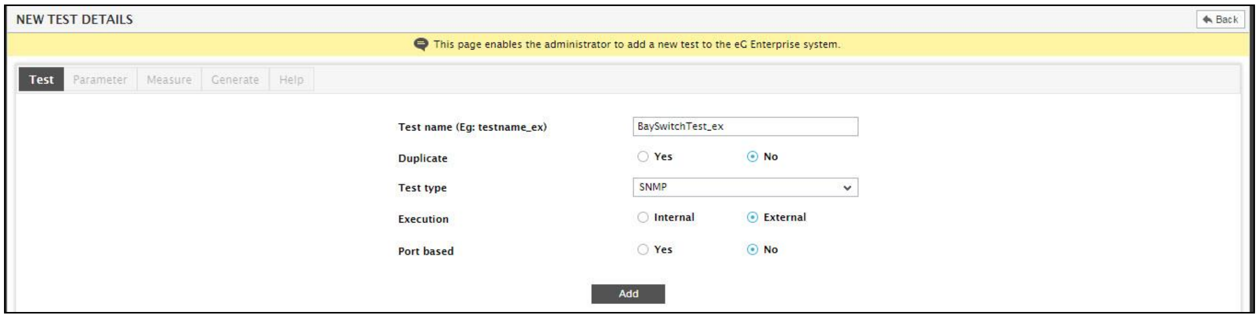
While adding a new test using the Integration Console, ensure that the **Test name** always ends with **_ex.** If not, an error message (see Figure 2.3) will appear upon clicking the **Add** button in Figure 2.63.

- **Duplicate** - Since the new test is not a duplicate of any existing test, set the **Duplicate** flag to **No**.
- **Execution** – External, as an external agent will be executing the test

Note

Using the Integration Console plugin, you can add an internal or an external test, but you cannot add tests that need to be run by a remote agent – i.e., tests that need to be executed in an agentless manner.

- **Port based** – Indicate whether the target server is listening on a port or not. In our case, the Nortel (Bay Networks) switch is not listening on a port.
- **Test type** - Snmp


NEW TEST DETAILS

Back

This page enables the administrator to add a new test to the eG Enterprise system.

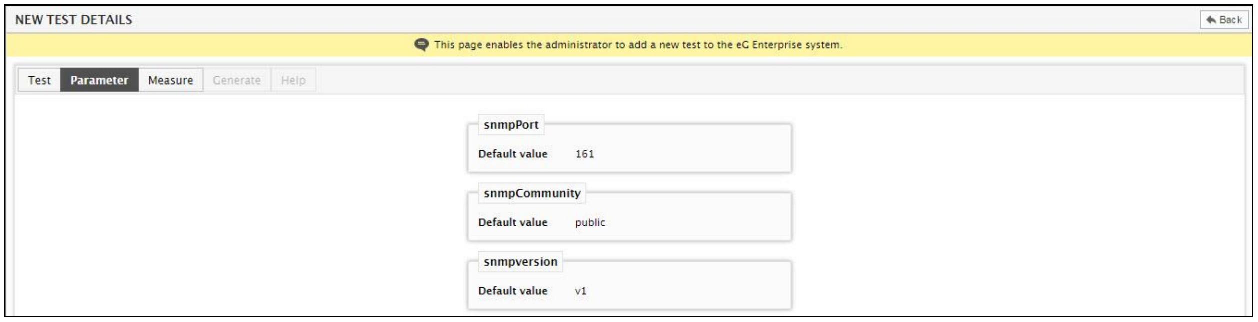
Test | Parameter | Measure | Generate | Help

Test name (Eg: testname_ex)	BaySwitchTest_ex
Duplicate	Yes [✓ No]
Test type	SNMP
Execution	Internal [✓ External]
Port based	Yes [✓ No]

Add

Figure 2.63: Adding a new test of type Snmp

Then, click the **Add** button to add the new test to the eG Enterprise system. The **Parameter** tab page then opens automatically (see Figure 2.64).


NEW TEST DETAILS

Back

This page enables the administrator to add a new test to the eG Enterprise system.

Test | Parameter | Measure | Generate | Help

snmpPort
Default value 161

snmpCommunity
Default value public

snmpversion
Default value v1

Figure 2.64: Viewing a summary of the details of the BaySwitchTest_ex

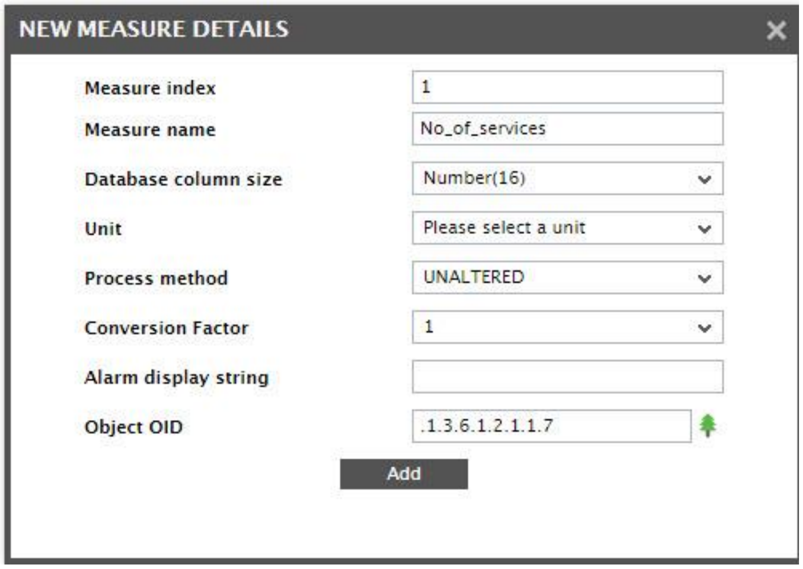
Note that, unlike other test types, by default the **Snmp** test type takes three parameters: **snmpPort**, **snmpCommunity**, and **snmpversion** (see Figure 2.64), with default values 161, public, and v1 respectively. You cannot delete these parameters, but they can be modified. Next, proceed to configure the measures for the **BaySwitchTest_ex** by clicking on the **Measure** tab page in Figure 2.64.

Clicking the **Add New Measure** button in the **Measure** tab page will invoke the **NEW MEASURE DETAILS** pop-up depicted by Figure 2.65. This figure shows how a measurement of the **Snmp** type is specified. The first measure, **No_of_services**, represents the number of services running on the Nortel (Bay Networks) switch. Now, enter **No_of_services** as the **Measure name**. Then, mention the **Database column size**, the **Unit**, the **Conversion Factor**, and the **Process method** as shown in Figure 2.65. Once again, leave the **Alarm display string** blank.

Refer

To know more about the **Process method**, refer to Page 41 of this document.

To know more about the **Conversion factor**, refer to Page 12 of this document.



NEW MEASURE DETAILS

Measure index	1
Measure name	No_of_services
Database column size	Number(16)
Unit	Please select a unit
Process method	UNALTERED
Conversion Factor	1
Alarm display string	
Object OID	.1.3.6.1.2.1.1.7

Figure 2.65: Specification of the No_of_services measure for the BaySwitchTest_ex

In addition to the above, an **Object OID** text box exists (see Figure 2.65). In this text box, enter the object ID of the specified measure. This object ID can be arrived at in either of the following ways:

- By manually "walking" the MIB tree of the application or network device of interest
- By uploading the SNMP MIB file of the application/network device to the eG manager and browsing the MIB tree using the eG administrative interface

In this example, we will discuss both the methodologies.

Determining the OID using SNMP Walk

For the purpose of our example, let us use the standard MIB- interface supported by the Bay switch. The figure below depicts a part of the MIB of relevance to this example:

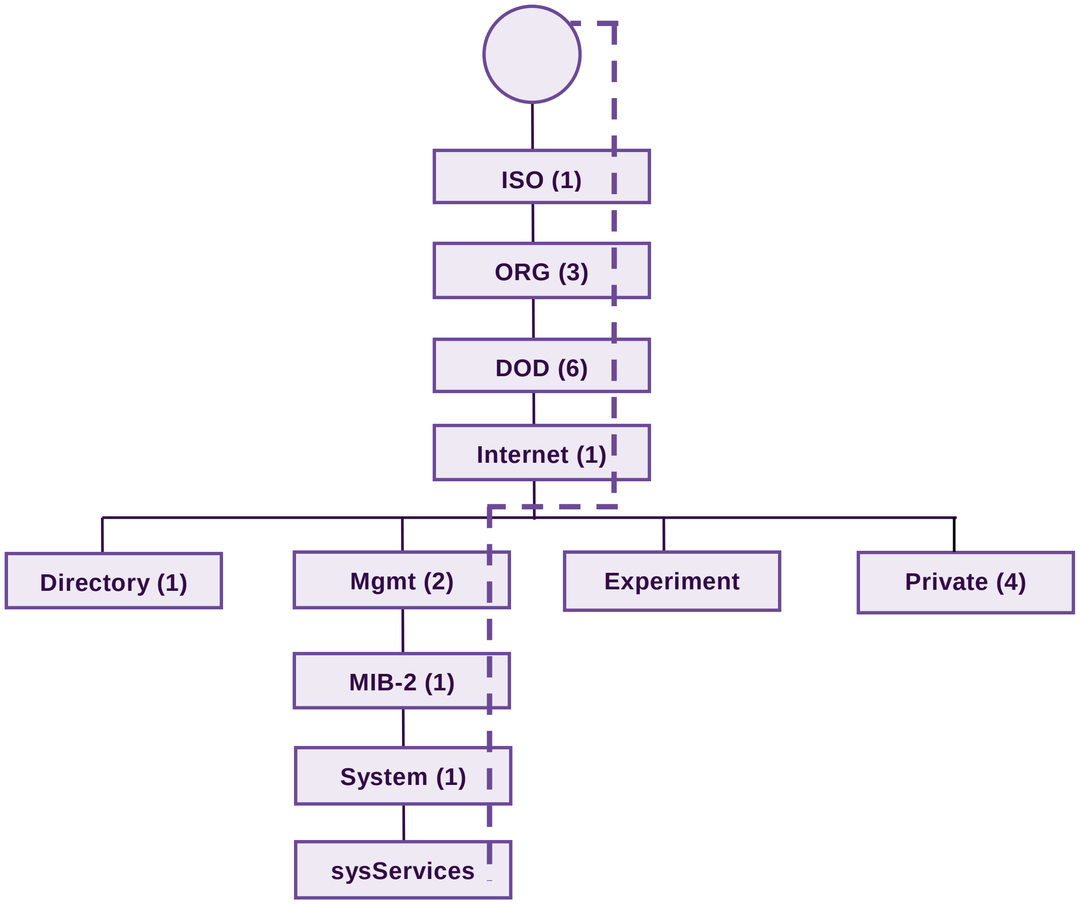


Figure 2.66: A portion of the MIB tree of the Bay switch

As shown in Figure 2.66, the MIB tree comprises of various nodes and sub-nodes, also referred to as "objects". Note that every object is accompanied by a numeric value. These numeric values, when put together in sequence using a dotted notation, provide the unique "object ID" of an object.

For example, the ID of the Internet object in the above tree would be: **.1.3.6.1**. 1, 3 and 6 are the numbers representing the objects that precede the Internet object (i.e. ISO, ORG and DOD respectively) in the MIB tree (see Figure 2.66). The final 1 in the object ID represents the Internet object itself. By arranging these numbers in the order of their occurrence in the MIB tree using the dotted notation, you will arrive at the ID of the Internet object.

You can arrive at the object ID of the **No_of_services** measure also in the same manner. The **sysServices** object in the MIB tree (see Figure 2.66) returns the number of services currently running on the Bay switch. Therefore, the ID of this object needs to be specified as the **Object OID** of the **No_of_services** measure. The dashed lines (-------) in Figure 2.66 trace the path from the root of the MIB tree to the **No_of_services** object. Now, do the following:

- Follow the dashed lines closely and identify the objects through which the lines pass. In our example, note that the lines pass through the following objects:
  - ISO
  - ORG
  - DOD
  - Internet
  - Mgmt

  - ➔ MIB-2
  - ➔ System
  - ➔ sysServices

- ➢ Pick the numbers representing the objects. In our example, the numbers are:
  - ➔ ISO : 1
  - ➔ ORG : 3
  - ➔ DOD : 6
  - ➔ Internet : 1
  - ➔ Mgmt : 2
  - ➔ MIB-2 : 1
  - ➔ System : 1
  - ➔ sysServices : 7
- ➢ Note the order in which the above-mentioned objects appear in the MIB tree, and arrange the corresponding numbers in the same order using the dotted notation
- ➢ You will now have the ID for the **No_of_services** measure, which is: **.1.3.6.1.2.1.1.7**.

Now, specify this ID in the **Object OID** text box of Figure 2.65.

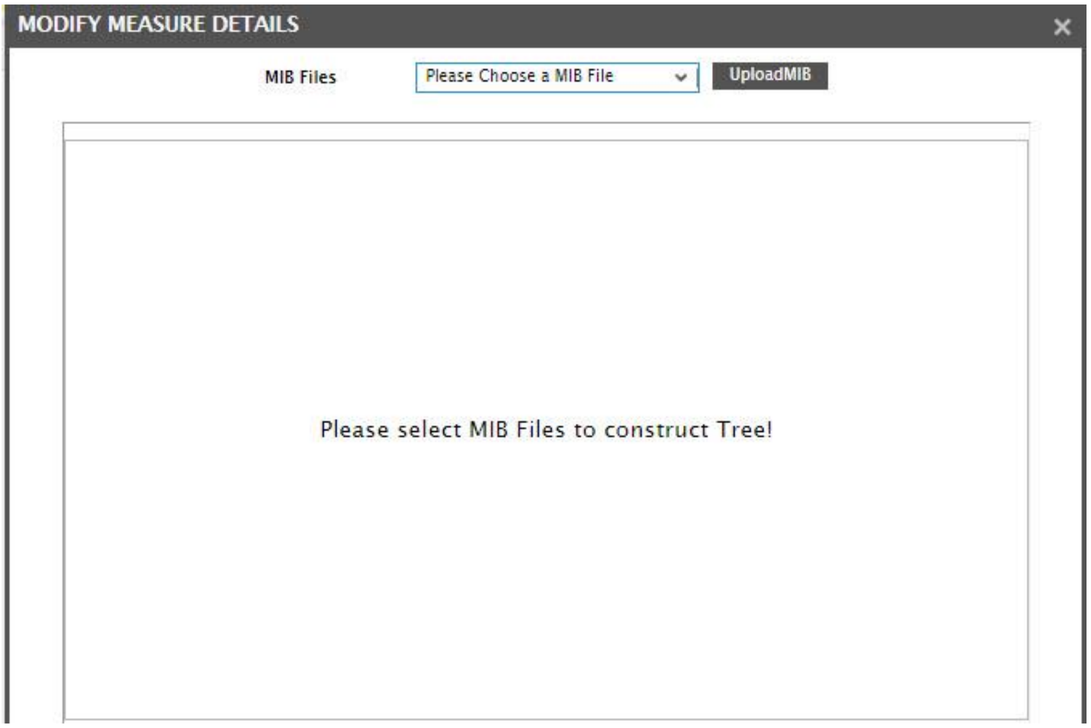
Finally, click the **Add** button to add the new measure.

Determining the OID using the MIB Browser

If you do not want to use the manual procedure for deducing the OID, then, you can determine the same quickly and easily using the **MIB Browser** that is built-into the eG Enterprise system for browsing uploaded MIB files online.

To use the MIB browser, do the following:

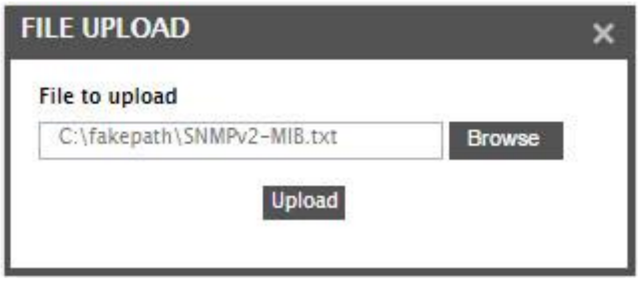
1. For the purpose of our example, let us use the MIB browser for specifying the **Object OID** of the **No_of_services** measure. For that, click on the button next to the **Object OID** text box in Figure 2.65. The **MIB Browser** will then appear as depicted by Figure 2.67. To browse the MIB file that we uploaded, first, select it from the **MIB Files** list in Figure 2.67. The **MIB Files** list contains all MIB files that have been uploaded to the eG manager. If the MIB file of the Bay switch has not been uploaded to the manager yet, then click the **Upload MIB** button in Figure 2.67.



Please select MIB Files to construct Tree!

Figure 2.67: The MIB Browser

2. A pop-up window depicted by Figure 2.68 will then appear. In the **File to upload** text box, specify the full path to the MIB file to be uploaded. You can use the **Browse** button to locate the MIB file of interest to you.

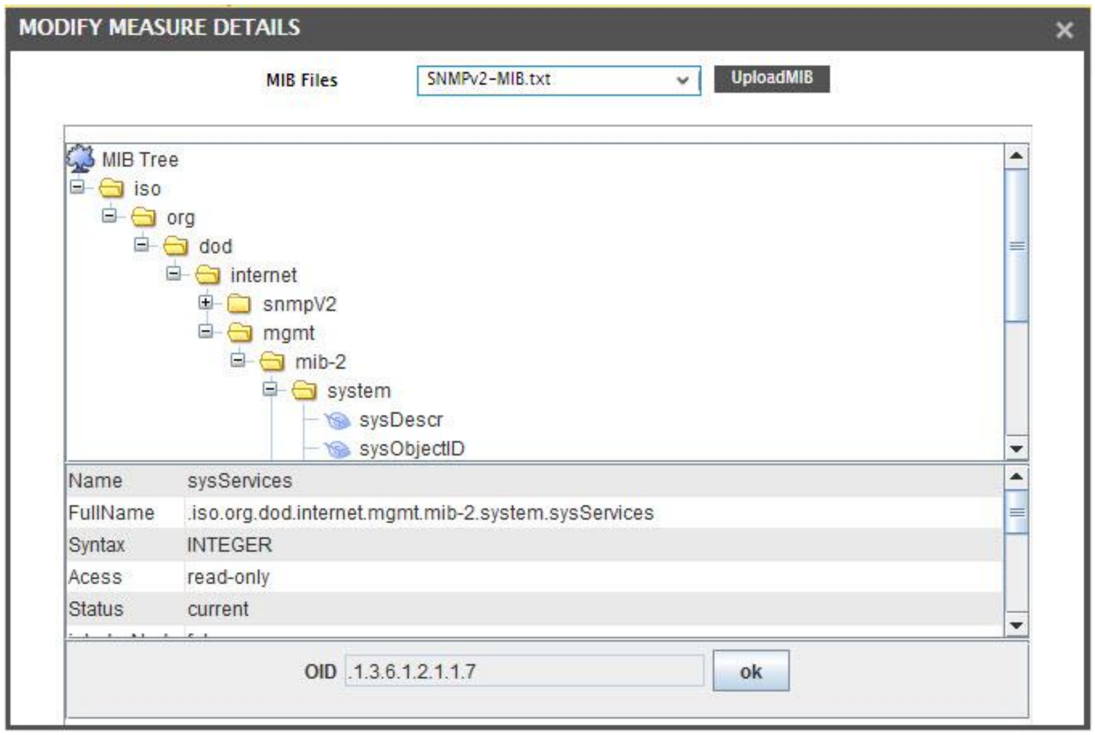


C:\fakepath\SNMPv2-MIB.txt

Figure 2.68: Specifying the full path to the MIB file to be uploaded

3. Once the MIB file path is specified, click the **Upload** button in Figure 2.67 upload the specified file to the eG manager.

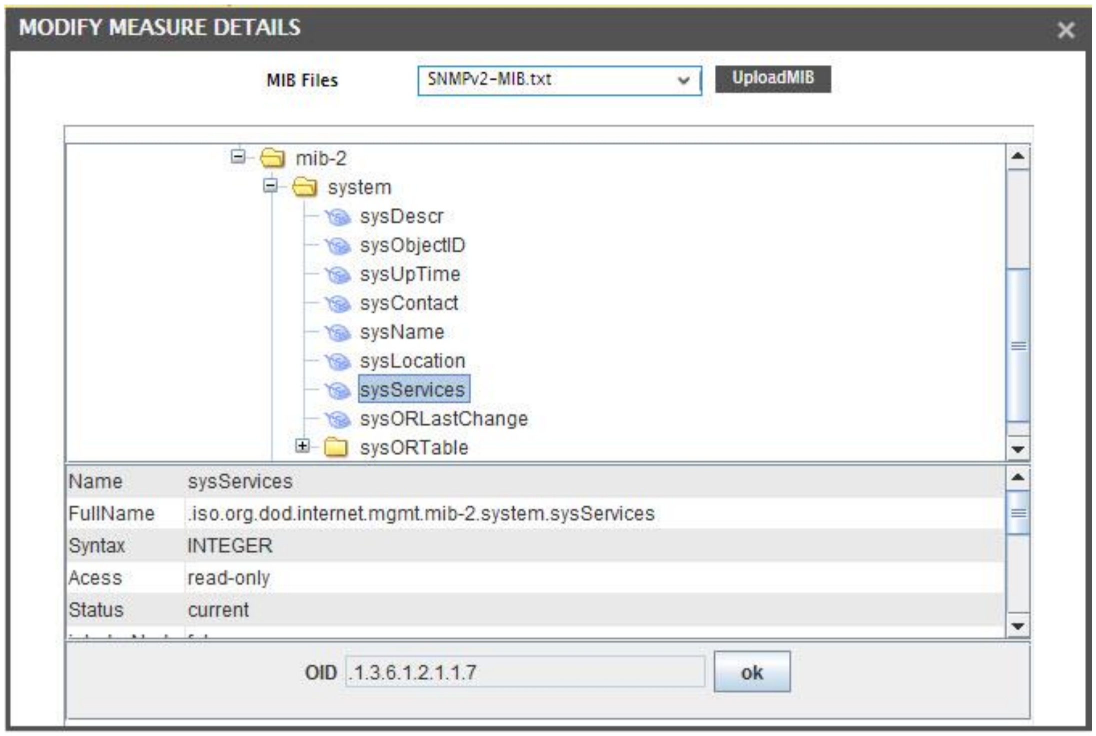
4. If upload is successful, then the newly uploaded MIB file will automatically appear selected in the **MIB Files** list in the **MIB Browser**, as depicted by Figure 2.69. Upon selection of the MIB file, the MIB browser automatically constructs a MIB tree using the SNMP MIB object definitions in the file. To determine the OID of the **No_of_services** measure, drill down the MIB tree by expanding each of the nodes in the sequence, *iso -> org -> dod -> internet -> mib-2-> system -> sysservices* (as depicted by Figure 2.69 and Figure 2.70).



Name	sysServices
FullName	.iso.org.dod.internet.mgmt.mib-2.system.sysServices
Syntax	INTEGER
Acess	read-only
Status	current

OID .1.3.6.1.2.1.1.7

Figure 2.69: The MIB Files list displaying the newly uploaded MIB file



Name	sysServices
FullName	.iso.org.dod.internet.mgmt.mib-2.system.sysServices
Syntax	INTEGER
Acess	read-only
Status	current

OID .1.3.6.1.2.1.1.7

Figure 2.70: Expanding the MIB tree to figure out the OID of the No_of_services measure

5. Upon selecting a node, the MIB browser automatically determines the OID of that node and displays the same against the **OID** field in Figure 2.70. Accordingly, once you drill down to the *sysServices* node, its complete **OID** will be automatically deduced and displayed against the **OID** field. To insert this OID into the **Object OID** text box

in the **NEW MEASURE DETAILS** page of Figure 2.65, just click the **ok** button in Figure 2.70.

You will then be prompted to confirm whether/not you want to add more measures. Click **No** here to stop adding more measures. This will automatically lead you to the **Generate** tab page (see Figure 2.71).

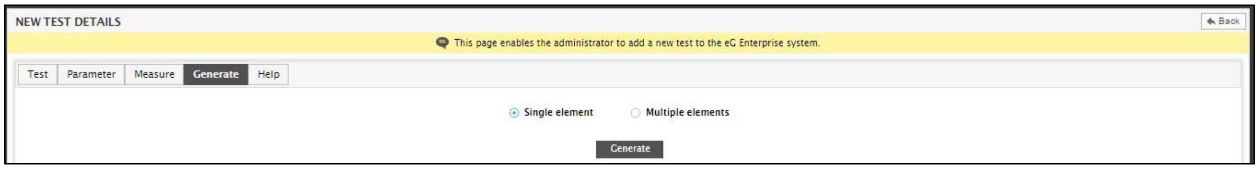


Figure 2.71: Generating the test of type SNMP

The **Generate** tab page of Figure 2.71 reveals two options: **Multiple elements** and **Single element**. To configure a descriptor-based test, select the **Multiple elements** option. In order to configure a non-descriptor-based test, select the **Single element** option. In the case of the **BayswitchTest_ex**, the measurements do not involve access to the SNMP table objects. Therefore, this test is a non-descriptor-based test. Hence, choose the **Single element** option for our example.

Finally, click the **Generate** button to integrate the test's implementation into the eG Enterprise system.

When a test's measurements are successfully configured, the eG Enterprise system prompts the user to specify the default threshold settings for each of the measurements made by the newly added test.

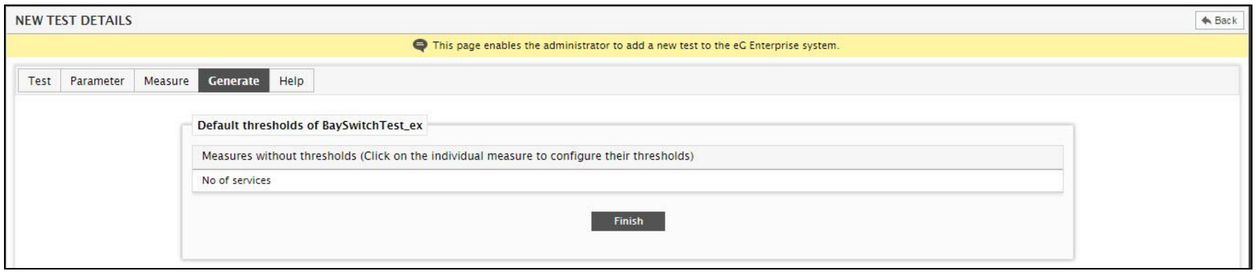

NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Default thresholds of BaySwitchTest_ex
Measures without thresholds (Click on the individual measure to configure their thresholds)
No of services
Finish

Figure 2.72: Configuring thresholds for the non-descriptor-based SNMP test newly created

2.5.2 Adding a Descriptor-based SNMP Test

To illustrate a descriptor-based **Snmp** test, consider another example. Say you have a BEA Tuxedo domain server running. This example involves the creation of a **TuxDomainTest_ex**, which will use the Tuxedo server's SNMP MIB to report the number of machines and servers running in a Tuxedo domain.

Begin adding this test by selecting the **Test** option from the **Integration Console** tile (see Figure 2.1). In the **INTEGRATION CONSOLE - TEST** page that appears next (see Figure 2.2), click the **Add New Test** button. The **NEW TEST DETAILS** page (see Figure 2.73) appears, wherein the following details need to be provided for our example:

- **Test name** –TuxDomainTest_ex

Note

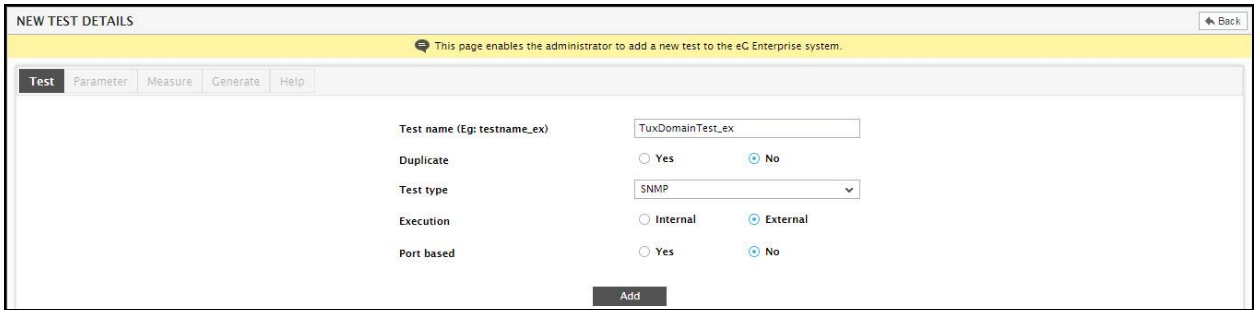
While adding a new test using the Integration Console, ensure that the **Test name** always ends with **_ex**. If not, an error message (see Figure 2.3) will appear upon clicking the **Add** button in Figure 2.73.

- **Duplicate** - Since the new test is not a duplicate of any existing test, set the **Duplicate** flag to **No**.
- **Execution** – External, as an external agent will be executing the test

Note

Using the Integration Console plugin, you can add an internal or an external test, but you cannot add tests that need to be run by a remote agent – i.e., tests that need to be executed in an agentless manner.

- **Port based** – Specify whether the target server listens on a port or not. In our case, the Tuxedo domain server is not listening on a port.
- **Test type** - Snmp

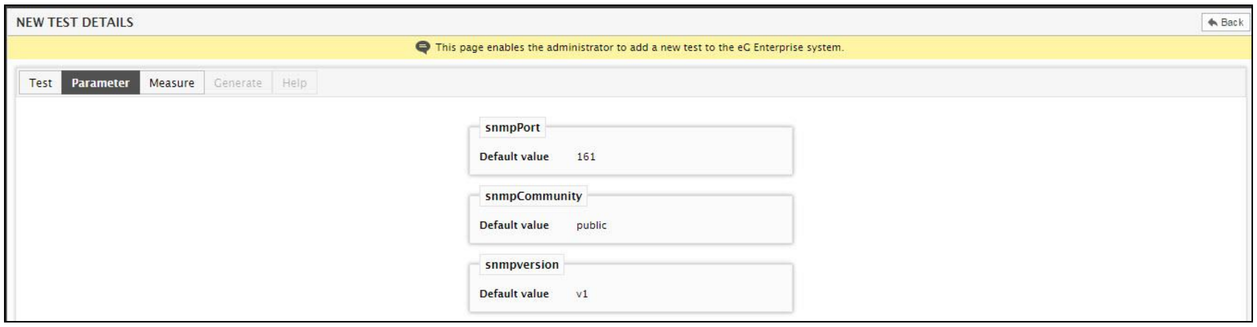

NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Test name (Eg: testname_ex)	TuxDomainTest_ex
Duplicate	Yes [✓ No]
Test type	SNMP
Execution	Internal [✓ External]
Port based	Yes [✓ No]

Figure 2.73: Adding a descriptor-based test of type SNMP

Then, click the **Add** button in Figure 2.73 to add the new test to the eG Enterprise system. This will automatically open the **Parameter** tab page (see Figure 2.74), where the default parameters of the SNMP-based test will be displayed.


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

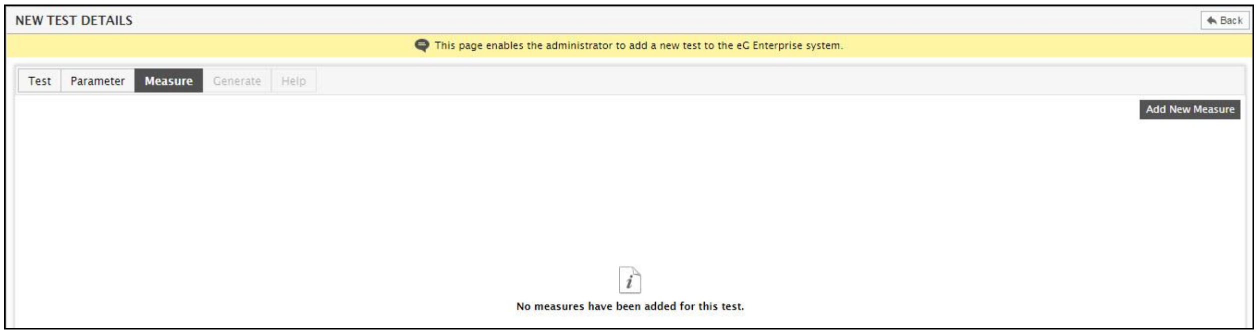
snmpPort
Default value 161

snmpCommunity
Default value public

snmpversion
Default value v1

Figure 2.74: Viewing the default parameters of the TuxedoDomainTest_ex

Since no additional parameters need be added to the new test, click the **Measure** tab page in Figure 2.74 to add measures for the test. Figure 2.75 will then appear indicating that no measures have been configured for the test yet.


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

No measures have been added for this test.

Figure 2.75: The Measure tab page indicating that no measures have been configured yet for the descriptor-based SNMP test

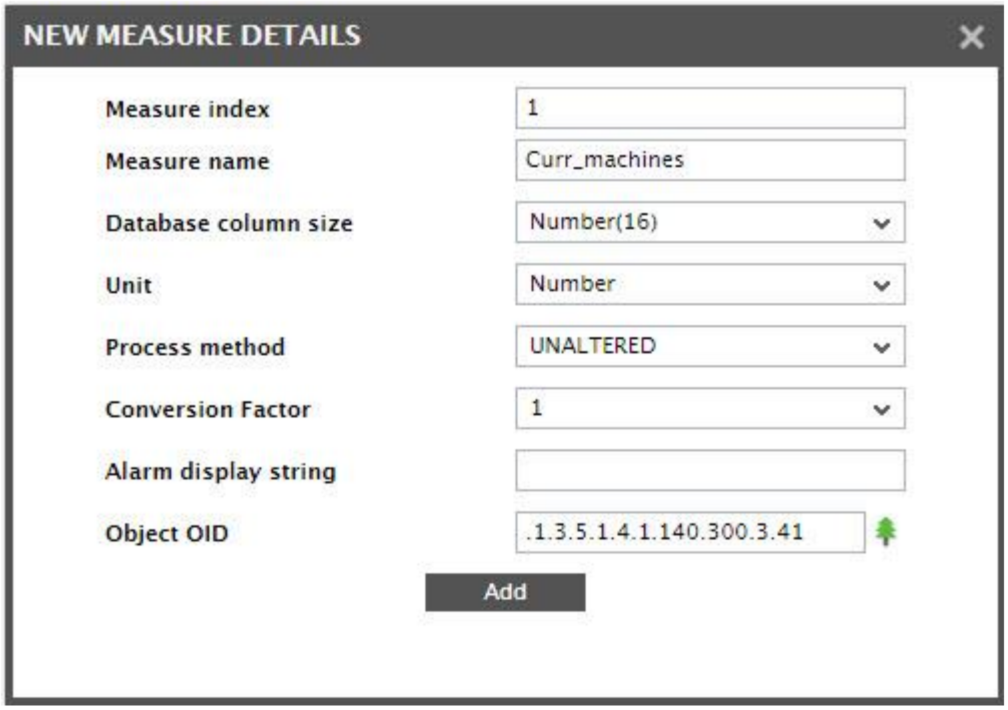
To add a new measure, click the **Add New Measure** button in Figure 2.75. The **NEW MEASURE DETAILS** page of Figure 2.76 will then appear, where you can specify the details of the new measure.

The first measure of the **TuxDomainTest_ex** in our example is **Curr_machines**. This represents the current number of machines in the domain. Therefore, in the **NEW MEASURE DETAILS** page of Figure 2.76, specify **Curr_machines** as the **Measure name**. Then, mention the **Database column size**, the **Unit**, the **Conversion Factor**, and the **Process method** as shown in Figure 2.76 below. As before, leave the **Alarm display string**, blank.

Refer

To know more about the **Process method**, refer to Page 41 of this document.

To know more about the **Conversion factor**, refer to Page 12 of this document.



NEW MEASURE DETAILS

Measure index	1
Measure name	Curr_machines
Database column size	Number(16)
Unit	Number
Process method	UNALTERED
Conversion Factor	1
Alarm display string	
Object OID	.1.3.5.1.4.1.140.300.3.41

Figure 2.76: Adding the Curr_machines measure to the eG Enterprise system

Then, in the **Object OID** text box, enter the object ID of the specified measure. Here again, you can either manually specify the OID, or use the MIB browser that eG Enterprise provides. Let us begin with the manual procedure.

Figure 2.77 depicts a part of the MIB for the Tuxedo domain server.

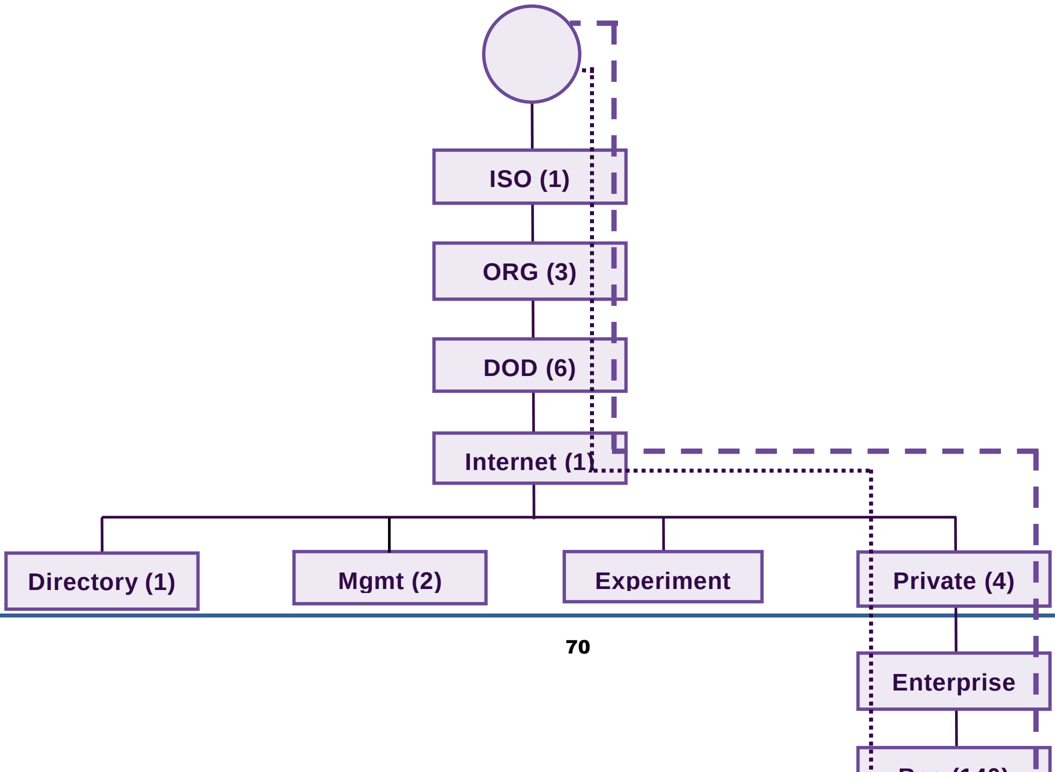


Figure 2.77: A portion of the MIB for the Tuxedo domain server

The **tuxTDomain** object in the tree is a table of entries, where each entry includes at least four attributes. The four objects that you can see below the **tuxTDomain** object in Figure 2.77 are the attributes that are relevant to our example. Of these:

- The **tuxTDomainID** attribute returns the domain identification string
- The **tuxTDomainCurMachines** attribute returns the number of machines currently available in the Tuxedo domain
- The **tuxTDomainCurServers** attribute returns the number of servers currently available in the Tuxedo domain
- The **tuxTDomainState** attribute returns the status of each of the servers in the Tuxedo domain

Since the **tuxTDomainCurMachines** object returns the number of machines currently available in the Tuxedo domain, the ID of this object needs to be specified as the **OBJECT OID** of the **Curr_machines** measure. The dashed lines (-------) in Figure 2.77 trace the path from the root of the MIB tree to the **tuxTDomainCurMachines** object. Now, do the following:

- Follow the dashed lines closely and identify the objects through which the lines pass. In our example, note that the lines pass through the following objects:
  - ➔ ISO
  - ➔ ORG
  - ➔ DOD
  - ➔ Internet
  - ➔ Private
  - ➔ Enterprise
  - ➔ Bea
  - ➔ Tuxedo
  - ➔ tuxTDomain

- ➔ tuxTDomainCurMachines

- ➢ Pick the numbers representing the objects. In our example, the numbers are:
  - ➔ ISO - 1
  - ➔ ORG - 3
  - ➔ DOD - 6
  - ➔ Internet - 1
  - ➔ Private – 4
  - ➔ Enterprise – 1
  - ➔ Bea – 140
  - ➔ Tuxedo – 300
  - ➔ tuxTDomain – 3
  - ➔ tuxTDomainCurMachines - 41
- ➢ Note the order in which the above-mentioned objects appear in the MIB tree and arrange the corresponding numbers in the same order using the dotted notation.
- ➢ You will now have the ID for the **Curr_machines** measure, which is: **.1.3.6.1.4.1.140.300.3.41**.

Now, specify this ID in the **Object OID** text box of Figure 2.76. Finally, click the **Add** button to add the new measure.

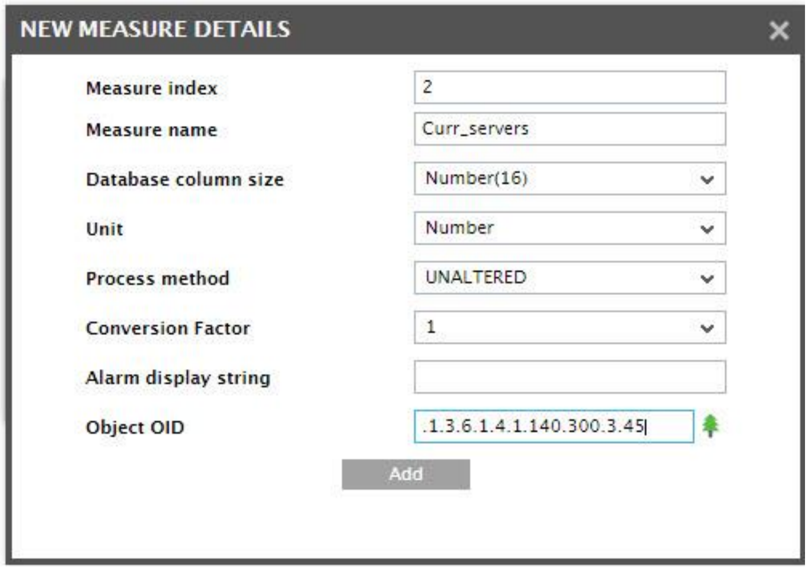
On the other hand, if you want to use the MIB browser to determine the **Object OID**, then click on the button next to the **Object OID** text box in Figure 2.76. The **MIB Browser** then appears. In the MIB browser, open the SNMP MIB file of the Tuxedo Domain server, view the MIB tree, and drill down the MIB tree to determine the OID of the object. The detailed procedure for navigating the MIB browser is available in page 64 of this document.

Note

In order to use the MIB browser for specifying the OID of the **Curr_machines** measure, you need to ensure that the SNMP MIB of the Tuxedo Domain Server is uploaded to the eG manager.

Once the first measure is added, click the **Add** button in Figure 2.76 to add the measure. When prompted to add more measures for the test, click the **Yes** button. Figure 2.78 will then appear, using which one more measure can be added to the **TuxDomainTest_ex**.

Figure 2.78 depicts the procedure for adding the **Curr_servers** measure, which reports the number of servers currently available in the Tuxedo domain.



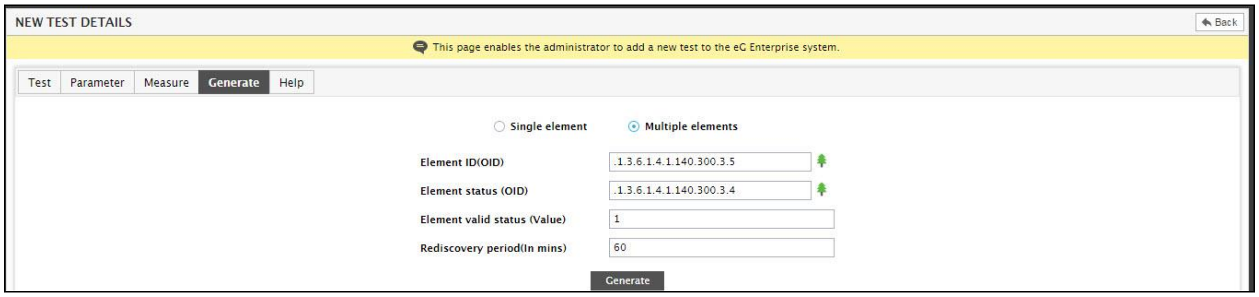
NEW MEASURE DETAILS

Measure index	2
Measure name	Curr_servers
Database column size	Number(16)
Unit	Number
Process method	UNALTERED
Conversion Factor	1
Alarm display string	
Object OID	.1.3.6.1.4.1.140.300.3.45

Figure 2.78: Adding the Curr_servers measure to the eG Enterprise system

Since the **Curr_servers** measure corresponds to the **tuxTDomainCurServers** attribute, the ID of this attribute has to be specified as the **Object OID** of the **Curr_servers** measure. The dotted lines (**.....**) in Figure 2.77 trace the path from the root of the MIB tree to the **tuxTDomainCurServers** object. Follow this path to arrive at the **Object OID** of the **Curr_servers** measure. Accordingly, you will have the ID: **.1.3.6.4.1.140.300.3.45**. Specify the same against the **Object OID** text box, click the **Add** button, and add the second measure also to the eG Enterprise system. When prompted again to add more measures, click the **No** button. This will automatically lead you to the **Generate** tab page (see Figure 2.79).

As before, the **Generate** tab page will reveal two options: **Multiple elements** and **Single element**. In our example, since the measurements of the **TuxDomainTest_ex** involve access to SNMP table objects, the test is a descriptor-based test. Hence, choose the **Multiple elements** option.



NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Single element [✓ Multiple elements]

Element ID(OID)	.1.3.6.1.4.1.140.300.3.5
Element status (OID)	.1.3.6.1.4.1.140.300.3.4
Element valid status (Value)	1
Rediscovery period(In mins)	60

Figure 2.79: Configuring a descriptor-based TuxedoDomainTest_ex

In the resulting page (see Figure 2.79), provide the following details:

- **Element ID(OID)**: Here, specify the ID of the object that supplies the descriptors for a test. For our example, the descriptor is the identification string of the Tuxedo domain. To manually specify the OID that reports the identification string, once again refer to the MIB tree in Figure 2.77. The object that returns this string value is the **tuxTDomainID** object of the MIB tree. Using the MIB tree, you can manually arrive at the ID of this object, and the same will be: **.1.3.6.1.4.1.140.300.3.5**.

  To use the MIB browser instead, click on the 🌲 button next to the **Element ID (OID)** text box in Figure 2.79. In the MIB browser that then appears, open the SNMP MIB file of the Tuxedo Domain server, view the MIB tree, and drill down the MIB tree to determine the OID of the object that returns the descriptors of the **TuxDomainTest_ex**. For the detailed procedure, refer to page 64 of this document.

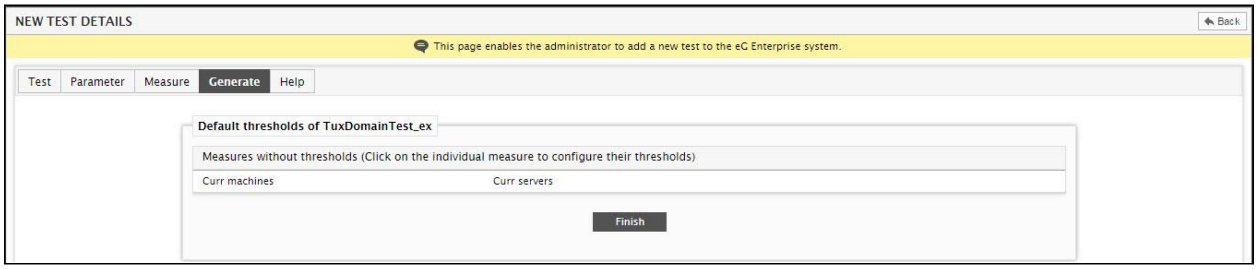
- **Element status(OID)**: In this text box, specify the ID of the object that returns the status of the descriptor. This specification combined with that of the **Element Valid Status (Value)** field form a filter condition that will enable you to view the measures pertaining to only those elements that are in a particular state. Such a condition is optional. Therefore, if filtering is not required, you can specify **none** in both these text boxes. In our example, we need to view the measures pertaining to only the "active" servers in the Tuxedo domain. Therefore, a filter condition is a must. Hence, specify the ID of the **tuxTDomainState** object that returns the status (whether active or inactive) of the servers in the Tuxedo domain. Its ID, as inferred from the MIB tree (see Figure 2.77) is **.1.3.6.1.4.1.140.300.3.4**. To browse the MIB tree using the MIB browser, click on the button next to the **Element status (OID)** text box in Figure 2.79. For the detailed procedure to use the browser, refer to page 64 of this document.

Note

In order to use the MIB browser for specifying the **Element ID (OID)** and the **Element status (OID)**, you need to ensure that the SNMP MIB of the Tuxedo Domain Server is uploaded to the eG manager.

- **Element valid status (Value)**: Specify the value that indicates the state of the descriptor. As mentioned already, if filtering is not required, you can specify **none** here. In our example, the value 1 indicates that the server is active and value 0 indicates that it is inactive. As we require only the measures pertaining to active servers, specify 1 here.
- **Rediscovery period (In mins)**: Specify the frequency with which rediscovery needs to occur. By default, this is 60 minutes. For our example, the default frequency is to be retained.

Now, click the **Generate** button to integrate the test's implementation into the eG Enterprise system. When a test's measurements are successfully configured, the eG Enterprise system prompts the user to specify the default threshold settings for each of the measurements made by the newly added test.


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Default thresholds of TuxDomainTest_ex

Measures without thresholds (Click on the individual measure to configure their thresholds)

Curr machines Curr servers

Figure 2.80: Configuring thresholds for the measures of the descriptor-based SNMP test

2.6 Adding a JMX-based Test

Most enterprises have custom developed application components running directly on the JVM directly or hosted on standard, off-the-shelf middleware application servers (such as WebLogic, WebSphere, JBoss, etc.). While monitoring the JVM or the web application servers provides visibility into the core engines that support the Java applications, it does not provide any information on the custom application components. **Java Management eXtensions** (**JMX**) offers a standard way by which applications can expose custom metrics for monitoring tools. Many custom applications use JMX to publish information about their functioning to third party monitoring applications.

The eG Integration Console has now been enhanced to collect and report on applications that offer JMX-based interfaces. Administrators now have a new **Jmx** option, using which they can specify the specific JMX attributes that the eG agents must collect to monitor their custom applications. Administrators can specify the JMX attribute name and the specific MBeans to be monitored, and the eG agent takes care of periodically polling these attributes and reporting the metrics back to the eG Enterprise console. This capability offers administrators a quick and easy way to integrate monitoring of their custom Java applications into the enterprise management console.

This section takes the help of an example to help you understand how a test of type **Jmx** can be built. The **Jmx** test to be added for this purpose will be connecting to a custom Java application and reporting the following to indicate how the JVM of that application uses its heap memory.

The sections that follow will discuss how this can be achieved.

2.6.1 Enabling JMX Support for the JRE of the Target Application

Prior to adding a new **Jmx** test, you need to enable the JMX support for the JRE of the target application. The steps for the same differ according to the authentication/security status of JMX. By default, JMX requires no authentication or security (SSL). In this case therefore, to use JMX for pulling out metrics from a target application, the following will have to be done:

1. Login to the application host.
2. The **<JAVA_HOME>\jre\lib\management** folder used by the target application will typically contain the following files:
    - *management.properties*
    - *jmxremote.access*
    - *jmxremote.password.template*
    - *snmp.acl*
3. Edit the *managerment.properties* file, and append the following lines to it:

```
com.sun.management.jmxremote.port=<Port No>
com.sun.management.jmxremote.ssl=false
com.sun.management.jmxremote.authenticate=false
```

   For instance, if the JMX listens on port 9005, then the first line of the above specification would be:

```
com.sun.management.jmxremote.port=9005
```

4. Then, save the file.
5. Next, edit the start-up script of the target application, and add the following line to it:

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

6. For instance, on a Windows host, the *<management.properties_file_path>* can be expressed as: **D:\bea\jrockit_150_11\jre\lib\management**.
7. On other hand, on a Unix/Linux/Solaris host, a sample *<management.properties_file_path>* specification will be as follows: **/usr/jdk1.5.0_05/jre/lib/management**.
8. Save this script file too.
9. Next, during test configuration, do the following:
    - Set **JMX** as the mode;

- Set the port that you defined in step 3 above (in the *management.properties* file) as the jmx remote port;
- • Set the user and password parameters to *none*.
- Update the test configuration.

On the other hand, if JMX requires **only authentication** (and no security), then the following steps will apply:

1. Login to the application host. If the application is executing on a Windows host, then, login to the host as a local/domain administrator.
2. As stated earlier, the **<JAVA_HOME>\jre\lib\management** folder used by the target application will typically contain the following files:
   - *management.properties*
   - *jmxremote.access*
   - *jmxremote.password.template*
   - *snmp.acl.template*
3. First, copy the *jmxremote.password.template* file to any other location on the host, rename it as as *jmxremote.password*, and then, copy it back to the **<JAVA_HOME>\jre\lib\management** folder.
4. Next, edit the *jmxremote.password* file and the *jmxremote.access* file to create a user with *read-write* access to the JMX. To know how to create such a user, refer to Section 2.6.1.1 of this document.
5. Then, proceed to make the *jmxremote.password* file secure by granting a single user "full access" to that file. For monitoring applications executing on Windows in particular, only the *Owner* of the *jmxremote.password* file should have full control of that file. To know how to grant this privilege to the *Owner* of the file, refer to Section 2.6.1.2.
6. In case of applications executing on Solaris / Linux hosts on the other hand, any user can be granted full access to the *jmxremote.password* file, by following the steps below:
   - Login to the host as the user who is to be granted full control of the *jmxremote.password* file.
   - Issue the following command:

     chmod 600 jmxremote.password

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

7. Next, edit the *management.properties* file, and append the following lines to it:

```
com.sun.management.jmxremote.port=<Port No>
com.sun.management.jmxremote.ssl=false
com.sun.management.jmxremote.authenticate=true
com.sun.management.jmxremote.access.file=<Path of jmxremote.access>
com.sun.management.jmxremote.password.file=<Path of jmxremote.password>
```

For instance, assume that the JMX remote port is 9005, and the *jmxremote.access* and *jmxremote.password* files exist in the following directory on a Windows host: **D:\bea\jrockit_150_11\jre\lib\management**. The specification above will then read as follows:

```
com.sun.management.jmxremote.port=9005
com.sun.management.jmxremote.access.file=D:\\bea\\jrockit_150_11\\jre\\lib\\managem
ent\\jmxremote.access
```

```
com.sun.management.jmxremote.password.file=D:\\bea\\jrockit_150_11\\jre\\lib\\manag
ement\\jmxremote.password
```

8. If the application in question is executing on a Unix/Solaris/Linux host, and the *jmxremote.access* and *jmxremote.password* files reside in the **/usr/jdk1.5.0_05/jre/lib/management** folder of the host, then the last 2 lines of the specification will be:

```
com.sun.management.jmxremote.access.file=/usr/jdk1.5.0_05/jre/lib/management/jmxrem
ote.access
com.sun.management.jmxremote.password.file=/usr/jdk1.5.0_05/jre/lib/management/jmxr
emote.password
```

9. Finally, save the file.
10. Then, edit the start-up script of the target application, include the following line in it, and save the file:

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

11. For instance, on a Windows host, the *<management.properties_file_path>* can be expressed as: **D:\bea\jrockit_150_11\jre\lib\management.**
12. On other hand, on a Unix/Linux/Solaris host, a sample *<management.properties_file_path>* specification will be as follows: **/usr/jdk1.5.0_05/jre/lib/management**.

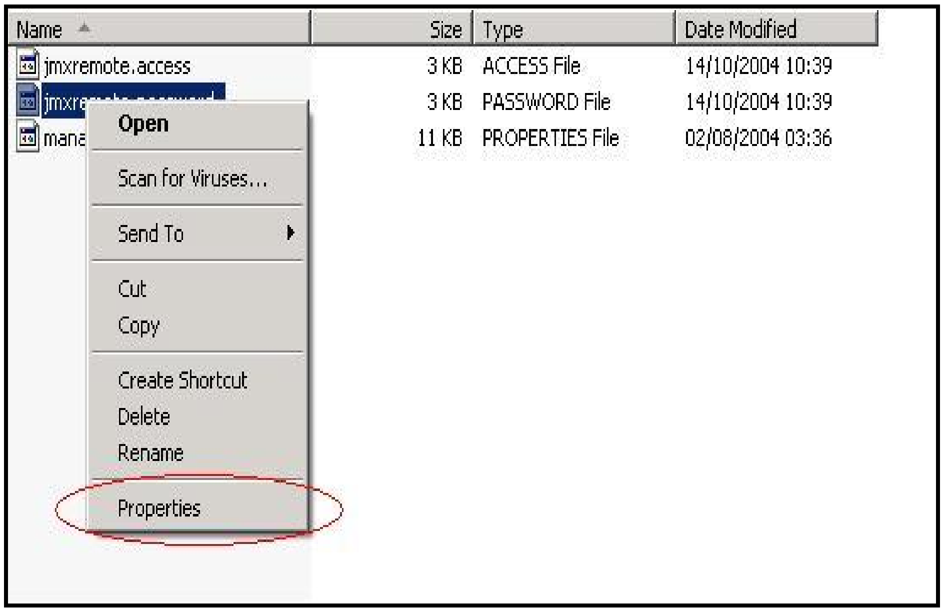
Note

eG Enterprise cannot use JMX that requires both authentication and security (SSL), for monitoring the target Java application.

2.6.1.1 Securing the 'jmxremote.password' file

To enable the eG agent to use JMX (that requires **authentication only**) for monitoring a Windows-based Java application, you need to ensure that the *jmxremote.password* file in the **<JAVA_HOME>\jre\lib\management** folder used by the target application is accessible **only by the Owner of that file**. To achieve this, do the following:

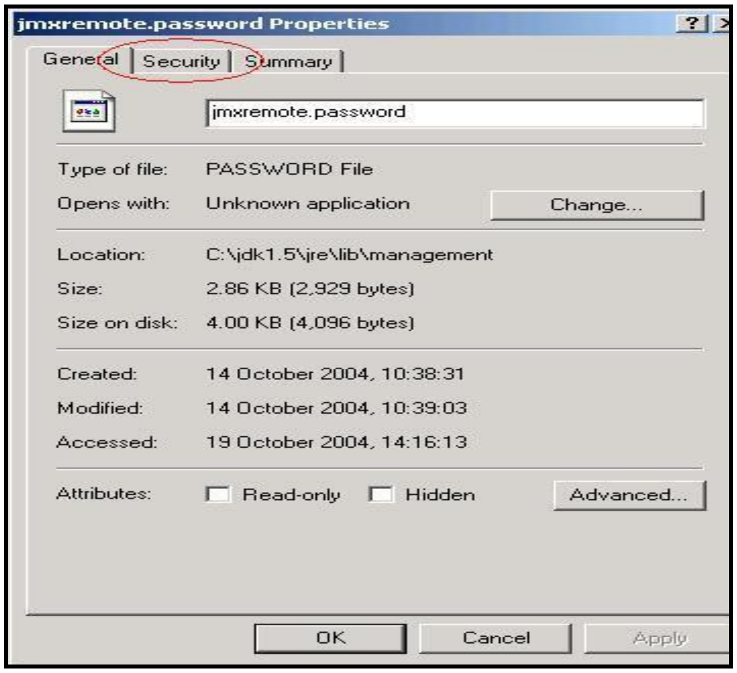
1. Login to the Windows host as a local/domain administrator.
2. Browse to the location of the *jmxremote.password* file using Windows Explorer.
3. Next, right-click on the *jmxremote.password* file and select the **Properties** option (see Figure 2.81).



Name	Size	Type	Date Modified
jmxremote.access	3 KB	ACCESS File	14/10/2004 10:39
jmxre	3 KB	PASSWORD File	14/10/2004 10:39
mana	11 KB	PROPERTIES File	02/08/2004 03:36

Figure 2.81: Selecting the Properties option

4. From Figure 2.82 that appears next, select the **Security** tab.



jmxremote.password

Type of file: PASSWORD File
Opens with: Unknown application

Location: C:\jdk1.5\jre\lib\management
Size: 2.86 KB (2,929 bytes)
Size on disk: 4.00 KB (4,096 bytes)

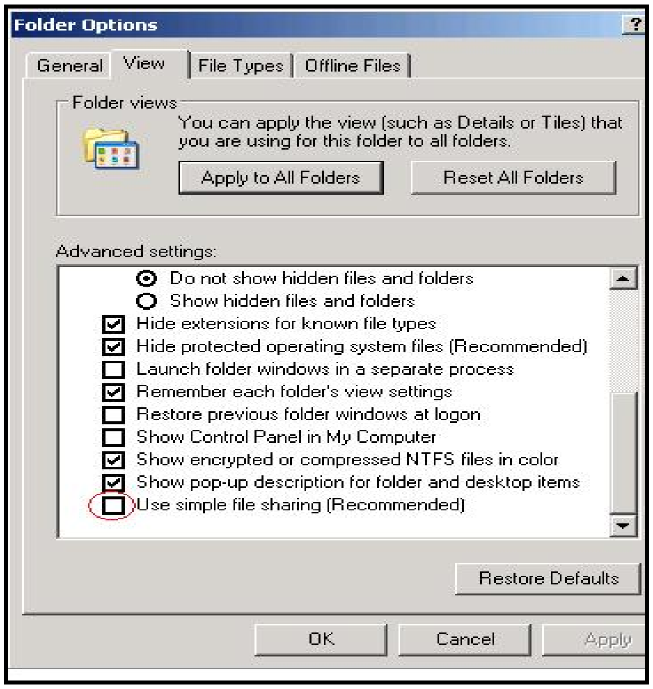
Created: 14 October 2004, 10:38:31
Modified: 14 October 2004, 10:39:03
Accessed: 19 October 2004, 14:16:13

Attributes: ☐ Read-only ☐ Hidden

Figure 2.82: The Properties dialog box

However, if you are on Windows XP and the computer is not part of a domain, then the **Security** tab may be missing. To reveal the **Security** tab, do the following:

- Open Windows Explorer, and choose **Folder Options** from the **Tools** menu.
- Select the **View** tab, scroll to the bottom of the **Advanced Settings** section, and clear the check box next to **Use Simple File Sharing**.



You can apply the view (such as Details or Tiles) that you are using for this folder to all folders.

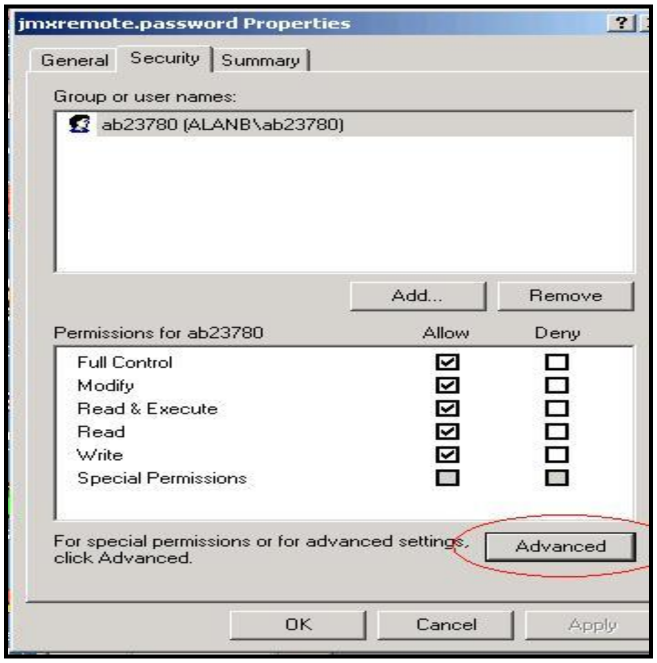
Advanced settings:

- [✓ Do not show hidden files and folders]
- Show hidden files and folders
- [✓] Hide extensions for known file types
- [✓] Hide protected operating system files (Recommended)
- [ ] Launch folder windows in a separate process
- [✓] Remember each folder's view settings
- [ ] Restore previous folder windows at logon
- [ ] Show Control Panel in My Computer
- [✓] Show encrypted or compressed NTFS files in color
- [✓] Show pop-up description for folder and desktop items
- [ ] Use simple file sharing (Recommended)

Figure 2.83: Deselecting the 'Use simple file sharing' option

- Click **OK** to apply the change
- When you restart Windows Explorer, the **Security** tab would be visible.

5. Next, select the **Advanced** button in the **Security** tab of Figure 2.84.



Group or user names:

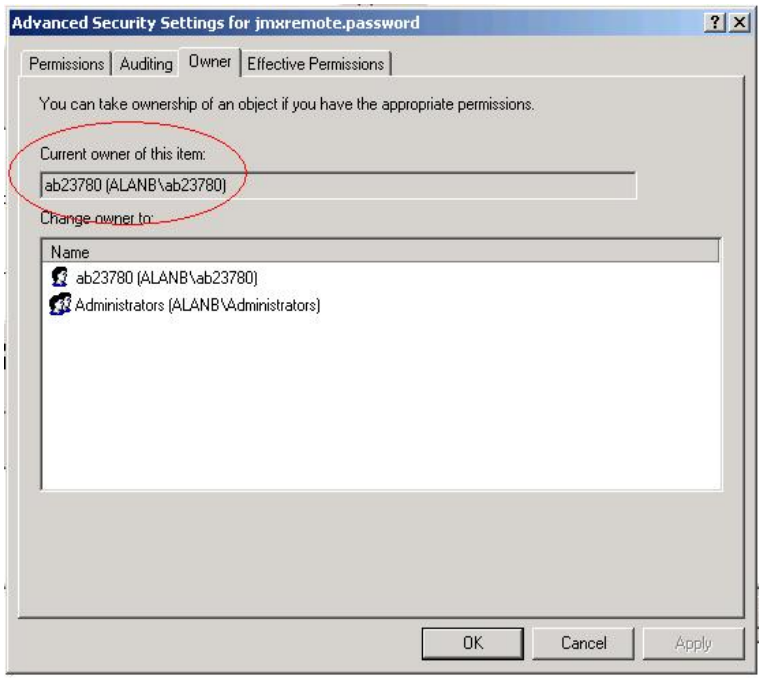
ab23780 (ALANB\ab23780)

Permissions for ab23780	Allow	Deny
Full Control	✓	
Modify	✓	
Read & Execute	✓	
Read	✓	
Write	✓	
Special Permissions		

For special permissions or for advanced settings, click Advanced.

Figure 2.84: Clicking the Advanced button

6. Select the **Owner** tab to see who the owner of the file is.


Advanced Security Settings for jmxremote.password

You can take ownership of an object if you have the appropriate permissions.

Current owner of this item:

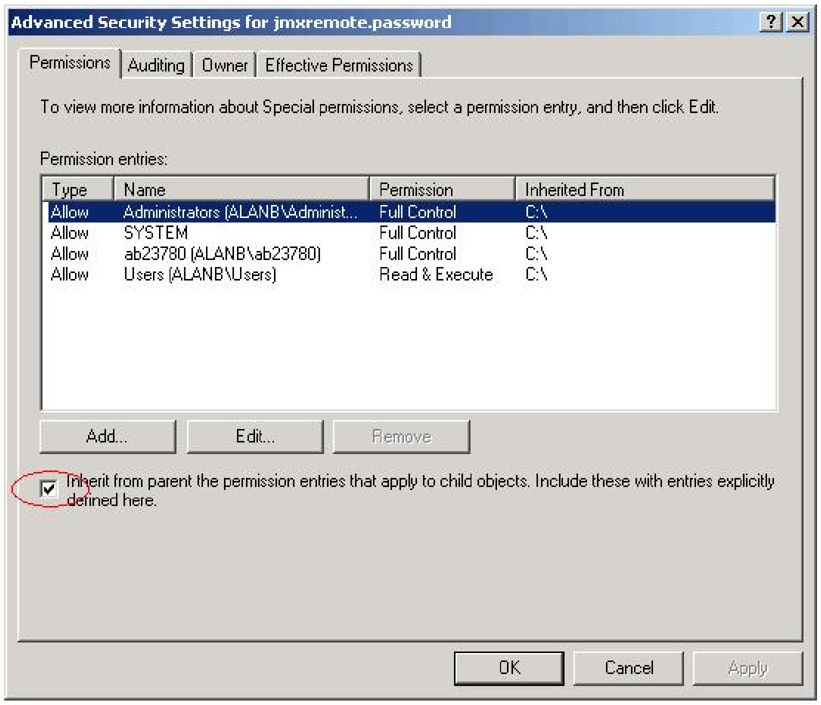
ab23780 (ALANB\ab23780)

Change owner to:

Name
ab23780 (ALANB\ab23780)
Administrators (ALANB\Administrators)

Figure 2.85: Verfying whether the Owner of the file is the same as the application Owner

7. Then, proceed to select the **Permissions** tab from Figure 2.86 to set the permissions. If the *jmxremote.password* file has inherited its permissions from a parent directory that allows users or groups other than the **Owner** to access the file, then clear the **Inherit from parent the permission entries that apply to child objects** check box in Figure 2.86.



To view more information about Special permissions, select a permission entry, and then click Edit.

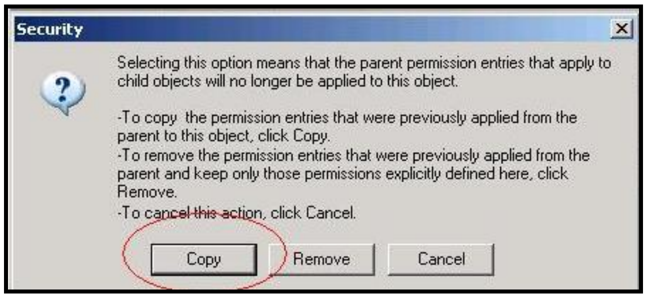
Permission entries:

Type	Name	Permission	Inherited From
Allow	Administrators (ALANB\Administ...	Full Control	C:\
Allow	SYSTEM	Full Control	C:\
Allow	ab23780 (ALANB\ab23780)	Full Control	C:\
Allow	Users (ALANB\Users)	Read & Execute	C:\

[✓] Inherit from parent the permission entries that apply to child objects. Include these with entries explicitly defined here.

Figure 2.86: Disinheriting permissions borrowed from a parent directory

8. At this point, you will be prompted to confirm whether the inherited permissions should be copied from the parent or removed. Press the **Copy** button in Figure 2.87.

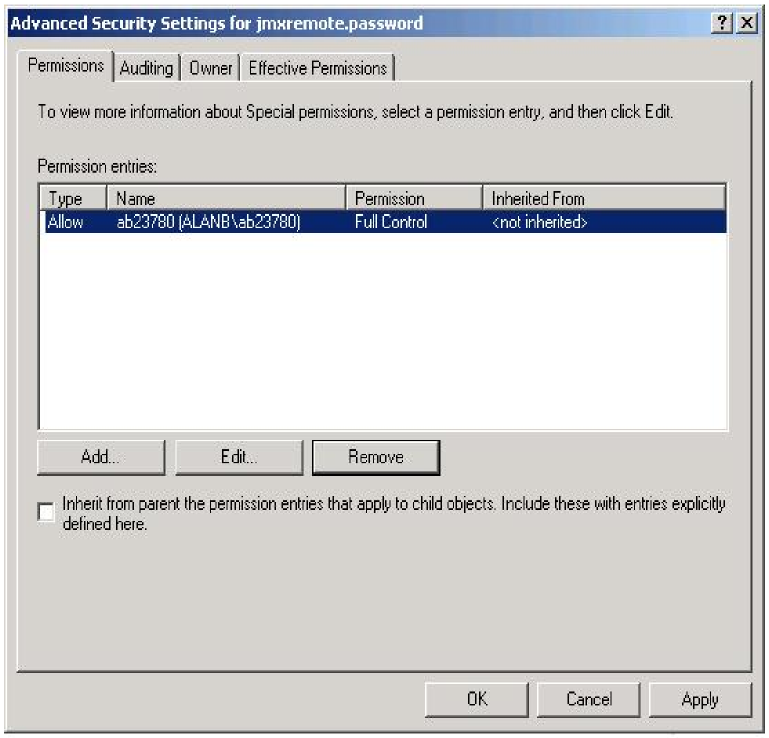


Selecting this option means that the parent permission entries that apply to child objects will no longer be applied to this object.

-To copy the permission entries that were previously applied from the parent to this object, click Copy.
-To remove the permission entries that were previously applied from the parent and keep only those permissions explicitly defined here, click Remove.
-To cancel this action, click Cancel.

Figure 2.87: Copying the inherited permissions

9. Next, remove all permission entries that allow the *jmxremote.password* file to be accessed by users or groups other than the file **Owner**. For this, click the user or group and press the **Remove** button in Figure 2.87. At the end of this exercise, only a single permission entry granting **Full Control** to the owner should remain in Figure 2.88.


Advanced Security Settings for jmxremote.password

To view more information about Special permissions, select a permission entry, and then click Edit.

Permission entries:

Type	Name	Permission	Inherited From
Allow	ab23780 (ALANB\ab23780)	Full Control	<not inherited>

☐ Inherit from parent the permission entries that apply to child objects. Include these with entries explicitly defined here.

Figure 2.88: Granting full control to the file owner

10. Finally, click the **Apply** and **OK** buttons to register the changes. The password file is now secure, and can only be accessed by the file owner.

2.6.1.2 Configuring the eG Agent to Support JMX Authentication

If the eG agent needs to use JMX for monitoring a Java application, and this JMX requires **authentication only** (and not security), then any new **JMX** test created using the Integration Console component should be configured with the credentials of a valid user to JMX, with *read-write rights*. The steps for creating such a user are detailed below:

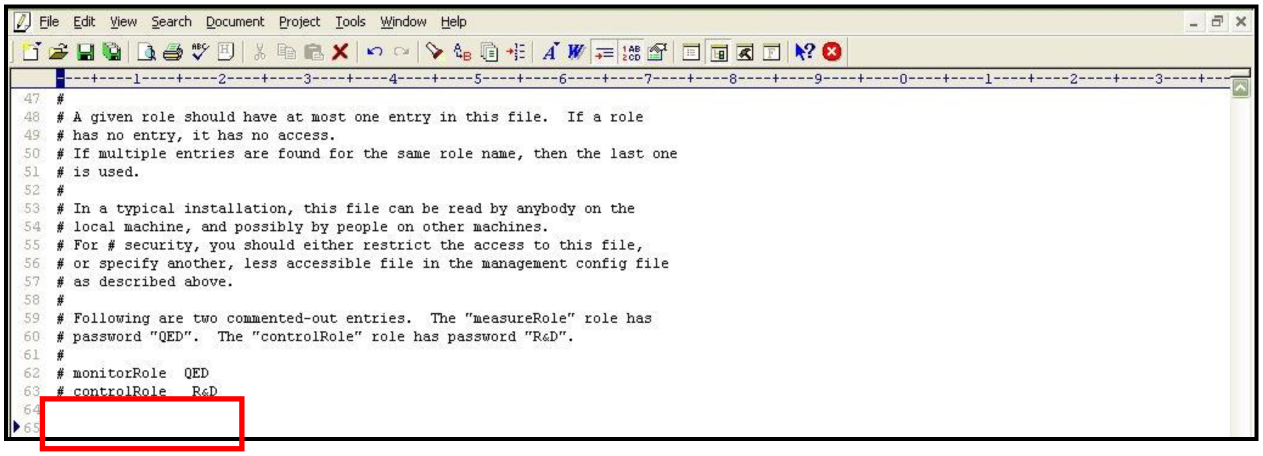
1. Login to the application host. If the application being monitored is on a Windows host, then login as a local/domain administrator to the host.

2. Go to the **<JAVA_HOME>\jre\lib\management** folder used by the target application to view the following files:

    - *management.properties*
    - *jmxremote.access*
    - *jmxremote.password.template*
    - *snmp.acl.template*

3. Copy the *jmxremote.password.template* file to a different location, rename it as *jmxremote.password*, and copy it back to the **<JAVA_HOME>\jre\lib\management** folder.

4. Open the *jmxremote.password* file and scroll down to the end of the file. By default, you will find the commented entries indicated by Figure 2.89 below:

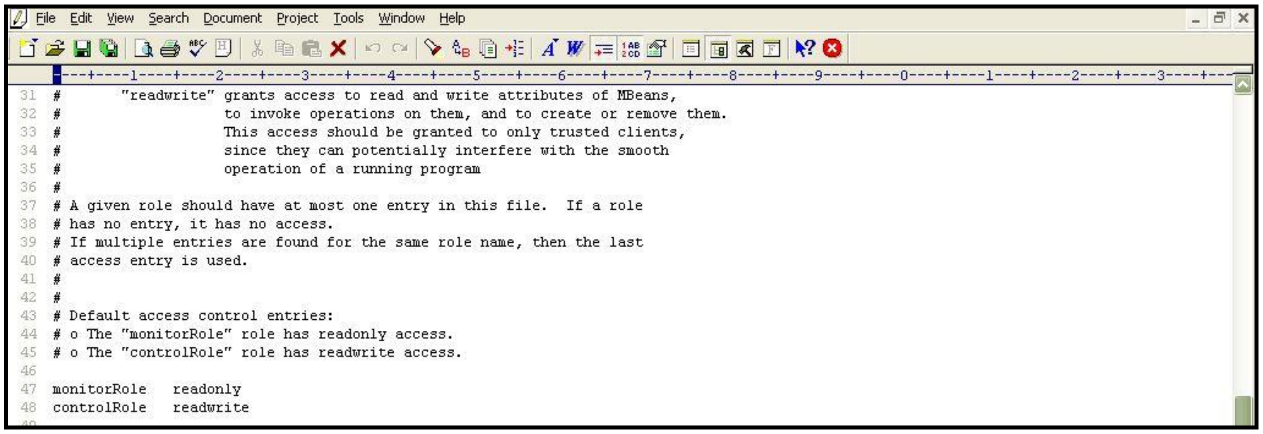


```
#
# A given role should have at most one entry in this file.  If a role
# has no entry, it has no access.
# If multiple entries are found for the same role name, then the last one
# is used.
#
# In a typical installation, this file can be read by anybody on the
# local machine, and possibly by people on other machines.
# For # security, you should either restrict the access to this file,
# or specify another, less accessible file in the management config file
# as described above.
#
# Following are two commented-out entries.  The "measureRole" role has
# password "QED".  The "controlRole" role has password "R&D".
#
# monitorRole  QED
# controlRole   R&D
```

Figure 2.89: Scrolling down the jmxremote.password file to view 2 commented entries

5. The two entries indicated by Figure 2.89 are sample *username password* pairs with access to JMX. For instance, in the first sample entry of Figure 2.89, *monitorRole* is the *username* and *QED* is the *password* corresponding to *monitorRole*. Likewise, in the second line, the *controlRole* user takes the password *R&D*.

6. If you want to use one of these pre-defined *username password* pairs during test configuration, then simply uncomment the corresponding entry by removing the *#* symbol preceding that entry. However, prior to that, you need to determine what privileges have been granted to both these users. For that, open the *jmxremote.access* file in the editor.

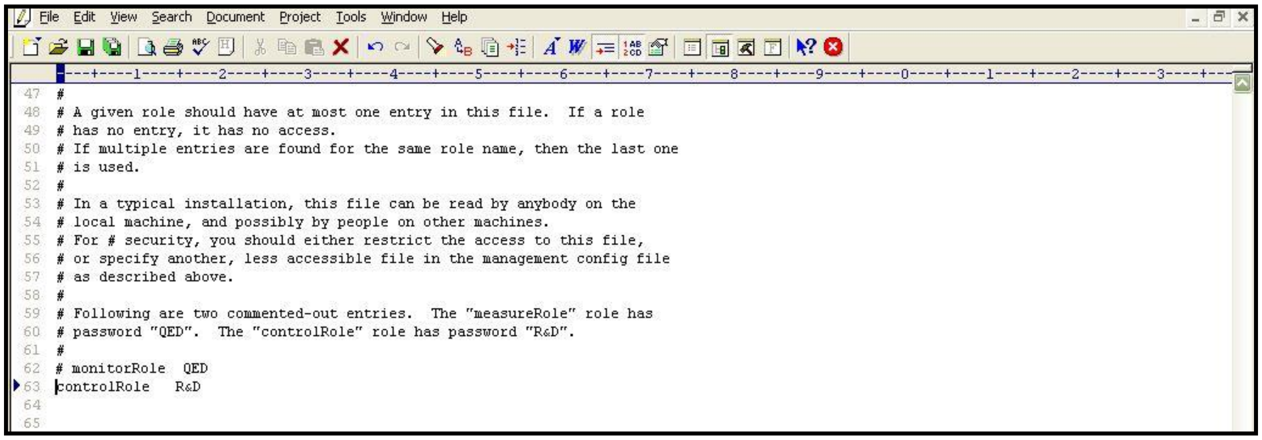


```
#       "readwrite" grants access to read and write attributes of MBeans,
#                   to invoke operations on them, and to create or remove them.
#                   This access should be granted to only trusted clients,
#                   since they can potentially interfere with the smooth
#                   operation of a running program
#
# A given role should have at most one entry in this file.  If a role
# has no entry, it has no access.
# If multiple entries are found for the same role name, then the last
# access entry is used.
#
#
# Default access control entries:
# o The "monitorRole" role has readonly access.
# o The "controlRole" role has readwrite access.

monitorRole   readonly
controlRole   readwrite
```

Figure 2.90: The jmxremote.access file

7. Scrolling down the file (as indicated by Figure 2.90) will reveal 2 lines, each corresponding to the sample *username* available in the *jmxremote.password* file. Each line denotes the access rights of the corresponding user. As is evident from Figure 2.90, the user *monitorRole* has only *readonly* rights, while user *controlRole* has *readwrite* rights. Since the eG agent requires *readwrite* rights to be able to pull out key JVM-related statistics using JMX, we will have to configure the test with the credentials of the user *controlRole*.

8. For that, first, edit the *jmxremote.password* file and uncomment the *controlRole <password>* line as depicted by Figure 2.91.



```
#
# A given role should have at most one entry in this file.  If a role
# has no entry, it has no access.
# If multiple entries are found for the same role name, then the last one
# is used.
#
# In a typical installation, this file can be read by anybody on the
# local machine, and possibly by people on other machines.
# For # security, you should either restrict the access to this file,
# or specify another, less accessible file in the management config file
# as described above.
#
# Following are two commented-out entries.  The "measureRole" role has
# password "QED".  The "controlRole" role has password "R&D".
#
# monitorRole  QED
controlRole   R&D
```

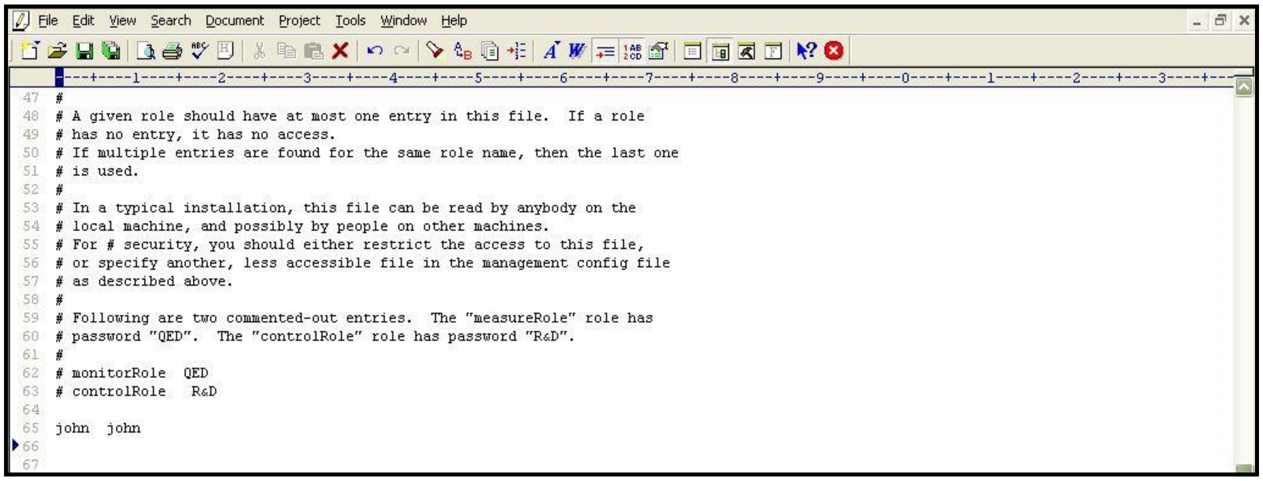
Figure 2.91: Uncommending the ‘controlRole’ line

9. Then, save the file. You can now proceed to configure the tests with the user name *controlRole* and password *R&D*.

10. Alternatively, instead of going with these default credentials, you can create a new *username password* pair in the *jmxremote.password* file, assign *readwrite* rights to this user in the *jmxremote.access* file, and then configure the eG tests with the credentials of this new user. For instance, let us create a user *john* with password *john* and assign *readwrite* rights to *john*.

11. For this purpose, first, edit the *jmxremote.password* file, and append the following line (see Figure 2.92) to it:

    john   john



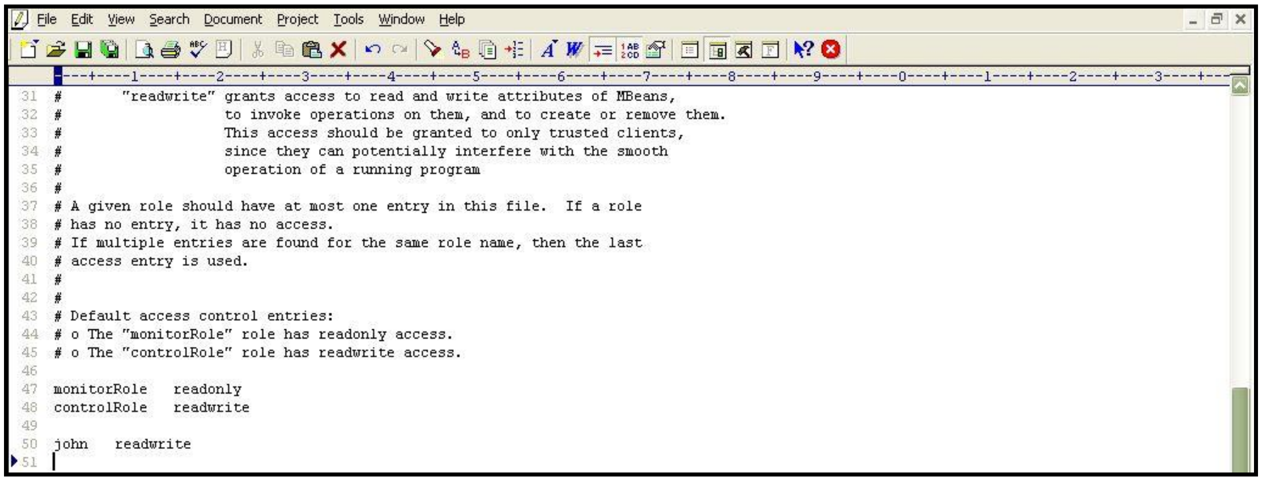
```
#
# A given role should have at most one entry in this file.  If a role
# has no entry, it has no access.
# If multiple entries are found for the same role name, then the last one
# is used.
#
# In a typical installation, this file can be read by anybody on the
# local machine, and possibly by people on other machines.
# For # security, you should either restrict the access to this file,
# or specify another, less accessible file in the management config file
# as described above.
#
# Following are two commented-out entries.  The "measureRole" role has
# password "QED".  The "controlRole" role has password "R&D".
#
# monitorRole  QED
# controlRole   R&D

john  john
```

Figure 2.92: Appending a new username password pair

12. Save the *jmxremote.password* file.

13. Then, edit the *jmxremote.access* file, and append the following line (see Figure 2.93) to it:

    john   readwrite



```
#       "readwrite" grants access to read and write attributes of MBeans,
#                   to invoke operations on them, and to create or remove them.
#                   This access should be granted to only trusted clients,
#                   since they can potentially interfere with the smooth
#                   operation of a running program
#
# A given role should have at most one entry in this file.  If a role
# has no entry, it has no access.
# If multiple entries are found for the same role name, then the last
# access entry is used.
#
#
# Default access control entries:
# o The "monitorRole" role has readonly access.
# o The "controlRole" role has readwrite access.

monitorRole   readonly
controlRole   readwrite

john   readwrite

```

Figure 2.93: Assigning rights to the new user in the jmxremote.access file

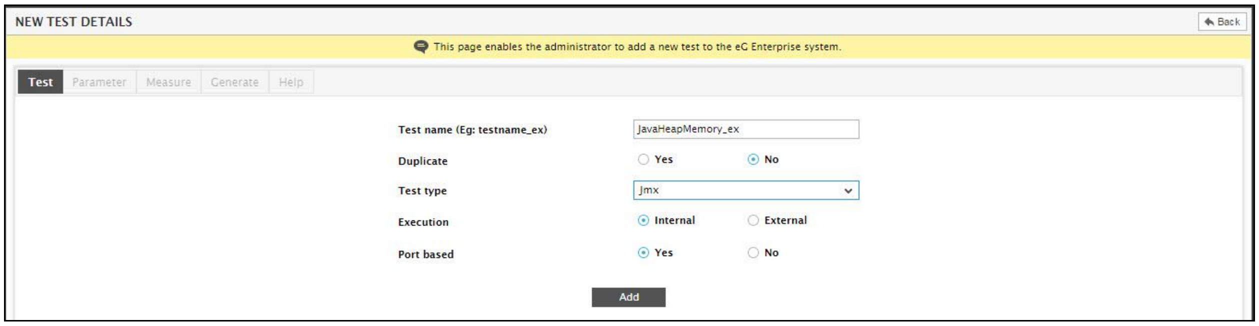
14. Then, save the *jmxremote.access* file.

15. Finally, proceed to configure the tests with the user name and password, *john* and *john*, respectively.

2.6.2 Adding a New Jmx Test

The next step involves the addition of the new test. For this purpose, first select the **Test** option from the **Integration Console** tile. Click the **Add New Test** button in the **INTEGRATION CONSOLE – TEST** page that appears next.

Figure 2.94 will then appear.



This page enables the administrator to add a new test to the eG Enterprise system.

Test name (Eg: testname_ex)	JavaHeapMemory_ex
Duplicate	Yes [✓ No]
Test type	Jmx
Execution	[✓ Internal] External
Port based	[✓ Yes] No

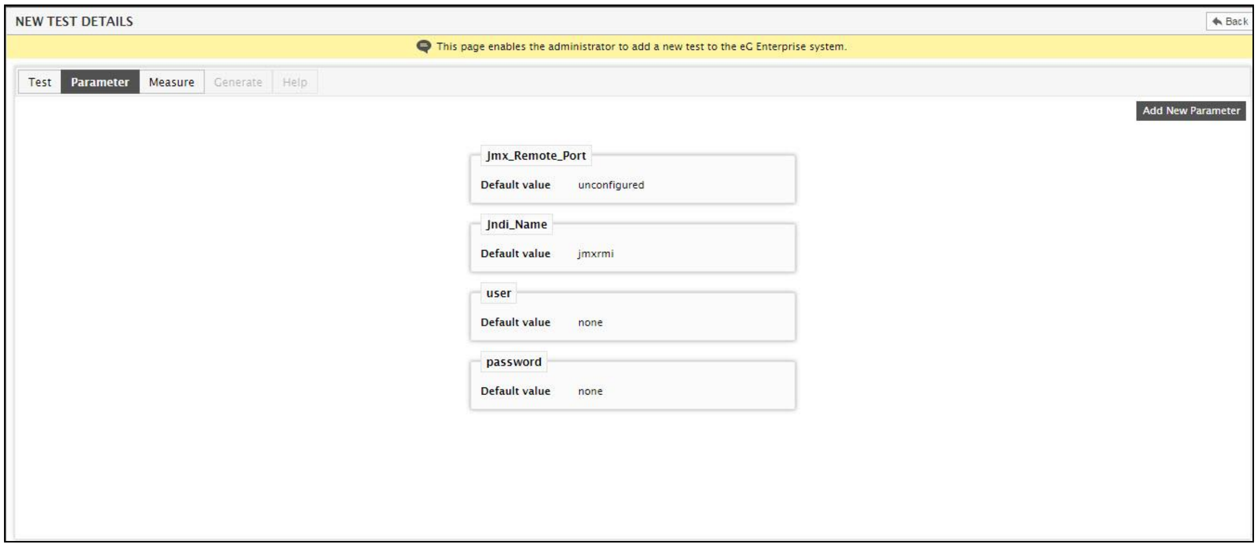
Figure 2.94: Adding a new JMX test

In Figure 2.94, provide the **Test name**, set **Duplicate** to **No** as the new test is not a duplicate of any existing IC test, indicate the **Execution** mode, and also mention whether the test is a **Port based** test or not. Finally, select **Jmx** from the **Test type** list and click the **Add** button to add the test.

Note

Using the Integration Console plugin, you can add an internal or an external test, but you cannot add tests that need to be run by a remote agent – i.e., tests that need to be executed in an agentless manner.

This will automatically lead you to the **Parameter** tab page (see Figure 2.95).


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Jmx_Remote_Port	
Default value	unconfigured
Jndi_Name	
Default value	jmxrmi
user	
Default value	none
password	
Default value	none

Figure 2.95: Viewing the default parameters of the Jmx test

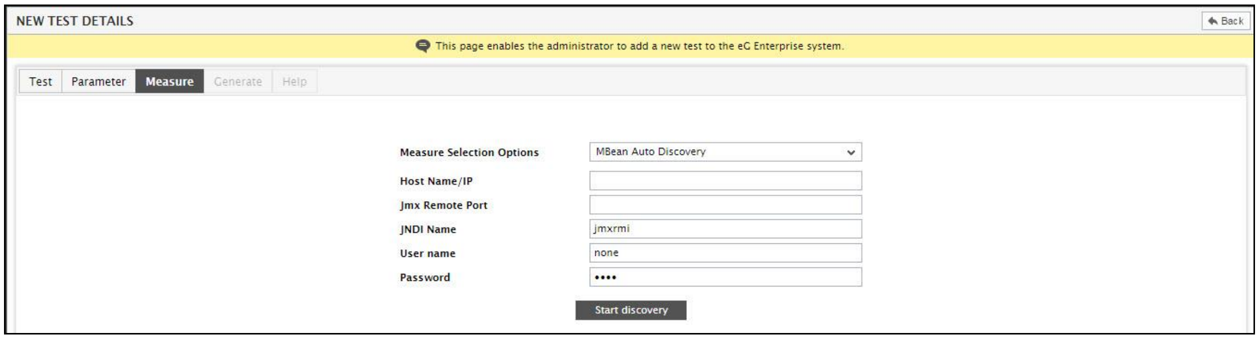
By default, any new test of type **Jmx** will take the following parameters:

- **JMX_REMOTE_PORT**: By default, this parameter is set to *unconfigured*. This implies that while configuring the *JavaHeapMemory_ex* test for the *javaapp* application in our example, you should manually configure this parameter with the port number at which **JMX** listens for requests from remote hosts. During test configuration, ensure that you specify the same port that you configured in the *management.properties* file; by default, this file will be in the **<JAVA_HOME>\jre\lib\management** folder used by the target application (see page 75).

- **JNDI_NAME**: The **JNDINAME** is a lookup name for connecting to the JMX connector. By default, this is set to *jmxrmi*. If you have registered the JMX connector in the RMI registery using a different lookup name, then, while managing the *javaapp* application and configuring its tests, change this default value to reflect the change in the lookup name.

- **USER** and **PASSWORD**: By default, both these parameters are set to *none*. However, if JMX requires **authentication only** (but no security), then, at the time of configuring this test for the *Javaapp* application (in our example), ensure that the **USER** and **PASSWORD** parameters are configured with the credentials of a user with *read-write* access to JMX. To know how to create this user, refer to Section 2.6.1.2.

Since no additional parameters need to be added for the **JavaHeapMemory_ex** test in our example, simply click on the **Measure** tab page in Figure 2.95 to add a new measure. Figure 2.96 will then appear.


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Measure Selection Options	MBean Auto Discovery
Host Name/IP	
Jmx Remote Port	
JNDI Name	jmxrmi
User name	none
Password	••••

Figure 2.96: Adding a new measure for the new Jmx test

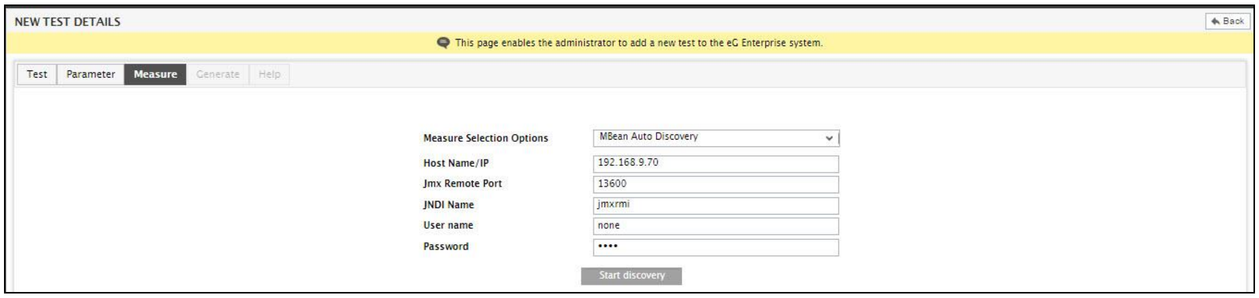
Figure 2.96 allows you the flexibility to choose from multiple methods for configuring the measures of the new Jmx test; these methods are as follows:

- **MBean Auto Discovery**
- **Load MBeans from File**
- **Manual Entry**

To help you understand when and how each of these methods should be used, we will be taking the example of the *Heap_memory_used* measure of this test and will illustrate how this measure can be added using each of the methodologies listed above.

2.6.2.1 Adding a Measure Using MBean Auto Discovery

The **MBean Auto Discovery** option enables the eG manager to automatically discover the domains and MBeans supported by a target Java application, so that you can configure any of the Mbean attributes as a measure with minimal manual effort. Select this option if you do not know the exact Mbean attribute name to be configured as a measure for a new JMX test. When this option is chosen, you will have to additionally specify the following in the **NEW TEST DETAILS** page as depicted by Figure 2.97 below:


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Measure Selection Options	MBean Auto Discovery
Host Name/IP	192.168.9.70
Jmx Remote Port	13600
JNDI Name	jmxrmi
User name	none
Password	••••

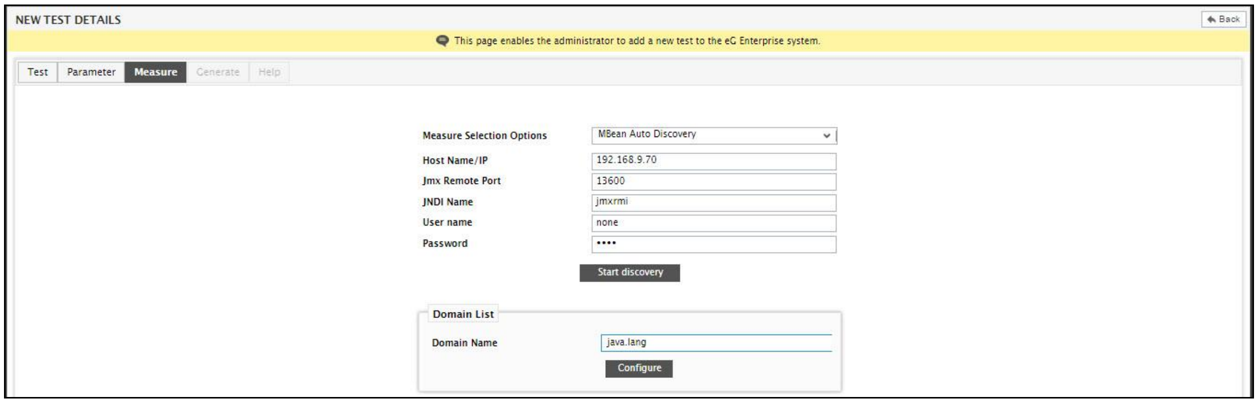
Figure 2.97: Configuring the auto-discovery of MBeans

- **Host Name / IP**: Specify the host name / IP address of the system hosting the *javaapp* in our example;
- **Jmx Remote Port**: Indicate the port at which the JMX of the *javaapp* in our example listens;
- **JNDI name**: Specify the lookup name for connecting to the JMX connector. By default, this is set to *jmxrmi*.
- **User name** and **Password**: By default, both these parameters are set to *none*. However, if JMX requires

authentication only (but no security), then, ensure that these fields are configured with the credentials of a user with *read-write* access to JMX. To know how to create this user, refer to Section 2.6.1.2.

Finally, click the **Start discovery** button.

Doing so will automatically populate the **Domain Name** drop-down list with the complete list of domains supported by the Java application in our example. From this list, pick the domain that contains the MBeans of interest to us – for our example, pick **java.lang** as the domain (see Figure 2.98).


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Measure Selection Options	MBean Auto Discovery
Host Name/IP	192.168.9.70
Jmx Remote Port	13600
JNDI Name	jmxrmi
User name	none
Password	••••

Domain List

Domain Name: java.lang

Figure 2.98: Selecting a domain for MBean discovery

To proceed with the configuration, click on the **Configure** button in Figure 2.98.

Doing so will invoke Figure 2.99 using which a new measure can be added. Let us begin by adding the *Heap_memory_used* measure. In order to achieve this, follow the steps given below:

1. Specify *Heap_memory_used* as the **Measure name.**
2. Select **Number(20.4)** as the **Database column size**.
3. Pick **MB** as the **Unit** of the measurement.
4. Pick **UNALTERED** as the **Process method.**

Refer

To know more about the **Process method**, refer to Page 41 of this document.

5. Since **java.lang** has been chosen as the **Domain name**, the eG manager will automatically discover all the MBeans that are available within **java.lang**, and will make them available for selection in the **Mbean name** list. From this list, choose the MBean that reports memory-related statistics for the *javaapp* application in our example. This would be **Memory** for our example.

Note

Once a **Domain** is chosen for a test, all the measures of that test should be based on the MBeans and attributes within that domain only.

6. Upon selection of an **Mbean name**, all the attributes of the chosen Mbean will be automatically discovered and displayed as the options of the **Attribute name** list. Now, from the **Attribute name** list, select the Mbean attribute that reports how much heap memory has been utilized by the *javaapp* application in our example – for our example, the attribute to be chosen is **HeapMemoryUsage>used**. **Note that, by default, eG Enterprise monitors the chosen 'Attribute' across all those MBeans to which it applies.**

7. The chosen attribute reports heap memory usage in Bytes. However, the **Unit** of measurement that we have set for the *Heap_memory_used* measure is **MB**. This implies that the eG manager should first convert the number of bytes into MB and then display the final output in the eG monitoring console. To enable bytes to MB conversion, you need to pick a **Conversion Factor**. The options available by default in the **Conversion Factor** drop-down list, do not allow 'Bytes to MB' conversion. You can however include this capability into the eG Enterprise system by following the steps discussed below:

   - Edit the **eg_ui.ini** file in the **<EG_INSTALL_DIR>\manager\config** directory.

   - To include a new conversion factor, you will have to append an entry of the following format to the **[CONVERSION_FACTORS]** section of the file:

     DisplayName=Value

   - For instance, to support 'Bytes to MB' conversion, append the following entry to the **[CONVERSION_FACTORS}** section:

     /1048576 (Bytes to MB)=0.00000095367431640625

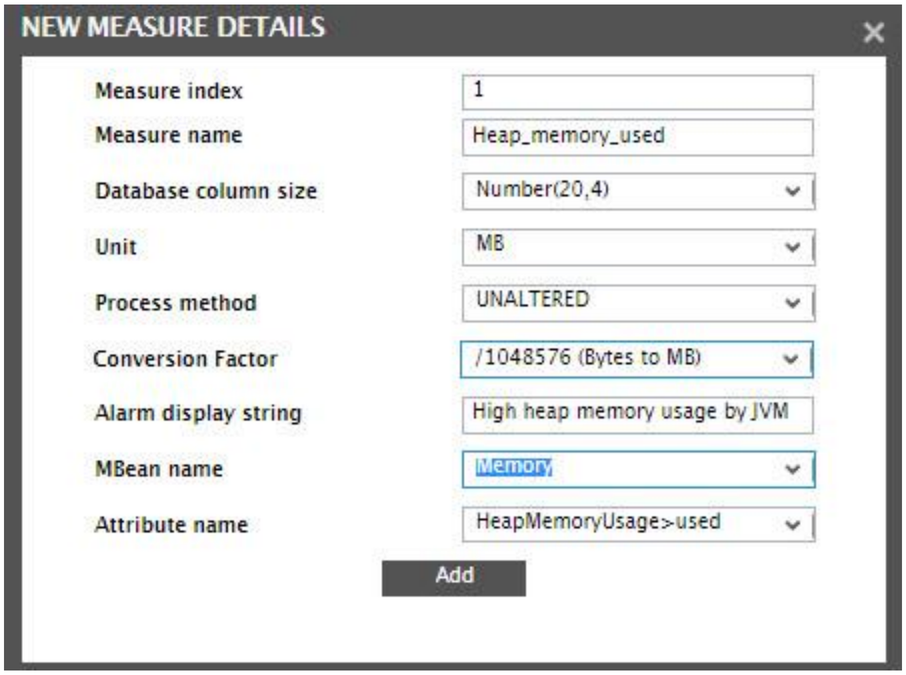
   - In this case, the *DisplayName,* /**1048576 (Bytes to MB)**, will be displayed as an option in the **Conversion Factor** drop-down list. If this option is chosen, then, at test run time, the conversion value of **0.00000095367431640625** will be multiplied with the actual measure value that is reported in Bytes to convert it into MB. **Care should be taken while specifying the conversion value, as incorrect values will result in wrong measures being reported by the test.**

   - Once the new entry is appended to the **[CONVERSION_FACTORS]** section, save the file.

   - Once this is done, you will find the string /**1048576 (Bytes to MB)** appear as an option in the **Conversion Factor** list. For the purpose of our example, pick this as the conversion factor.

8. Provide a brief description of the alarm in the **Alarm display string** text box.

9. Finally, click the **Add** button to add the new measure.



Measure index	1
Measure name	Heap_memory_used
Database column size	Number(20,4)
Unit	MB
Process method	UNALTERED
Conversion Factor	/1048576 (Bytes to MB)
Alarm display string	High heap memory usage by JVM
MBean name	Memory
Attribute name	HeapMemoryUsage>used

Figure 2.99: Adding the Heap_memory_used measure of the Jmx test using the MBean auto-discovery method

2.6.2.2 Adding a New Measure by Loading MBeans from a File

For **Mbean Autodiscovery** to work, the eG manager should be able to access the target application. However, in environments where connectivity issues exist between the eG manager and the target Java application say, owing to strict firewall rules, autodiscovery of MBeans may not be possible. In such situations, you can enable the eG agent to load the MBeans of an application onto a file, and then make that file available on the eG manager host, so that the eG manager can download the MBeans from the file. The first step towards this end is to generate the file to which MBeans are to be loaded.

The eG agent is bundled with an **AgentMbeanDiscovery.bat** file, which is available in the **<EG_INSTALL_DIR>\lib** directory of the eG agent host. If this batch file is executed, it requests for information that enables access to the target Java application; upon accessing the application, the batch file automatically discovers the MBeans of that application and loads them on to a file.

Like the eG agent, the eG manager is also bundled with a batch file named **MgrMbeanDiscovery.bat**. This file is available in the **<EG_INSTALL_DIR>\lib** directory of the eG manager host. The **MgrMbeanDiscovery.bat** file can be used instead of the **AgentMbeanDiscovery.bat**, only if all the following conditions prevail:

- The target application should be available on the eG manager host, but the eG manager should not be able to autodiscover the MBeans using the procedure discussed in Section 2.6.2.1
- No agent should be available on the eG manager host;
- No other agent should be able to access the target application.

For instance, you could have a redundant manager setup, where the target Java application executes on a secondary manager host; this host may not have any agent executing on it. Moreover, for security reasons, the firewall could have been configured in such a way that both the primary manager and the eG agents are denied access to the target application.

In such a case, you may first consider auto-discovering the MBeans using the procedure discussed in Section 2.6.2.1.

Since this requires access to the eG administrative interface, it cannot be performed from the secondary manager; also, since firewall rules disallow the primary manager-Java application communication, autodiscovery cannot be performed via the primary manager either. Moreover, as no agent has been deployed on the secondary manager host and because the target application is inaccessible to all other agents, the **AgentMbeanDiscovery.bat** file too cannot be used. In such a case, you can use the **MgrMbeanDiscovery.bat** file on the secondary manager host to load the Mbeans of the application to a file.

Follow the steps below to load the MBeans of the application to a file using the **AgentMbeanDiscovery.bat** file:

1. Go to the command prompt of the agent host.
2. Switch to the **<EG_INSTALL_DIR>\lib** directory.
3. To load the MBeans of the *javaapp* application on to a file, issue the following command at the prompt: **AgentMbeanDiscovery.bat**
4. Upon execution, the batch file will request you to specify the following:

```
Enter the Target Host IP:
Enter the Jmx port:
Enter the JNDIPath:
Enter the Username:
Enter the Password:
```

Specify the IP address of the target application, the JMX port at which the application listens, the JNDI lookup name of the JMX connector, and the username and password of the JMX.

5. Based on the information provided, the batch file auto-discovers the MBeans of the application, generates a file of the format *MBeans_IpofApplication_JMXPort.txt* in the **<EG_AGENT_INSTALL_DIR>\lib** directory, and copies the discovered MBeans to the file.
6. Finally, make sure that the file so created is available in the **<EG_MANAGER_INSTALL_DIR>\tmp** folder on the eG manager host. To achieve this, you can opt for either of the following:
   a. You can quickly upload the file to the **<EG_MANAGER_INSTALL_DIR>\tmp** on the eG manager host from the eG administrative interface;
   b. You can manually copy the file so created to the **<EG_MANAGER_INSTALL_DIR>\tmp** directory on the eG manager host

Note

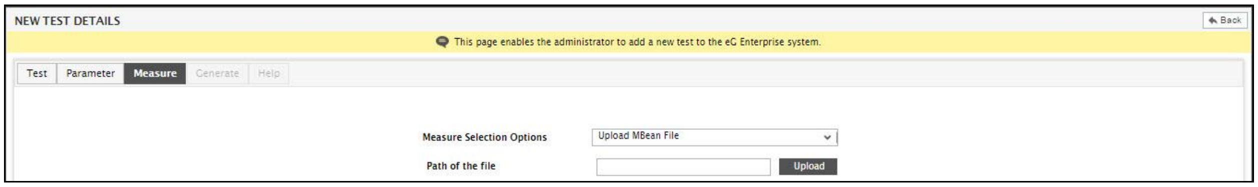
The **MgrMbeanDiscovery.bat** file is to be executed the same way as the **AgentMbeanDiscovery.bat**. However, the difference in the case of the **MgrMbeanDiscovery.bat** file is, you need to login to the eG manager host to run the batch file, and the command to be executed is: **MgrMbeanDiscovery.bat**. Also, since the MBean file will be created in the **<EG_MANAGER_INSTALL_DIR>\lib** directory, you do not have to once again copy/upload the file to the eG manager host.

If you choose option (a) – i.e., if you choose to upload the file via the eG administrative interface – then follow the procedure discussed in Section 2.6.2.2.1 to upload the file and to use it for configuring a measure. On the other hand, if you have manually copied the file to the eG manager host (as suggested by option (b)), then, such a file will pre-exist on the eG manager host at the time of measure configuration. To know how to use an existing MBean file for configuring a measure, refer to Section 2.6.2.2.2.

2.6.2.2.1 Uploading an MBean File to the eG Manager Host and Using that File to Add a New Measure

For our example, we will be configuring the *Heap_memory_used* measure using the Mbean attributes loaded from a file. Now, proceed as discussed below:

1. Select the **Upload MBean File** option in Figure 2.100 to upload the file to the eG manager host.



NEW TEST DETAILS

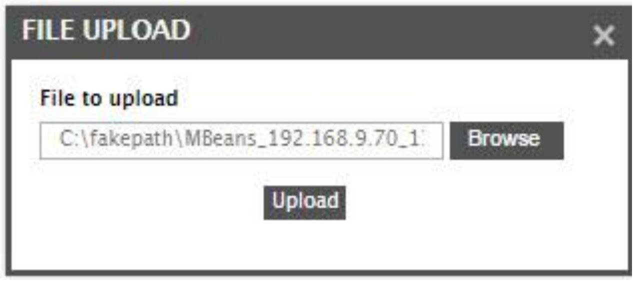
This page enables the administrator to add a new test to the eG Enterprise system.

Measure Selection Options: Upload MBean File

Path of the file:

Figure 2.100: Selecting the option to load MBeans from a file

2. Next, specify the full path to the MBeans file that was generated previously in the **Path of the file** text box. To browse for the file path and upload it, click the **Upload** button in Figure 2.100. Figure 2.101 will then appear.



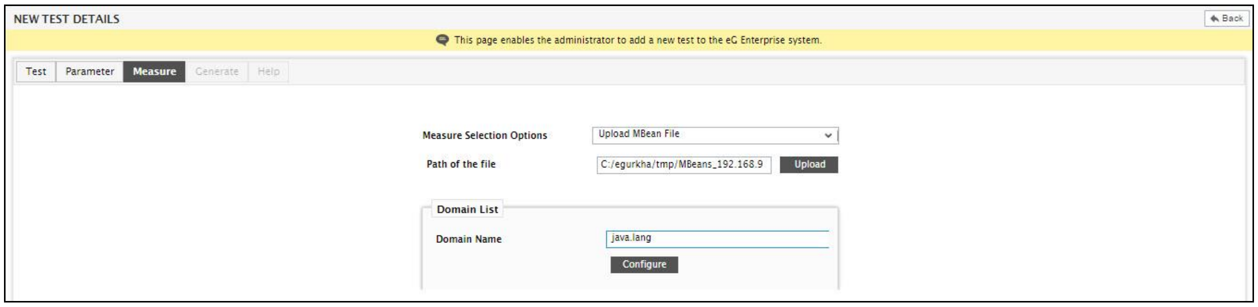
FILE UPLOAD

File to upload: C:\fakepath\MBeans_192.168.9.70_1

Figure 2.101: Browsing for the file and uploading it

3. Click on the **Browse** button in Figure 2.101 to browse for the file path, and then click the **Upload** button therein to upload the file.

4. If the file is successfully uploaded to the eG manager, then the path to which the file has been uploaded will be displayed in the **Path of the file** text box as depicted by Figure 2.102.



NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Measure Selection Options: Upload MBean File

Path of the file: C:/egurkha/tmp/MBeans_192.168.9

Domain List

Domain Name: java.lang

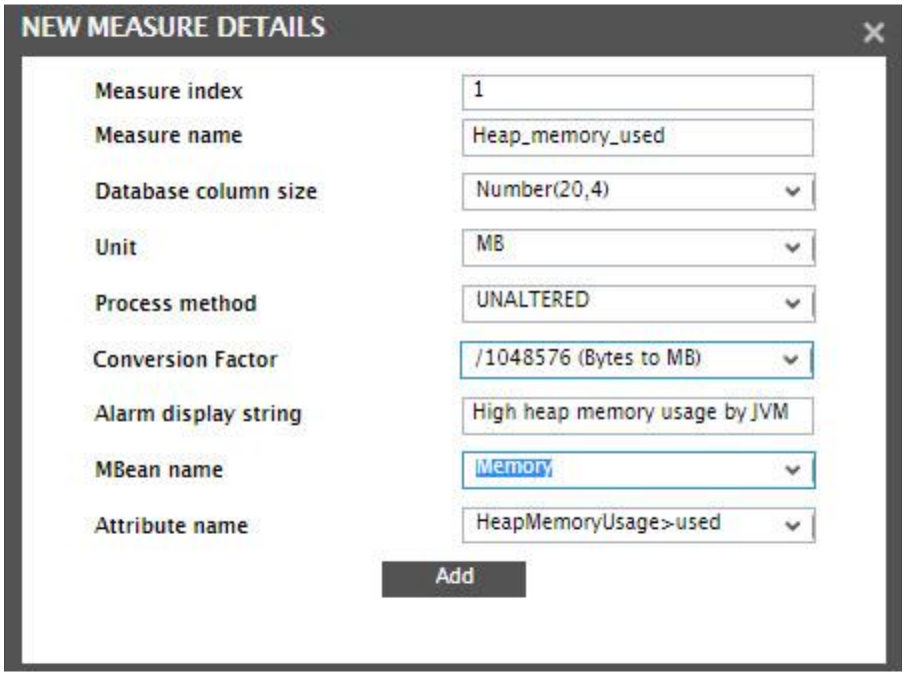
Figure 2.102: The path to which the MBean file is uploaded being displayed against Path of the file

5. If a valid MBean file has been uploaded, then the eG Enterprise system will automatically discover the domains from the uploaded file and display the same in the **Domain Name** list of Figure 2.102. To proceed with the measure configuration, pick a domain from the list. For our example, pick **java.lang** as the **Domain Name.**

Note

If a test reports multiple measures, then the MBeans and/or attribute names mapped to all such measures should belong to the same **Domain**.

6. Then, click the **Configure** button in Figure 2.102, so that the **NEW MEASURE DETAILS** window appears (see Figure 2.103).



NEW MEASURE DETAILS

Measure index	1
Measure name	Heap_memory_used
Database column size	Number(20,4)
Unit	MB
Process method	UNALTERED
Conversion Factor	/1048576 (Bytes to MB)
Alarm display string	High heap memory usage by JVM
MBean name	Memory
Attribute name	HeapMemoryUsage>used

Figure 2.103: Configuring the Heap_memory_used measure by loading MBeans from a file uploaded to the eG manager

7. Configure the *Heap_memory_used* measure in the same way as discussed in Section 2.6.2.2.1.

8. Finally, click the **Add** button.

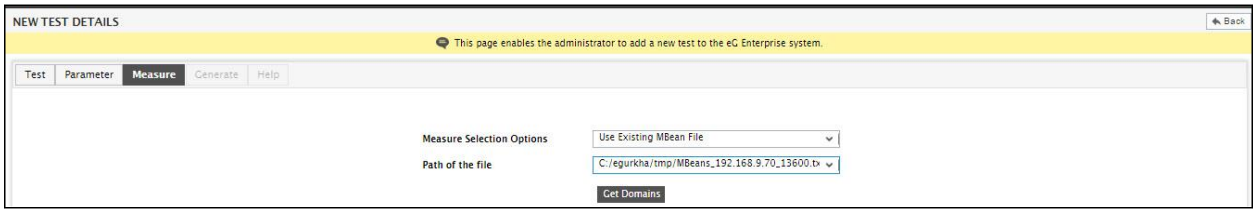
2.6.2.2.2 Adding a Measure Using an Existing MBean File

An MBean file may pre-exist in the **<EG_MANAGER_INSTALL_DIR>\tmp** directory in the eG manager host in the following situations:

- The MBean file may have been manually copied to the eG manager host;
- The MBean file may have already been uploaded to the eG manager using steps 1 to 5 of Section 2.6.2.2.2.
- The MBean file may have been created using the **MgrMbeanDiscovery.bat** file (and not the **AgentMbeanDiscovery.bat**)

To use an existing MBean file to configure a new measure for a **JMX** test, do the following:

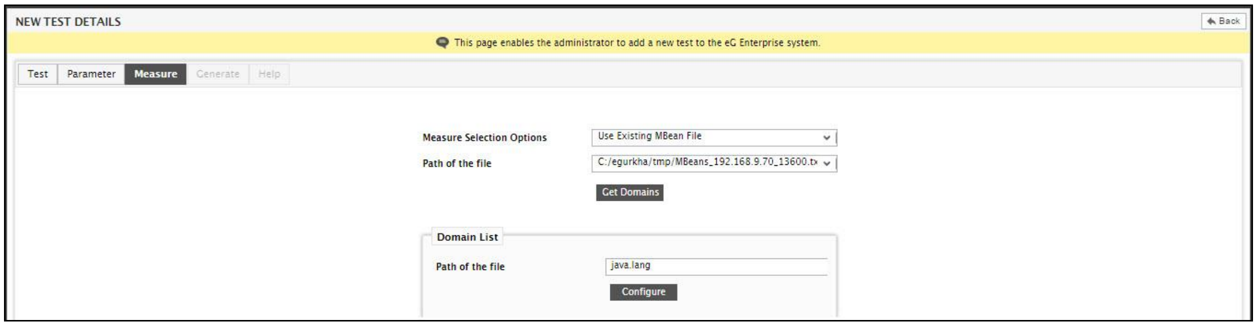
1. Select the **Use Existing MBean File** option (see Figure 2.104). Once this option is chosen, then all the MBean files that pre-exist in the **<EG_MANAGER_INSTALL_DIR>\tmp** directory in the eG manager host will populate the **Path of the file** drop-down. From this drop-down, select the file that you want to use for measure configuration.


NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Get Domains

Measure Selection Options: Use Existing MBean File
Path of the file: C:/egurkha/tmp/MBeans_192.168.9.70_13600.tx

Figure 2.104: Using an existing MBean file

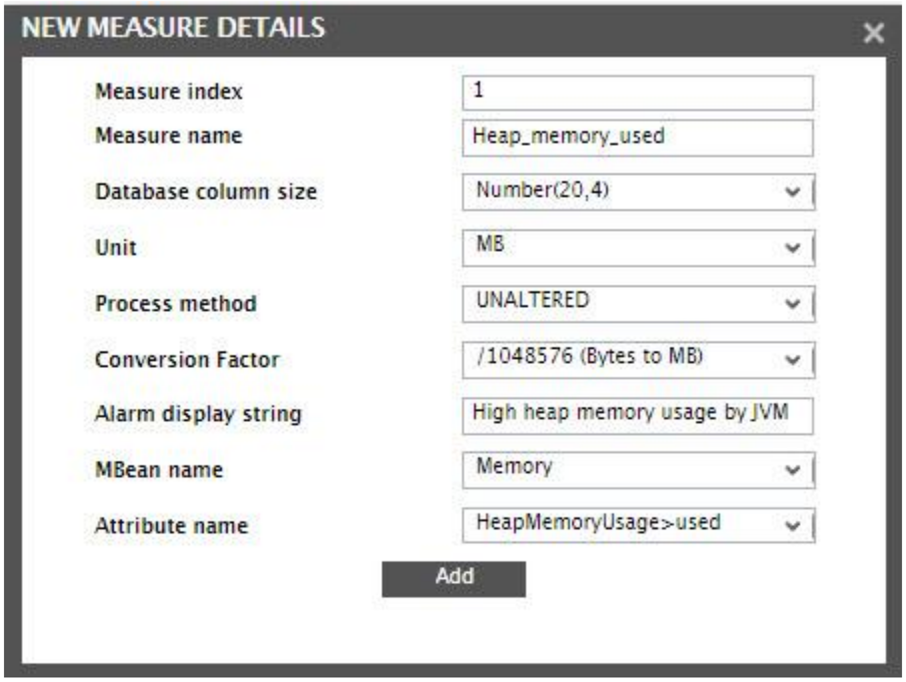
2. Then, click the **Get Domains** button to get the domains. The eG Enterprise system will then automatically discover the domains from the chosen file and display the same in the **Domain Name** list of Figure 2.105. Pick a domain and and click the **Configure** button to proceed with the measure configuration.



Measure Selection Options: Use Existing MBean File
Path of the file: C:/egurkha/tmp/MBeans_192.168.9.70_13600.tx
Path of the file: java.lang

Figure 2.105: Selecting a domain from the domains discovered from an existing MBean file

3. When Figure 2.106 appears, add the *Heap_meory_used* measure using the steps 1-9 in Section 2.6.2.1 of this document.


NEW MEASURE DETAILS

Measure index	1
Measure name	Heap_memory_used
Database column size	Number(20,4)
Unit	MB
Process method	UNALTERED
Conversion Factor	/1048576 (Bytes to MB)
Alarm display string	High heap memory usage by JVM
MBean name	Memory
Attribute name	HeapMemoryUsage>used

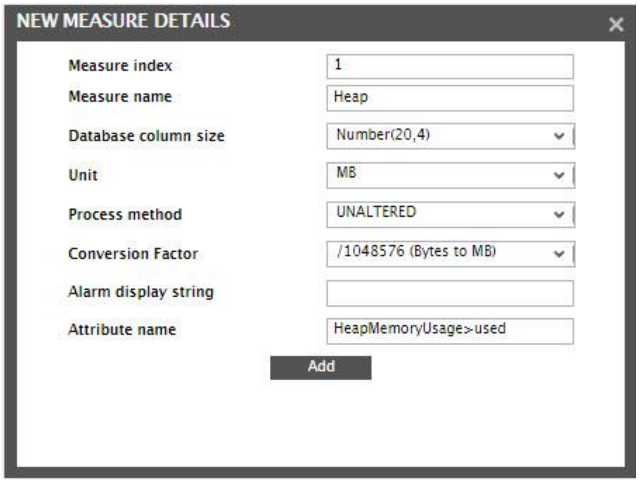
Figure 2.106: Adding the Heap_memory_used measure by discovering domains from an existing MBean file

4. Finally, click the **Add** button in Figure 2.106 to add the measure.

2.6.2.3 Adding a Measure Using the Manual Entry Method

Use the **Manual Entry** method if you know the exact domain, Mbean, and attribute that can report the measure you want, and do not wish to needlessly auto-discover MBeans or load them from a file for this purpose.

For the purpose of our example, let us add the *Heap_memory_used* measure, but this time using the manual entry method. To achieve this, first pick the **Manual Entry** option from Figure 2.97. Then, proceed to configure the measure as depicted by Figure 2.107.


NEW MEASURE DETAILS

Measure index	1
Measure name	Heap
Database column size	Number(20,4)
Unit	MB
Process method	UNALTERED
Conversion Factor	/1048576 (Bytes to MB)
Alarm display string	
Attribute name	HeapMemoryUsage>used

Figure 2.107: Configuring the Heap_memory_used measure manually

In Figure 2.107, specify the following:

- Specify *Heap_memory_used* as the **Measure name.**
- Pick **Number(20,4)** as the **Database column size**.
- Choose **MB** as the **Unit** of measurement.
- Select **UNALTERED** as the **Process Method**.

Refer

To know more about the **Process method**, refer to Page 41 of this document.

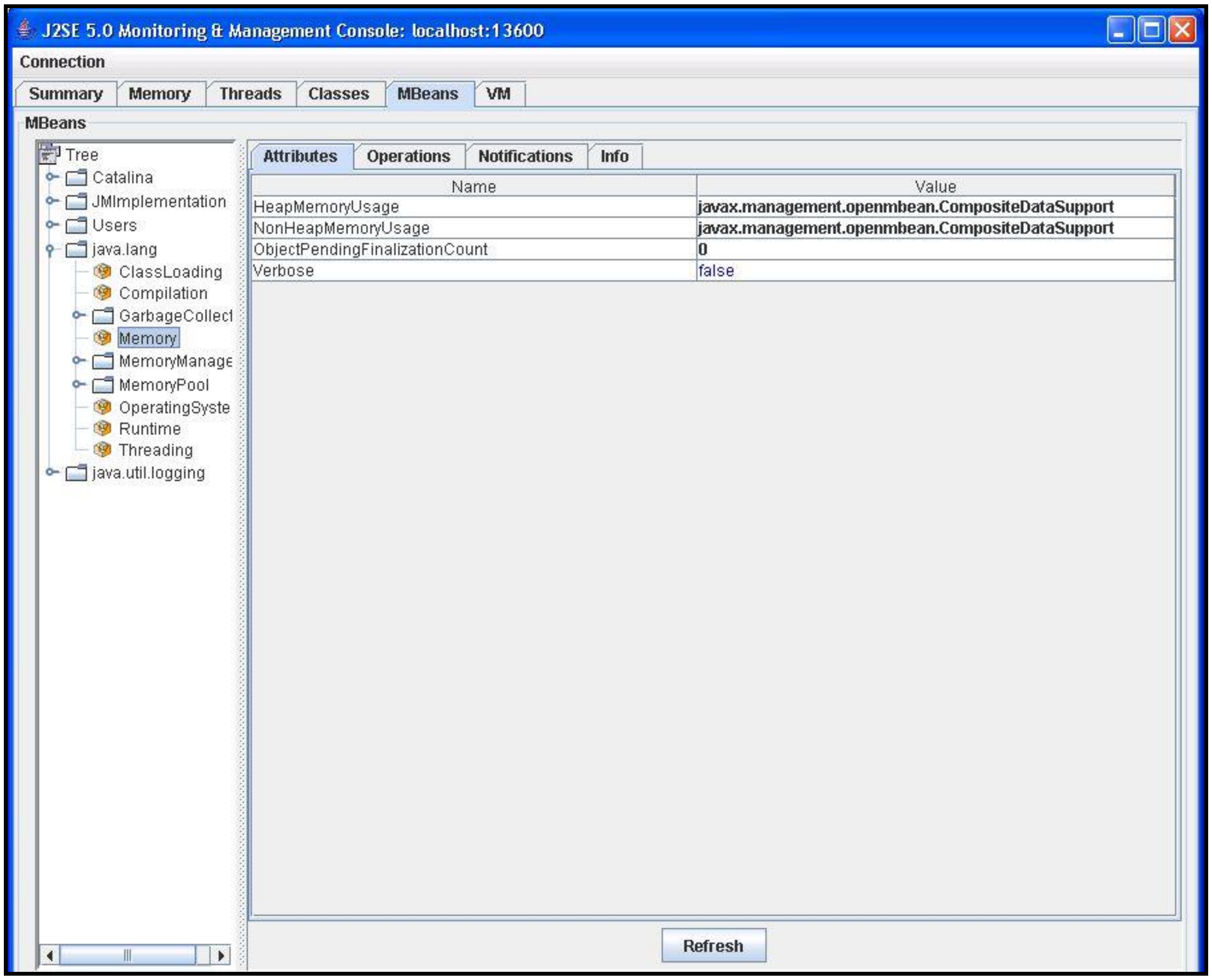
- Specify **HeapMemoryUsage>committed** as the **Attribute name** that reports the initial heap memory. **Note that, by default, eG Enterprise monitors the chosen 'Attribute' across all those MBeans to which it applies**.

Note

If a test reports multiple measures, then the MBean and/or attribute name that is included in each measure specification should belong to the same domain that was chosen for the first measure.

To identify the name of the attribute, do the following:

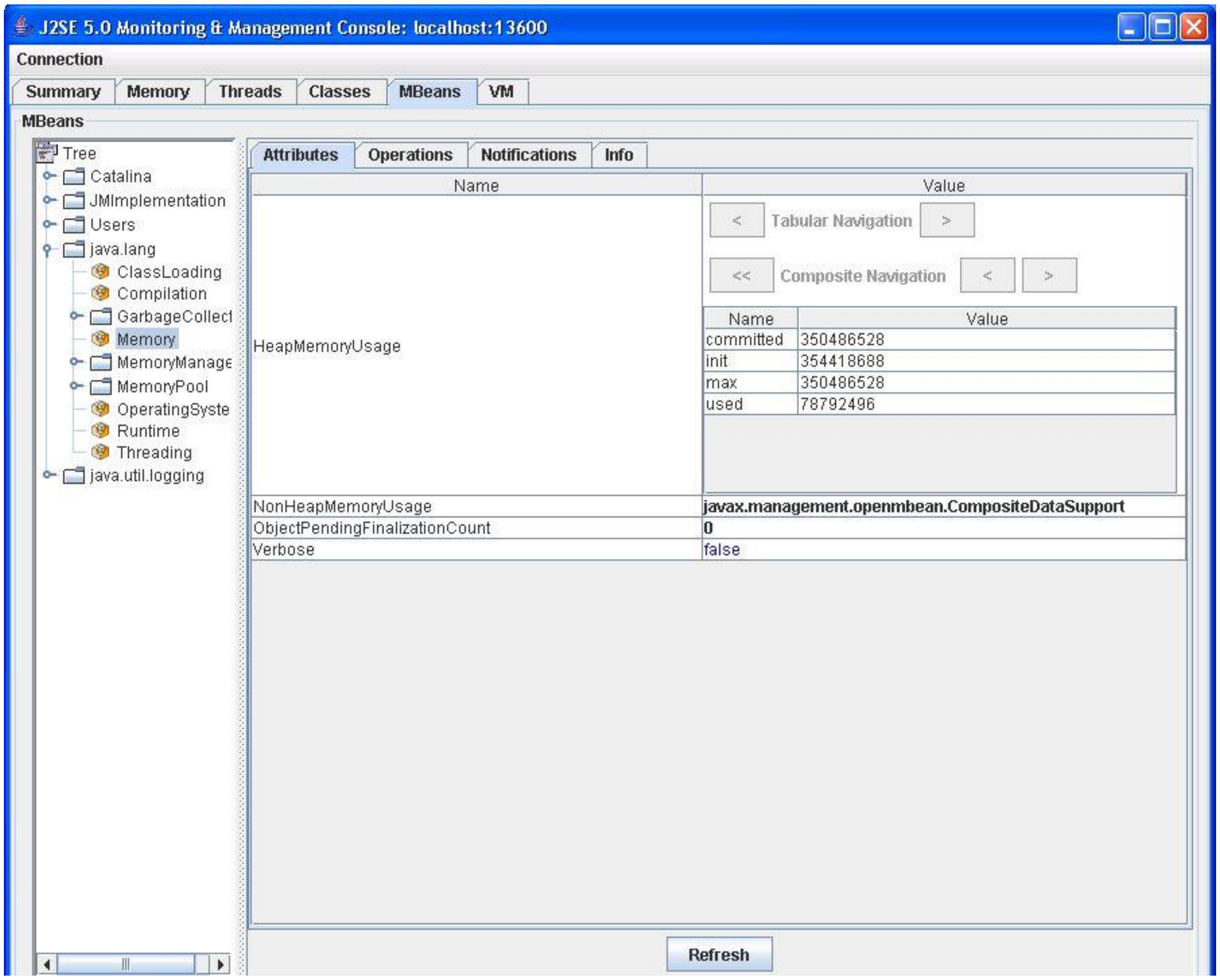
- Go to the command prompt on the target Java application host;
- Switch to the **<JAVA_INSTALL_DIR>\bin** directory.
- Issue the **jconsole** command.
- Figure 2.108 will then appear.



Name	Value
HeapMemoryUsage	javax.management.openmbean.CompositeDataSupport
NonHeapMemoryUsage	javax.management.openmbean.CompositeDataSupport
ObjectPendingFinalizationCount	0
Verbose	false

Figure 2.108: Jconsole

- In the **Tree** structure in the left panel of Figure 2.108 locate the **java.lang** domain to which the Mbean of interest to our example belongs.
- Expand the **java.lang** domain; since the *Heap_memory_used* measure in our example is a *memory* related measure, click on the **Memory** Mbean within the **java.lang** domain.
- The right panel of Figure 2.108 will change to display the **Attributes** of the **Memory** Mbean. The top attribute is the **HeapMemoryUsage** attribute, which reports **commited** heap memory as its sub-attribute.
- Therefore, to drill down to the **heap memory committed** attribute, click on the **HeapMemoryUsage** attribute in the right panel.
- Figure 2.109 will then appear.



Name	Value
HeapMemoryUsage	
NonHeapMemoryUsage	javax.management.openmbean.CompositeDataSupport
ObjectPendingFinalizationCount	0
Verbose	false

Name	Value
committed	350486528
init	354418688
max	350486528
used	78792496

Figure 2.109: Drilling down to the 'used' attribute

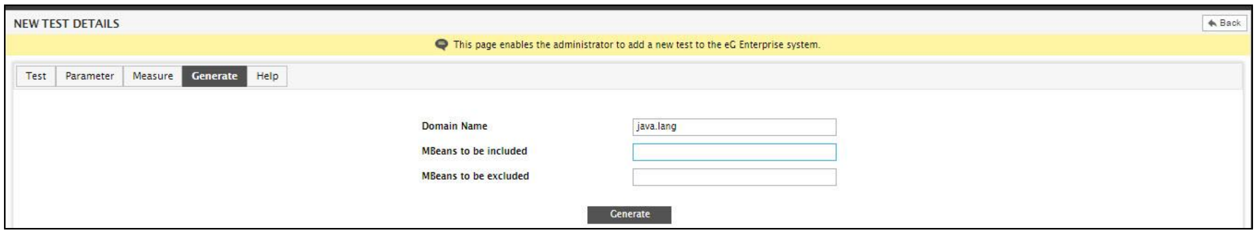
- Figure 2.109 displays the sub-attributes of the **HeapMemoryUsage** attribute. One of these sub-attrbutes is the **used** attribute.
- Therefore, against the **Attribute name** field in Figure 2.107, type both the main attribute – **HeapMemoryUsage**- and its sub-attribute – **used**- separated by the '>' symbol.

- Since the specified attribute reports the used heap memory in bytes, you need to select a **Conversion Factor** to convert the bytes into MB (which is the **Unit** of measurement of this measure). Therefore, select */1048576 (Bytes to MB)* as the **Conversion Factor**.
- If required, provide an alarm description in the **Alarm display string** text box.
- Finally, click the **Add** button in Figure 2.107.

When prompted to configure additional measures for the test, click **No** to indicate that no more measures need be added. Doing so will automatically lead you to the **Generate** tab page (see Figure 2.110).

If the measures were added using auto-discovered MBeans or the ones loaded from a file, then the **Domain** that you chose at the time of adding the measures, will be displayed against **Domain Name** in Figure 2.110. In the case of our example therefore, **java.lang** will be displayed as the default **Domain Name**, if auto-discovered/loaded MBeans were used for configuring the measures.

On the contrary, if the measures were added using the manual entry method, then no **Domain Name** will be displayed here. In this case, you will have to manually enter the **Domain Name**. This means that if the *Heap_memory_used* measure in our example had been configured by manually entering the attribute name, then, you will have to manually specify **java.lang** as the **Domain Name** here.

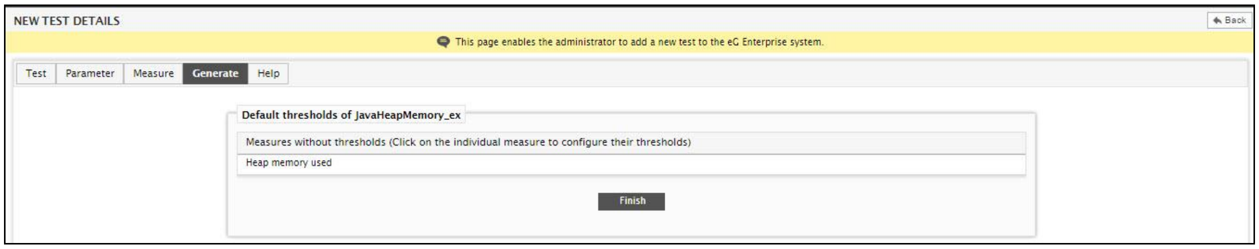

NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Generate

Domain Name	java.lang
MBeans to be included	
MBeans to be excluded	

Figure 2.110: Generating the Jmx test

Sometimes, the **Attribute** that reports the value of a measure could be associated with more than one MBean. In such a case, by default, eG Enterprise monitors the **Attribute** across all the MBeans with which it is mapped. If you want to override this default setting – i.e., if you want the new test to monitor the given **Attribute** for specific MBeans only and not all of them – then, you can use the **MBeans to be included** and **MBeans to be excluded** text boxes for this purpose. To instruct the new test to include a few specific MBeans alone in its monitoring scope, provide a comma-separated list of MBeans in the **MBeans to be included** text box. Similarly, if you want the test to exclude specific MBeans from monitoring, provide a comma-separated list of these MBeans in the **MBeans to be excluded** text box. Note that these specifications too are not measure-specific and apply to all the measures configured for a test.

Click on the **Generate** button in Figure 2.110 to generate the measures of the test. Figure 2.111 then appears allowing you to define the thresholds for the measure that we had configured earlier for the *JavaHeapMemory_ex* test in our example.


NEW TEST DETAILS
Back
This page enables the administrator to add a new test to the eG Enterprise system.
Test
Parameter
Measure
Generate
Help
Finish

Default thresholds of JavaHeapMemory_ex

Measures without thresholds (Click on the individual measure to configure their thresholds)

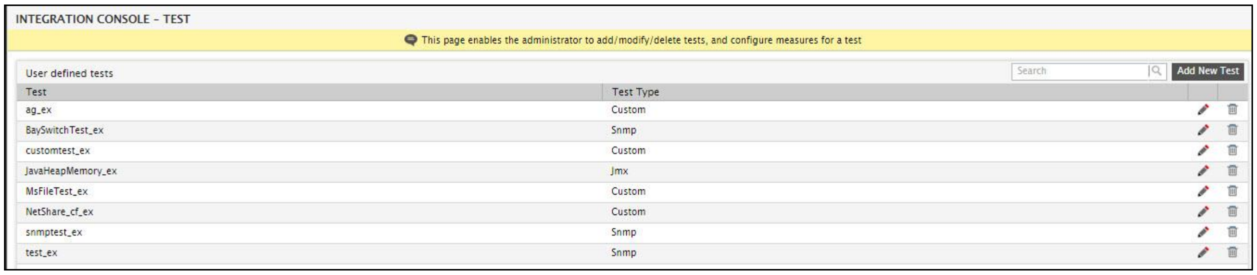
Heap memory used

Figure 2.111: Defining the thresholds for the measures configured for the JavaHeapMemory_ex test

2.7 Modifying/Deleting Tests Added Using the Integration Console

To delete a test or to modify the configuration and measures of a test that is added using the Integration Console, follow the steps below:

1. Select the **Test** option from the **Integration Console** tile in Figure 2.1.
2. Figure 2.112 will then appear, listing all the tests that have been added using the Integration Console.


INTEGRATION CONSOLE – TEST

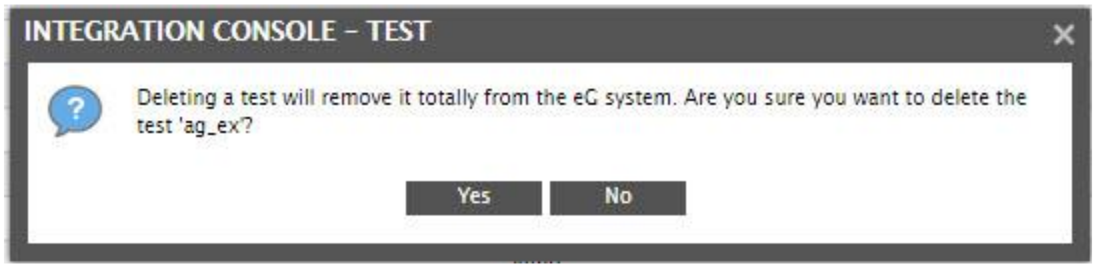
This page enables the administrator to add/modify/delete tests, and configure measures for a test

User defined tests

Test	Test Type
ag_ex	Custom
BaySwitchTest_ex	Snmp
customtest_ex	Custom
JavaHeapMemory_ex	Jmx
MsFileTest_ex	Custom
NetShare_cf_ex	Custom
snmptest_ex	Snmp
test_ex	Snmp

Figure 2.112: List of tests that pre-exist

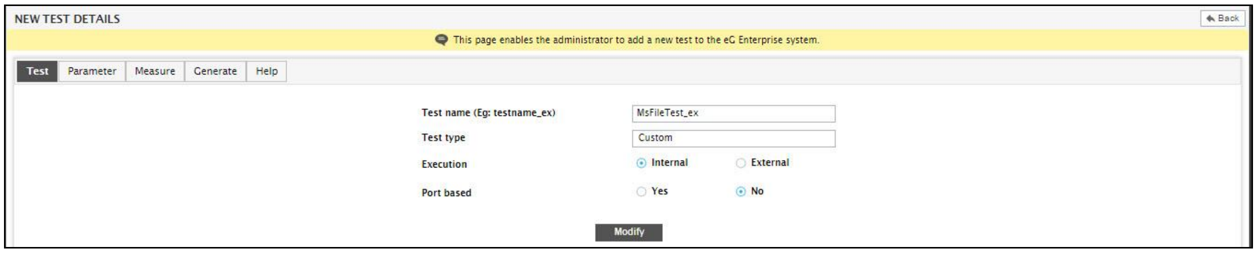
3. To delete a test, click the button corresponding to that test in Figure 2.112. You will then be prompted to confirm deletion (see Figure 2.113).


INTEGRATION CONSOLE – TEST

Deleting a test will remove it totally from the eG system. Are you sure you want to delete the test 'ag_ex'?

Figure 2.113: A message box that appears requesting your confirmation to delete a test

4. Click **Yes** in Figure 2.113 to proceed with the deletion. Click **No** to cancel the deletion.
5. To modify a test, click the button corresponding to that test in Figure 2.112.
6. Figure 2.114 will then appear displaying the test's specifications.

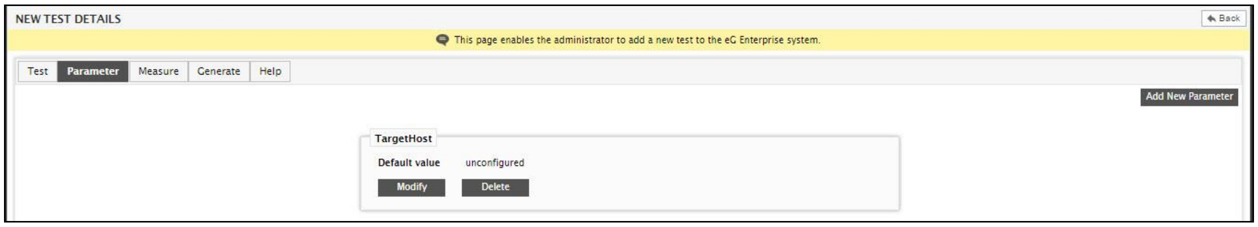

NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Test name (Eg: testname_ex)	MsFileTest_ex
Test type	Custom
Execution	[✓ Internal] External
Port based	Yes [✓ No]

Figure 2.114: Modifying a test's specification

7. For any test chosen for modification, you cannot change the **Test name** or the **Test type**. However, all other test details displayed in the **Test** tab page of Figure 2.114 can be altered. This includes the **Execution** mode, the **Port** configuration, the **OS type** (in case of a Script/batch file test), and the **DB type** (in case of a SQL query-based test).
8. Once the changes are made, click the **Modify** button to make sure that the changes take effect.
9. Then, click the **Parameter** tab page in Figure 2.115, if you want to add new parameters or modify/delete existing ones.


NEW TEST DETAILS

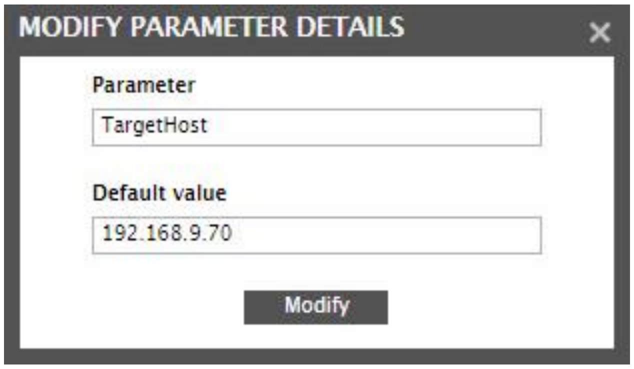
This page enables the administrator to add a new test to the eG Enterprise system.

TargetHost

Default value unconfigured

Figure 2.115: Adding/modifying test parameters

10. If any parameter pre-exists for a test, the same will be displayed in the **Parameter** tab page. While some of these parameters could have been user-defined, some others could be default parameters that the test type supports. For instance, tests of type **Jmx** and **Snmp** come bundled with a default set of parameters. You can modify/delete a user-defined parameter, but can only modify (and not delete) a default parameter. For example, the **TargetHost** parameter in Figure 2.115 above is a user-defined parameter; this is why, it is accompanied by a **Modify** and a **Delete** button. To delete this parameter, simply click the **Delete** button corresponding to it in Figure 2.115. To modify this parameter, click the **Modify** button corresponding to it. Figure 2.116 will then appear.


MODIFY PARAMETER DETAILS

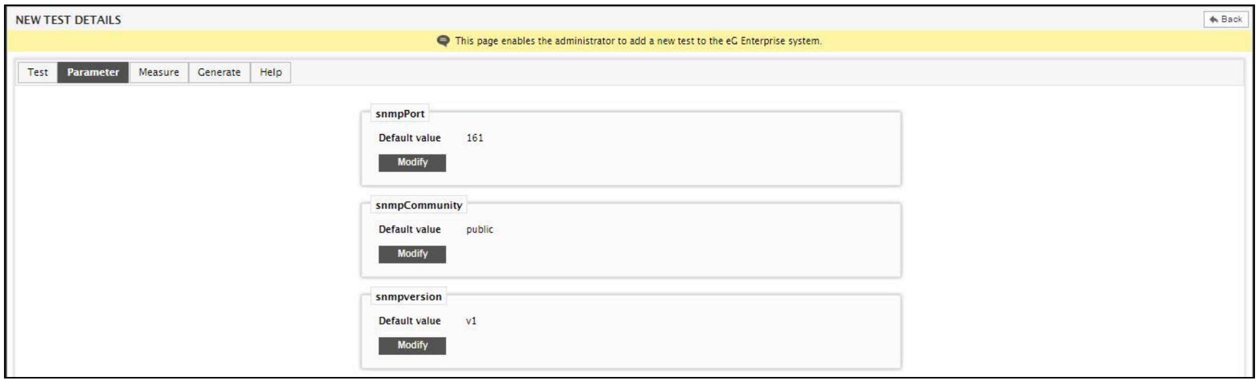
Parameter: TargetHost

Default value: 192.168.9.70

Figure 2.116: Modifying a user-defined parameter

11. Using Figure 2.116, you can change the **Parameter** name and/or the **Default value** of a user-defined parameter. Once the changes are made, click the **Modify** button to save the changes.

12. Now, take a look at Figure 2.117 below, which depicts the **Parameter** tab page of a test of type **Snmp**. As you can see, only the default parameters of the **Snmp** test are displayed in Figure 2.116. This is why, these parameters are accompanied only by a **Modify** button and not a **Delete** button. To modify a default parameter, click the **Modify** button corresponding to it in Figure 2.117.


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

snmpPort

Default value 161

snmpCommunity

Default value public

snmpversion

Default value v1

Figure 2.117: Viewing the default parameters of an Snmp test

13. Figure 2.118 will then appear, using which you can change the **Default value** of the parameter. **Note that you cannot change the name of a default parameter**.

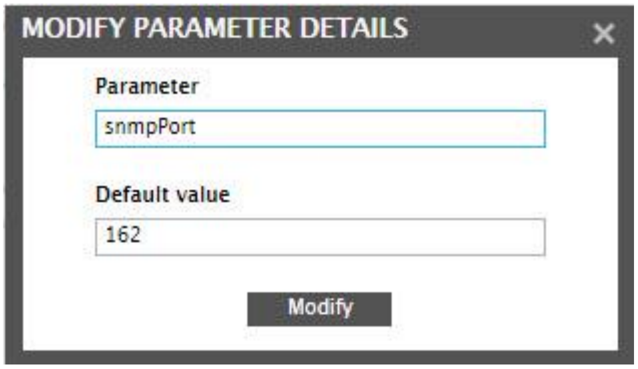


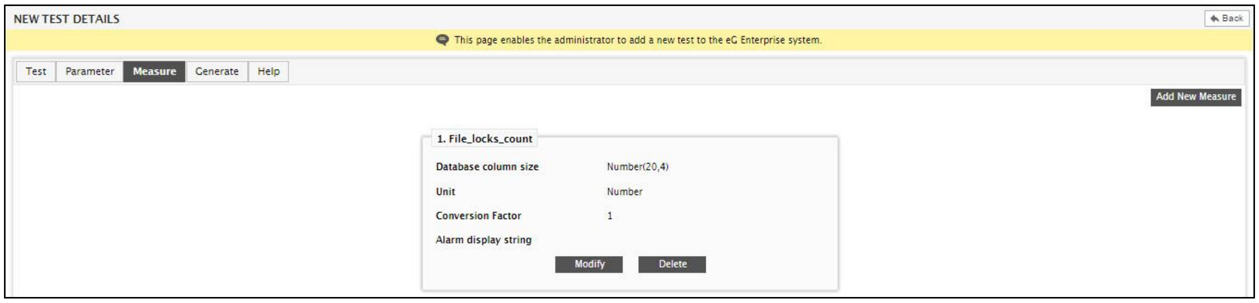
Figure 2.118: Modifying the default value of a default parameter

14. Finally, click the **Modify** button in Figure 2.118 to make the changes.

15. If required, you can also add new parameters to a test in the **Modify** mode. For this, just click the **Add New Parameter** button in Figure 2.117.

16. Next, click the **Measure** tab page in Figure 2.117 to make changes to the measure configurations of the test.

17. Figure 2.119 will then appear. If measures have already been configured for the test, then the same will be displayed in the **Measure** tab page, as depicted by Figure 2.119.



This page enables the administrator to add a new test to the eG Enterprise system.

1. File_locks_count

Database column size	Number(20,4)
Unit	Number
Conversion Factor	1
Alarm display string	

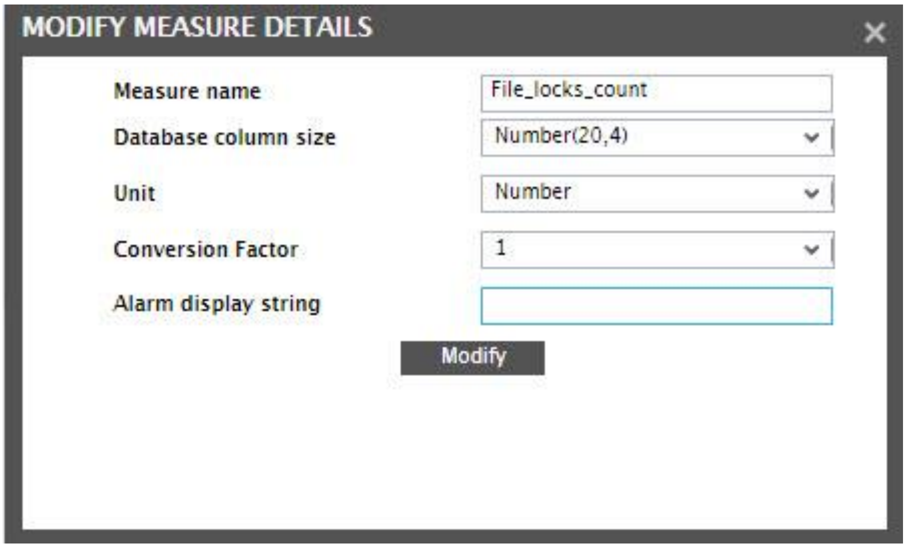
Figure 2.119: Viewing the measures configured for a test

18. The details displayed for a measure will change according to the type of test to which that measure pertains. For instance, Figure 2.119 above displays the details of a measure reported by a **Custom** test. A test of type **Script/Batch File** will additionally display a **Process Method** for each of its measures. Likewise, a test of type **Snmp** will additionally display an **Object OID**, a test of type **Perfmon** will include a **Counter name** as part of its measure specifications, and measures reported by a **Jmx** test will additionally support an **Attribute name**. You can delete a measure displayed in Figure 2.119 by clicking the **Delete** button corresponding to it. You can even add new measures by clicking the **Add New Measure** button in Figure 2.119. To modify an existing measure, click the **Modify** button corresponding to it in Figure 2.119.

Note

If you want to add more measures to a **Jmx** test in the **Modify** mode, note that the new measures should belong to same **Domain** and should follow the same MBean discovery methodology chosen for the old measures. In other words, if the **Domain** chosen for the old measures was **java.lang**, then the new measures should also belong to the **java.lang** domain. Likewise, if the MBeans for the old measures were loaded to a file and read from it, the new measures can also be added using only the MBeans so read.

19. This will invoke Figure 2.120. Using Figure 2.120, you can modify all the details of a measure. Then, click the **Modify** button in Figure 2.120 to save the changes.

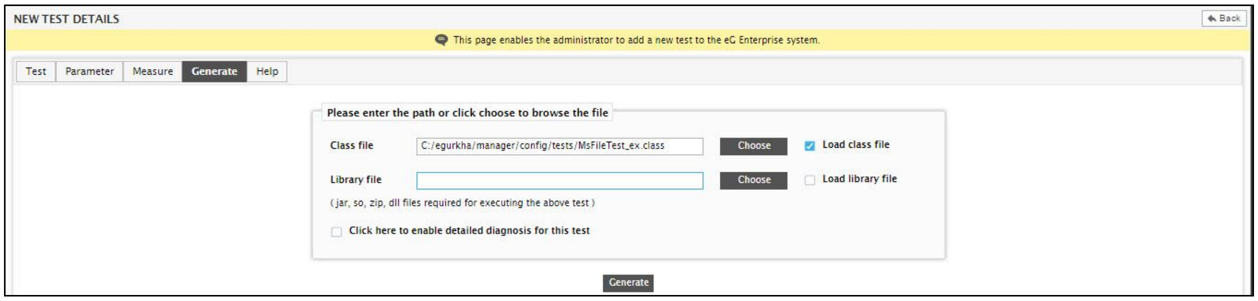


MODIFY MEASURE DETAILS

Measure name	File_locks_count
Database column size	Number(20,4)
Unit	Number
Conversion Factor	1
Alarm display string	

Figure 2.120: Modifying a measure

20. Finally, click the **Generate** tab page in Figure 2.119 to regenerate the test. Figure 2.121 will then appear.



NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Please enter the path or click choose to browse the file

Class file: C:/egurkha/manager/config/tests/MsFileTest_ex.class [✓ Load class file]

Library file: ☐ Load library file

(jar, so, zip, dll files required for executing the above test)

☐ Click here to enable detailed diagnosis for this test

Figure 2.121: Modifying the test implementation

21. The contents of the **Generate** tab page too will change according to the test type. However, regardless of the test type, you can change all details displayed in the **Generate** tab page, if required. For instance, for a **Custom** test, you can change the following:

- The **Class file** specification
- The **Library file** specification

- Enable/disable **detailed diagnosis** for the test
- Alter the detailed diagnosis specification, if it is enabled;

For a **Script/Batch File** test, you can change the **Path of the file** displayed in the **Generate** tab page.

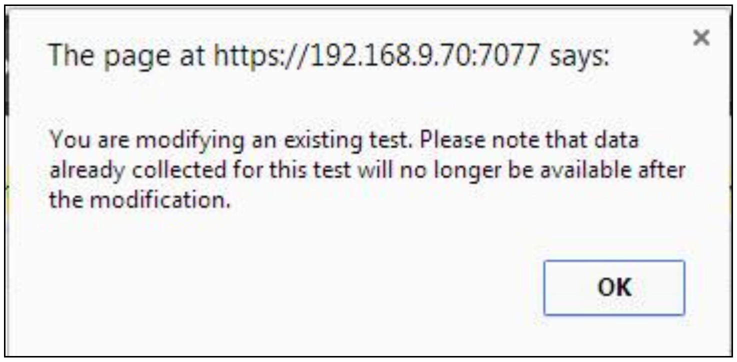
For a **SQL query** test, you can modify the query to be executed or the stored procedure call.

For a **Perfmon** test, you can change the name of the performance **Object**, and **Instances to be included** or **excluded** from monitoring.

For an **Snmp** test, you can indicate whether the measures pertain to a **Single element** or **Multiple elements**. If **Multiple elements** is chosen, you can also change the **Element ID**, the **Element status**, the **Element valid status**, and the **Rediscovery period**.

For a **Jmx** test, you can modify the **MBeans to be included** and/or **excluded**. The **Domain Name** however, cannot be changed.

22. Finally, click the **Generate** button to generate the test.
23. If a test specification is modified and the test is regenerated, the performance data previously collected by the test will no longer be available. A warning message to this effect will be displayed when the **Generate** button is clicked (see Figure 2.122). Click the **OK** button in Figure 2.122 to go ahead with the modifications.


The page at https://192.168.9.70:7077 says:

You are modifying an existing test. Please note that data already collected for this test will no longer be available after the modification.

Figure 2.122: A warning message that appears when a test is modified and regenerated

2.8 Adding Help Pages for the New Test

eG Enterprise embeds a context-sensitive online help system, which enables users to instantly invoke help pages for assistance while configuring the tests run by the eG agent or understanding the measures reported by the tests. By default, the eG manager comes bundled with help pages for the tests it supports out-of-the-box. Each test is associated with an **Admin** and a **Monitor** help page. While the **Admin** help page describes how the test parameters are to be configured, the **Monitor** help page lists the measures reported by the test and explains the significance of each measure.

For new tests added via the Integration Console plugin however, no such help pages pre-exist. To enable users to include help pages for these new tests into the eG Enterprise system, you can do either of the following:

- Use the Integration Console itself to create new **Admin** and **Monitor** help pages for the new test, OR;
- Create **Admin** and **Monitor** help pages using a third-party HTML editor (eg., Editplus, Adobe Dreamweaver, Microsoft Frontpage, etc.), and use the Integration Console to upload these help pages to the eG manager;

The sub-sections that follow will discuss each of these options in detail:

2.8.1 Creating New Help Pages Using the Integration Console

Let us now define an **Admin** and a **Monitor** help page for the **MsFileTest_ex** that we created in Section 2.1.1. To begin help page creation, do either of the following:

- Typically, until a new test is properly generated and thresholds are set for its measures in the eG Enterprise system, you cannot define help pages for that test. This is why, the **Help** tab page in the **NEW TEST DETAILS** page will remain disabled till the **Finish** button is clicked in Figure 2.111. Clicking the **Finish** button will invoke the message box depicted by Figure 2.123. Click the **Yes** button in the message box to begin help page creation. This will lead you to the **Help** tab page of Figure 2.124.

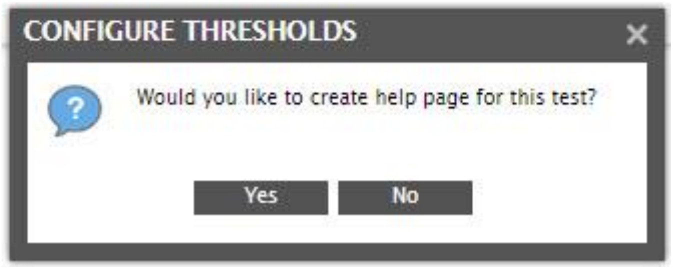
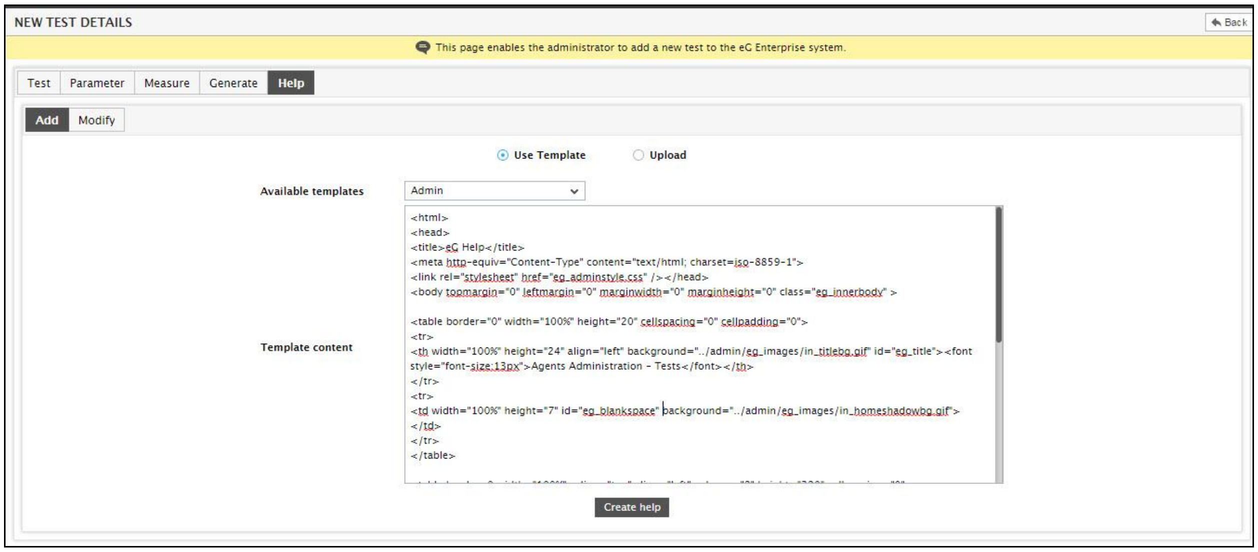


Figure 2.123: A message box requesting your confirmation to define a help page for the new test

- On the other hand, if you click the **No** button in the message box of Figure 2.123, you will exit the **NEW TEST DETAILS** page and return to the **INTEGRATION CONSOLE – TEST** page, where all the new IC tests will be displayed. To create a help page for a test that has already been generated, you need to **Modify** the specifications of that test. To do this, click the button corresponding to that test in the **INTEGRATION CONSOLE – TEST** page. This will take you to the **NEW TEST DETAILS** page, where you can click the **Help** tab page (see Figure 2.124) to begin creating help pages for that test.



This page enables the administrator to add a new test to the eG Enterprise system.

(● Use Template) ○ Upload

Available templates: Admin

Template content:

```
<html>
<head>
<title>eG Help</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" href="eg_adminstyle.css" /></head>
<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0" class="eg_innerbody" >

<table border="0" width="100%" height="20" cellspacing="0" cellpadding="0">
<tr>
<th width="100%" height="24" align="left" background="../admin/eg_images/in_titlebg.gif" id="eg_title"><font
style="font-size:13px">Agents Administration - Tests</font></th>
</tr>
<tr>
<td width="100%" height="7" id="eg_blankspace" background="../admin/eg_images/in_homeshadowbg.gif">
</td>
</tr>
</table>
```

Figure 2.124: The Help tab page

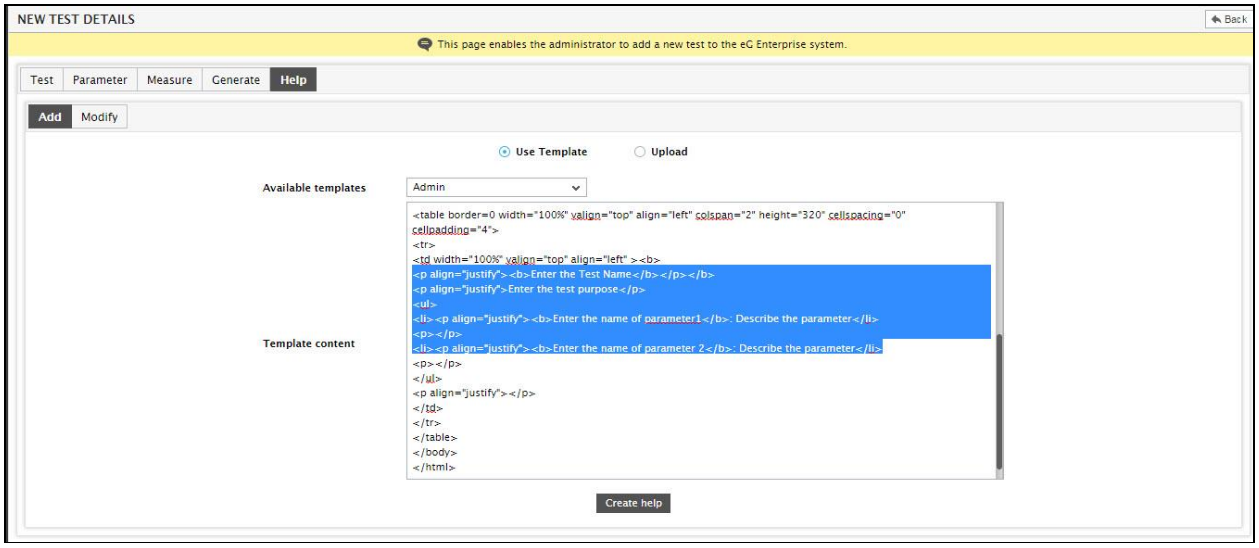
The **Help** tab page, as you can see, comes with two sub-tabs: **Add** and **Modify**. By default, the **Add** tab page will be selected. To create a new help page, you should use the **Add** tab page only.

The **Add** tab page provides help page templates in HTML format, which can be easily customized to create the help pages you want. To create an **Admin** help page for the **MsFileTest_ex**, do the following:

1. Select the **Use Template** option from the **Add** tab page.
2. Then, from the **Available templates** list, pick the **Admin** option.
3. This will bring up the **Admin** help page template as shown in Figure 2.125.
4. The **Admin** help page template of Figure 2.125 embeds directions on how to edit the template. You just need to scan the template for these instructions and follow them. For instance, search the template for the string **Enter the Test Name**. Once it is found, remove the string and in its place type the test name, **MsFileTest_ex**, as directed. Likewise, look for the following strings in the template, remove them one after another from the template, and in the place of each, provide the inputs indicated by the corresponding string.

String	Help page information
Enter the test purpose	Describe the purpose of the **MsFileTest_ex**
Enter the name of parameter1	Specify the first parameter that is to be configured for the **MsFileTest_ex**. If you can recall, while creating the **MsFileTest_ex**, we had not configured any special parameters for the test. However, by default, any non-port-based test added using IC will take TEST PERIOD and HOST as its parameters. Therefore, type test period as the first parameter.
Describe the parameter	This string will appear after **parameter1** and **parameter2**. Provide a description of the corresponding parameter here.
Enter the name of parameter2	Enter host as parameter 2

The strings in the table above have been highlighted in the template in Figure 2.125.


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

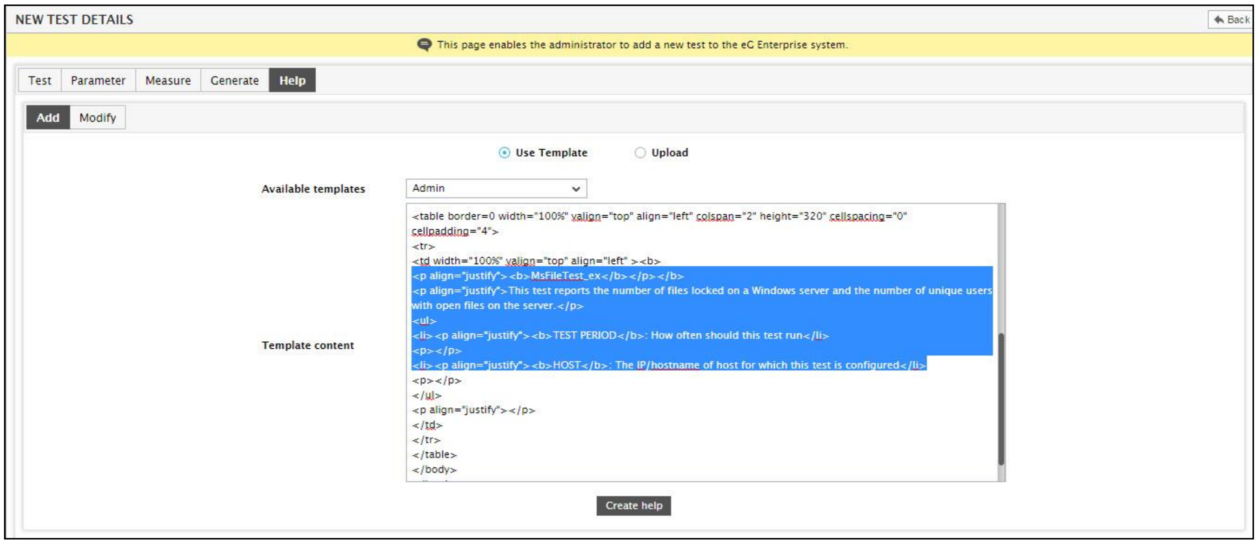
Available templates: Admin

Template content:

```
<table border=0 width="100%" valign="top" align="left" colspan="2" height="320" cellspacing="0"
cellpadding="4">
<tr>
<td width="100%" valign="top" align="left" ><b>
<p align="justify"><b>Enter the Test Name</b></p></b>
<p align="justify">Enter the test purpose</p>
<ul>
<li><p align="justify"><b>Enter the name of parameter1</b>: Describe the parameter</li>
<p></p>
<li><p align="justify"><b>Enter the name of parameter 2</b>: Describe the parameter</li>
<p></p>
</ul>
<p align="justify"></p>
</td>
</tr>
</table>
</body>
</html>
```

Figure 2.125: The strings containing instructions on how to edit the Admin template

5. Once the HTML block highlighted in Figure 2.125 is edited based on the instructions provided in step 4, the same block will look as depicted by Figure 2.126.



This page enables the administrator to add a new test to the eG Enterprise system.

(● Use Template) ○ Upload

Available templates: Admin

Template content:

```
<table border=0 width="100%" valign="top" align="left" colspan="2" height="320" cellspacing="0"
cellpadding="4">
<tr>
<td width="100%" valign="top" align="left" ><b>
<p align="justify"><b>MsFileTest_ex</b></p></b>
<p align="justify">This test reports the number of files locked on a Windows server and the number of unique users
with open files on the server.</p>
<ul>
<li><p align="justify"><b>TEST PERIOD</b>: How often should this test run</li>
<p></p>
<li><p align="justify"><b>HOST</b>: The IP/hostname of host for which this test is configured</li>
<p></p>
</ul>
<p align="justify"></p>
</td>
</tr>
</table>
</body>
```

Figure 2.126: The edited HTML block in the Admin template

6. By default, the template allows you to provide details for a maximum of two parameters. If your test supports say, one more parameter, and you want to include the details of this additional parameter in the help page, then, insert a line of the following format just above the *</ul>* tag in the help page template:

   *<li><p align="justify"><b>**Enter the name of parameter 3**</b>: **Description of parameter 3**</li>*

   <p></p>

   For instance, if you are adding an additional parameter named filename into the help page, then, the new line you insert should be as follows:

   *<li><p align="justify"><b>**FILENAME**</b>: **The name of the file to be monitored**</li>*

   <p></p>

   Multiple lines of the above format can be inserted for every additional parameter the test takes.

7. Likewise, if your test supports only one parameter, then you will have to explicitly remove the entire row of information available for the second parameter from the template. For instance, if you want to remove the host parameter from the **MsFileTest_ex** in our example, then, simply delete the following line of code:

   *<li><p align="justify"><b>**HOST**</b>: **The host for which the test is being configured**</li>*

   <p></p>

8. No additional parameters exist for the **MsFileTest_ex** in our example. Similarly, no lines of code need be removed from the template for the purpose of our example. Therefore, simply proceed to click the **Create help** button to generate the **Admin** help page for **MsFileTest_ex**. If the help page is generated successfully, then a message to that effect will appear.

9. The eG Enterprise system will automatically assign a name of the format, *<TestName>_Admin*, for any **Admin** help page you create for an IC-based test using IC. This help page will be automatically saved to the **<EG_MANAGER_INSTALL_DIR>\tomcat\webapps\final\eghelp** directory.

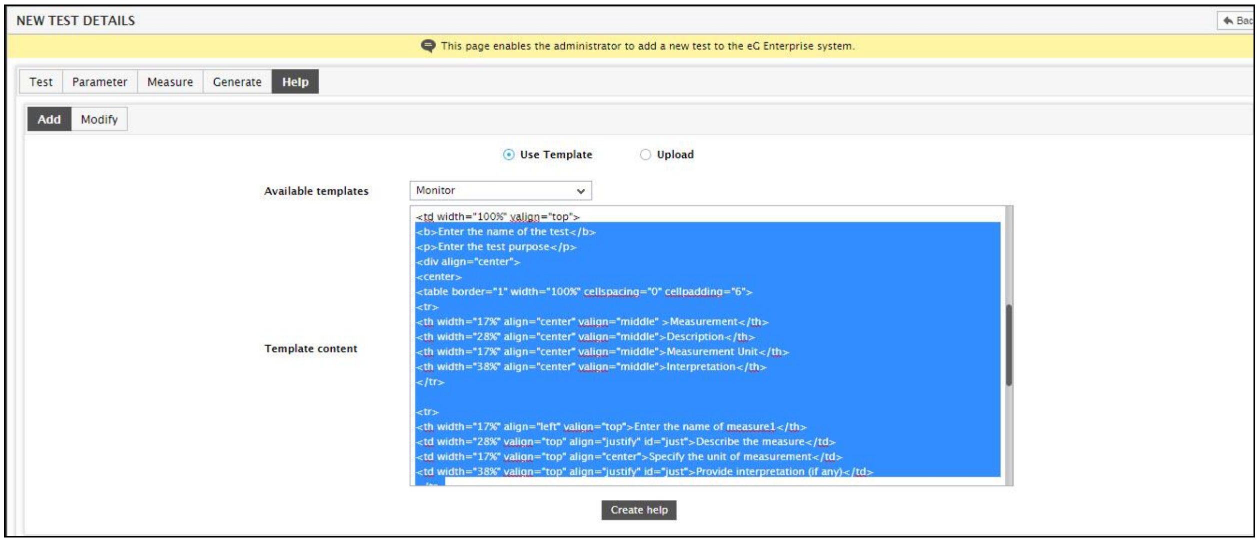
Let us now proceed to create a **Monitor** hlep page for the **MsFileTest_ex**. For this, do the following:

1. First, select the **Use Template** option from Figure 2.124.

2. Then, from the **Available templates** list, pick the **Monitor** option.
3. This will bring up the **Monitor** help page template.
4. Like the **Admin** help page template, the **Monitor** help page template also includes instructions for editing the template. You just need to scan the template for these instructions and follow them. For instance, search the template for the string **Enter the name of the test**. Once it is found, remove the string and in its place type the test name, **MsFileTest_ex**, as directed. Likewise, look for the following strings in the template, remove them one after another from the template, and in the place of each, provide the inputs indicated by the corresponding string.

String	Help page information
Enter the test purpose	Describe the purpose of the **MsFileTest_ex**
Enter the name of measure1	Specify the first measure that is reported by the **MsFileTest_ex**. For our example, type **File_locks_count** here.
Enter the name of measure2	Specify the second measure that is reported by the **MsFileTest_ex**. For our example, type **Unique_users_count** here.
Describe the measure	This will appear after **measure1** and **measure2**. Provide a description for the corresponding measure here.
Specify the unit of measurement	This will appear after **measure1** and **measure2.** Provide the unit of measurement that you have configured for the corresponding measure here. For the **File_locks_count** and the **Unique_users_count** measures in our example, the unit will be **Number.**
Provide interpretation (if any)	This will be available for both **measure1** and **measure2**. Here, you can explain how the high or low values of the corresponding measure will impact the performance of the target server. This is an optional specification. In other words, if you feel that no interpretation is necessary, then, you can just remove the string and leave the placeholder blank.

The strings in the table above have been highlighted in the template in Figure 2.127.



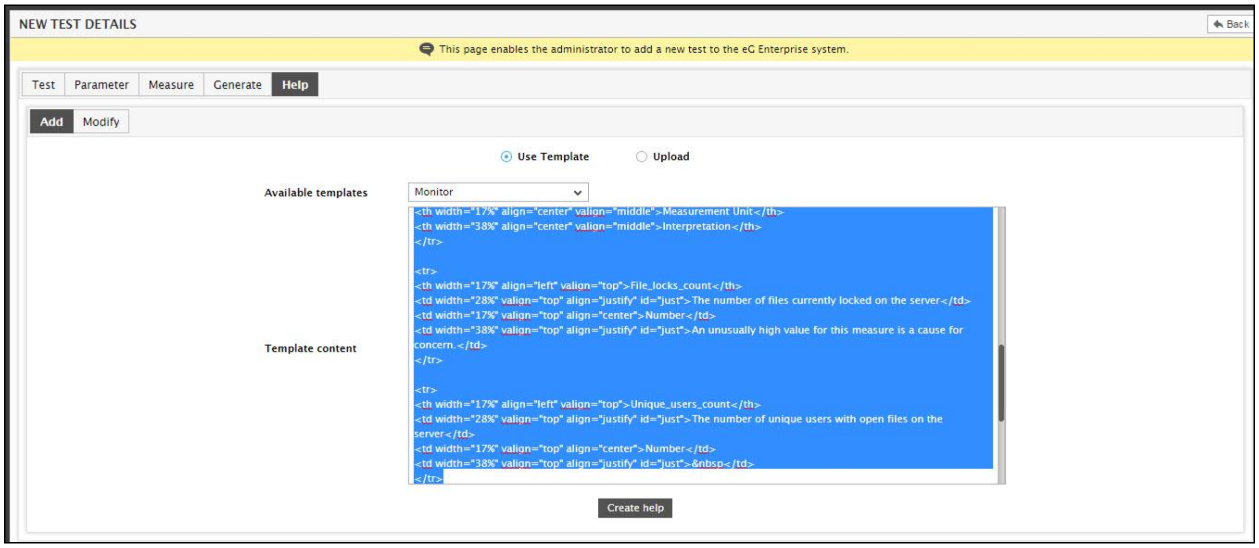
Available templates: Monitor

Template content:

```
<td width="100%" valign="top">
<b>Enter the name of the test</b>
<p>Enter the test purpose</p>
<div align="center">
<center>
<table border="1" width="100%" cellspacing="0" cellpadding="6">
<tr>
<th width="17%" align="center" valign="middle" >Measurement</th>
<th width="28%" align="center" valign="middle">Description</th>
<th width="17%" align="center" valign="middle">Measurement Unit</th>
<th width="38%" align="center" valign="middle">Interpretation</th>
</tr>

<tr>
<th width="17%" align="left" valign="top">Enter the name of measure1</th>
<td width="28%" valign="top" align="justify" id="just">Describe the measure</td>
<td width="17%" valign="top" align="center">Specify the unit of measurement</td>
<td width="38%" valign="top" align="justify" id="just">Provide interpretation (if any)</td>
```

Figure 2.127: The strings containing instructions on how to edit the Monitor template

5. Once the HTML block highlighted in Figure 2.127 is edited based on the instructions provided at step 4, the same block will look as depicted by Figure 2.128.



Available templates: Monitor

Template content:

```
<th width="17%" align="center" valign="middle">Measurement Unit</th>
<th width="38%" align="center" valign="middle">Interpretation</th>
</tr>

<tr>
<th width="17%" align="left" valign="top">File_locks_count</th>
<td width="28%" valign="top" align="justify" id="just">The number of files currently locked on the server</td>
<td width="17%" valign="top" align="center">Number</td>
<td width="38%" valign="top" align="justify" id="just">An unusually high value for this measure is a cause for
concern.</td>
</tr>

<tr>
<th width="17%" align="left" valign="top">Unique_users_count</th>
<td width="28%" valign="top" align="justify" id="just">The number of unique users with open files on the
server</td>
<td width="17%" valign="top" align="center">Number</td>
<td width="38%" valign="top" align="justify" id="just">&nbsp</td>
</tr>
```

Figure 2.128: The edited HTML block in the Monitor template

6. By default, the template allows you to provide details for a maximum of two measures. If your test supports say, one more measure, and you want to include the details of this additional measure in the help page, then, insert a block of HTML code of the following format, just above the *</table>* tag in the help page template:

<tr>

*<th width="17%" align="left" valign="top">***Enter the measure name***</th>*

*<td width="28%" valign="top" align="justify" id="just">***Briefly describe the measure***</td>*

*<td width="17%" valign="top" align="center">***Specify the unit of measurement for the measure***</td>*

*<td width="38%" valign="top" align="justify" id="just">**Provide interpretation (if any)** </td>*

</tr>

For instance, say that you are adding an additional measure named **Open_files**; this measure reports the total number of files that have been opened on the server by users over the network. The HTML code block that you insert into the **Monitor** template for this purpose should be as follows:

<tr>

*<th width="17%" align="left" valign="top">***Open_files** *</th>*

*<td width="28%" valign="top" align="justify" id="just">**The total number of files that have been opened on the server by users over the network** </td>*

*<td width="17%" valign="top" align="center">***Number** *</td>*

*<td width="38%" valign="top" align="justify" id="just">**This measurement is an indicator of the workload on the file server.** </td>*

</tr>

Multiple blocks of the above format can be inserted for every additional measure that you configure for the test.

7. Likewise, if your test reports only one measure, then you will have to explicitly remove the entire block of code that is provided for the second measure from the template. For instance, if you want to remove the **Unique_users_count** measure from the help page of the **MsFileTest_ex** in our example, then, simply delete the following line of code:

   <tr>

   <th width="17%" align="left" valign="top">Unique_users_count</th>

   <td width="28%" valign="top" align="justify" id="just">The number of distinct users with open files</td>

   <td width="17%" valign="top" align="center">Number</td>

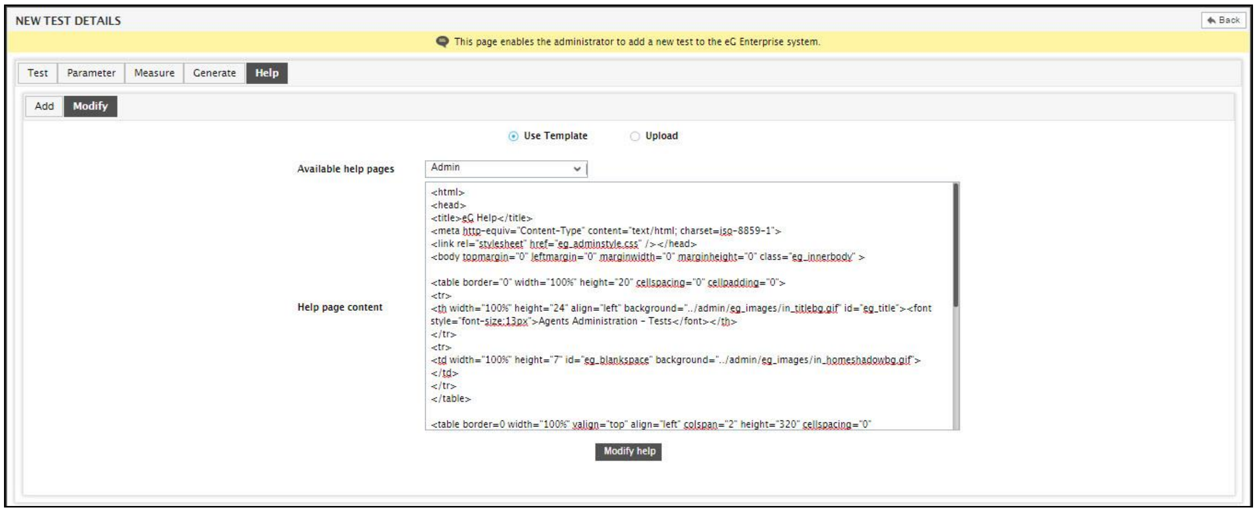
   <td width="38%" valign="top" align="justify" id="just">Provide interpretation (if any)</td>

   </tr>

8. No additional measures are reported by the **MsFileTest_ex** in our example. Similarly, no lines of code need be removed from the template for the purpose of our example. Therefore, simply proceed to click the **Create help** button to generate the **Monitor** help page for **MsFileTest_ex**. If the help page is generated successfully, then a message to that effect will appear.

9. The eG Enterprise system will automatically assign a name of the format, *<TestName>_Monitor*, for any **Monitor** help page that you create using IC. This help page will be automatically saved to the **<EG_MANAGER_INSTALL_DIR>\tomcat\webapps\final\eghelp** directory.

With that, both the **Admin** and **Monitor** help pages have been created for the **MsFileTest_ex**. You can modify the help pages so created by clicking the **Modify** tab page in Figure 2.124. This will open Figure 2.129.


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

[✓ Use Template] Upload

Available help pages: Admin

Help page content:

```
<html>
<head>
<title>eG Help</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" href="eg_adminstyle.css" /></head>
<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0" class="eg_innerbody" >

<table border="0" width="100%" height="20" cellspacing="0" cellpadding="0">
<tr>
<th width="100%" height="24" align="left" background="../admin/eg_images/in_titlebg.gif" id="eg_title"><font
style="font-size:13px">Agents Administration - Tests</font></th>
</tr>
<tr>
<td width="100%" height="7" id="eg_blankspace" background="../admin/eg_images/in_homeshadowbg.gif">
</td>
</tr>
</table>

<table border=0 width="100%" valign="top" align="left" colspan="2" height="320" cellspacing="0"
```

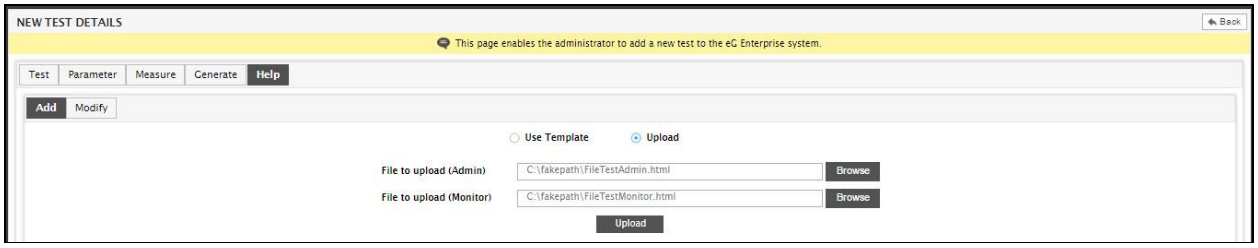
Figure 2.129: Modifying a help page created using the Integration Console

For modifying an **Admin** or a **Monitor** help page, follow the same procedure that has been outlined for creating a help page.

2.8.2 Uploading Help Pages that Pre-exist to the eG Manager

To achieve this, do the following:

1. First, select the **Upload** option from Figure 2.130.


NEW TEST DETAILS

This page enables the administrator to add a new test to the eG Enterprise system.

Use Template [✓ Upload]

File to upload (Admin): C:\fakepath\FileTestAdmin.html

File to upload (Monitor): C:\fakepath\FileTestMonitor.html

Figure 2.130: Uploading the help pages

2. In the **File to upload (Admin)** text box, specify the full path to the **Admin** help page to be uploaded. You can use the **Browse** button to locate the help page you need.
3. Likewise, in the **File to upload (Monitor)** text box, specify the full path to the **Monitor** help page to be uploaded. Here again, you can use the **Browse** button alongside to locate the monitor help page.
4. Once the help page locations are specified, click the **Upload** button to upload them to the eG manager.
5. Once uploaded, the **Admin** help page will be automatically renamed as *TestName>_Admin*, and the **Monitor** help page will be automatically renamed as *<TestName>_Monitor.* Moreover, both help pages will be uploaded to the **<EG_MANAGER_INSTALL_DIR>\tomcat\webapps\final\eghelp** directory.

Note

- It is not mandatory to upload both the **Admin** and **Monitor** help pages of a test simultaneously.
- At any given point in time, you can upload only one **Admin** help page and/or one **Monitor** help page to the eG manager.

3

Adding/Modifying Layers Using the Integration Console

Once a new test is added using IC, you need to associate that test to a layer for the test functionality to be implemented in real-time. A test can be associated with a layer that pre-exists or a brand new layer.

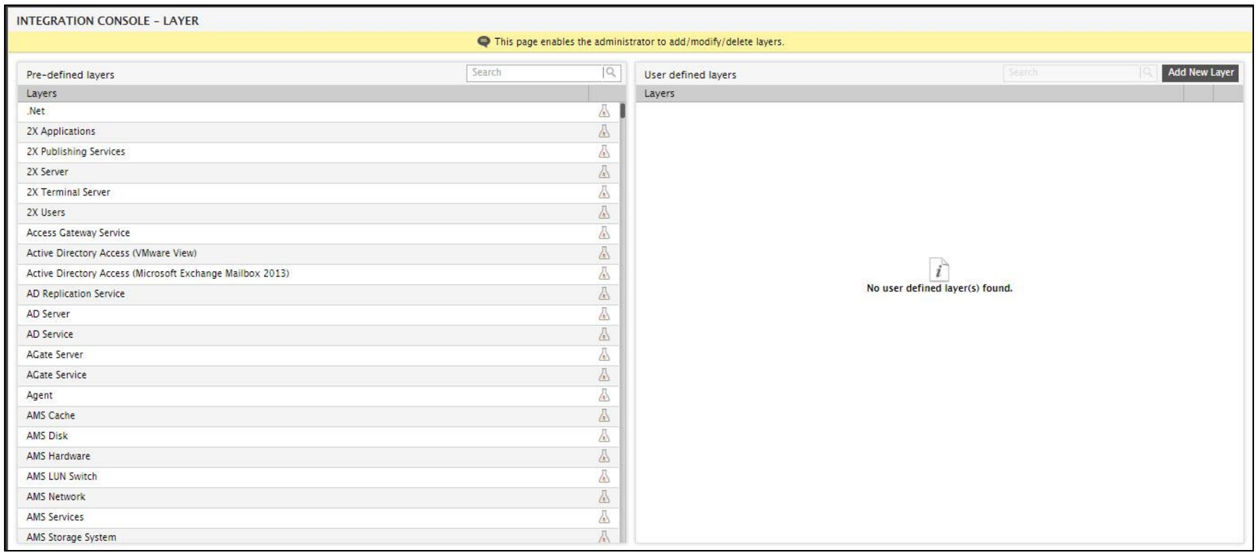
This chapter takes the help of an example to explain how a new layer can be created using IC, how it can be modified (if required), and how a test can be associated with a new/existing layer.

3.1 Adding a New Layer and Associating Tests with the User-Defined Layer

In this section, we will be creating a new layer named *TUXEDO_SERVICES* and will be associating a new test named **TuxDomainTest_ex** with it. This test has been engineered to report the number of machines and servers running in a Tuxedo domain.

To create a new layer, do the following:

1. Select the **Layer** option from the **Integration Console** tile of the **Admin** tile menu (see Figure 2.1).
2. Figure 3.1 will then appear with two panels: one listing pre-defined layers and another listing user-defined layers. The eG Enterprise system represents every application/device that it monitors out-of-the-box using a hierarchical set of layers. The **Pre-defined layers** panel lists all those layers that are by default built into the eG Enterprise system for representing components that are monitored out-of-the-box. The **User-defined layers** panel lists only those layers that are custom-defined by users to extend the monitoring capabilities of the eG Enterprise solution.



INTEGRATION CONSOLE – LAYER

This page enables the administrator to add/modify/delete layers.

Pre-defined layers

Layers

.Net
2X Applications
2X Publishing Services
2X Server
2X Terminal Server
2X Users
Access Gateway Service
Active Directory Access (VMware View)
Active Directory Access (Microsoft Exchange Mailbox 2013)
AD Replication Service
AD Server
AD Service
ACate Server
ACate Service
Agent
AMS Cache
AMS Disk
AMS Hardware
AMS LUN Switch
AMS Network
AMS Services
AMS Storage System

User defined layers

Layers

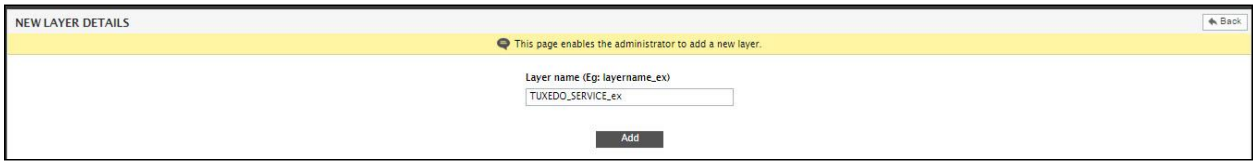
No user defined layer(s) found.

Figure 3.1: Viewing the list of pre-defined and user-defined layers.

3. To create a new layer, click the **Add New Layer** button in Figure 3.1. This will open Figure 3.2. In the **Layer name** text box in Figure 3.2, enter the name of the new layer. For the purpose of our example, type **TUXEDO_SERVICES_ex** against **Layer name.**

Note

The layer name should be suffixed by _*ex*.



NEW LAYER DETAILS

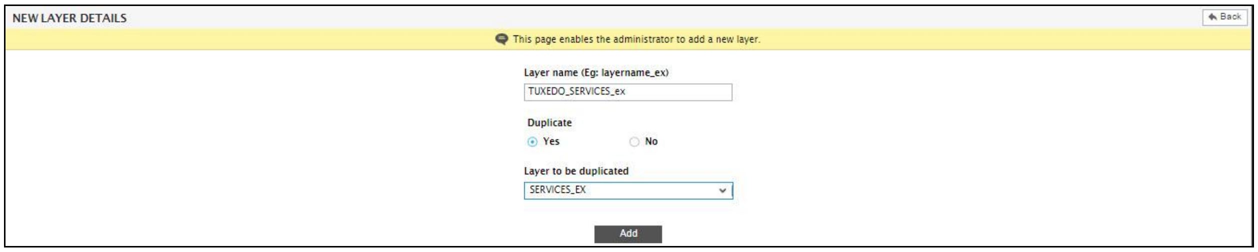
This page enables the administrator to add a new layer.

Layer name (Eg: layername_ex)

TUXEDO_SERVICE_ex

Figure 3.2: Adding a new layer

If more than one pre-defined layer pre-exists, then Figure 3.2 will additionally include a **Duplicate** flag. By default, the **Duplicate** flag is set to **No**. Set this flag to **Yes** only if you want the new layer to inherit the attributes of another user-defined layer. In this case, a **Layer to be duplicated** drop-down will appear, as depicted by Figure 3.3. From this drop-down, select the layer, the properties of which need to be acquired by the new layer being added.


NEW LAYER DETAILS

This page enables the administrator to add a new layer.

Layer name (Eg: layername_ex): TUXEDO_SERVICES_ex

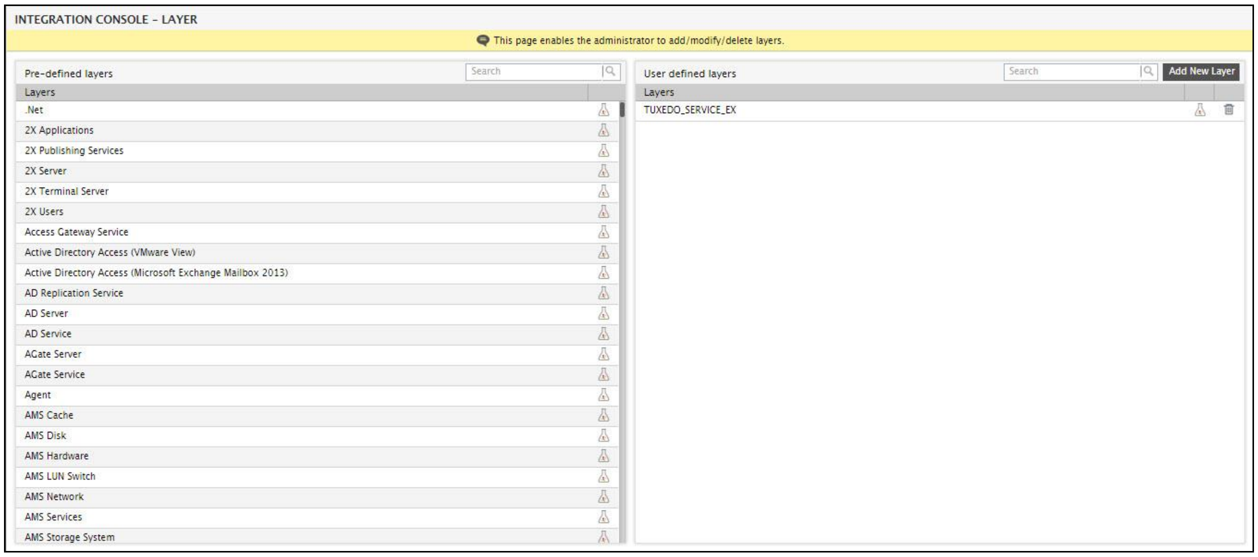
Duplicate: [✓ Yes] No

Layer to be duplicated: SERVICES_EX

Figure 3.3: Duplicating a layer

4. Then, click the **Add** button to add the new layer.

5. The newly added layer will now appear in the panel listing user-defined layers, as depicted by Figure 3.4.


INTEGRATION CONSOLE – LAYER

This page enables the administrator to add/modify/delete layers.

Pre-defined layers

Layers
.Net
2X Applications
2X Publishing Services
2X Server
2X Terminal Server
2X Users
Access Gateway Service
Active Directory Access (VMware View)
Active Directory Access (Microsoft Exchange Mailbox 2013)
AD Replication Service
AD Server
AD Service
AGate Server
AGate Service
Agent
AMS Cache
AMS Disk
AMS Hardware
AMS LUN Switch
AMS Network
AMS Services
AMS Storage System

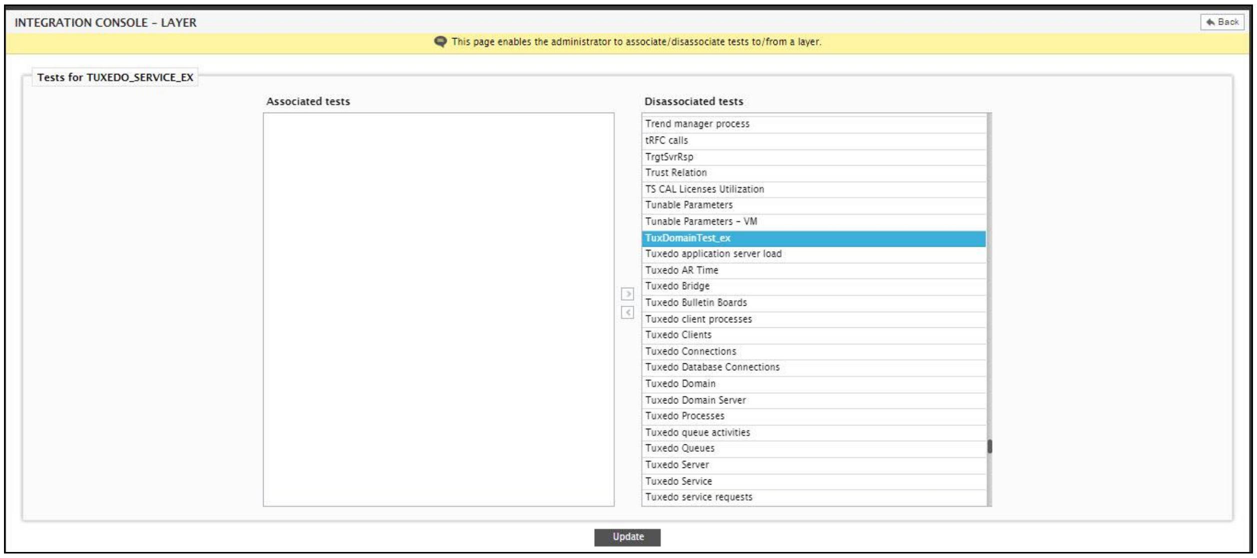
User defined layers

Layers
TUXEDO_SERVICE_EX

Figure 3.4: The new layer listed in the User defined layers panel

6. If you want, you can delete any user-defined layer by clicking the 🗑 button corresponding to it in Figure 3.4.

7. To associate a test with a user-defined layer, click the ⚗ button corresponding to the layer name in the **User-defined layers** panel of Figure 3.4. This will invoke Figure 3.5. From the **Disassociated tests** list of Figure 3.5, select the test(s) you want to associate with the new layer. For the purpose of our example, select the *TuxDomainTest_ex* test from the **Disassociated tests** list.



INTEGRATION CONSOLE – LAYER

This page enables the administrator to associate/disassociate tests to/from a layer.

Tests for TUXEDO_SERVICE_EX

Associated tests

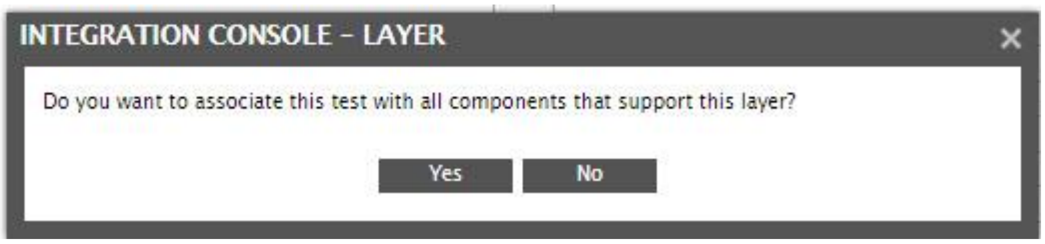
Disassociated tests

Trend manager process
tRFC calls
TrgtSvrRsp
Trust Relation
TS CAL Licenses Utilization
Tunable Parameters
Tunable Parameters – VM
TuxDomainTest_ex
Tuxedo application server load
Tuxedo AR Time
Tuxedo Bridge
Tuxedo Bulletin Boards
Tuxedo client processes
Tuxedo Clients
Tuxedo Connections
Tuxedo Database Connections
Tuxedo Domain
Tuxedo Domain Server
Tuxedo Processes
Tuxedo queue activities
Tuxedo Queues
Tuxedo Server
Tuxedo Service
Tuxedo service requests

Figure 3.5: Selecting the test to be associated with the user-defined layer

8. Click the **<** button in Figure 3.5 to move the selected test to the **Associated tests** list.

9. This will invoke a message box depicted by Figure 3.6 below. By default, once a test is associated with a layer, that layer will get automatically associated with all components that support that layer. Sometimes, you may want a test to be associated with only a few components that support that layer and not all of them. In this case, click the **No** button in the message box of Figure 3.6. If this is done, then the test will not be associated with that layer. On the other hand, if you want the test to be associated with all components that support the layer, click the **Yes** button. This will transfer the selection to the **Associated tests** list.



INTEGRATION CONSOLE – LAYER

Do you want to associate this test with all components that support this layer?

Figure 3.6: A message box requesting your confirmation to associate the test with all components tha tsupport the chosen layer

10. If for any reason, you want to disassociate a test from a layer, simply select the test from the **Associated tests** list and click the **>** button in Figure 3.5.

11. Finally, click the **Update** button to save the changes.

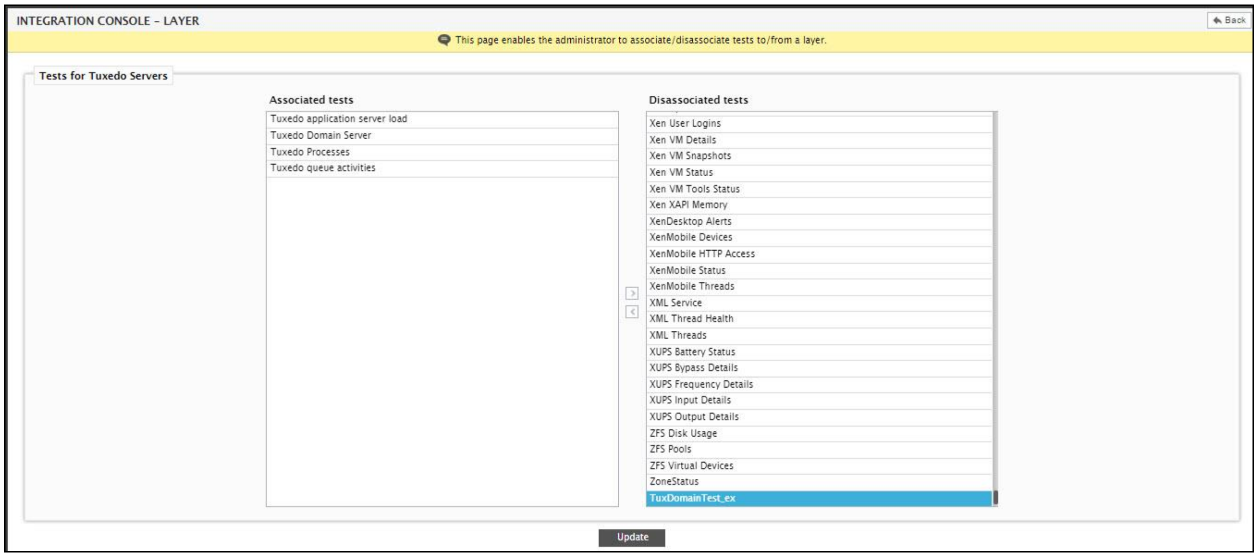
3.2 Associating Tests with a Pre-defined Layer

Let us consider another example. Let us try to associate the *TuxDomainTest_ex* in the example of Section 3.2 with the **Tuxedo Servers** layer that pre-exists in the eG Enterprise system. For this, do the following:

1. Focus on the **Pre-defined layers** list in Figure 3.1. As can be inferred from Figure 3.1, **you can neither modify nor delete a pre-defined layer**. However, you can associate/disassociate tests with a pre-defined layer.

2. To associate tests with the **Tuxedo Servers** layer in our example, click the button corresponding to that layer in the **Pre-defined layers** list of Figure 3.1.

3. This will open Figure 3.7.


INTEGRATION CONSOLE – LAYER

Back

This page enables the administrator to associate/disassociate tests to/from a layer.

Tests for Tuxedo Servers

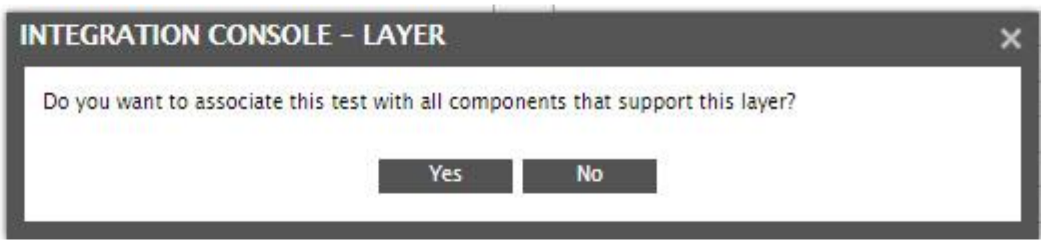
Associated tests
- Tuxedo application server load
- Tuxedo Domain Server
- Tuxedo Processes
- Tuxedo queue activities

Disassociated tests
- Xen User Logins
- Xen VM Details
- Xen VM Snapshots
- Xen VM Status
- Xen VM Tools Status
- Xen XAPI Memory
- XenDesktop Alerts
- XenMobile Devices
- XenMobile HTTP Access
- XenMobile Status
- XenMobile Threads
- XML Service
- XML Thread Health
- XML Threads
- XUPS Battery Status
- XUPS Bypass Details
- XUPS Frequency Details
- XUPS Input Details
- XUPS Output Details
- ZFS Disk Usage
- ZFS Pools
- ZFS Virtual Devices
- ZoneStatus
- TuxDomainTest_ex

Figure 3.7: Selecting the test to be associated with a pre-defined layer

4. The **Associated tests** list in Figure 3.7 will display all those tests that are already associated with the **Tuxedo Servers** layer. To associate the *TuxDomainTest_ex* with this layer, first select that test from the **Disassociated tests** list of Figure 3.7. Then, click the **<** button.

5. This will invoke a message box depicted by Figure 3.8 below. By default, once a test is associated with a layer, that layer will get automatically associated with all components that support that layer. Sometimes, you may want a test to be associated with only a few components that support that layer and not all of them. In this case, click the **No** button in the message box of Figure 3.8. If this is done, then the test will not be associated with that layer. On the other hand, if you want the test to be associated with all components that support the layer, click the **Yes** button. This will transfer the selection to the **Associated tests** list.


INTEGRATION CONSOLE – LAYER

Do you want to associate this test with all components that support this layer?

Figure 3.8: A message box requesting your confirmation to associate the test with all components tha tsupport the chosen layer

6. If for any reason, you want to disassociate a test from a layer, simply select the test from the **Associated tests** list and click the **>** button in Figure 3.7.

7. Finally, click the **Update** button to save the changes.

4

Adding/Modifying New Component Types Using the Integration Console

eG Enterprise provides out-of-the-box monitoring support to over 150 applications/devices, 10+ operating systems, and 9+ virtualization platforms. Specialized monitoring models are available in eG for each of these applications/devices/systems/hypervisors. If eG does not offer a monitoring model out-of-the-box for any application/device that is operational in your environment, then you can use the Integration Console to build a monitoring model for that type of component. This can be achieved by:

- Creating a new component-type for the custom application/device using the Integration Console;
- Building a layer stack for the new component-type;
- Associating/Disassociating performance and configuration tests for that component-type;

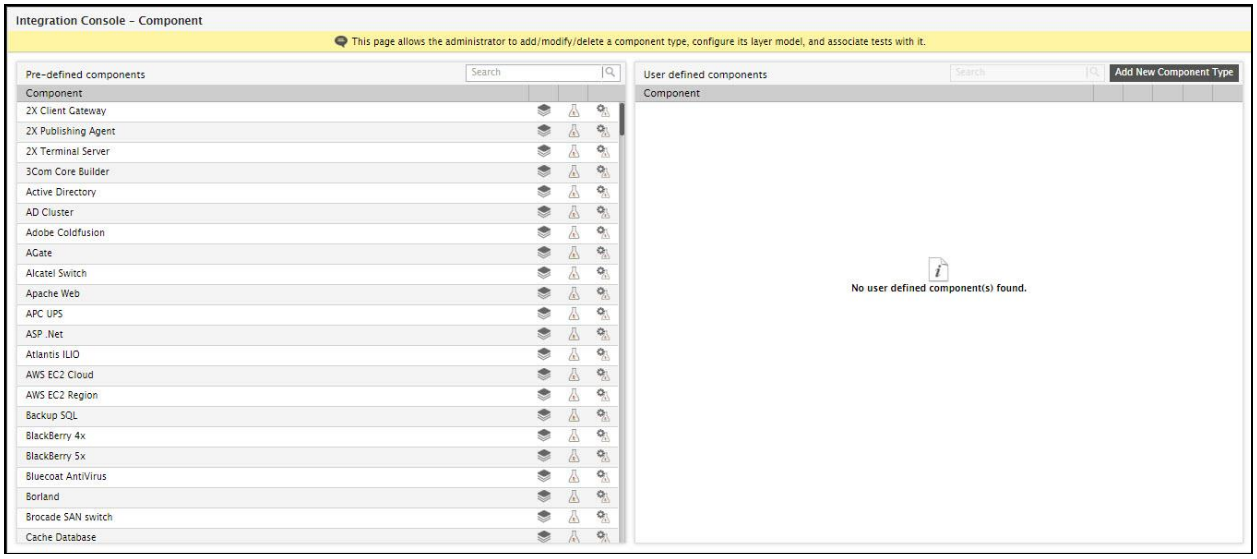
This chapter discusses each of these steps in detail, using an illustrated example. In this example, we will be creating a monitoring model for the *Tuxedo_Domain_Server* component. As part of this exercise, we will be:

- Creating a new *Tuxedo_Domain_Server* component-type;
- Building a layer model for this component-type by grouping together some user-defined and pre-defined layers;
- Disassociating some tests that are by default mapped to the layers supported by this component-type;
- Associating configuration tests with this component-type

4.1 Creating a New Component-type Using the Integration Console

To achieve this, follow the steps below:

1. Select the **Component** option from the **Integration Console** tile of Figure 2.1.
2. Figure 4.1 will then appear.



Integration Console – Component

This page allows the administrator to add/modify/delete a component type, configure its layer model, and associate tests with it.

Pre-defined components

Component
2X Client Gateway
2X Publishing Agent
2X Terminal Server
3Com Core Builder
Active Directory
AD Cluster
Adobe Coldfusion
AGate
Alcatel Switch
Apache Web
APC UPS
ASP .Net
Atlantis ILIO
AWS EC2 Cloud
AWS EC2 Region
Backup SQL
BlackBerry 4x
BlackBerry 5x
Bluecoat AntiVirus
Borland
Brocade SAN switch
Cache Database

User defined components

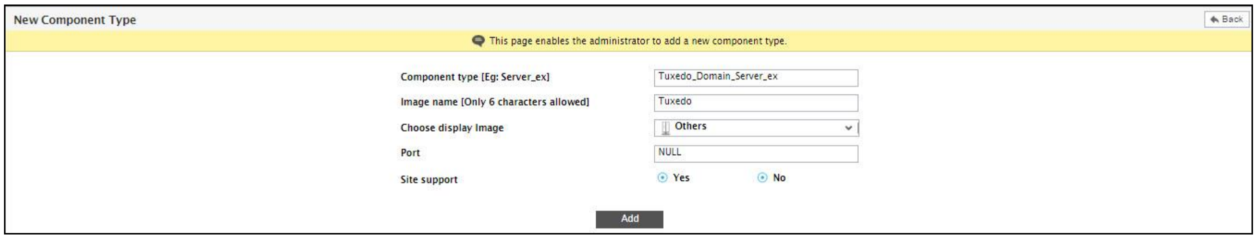
Add New Component Type

Component

No user defined component(s) found.

Figure 4.1: Viewing the user-defined and pre-defined component types

3. Figure 4.1 displays two panels: one is the **Pre-defined components** panel, which lists all component-types that are supported out-of-the-box by the eG Enterprise system, and the other is the **User-defined components** panel, which lists all custom defined component-types (if any), added using the Integration Console.
4. To add a new component type, click the **Add New Component Type** button in Figure 4.2. Figure 4.2 will then appear.



New Component Type

Back

This page enables the administrator to add a new component type.

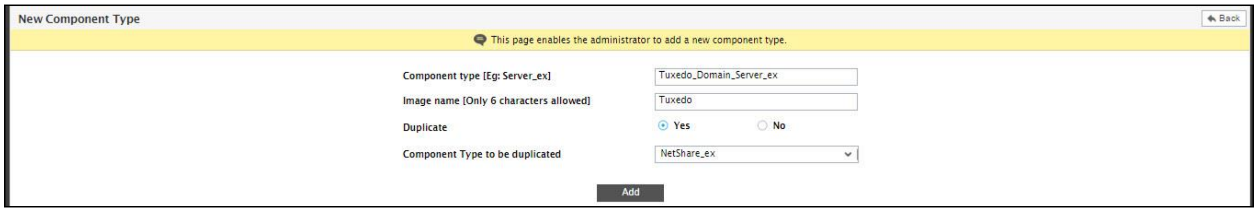
Component type [Eg: Server_ex]	Tuxedo_Domain_Server_ex
Image name [Only 6 characters allowed]	Tuxedo
Choose display Image	Others
Port	NULL
Site support	Yes No

Add

Figure 4.2: Adding a new component type using Integration Console

5. Provide the details of the new component-type in Figure 4.2. This includes:
   - **Component type**: The name of the new component-type, suffixed by _ex. In the case of our example, this will be *Tuxedo_Domain_Server_ex.*
   - **Image name:** Specify the **Image name** that will be displayed below the component type image in the eG user interface.

- **Choose display image:** Select the image that you want to use to represent the new component-type in the eG user interface.
- **Port**: If the component-type is port-based, then specify the port number at which the component listens by default. In the case of our example, leave the port as *NULL*.
- **Site support**: Indicate whether/not the component-type supports web sites. If so, set this flag to **Yes**. If not, set this flag to **No**. In the case of our example, set this flag as **No**.
- **Duplicate**: **This flag will appear only if one/more user-defined components pre-exist.** You can set this flag to **Yes** if you want the new component-type to inherit the properties of an existing user-defined component-type. In this case, you will be additionally required to pick the **Component Type to be duplicated** (see Figure 4.3).


New Component Type

Back

This page enables the administrator to add a new component type.

Component type [Eg: Server_ex]	Tuxedo_Domain_Server_ex
Image name [Only 6 characters allowed]	Tuxedo
Duplicate	[✓ Yes] No
Component Type to be duplicated	NetShare_ex

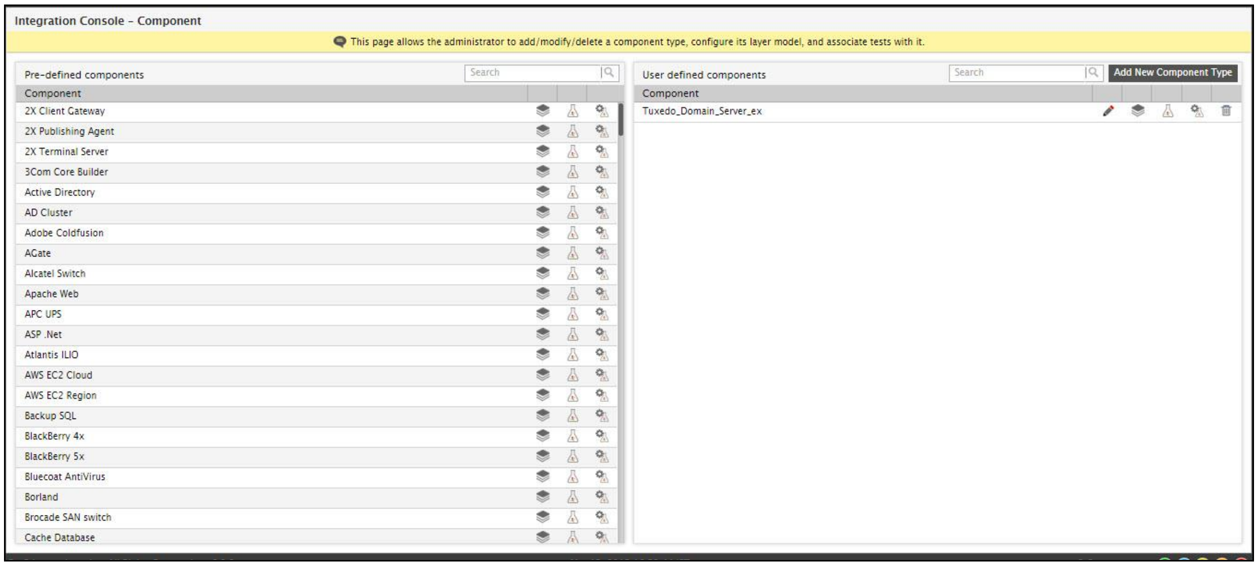
Add

Figure 4.3: Duplicating a component type

Once a component type to be duplicated is chosen, the **Display image**, **Port**, and **Site support** settings of that component-type will automatically apply to the component-type being added.

On the other hand, if the new component-type is not a duplicate of an existing user-defined component-type, set this flag to **No**. In this case, you will have to explicitly define a display image, port, and site support settings for the new component-type.

6. Finally, click the **Add** button in Figure 4.3 to add the new component-type.
7. The newly added component-type will then appear as a **User-defined component** (see Figure 4.4).


Integration Console – Component

This page allows the administrator to add/modify/delete a component type, configure its layer model, and associate tests with it.

Pre-defined components

Component
2X Client Gateway
2X Publishing Agent
2X Terminal Server
3Com Core Builder
Active Directory
AD Cluster
Adobe Coldfusion
AGate
Alcatel Switch
Apache Web
APC UPS
ASP .Net
Atlantis ILIO
AWS EC2 Cloud
AWS EC2 Region
Backup SQL
BlackBerry 4x
BlackBerry 5x
Bluecoat AntiVirus
Borland
Brocade SAN switch
Cache Database

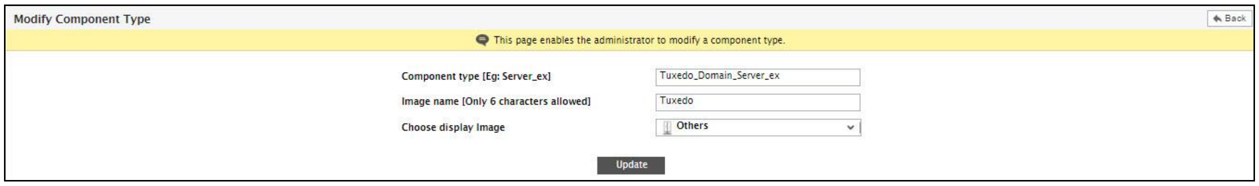
User defined components

Add New Component Type

Component
Tuxedo_Domain_Server_ex

Figure 4.4: The User-defined components panel displaying the newly added component-type

8. You can modify the details of the new component-type, if you so need, by clicking the button corresponding to it in Figure 4.4. Figure 4.5 will then appear. In the **Modify** mode, you can change the **Image name** and choose a different **display image** for the component. However, you cannot change the name of the **Component type**; nor can you change the **Port** and **Site support** settings. After making the required changes, click the **Update** button in Figure 4.5 to save the changes.


Modify Component Type

This page enables the administrator to modify a component type.

Component type [Eg: Server_ex]	Tuxedo_Domain_Server_ex
Image name [Only 6 characters allowed]	Tuxedo
Choose display Image	Others

Figure 4.5: Modifying the details of a user-defined component-type

9. You can even delete a user-defined component-type by clicking the button corresponding to that component-type in the **User defined components** list of Figure 4.4.

Pre-defined components can neither be modified nor deleted.

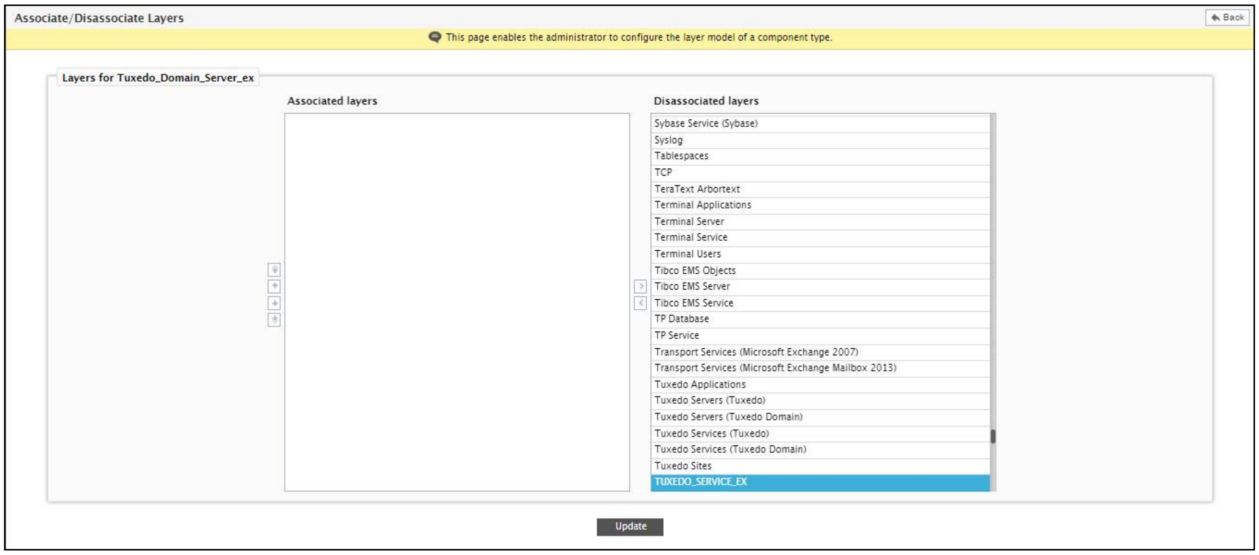
4.2 Building a Layer Model for a New Component Type

Let us now build a layer model for the new component-type we added in Section 4.1 – i.e., the **Tuxedo_Domain_Server_ex**. Say, the layer model of this component comprises of two **pre-defined layers** of the eG Enterprise system - **Operating System** and **Network** – and one **user-defined layer** named *TUXEDO_SERVICES_ex*.

To build such a model, follow the steps below:

1. Click the button corresponding to the **Tuxedo_Domain_Server_ex** component in the **User-defined components** list of Figure 4.4.

2. Figure 4.6 will then appear. From the **Disassociated layers** list of Figure 4.6, select the *TUXEDo-SERVICES_EX* layer that you want to associate with the **Tuxedo_Domain_Server_ex** component-type.



Associate/Disassociate Layers

This page enables the administrator to configure the layer model of a component type.

Layers for Tuxedo_Domain_Server_ex

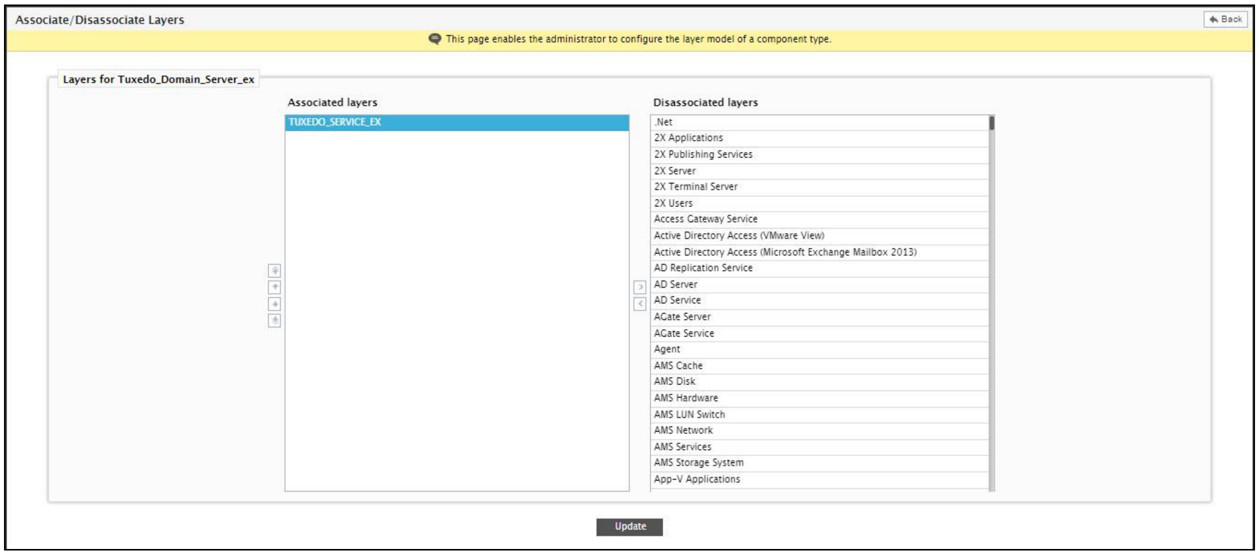
Associated layers

Disassociated layers

Sybase Service (Sybase)
Syslog
Tablespaces
TCP
TeraText Arbortext
Terminal Applications
Terminal Server
Terminal Service
Terminal Users
Tibco EMS Objects
Tibco EMS Server
Tibco EMS Service
TP Database
TP Service
Transport Services (Microsoft Exchange 2007)
Transport Services (Microsoft Exchange Mailbox 2013)
Tuxedo Applications
Tuxedo Servers (Tuxedo)
Tuxedo Servers (Tuxedo Domain)
Tuxedo Services (Tuxedo)
Tuxedo Services (Tuxedo Domain)
Tuxedo Sites
TUXEDO_SERVICE_EX

Figure 4.6: Selecting the layer to be associated with the new component-type

3. Then, click the **<** button in Figure 4.6 to move the selection to the **Associated layers** list (see Figure 4.7).



Associate/Disassociate Layers

This page enables the administrator to configure the layer model of a component type.

Layers for Tuxedo_Domain_Server_ex

Associated layers

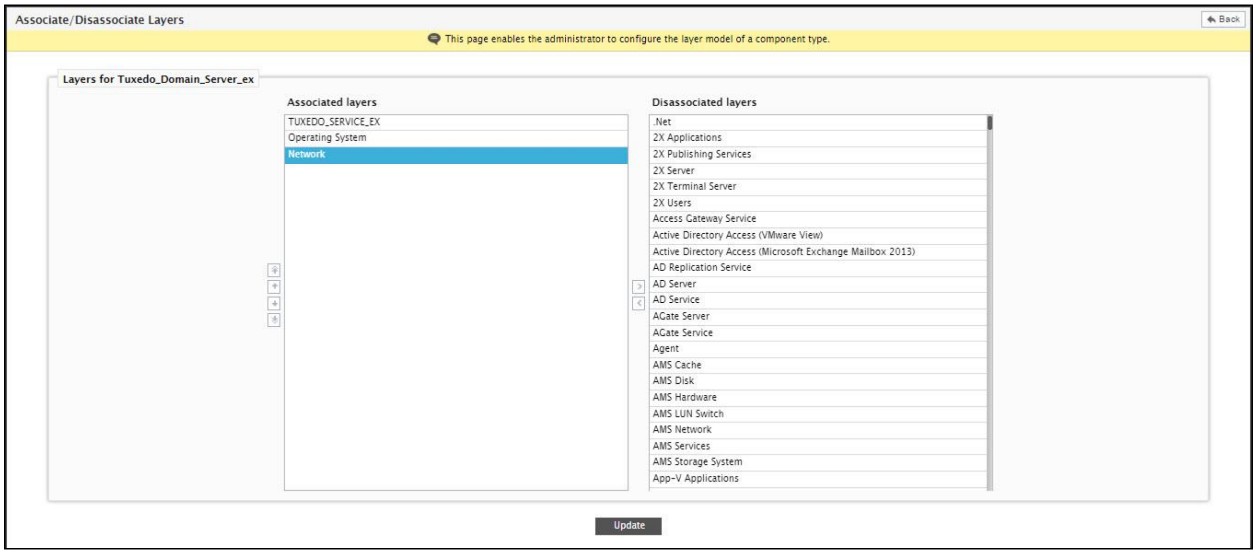
TUXEDO_SERVICE_EX

Disassociated layers

.Net
2X Applications
2X Publishing Services
2X Server
2X Terminal Server
2X Users
Access Gateway Service
Active Directory Access (VMware View)
Active Directory Access (Microsoft Exchange Mailbox 2013)
AD Replication Service
AD Server
AD Service
ACate Server
ACate Service
Agent
AMS Cache
AMS Disk
AMS Hardware
AMS LUN Switch
AMS Network
AMS Services
AMS Storage System
App-V Applications

Figure 4.7: Associating a layer with a new component-type

4. Similarly, associate the **Network** and **Operating System** layers too with the **Tuxedo_Domain_Server_ex** (see Figure 4.8).



Associate/Disassociate Layers

Back

This page enables the administrator to configure the layer model of a component type.

Layers for Tuxedo_Domain_Server_ex

Associated layers	Disassociated layers
TUXEDO_SERVICE_EX	.Net
Operating System	2X Applications
Network	2X Publishing Services
	2X Server
	2X Terminal Server
	2X Users
	Access Gateway Service
	Active Directory Access (VMware View)
	Active Directory Access (Microsoft Exchange Mailbox 2013)
	AD Replication Service
	AD Server
	AD Service
	AGate Server
	AGate Service
	Agent
	AMS Cache
	AMS Disk
	AMS Hardware
	AMS LUN Switch
	AMS Network
	AMS Services
	AMS Storage System
	App-V Applications

Update

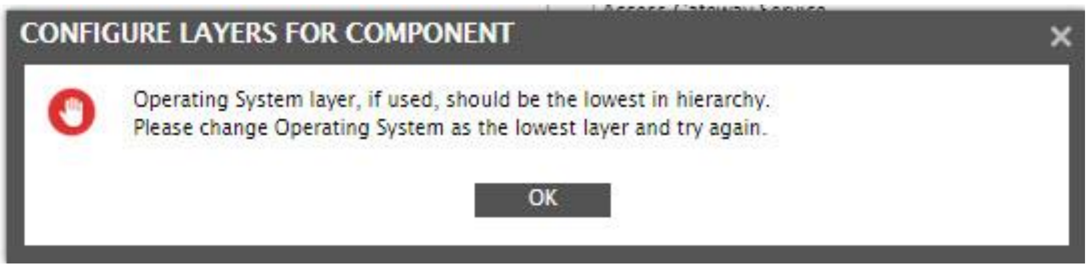
Figure 4.8: Associating multiple layers with the new component-type

5. If you need, you can even disassociate a layer from a component-type by selecting that layer from the **Associated layers** list and clicking the **>** button. For our example however, you need not disassociate any of the layers in the **Associated layers** list.

6. Now that the layer model of the **Tuxedo_Domain_Server_ex** is complete, click the **Update** button in Figure 4.8 to save the changes.

7. Typically, layers are to be associated in the same order in which they should appear in the layer model representation in the eG monitoring console. Moreover, since state of the layers below impact the state of the layers above, exercise caution when positioning layers in your layer model. By default however, the eG Enterprise system reserves the bottom-most position to the **pre-defined Operating System** layer. This is why, when a new layer model is built using the Integration Console, and the **Operating System** layer is included in that model, eG Enterprise expects this layer to be lowest layer in the layer hierarchy. If not, the eG Enterprise system throws an exception to this effect.

   In the case of our example too (see Figure 4.8), you can see that the **Operating System** layer is not the last layer. This is why, as soon as the **Update** button in Figure 4.8 is clicked, the following error message appears:



CONFIGURE LAYERS FOR COMPONENT

Operating System layer, if used, should be the lowest in hierarchy.
Please change Operating System as the lowest layer and try again.

OK

Figure 4.9: An error message prompting you to change the position of the Operating System layer

8. Click the **OK** button in Figure 4.9 to close the message, and then proceed to change the position of the

Operating System layer. Typically, to change the position of any layer in the layer model, you will have to use the direction buttons provided to the right of the **Associated layers** list in Figure 4.9. The table below lists these buttons and their purpose:

Direction Button	Purpose
⤓	Click to push a layer to the bottom of the layer model.
⤒	Click to push a layer to the top of the layer model.
↑	Click to push a layer a step up.
↓	Click to push a layer a step down.

9. To change the position of the **Operating System** layer in our example, select that layer from the **Associated** layers list of Figure 4.9 and click the ↓ button. When this is done, the **Operating System** layer instantly swaps positions with the **Network** layer, thus becoming the last layer of the model. If you now click the **Update** button to save the changes, you will notice that the error message of Figure 4.9 above does not re-appear.

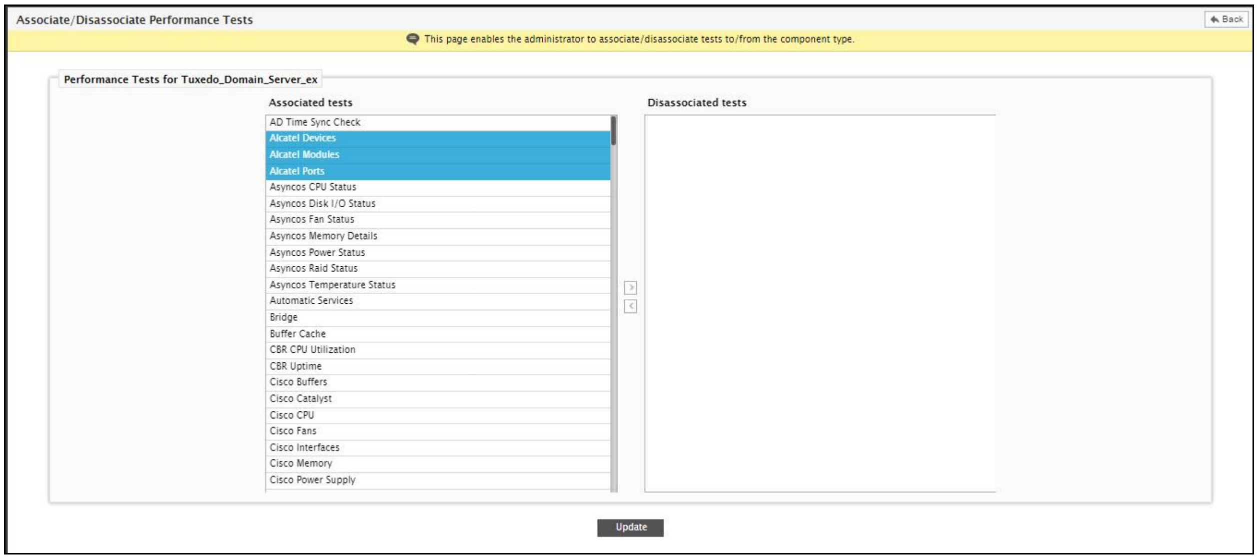
Note

You can modify the layer model definition of a **Pre-defined component** by clicking the button corresponding to that component in the **Pre-defined components** panel. The rest of the procedure is the same as outlined in steps 2-9 in this section.

4.3 Associating/Disassociating Tests from a New Component Type

Typically, once a layer model is defined for a new component-type, all tests that are mapped to each of those layers will automatically get associated with that component-type. Sometimes, you may want to exclude one or a few of these tests for a specific component-type. For instance, let us assume that 3 Alcatel switch-related tests are associated with the **Network** layer. Since this layer is now mapped to the **Tuxedo_Domain_Server_ex** component-type in our example, these 3 Alcatel tests will now run for the **Tuxedo_Domain_Server_ex** component as well. These Alcatel tests however will not provide any information that is of significance to a **Tuxedo_Domain_Server_ex** component. Hence, it is best that these tests are disassociated from the **Tuxedo_Domain_Server_ex** component alone. Let us see, how this can be achieved:

1. Click the button corresponding to the **Tuxedo_Domain_Server_ex** component-type in the **User-defined components** panel of Figure 4.4.
2. This will invoke Figure 4.10. From the **Associated tests** list of Figure 4.10, select the Alcatel tests that you want to exclude for this component.



Associate/Disassociate Performance Tests

This page enables the administrator to associate/disassociate tests to/from the component type.

Performance Tests for Tuxedo_Domain_Server_ex

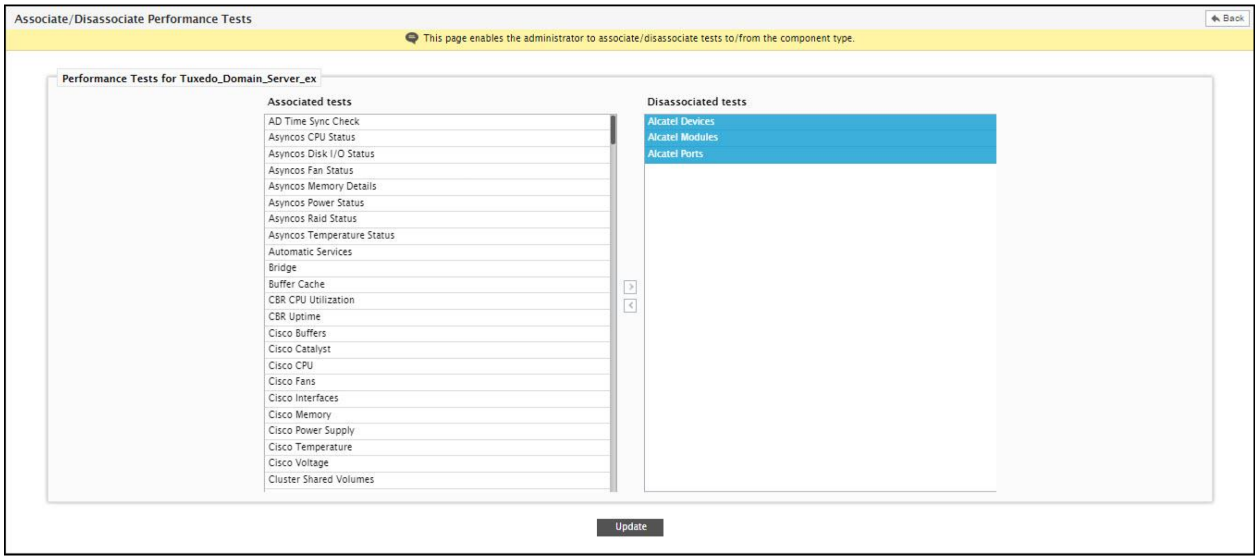
Associated tests

AD Time Sync Check
Alcatel Devices
Alcatel Modules
Alcatel Ports
Asyncos CPU Status
Asyncos Disk I/O Status
Asyncos Fan Status
Asyncos Memory Details
Asyncos Power Status
Asyncos Raid Status
Asyncos Temperature Status
Automatic Services
Bridge
Buffer Cache
CBR CPU Utilization
CBR Uptime
Cisco Buffers
Cisco Catalyst
Cisco CPU
Cisco Fans
Cisco Interfaces
Cisco Memory
Cisco Power Supply

Disassociated tests

Figure 4.10: Selecting the tests to be disassociated from the new component-type

3. Click the **>** button to disassociate the chosen tests. This will transfer the selection to the **Disassociated tests** list of Figure 4.11.



Associate/Disassociate Performance Tests

This page enables the administrator to associate/disassociate tests to/from the component type.

Performance Tests for Tuxedo_Domain_Server_ex

Associated tests

AD Time Sync Check
Asyncos CPU Status
Asyncos Disk I/O Status
Asyncos Fan Status
Asyncos Memory Details
Asyncos Power Status
Asyncos Raid Status
Asyncos Temperature Status
Automatic Services
Bridge
Buffer Cache
CBR CPU Utilization
CBR Uptime
Cisco Buffers
Cisco Catalyst
Cisco CPU
Cisco Fans
Cisco Interfaces
Cisco Memory
Cisco Power Supply
Cisco Temperature
Cisco Voltage
Cluster Shared Volumes

Disassociated tests

Alcatel Devices
Alcatel Modules
Alcatel Ports

Figure 4.11: Disassociating tests for a component-type

4. If you want, you can even associate some of the disassociated tests by selecting the tests from the **Disassociated tests** list and clicking the **<** button in Figure 4.11. However, this need not be done for our example.

Note

You cannot associate a port-based test with a non-port-based component or vice-versa.

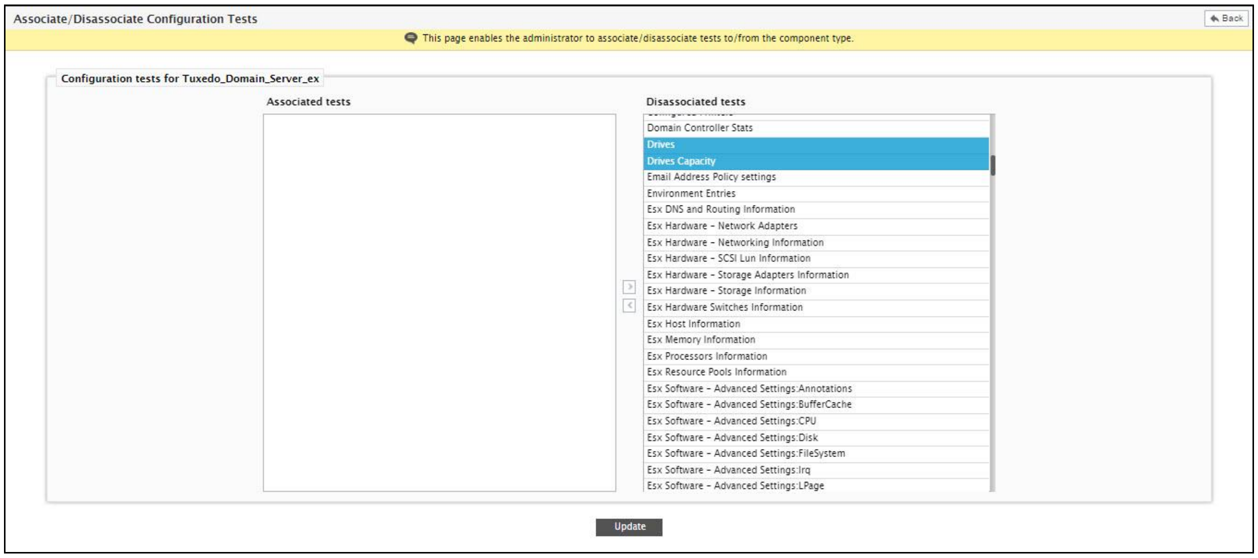
Finally, click the **Update** button in Figure 4.11 to save the changes.

Note

You can associate/disassociate performance tests for a **Pre-defined component** by clicking the button corresponding to that component in the **Pre-defined components** panel. The rest of the procedure is the same as outlined in steps 2-5 above.

Unlike performance tests, configuration tests are not mapped to any layer. This means that if you want one/more configuration tests to run on a new component-type, you will have to explicitly map these tests with that component-type. Let us see how the **Drives** and **Drives Capacity** configuration tests can be associated with the **Tuxedo_Domain_Server_ex** component in our example. To achieve this, do the following:

1. Click the button corresponding to the **Tuxedo_Domain_Server_ex** component in the **User-defined components** panel of Figure 4.4.
2. Figure 4.12 will then appear. From the **Disassociated tests** list of Figure 4.12, select the configuration tests that you want to associate with the **Tuxedo_Domain_Server_ex** component. Then, click the **<** button to associate the selected tests.


Associate/Disassociate Configuration Tests

This page enables the administrator to associate/disassociate tests to/from the component type.

Configuration tests for Tuxedo_Domain_Server_ex

Associated tests

Disassociated tests

Domain Controller Stats
Drives
Drives Capacity
Email Address Policy settings
Environment Entries
Esx DNS and Routing Information
Esx Hardware - Network Adapters
Esx Hardware - Networking Information
Esx Hardware - SCSI Lun Information
Esx Hardware - Storage Adapters Information
Esx Hardware - Storage Information
Esx Hardware Switches Information
Esx Host Information
Esx Memory Information
Esx Processors Information
Esx Resource Pools Information
Esx Software - Advanced Settings:Annotations
Esx Software - Advanced Settings:BufferCache
Esx Software - Advanced Settings:CPU
Esx Software - Advanced Settings:Disk
Esx Software - Advanced Settings:FileSystem
Esx Software - Advanced Settings:Irq
Esx Software - Advanced Settings:LPage

Figure 4.12: Selecting the configuration tests to be associated with a new component-type

Note

You cannot associate a port-based test with a non-port-based component or vice-versa.

3. Finally, click the **Update** button to save the changes.

Note

You can associate/disassociate configuration tests for a **Pre-defined component** by clicking the button corresponding to that component in the **Pre-defined components** panel. The rest of the procedure is the same as outlined in steps 2-3 above.

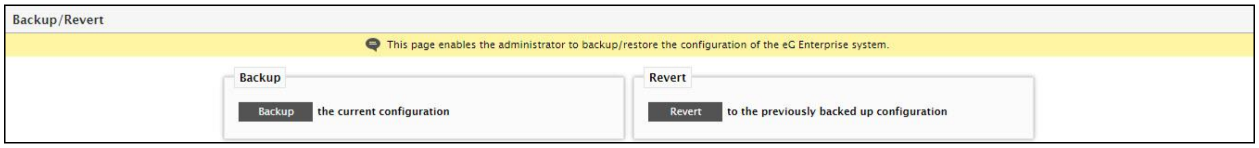
5

Backing Up and Restoring the Configuration of eG Enterprise

Using the eG Integration Console, you can take a backup of the configurations performed using the Integration Console module of the eG Enterprise system – for example, new tests / layers / components that were added using Integration Console. Similarly, you can also restore the backed up configuration any time you need.

To achieve this, select the **Backup** option from the **Integration Console** tile of the **Admin** tile menu (see Figure 2.1).

This will invoke Figure 5.1.


Backup/Revert

This page enables the administrator to backup/restore the configuration of the eG Enterprise system.

Backup
Backup
the current configuration
Revert
Revert
to the previously backed up configuration

Figure 5.1: Backing up/Restoring the configurations performed using IC

To backup the IC-based configuration changes, click the **Backup** button in Figure 5.1. To restore the backed up configurations, click the **Revert** button in Figure 5.1.

6

Conclusion

The eG Enterprise Suite has been specially designed keeping in mind the unique requirements of IT infrastructure operators. For more information on the eG family of products, please visit our web site at www.eginnovations.com.

For more details regarding eG Enterprise suite of products and the details of the metrics collected by the eG agents, please refer to the following documents:

- *Administering the eG Enterprise Suite*
- *Monitoring eG Enterprise*
- *The eG Installation Guide*
- *The eG Measurements Manuals*

We recognize that the success of any product depends on its ability to address real customer needs, and are eager to hear from you regarding requests for enhancements to the products, suggestions for modifications to the product, and feedback regarding what works and what does not. Please provide all your inputs as well as any bug reports via email to sales@eginnovations.com.