Monitoring Sun Java System Application Server

eG Innovations Product Documentation

www.eginnovations.com

eG Total Performance Visibility

Table of Contents

Table of Figures

Chapter 1: Introduction

The SunOne Application suite offers a comprehensive list of products for Internet infrastructures, i.e., web server, middleware application server, LDAP server, messaging server, and identity server, that are used in many domains such as banking, trading, healthcare, and logistics to support mission-critical services. IT infrastructures based on the SunOne Application suite follow the popular multi-tier architecture wherein the web server functions as the front-end receiving client requests, the application server hosts the business logic components, the identity server manages user policies, the directory server handles access rights and other user information lookups, and the database server stores and retrieves application data.

Routine monitoring of the infrastructure including the network, system, and application is imperative to ensure that the infrastructure functions at peak performance at all times. Since each Sun ONE application performs a different, specialized function, the monitoring has to be specific to each application – e.g., is the mail server delivering emails? is the application server's heap effectively sized?. More importantly, since the different Sun ONE applications inter-operate to support the end-user service, it is critical that the monitoring system track the inter-dependencies between applications in order to pin-point the exact source of a performance bottleneck in the infrastructure. eG Enterprise offers specialized monitoring models for each of the most popular application servers such as WebLogic, WebSphere, ColdFusion, Oracle 9i/10G, etc. A plethora of metrics relating to the health of the application servers can be monitored in real-time and alerts can be generated based on user-defined thresholds or auto-computed baselines. These metrics enable administrators to quickly and accurately determine server availability and responsiveness, resource usage at the host-level and at the application server level, how well the application server processes requests, how quickly the server completes transactions, overall server security, etc.

This document engages you in an elaborate discussion on how eG Enterprise monitors Sun Java System Application server in the market.

Chapter 2: How does eG Enterprise Monitor Sun Java System Application Server?

eG Enterprise is capable of monitoring the Sun Java System application server in both agent-based and agentless manners. Before attempting to monitor the server, first configure the server to work with the eG Enterprise. The procedure for achieving this is discussed in the below section.

2.1 Configuring the Sun Java System Application Server to work with eG

In order to ensure that the Sun Java System application server works smoothly with the eG product, the monitoring capability of a few key services has to be enabled. To achieve this, do the following:

1. Open the Internet Explorer and type the following URL in its address bar: *http://<WebSphereIP>:<WebSpherePort>* of the **Sun Java System server>:<Sun Java System server's admin port>/**, to open the Sun Java System application server's admin console.
2. Upon typing the URL, a dialog box requesting the **User Name** and **Password** of the administrator will appear (see Figure 2.1). Specify the same in the respective text boxes and click the **OK** button.

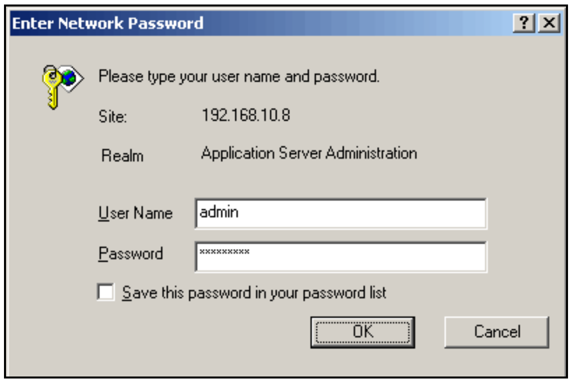
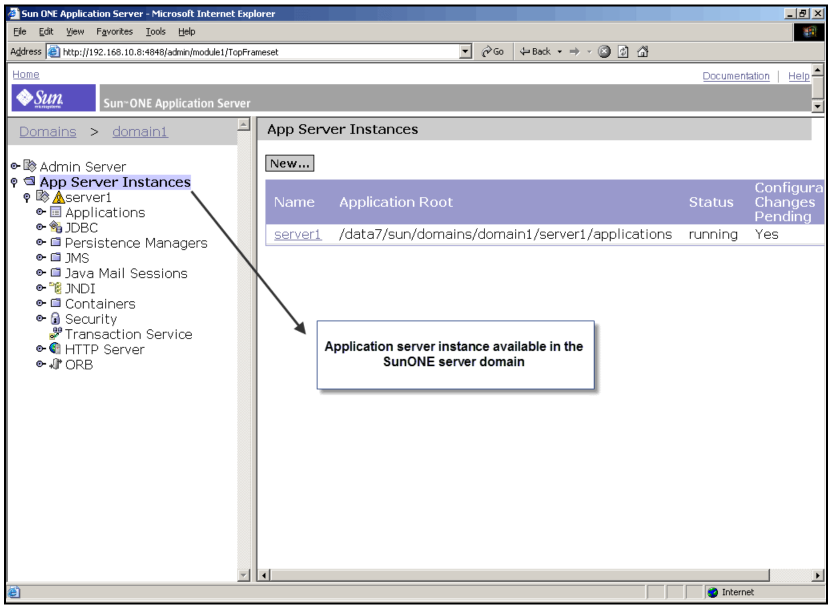

Enter Network Password
Please type your user name and password.
Site: 192.168.10.8
Realm Application Server Administration
User Name admin
Password *********
Save this password in your password list
OK
Cancel

Figure 2.1: Specifying the user name and password to log into the Sun Java System server's admin console

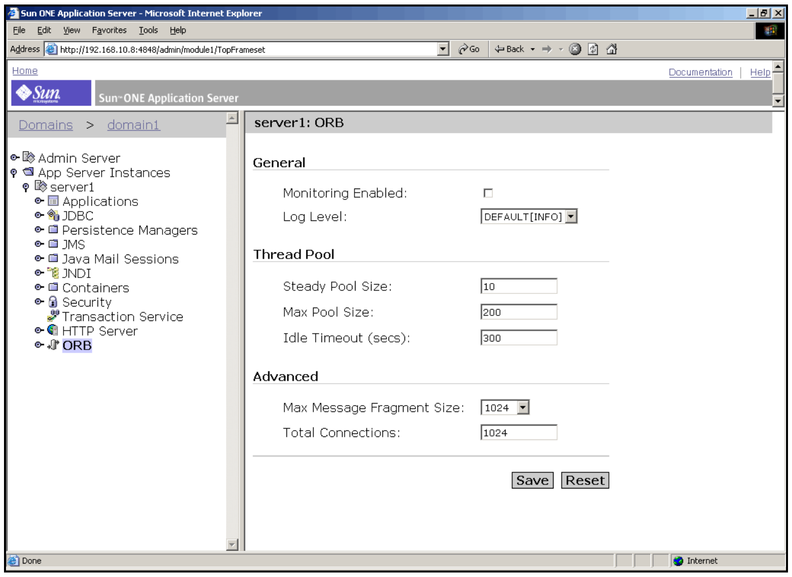
3. Upon successful login, the Sun Java System application server's admin console will open (see Figure 2.2).


Sun ONE Application Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://192.168.10.8:4848/admin/module1/TopFrameset
Go
Back
Home
Documentation | Help
Sun
Sun ONE Application Server
Domains > domain1
Admin Server
App Server Instances
server1
Applications
JDBC
Persistence Managers
JMS
Java Mail Sessions
JNDI
Containers
Security
Transaction Service
HTTP Server
ORB
App Server Instances
New...
Application server instance available in the SunONE server domain
Internet

Name	Application Root	Status	Configura Changes Pending
server1	/data7/sun/domains/domain1/server1/applications	running	Yes

Figure 2.2: The Sun Java System application server's admin console

4. The **App Server Instances** node in the left pane comprises of sub-nodes representing the application server instances present in the Sun Java System server domain . To monitor a particular application server instance, first, expand the sub-node corresponding to the same as indicated by Figure 2.2. This will display the services executing on the expanded application server instance.
5. Now, select the **ORB** service as indicated by Figure 2.3.



server1: ORB

General

Monitoring Enabled: ☐

Log Level: DEFAULT[INFO]

Thread Pool

Steady Pool Size: 10

Max Pool Size: 200

Idle Timeout (secs): 300

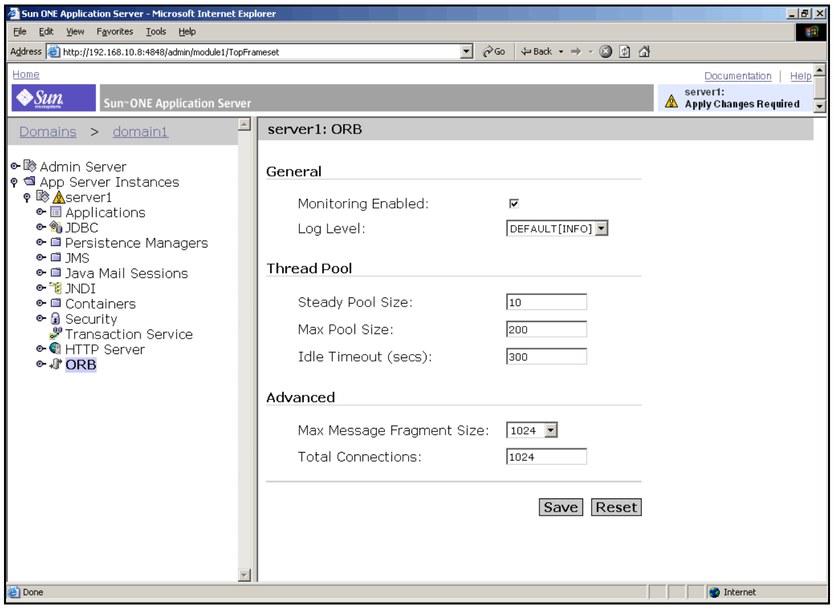
Advanced

Max Message Fragment Size: 1024

Total Connections: 1024

Figure 2.3: Enabling the monitoring capability of ORB

6. Figure 2.3 reveals that by default the **Monitoring Enabled** check box displayed in the right pane is deselected for the ORB service. This capability has to be enabled for the eG agents to effectively monitor the ORB service. To enable this capability, click on the check box and click the **Save** button below. Upon clicking, a link named **Apply Changes Required** (see Figure 2.4) will appear in the top right corner of the window. To apply the changes made, first, click on this link.



server1: ORB

General

Monitoring Enabled: ☑

Log Level: DEFAULT[INFO]

Thread Pool

Steady Pool Size: 10

Max Pool Size: 200

Idle Timeout (secs): 300

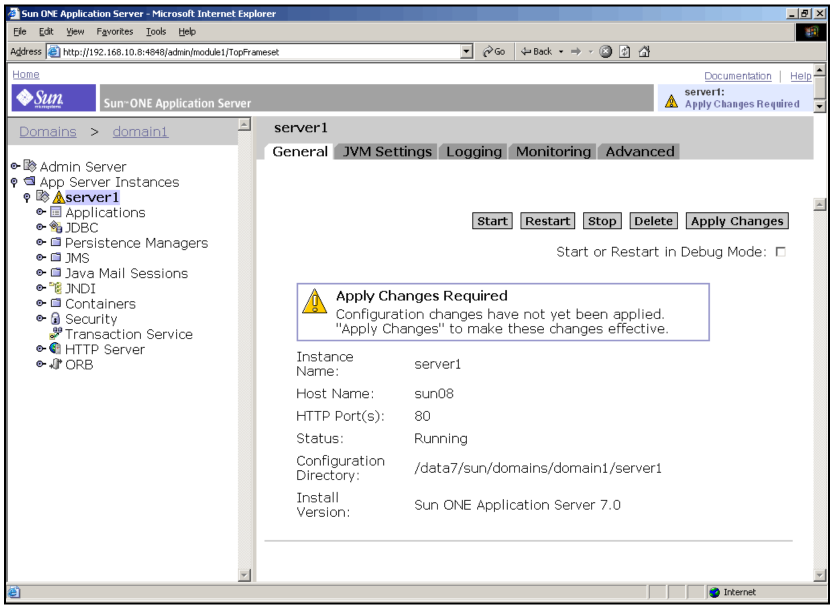
Advanced

Max Message Fragment Size: 1024

Total Connections: 1024

Figure 2.4: Clicking on the link

7. Once the link is clicked, a page depicted by Figure 2.5 will appear. Click on the **Apply Changes** button in this page to register the changes made.



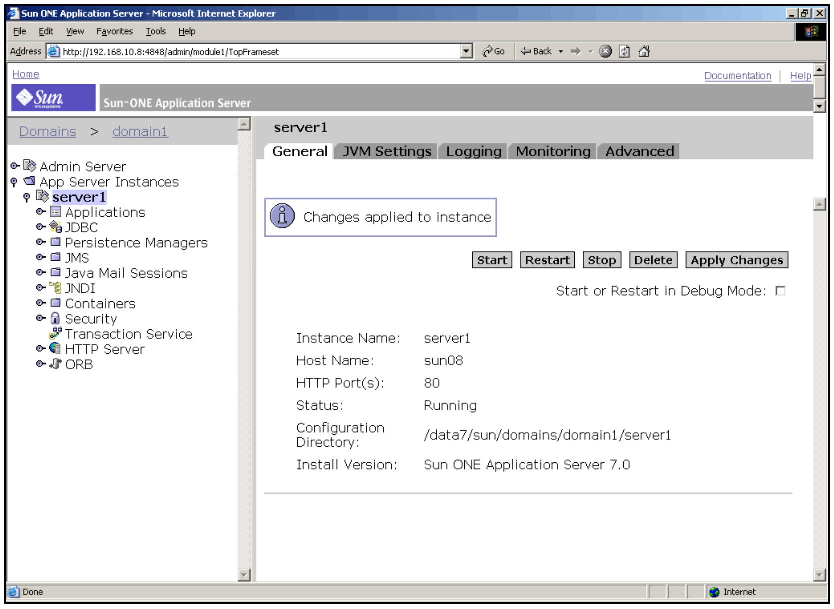
server1

Apply Changes Required
Configuration changes have not yet been applied. "Apply Changes" to make these changes effective.

Instance Name:	server1
Host Name:	sun08
HTTP Port(s):	80
Status:	Running
Configuration Directory:	/data7/sun/domains/domain1/server1
Install Version:	Sun ONE Application Server 7.0

Figure 2.5: Applying the changes

8. When this is done, the following confirmation message will appear (see Figure 2.6).



server1

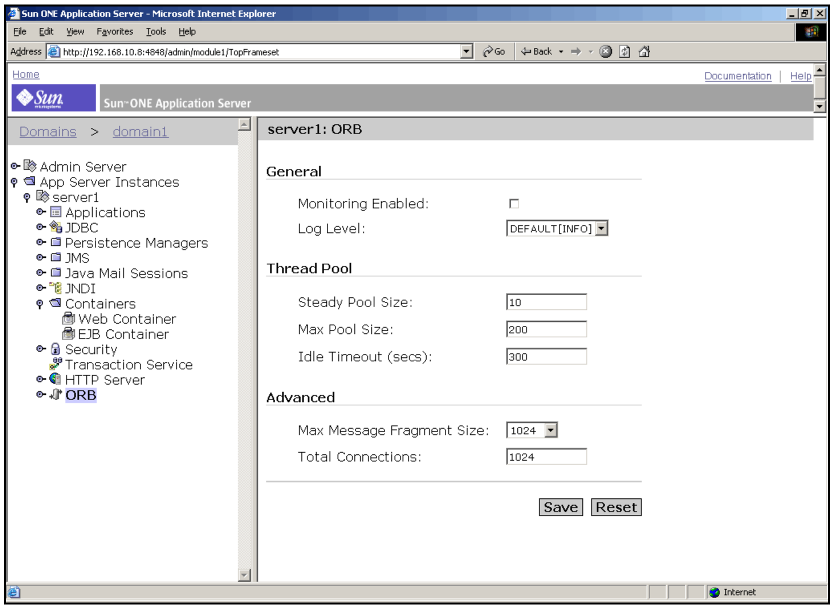
Changes applied to instance

Instance Name:	server1
Host Name:	sun08
HTTP Port(s):	80
Status:	Running
Configuration Directory:	/data7/sun/domains/domain1/server1
Install Version:	Sun ONE Application Server 7.0

Figure 2.6: A message confirming that changes have been applied to the instance

9. In some cases, a message prompting the user to restart the server will appear in this page. If such a message appears, click on the **Restart** button in this page to restart the server instance.

10. Repeat the procedure discussed above for the **Transaction Service** and the **EJB Container**, which can be seen in the left pane (see Figure 2.7). To view the **EJB Container** sub-node, expand the **Containers** node in the left pane (see Figure 2.7).



server1: ORB

General

Monitoring Enabled: ☐

Log Level: DEFAULT[INFO]

Thread Pool

Steady Pool Size: 10

Max Pool Size: 200

Idle Timeout (secs): 300

Advanced

Max Message Fragment Size: 1024

Total Connections: 1024

Figure 2.7: The EJB Container property

2.2 Configuring the eG Agent to Collect JVM-related Metrics from the Sun Java System Application Server 7.0

The **JVM** layer of the Sun Java System Application monitoring model is mapped to tests that report critical statistics related to the Sun Java System application server's JVM. These statistics typically reveal the following:

- The count of classes loaded/unloaded (**Java Classes** test)
- JVM thread usage (**JVM Threads** test)
- CPU and memory usage of the JVM (**JVM Cpu Usage** test and **JVM Memory Usage** test)
- The effectiveness of the JVM's garbage collection activity (**JVM Garbage Collections** test)
- The uptime of the JVM (**JVM Uptime** test)
- Whether JMX is currently enabled/disabled on the target WebLogic server (**JMX Connection**

to JVM test)

- The count and status of file descriptors (JVM File Descriptors test)

These tests connect to the JRE used by the Sun Java System application server to pull out the above-mentioned metrics. For these tests to work, the eG agent should be configured to connect to the JRE and collect the required metrics, using one of the following methodologies:

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

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

Note:

Version 7.0 of the Sun Java System Application server uses JDK 1.4 by default. Therefore, before enabling the optional JVM tests for this server, make sure you change to JDK 1.5.

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

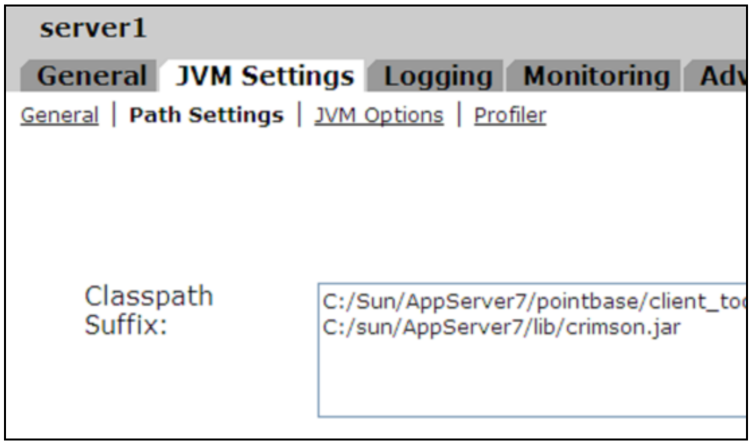
- First, determine whether the JMX requires no authentication at all, or requires authentication (but no security)
- If JMX does not require authentication, follow the steps below:
- Login to the target Sun Java System application server.
- Edit the *management.properties* file that is used by the JRE of the target *Sun Java System* application server, and configure the following in it:
- The JMX remote port
- Whether JMX is SSL-enabled or not
- Whether JMX requires authentication or not
- To know how to configure these, refer to the Monitoring Java Applications document.
- Save the file.

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

- Login to the target Sun Java System application server. If the server is executing on a Windows host, then, login to the host as a local/domain administrator.

- Next, copy the *jmxremote.password.template* file in the <JAVA_HOME>\jre\lib\management folder to any other location on the host, rename it as as *jmxremote.password*, and then, copy it back to the <JAVA_HOME>\jre\lib\management folder.
- Next, edit the *jmxremote.password* file and the *jmxremote.access* file to create a user with *read-write* access to the JMX. To know how to create such a user, refer to *Monitoring Java Applications* document.
- Then, proceed to make the *jmxremote.password* file secure by granting a single user "full access" to that file. To know how to achieve this, refer to the *Monitoring Java Applications* document.
- Edit the *management.properties* file that is used by the JRE of the target Sun Java System application server, and configure the following in it:
    - The JMX remote port
    - Whether JMX is SSL-enabled or not
    - Whether JMX requires authentication or not
    - The full path to the *jmxremote.access* file
    - The full path to the *jmxremote.password* file

  To know how to configure these, refer to the *Monitoring Java Applications* document.
- Then, save the file.
- Next, download the **crimson.jar** file from the URL, http://www.java2s.com/Code/Jar/ABC/Downloadcrimson113jar.htm , to any location on the Sun Java System application server host.
- Copy the **crimson.jar** file to the <SUN_ONE_APP_SERVER_HOME>\lib directory.
- Then, connect to the admin console of the Sun Java System application server using the URL:*<Sun Java SystemAppServerIP>:<Sun Java SystemPort>/*
- The default port number is: 4848
- Login to the console by providing valid credentials.
- Once the console opens, follow the node sequence click on **App Server Instances** and then click on the name of the Sun Java System application server instance being monitored.
- When Figure 2.8 appears, click on the **JVM Settings** tab page. Then, click on the **Path Settings** link. In the **Classpath Suffix** text area, add a line indicating the full path to the **crimson.jar** file.

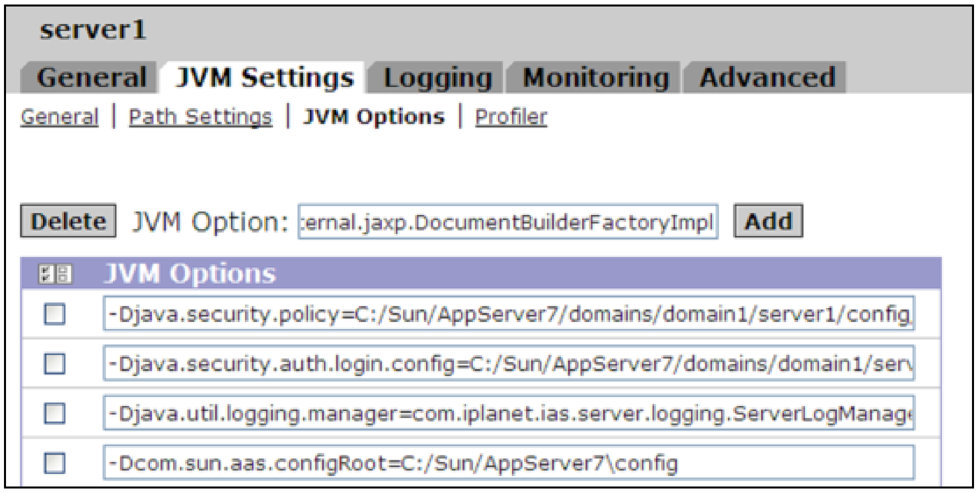


Classpath Suffix: C:/Sun/AppServer7/pointbase/client_to
C:/sun/AppServer7/lib/crimson.jar

Figure 2.8: Specifying the path to the crimson.jar file

- Save the settings and then, click he **JVM Options** link. In the **JVM Option** text box of Figure 2.9, specify the following:

```
-Dcom.sun.management.config.file=<Full path to the management.properties file of the
JRE used by the target Sun Java System Application server>
```



JVM Option: :ernal.jaxp.DocumentBuilderFactoryImpl

-Djava.security.policy=C:/Sun/AppServer7/domains/domain1/server1/config
-Djava.security.auth.login.config=C:/Sun/AppServer7/domains/domain1/serv
-Djava.util.logging.manager=com.iplanet.ias.server.logging.ServerLogManage
-Dcom.sun.aas.configRoot=C:/Sun/AppServer7\config

Figure 2.9: Configuring the path to the management.properties file

- Then, click the **Add** button in Figure 2.9 to add the path specification to the **JVM Options** section.
- Click the **General** link, click on **Apply changes**, and then restart the target server.
- If the server does not start and throws errors instead, check the **<App_ Server_ Home>\domains\domain1\server1\logs\server.log** for details.

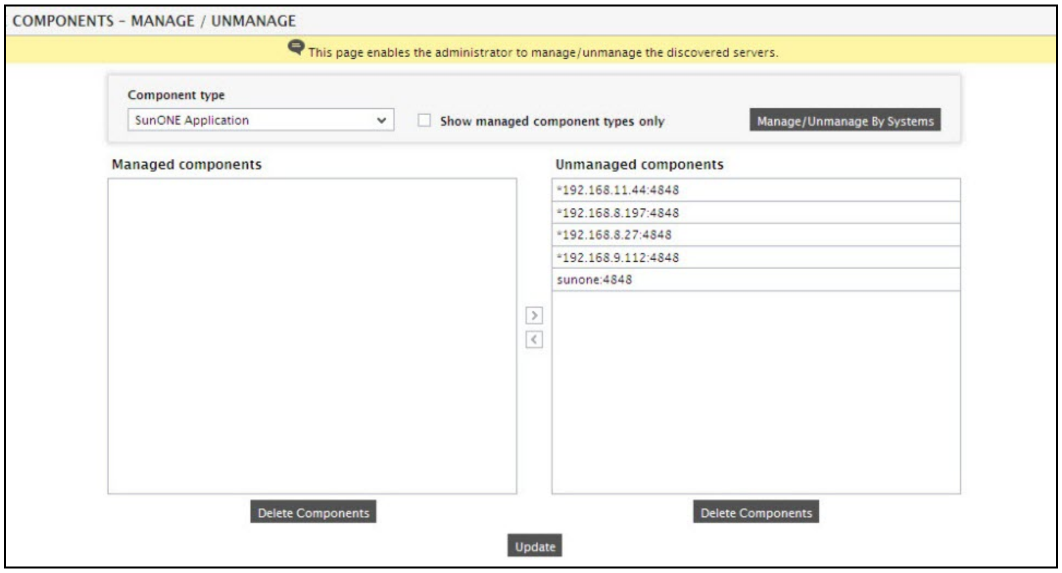
Note:

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

2.3 Managing the Sun Java System Application Server

eG Enterprise can automatically discover the Sun Java System Application Server in the environment and also lets you to manually add the Sun Java System Application Server for monitoring if the server is not auto-discovered. The following steps explain you how to manage the server that is auto-discovered using the eG administrative interface.

1. Log into the eG administrative interface.
2. If a Sun Java System Application Server is already discovered, then directly proceed towards managing it using the **COMPONENTS – MANAGE/UNMANAGE** page.
3. However, if it is yet to be discovered, then run discovery (Infrastructure -> Components -> Discover) to get it discovered or add the component manually using the **COMPONENTS**page (Infrastructure -> Components -> Add/Modify). Remember that components manually added are managed automatically. Discovered components, however, are managed using the **COMPONENTS – MANAGE / UNMANAGE** page.
4. To manage the component that is auto-discovered, follow the Infrastructure -> Components -> Manage/Unmanage in the **Infrastructure** tile of the **Admin** menu.
5. In the **COMPONENTS – MANAGE/UNMANAGE** page that appears next, select *SunONE Application* as the **Component type**. Then, the auto-discovered components will be displayed under the **Unmanaged Components** section.
6. Then, choose the component to be managed from the list. 2.3 and 2.3 clearly illustrate the process of managing the application server.



COMPONENTS - MANAGE / UNMANAGE

This page enables the administrator to manage/unmanage the discovered servers.

Component type: SunONE Application

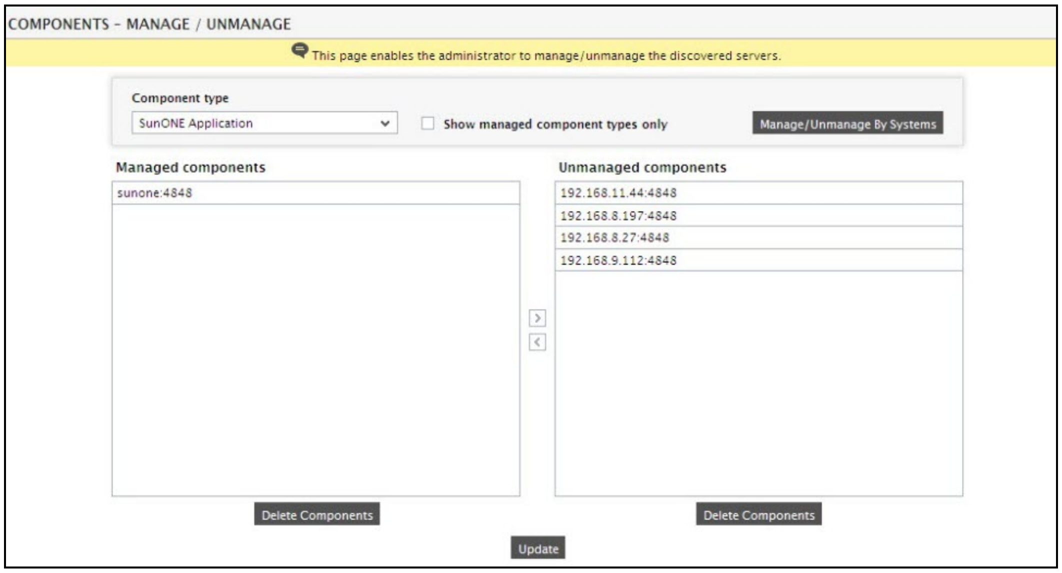
Show managed component types only

Managed components

Unmanaged components

*192.168.11.44:4848
*192.168.8.197:4848
*192.168.8.27:4848
*192.168.9.112:4848
sunone:4848

Figure 2.10: Selecting the Sun Java System Application server component to be managed



COMPONENTS - MANAGE / UNMANAGE

This page enables the administrator to manage/unmanage the discovered servers.

Component type: SunONE Application

Show managed component types only

Managed components

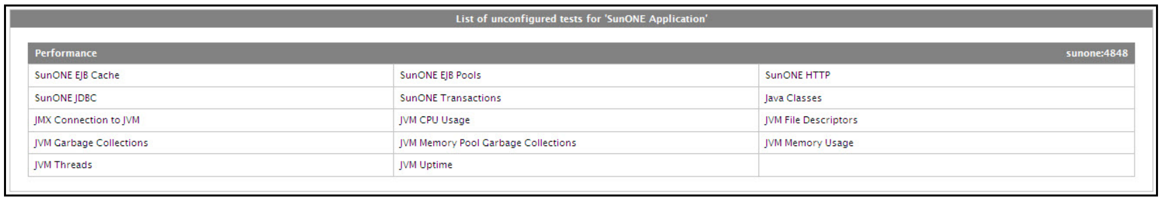
sunone:4848

Unmanaged components

192.168.11.44:4848
192.168.8.197:4848
192.168.8.27:4848
192.168.9.112:4848

Figure 2.11: Managing the selected Sun Java System Application server component

7. Next, try to sign out of the eG administrative interface. Upon doing so, a list of unconfigured tests will appear prompting you to configure the tests pertaining to the Sun Java System application server (see Figure 2.12) .

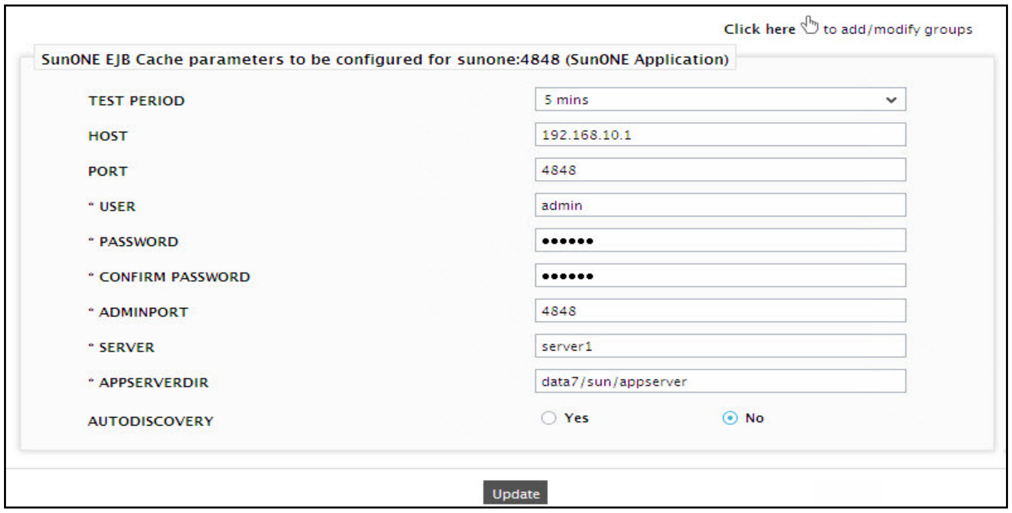


List of unconfigured tests for 'SunONE Application'

Performance		sunone:4848
SunONE EJB Cache	SunONE EJB Pools	SunONE HTTP
SunONE JDBC	SunONE Transactions	Java Classes
JMX Connection to JVM	JVM CPU Usage	JVM File Descriptors
JVM Garbage Collections	JVM Memory Pool Garbage Collections	JVM Memory Usage
JVM Threads	JVM Uptime	

Figure 2.12: List of unconfigured tests for the Sun Java System Application server

8. Click on the **SunONE EJB Cache** test in the table to configure it. Upon clicking, Figure 2.13 will appear. This test extracts the caching related metrics pertaining to such EJBs.



SunONE EJB Cache parameters to be configured for sunone:4848 (SunONE Application)

TEST PERIOD	5 mins
HOST	192.168.10.1
PORT	4848
* USER	admin
* PASSWORD	••••••
* CONFIRM PASSWORD	••••••
* ADMINPORT	4848
* SERVER	server1
* APPSERVERDIR	data7/sun/appserver
AUTODISCOVERY	Yes [✓ No]

Figure 2.13: Configuring the SunONE EJB Cache test

To know how to configure the test, refer to the **Monitoring Sun Java System Application Servers** chapter.

9. Finally, sign out of the administrative interface.

Chapter 3: Monitoring Sun Java System Application Servers

eG SunONE monitor offers extensive infrastructure monitoring capabilities for the SunONE application suite. A pre-built model for the SunONE application servers (see Figure 3.1) dictate what metrics are to be collected by eG agents, what thresholds are to be applied to the metrics, and how the metrics are to be correlated in order to assist with problem diagnosis.

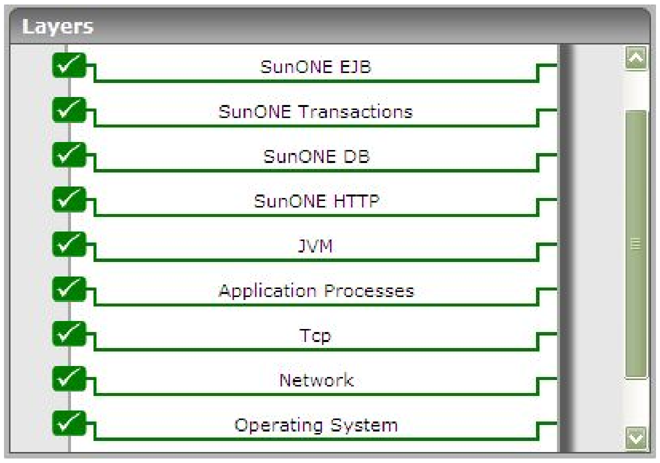


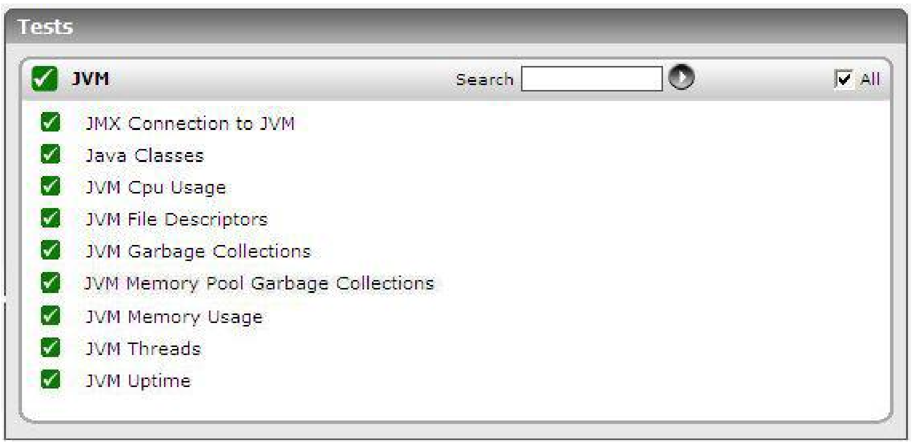
Figure 3.1: The layer model of a Sun Java System application server

Using the customized model for the Sun Java System Application server (see Figure 3.1), you can monitor the following:

- Server up/down time monitoring
- Server throughput tracking in terms of requests and data traffic
- JDBC connection pool usage levels and connection failure reports
- Thread pool execution status levels
- Transaction-related measures including the number of transactions that are completed, rolled back and in-progress
- EJB caching efficiency, bean pool-related metrics

3.1 The JVM Layer

This layer collectively reports the resource usage and overall health of the Sun Java System JVM.



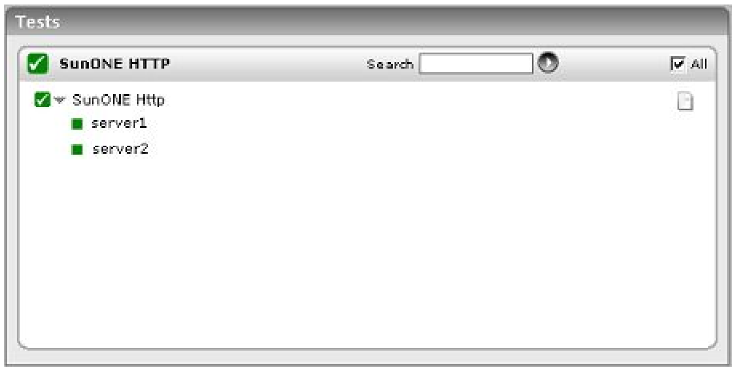
JMX Connection to JVM
Java Classes
JVM Cpu Usage
JVM File Descriptors
JVM Garbage Collections
JVM Memory Pool Garbage Collections
JVM Memory Usage
JVM Threads
JVM Uptime

Figure 3.2: The tests mapped to the JVM layer

All the tests displayed in Figure 3.2 have been dealt with in the Monitoring Java Applications document.

3.2 The SunONE HTTP Layer

To monitor the performance of the virtual web server on the Sun Java System server, use the Sun Java System Http test associated with this layer (see Figure 3.3).



SunONE Http
server1
server2

Figure 3.3: Tests associated with the SunONE HTTP layer

3.2.1 SunOne Http Test

This test measures the health of the virtual server configured for the Sun Java System Http server.

Target of the test : Any Sun Java System application server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every virtual server configured for the web server.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port number on which the Sun Java System application server is running.
User	A valid user name for the Sun Java System server to be monitored.
Password	Password for the Sun Java System server to be monitored.
Admin Port	The port number on which the “asadmin” tool runs.
Server	Name of the Sun Java System server to be monitored.
AppServerDir	The directory in which Sun Java System application server is installed.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Request rate	Rate of requests to the web server during the last measurement period.	Reqs/Sec	This measure reflects the server workload.
Data received	Rate at which the data is received by the server during the last measurement period	KB/Sec	
Data transmitted	Rate at which the data is transmitted by the server during the last measurement period.	KB/Sec	
Data transmit high watermark	The high-water-mark of the data transmit rate by this server	KB	
Open connections	Number of server threads/processes currently in use for serving requests	Number	Ideally, this measure must be low. Too many server threads or processes serving user requests could indicate a server-side bottleneck – either with the web server itself or with one of the

Measurement	Description	Measurement Unit	Interpretation
			backend components that the web server uses (e.g., database server).
Open connections high water mark	The high-water-mark of the open connections count	Number	Changes in the high water mark are indicative of times when the server has been a performance bottleneck. Note that the high water mark value gets reset every time the server is restarted.
200 responses	Percentage of responses with a status code in the range of 200-299 during the last measurement period	Percent	
300 responses	Percentage of responses with a status code in the range of 300-399 during the last measurement period	Percent	
400 errors	Percentage of errors with a status code in the range of 400-499 during the last measurement period	Percent	
500 errors	Percentage of errors with a status code in the range of 500-599 during the last measurement period	Percent	
Other responses	Percentage of responses with a status code that is greater than or equal to 600 during the last measurement period	Percent	

3.2.2 Web Service Test

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

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

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

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

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

- XML to tag the data
- SOAP to transfer a message
- WSDL to describe the availability of service.

The following are the major uses of the Web Services:

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

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

Target of the test : A Sun Java System application server

Agent deploying the test : An internal agent

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

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port number on which the Sun Java System application server is running.
WSDL Port	This test emulates a user accessing a specific web service(s) on the target server to determine the availability and responsiveness of the server. to enable this emulation, you need to configure the test with the URL of the web service that it should access. Specify this URL against the WSDL Port parameter. if required, you can even configure multiple WSDL URLs - one each for every web service that the test should attempt to access. If each WSDL URL configured requires special permissions for logging in, then, you need to configure the test with separate credentials for logging into every WSDL URL. Likewise, you need to provide instructions to the test on how to validate the content returned by every WSDL URL, and also set an encoding format for each wsdl url. to enable administrators to easily configure the above per WSDL URL, eg enterprise provides a special interface. to access this interface, click on the encircled '+' button alongside the url text box in the test configuration page. Alternatively, you can even click on the encircled '+' button adjacent to the WSDL URL parameter in the test configuration page. to know how to use this special interface, refer to Section **3.2.2.1**.
Operations	Once the WSDL URL(s) are specified, the operations that are offered by the web services and those that are to be monitored have to be configured. To select the required operations for monitoring, eG Enterprise provides a special interface. TO

Parameter	Description
	access this interface, click on the encircled '+' button alongside the Operations text box in the test configuration page. Alternatively, you can even click on the encircled '+' button adjacent to the Operations parameter in the test configuration page. To know how to use this special interface, refer to Section **3.2.2.2**.
Timeout	Specify the duration (in seconds) for which this test should wait for a response from the server. If there is no response from the server beyond the configured duration, the test will timeout. By default, this is set to 30 seconds.
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: • The eG manager license should allow the detailed diagnosis capability • Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
WSDL URL availability	Indicates whether the web service was able to respond successfully to the query made by the test.	Percent	Availability failures could be caused by several factors such as the web service process(es) being down, the web service being misconfigured, a network failure, etc. Temporary unavailability may also occur if the web service is overloaded. Availability is determined based on the response code returned by the service. A response code between 200 to 300 indicates that the service is available.
WSDL response time	Indicates the time taken by the eG agent to get the configured web service.	Secs	Response time being high denotes a problem. Poor response times may be due to the service being overloaded or

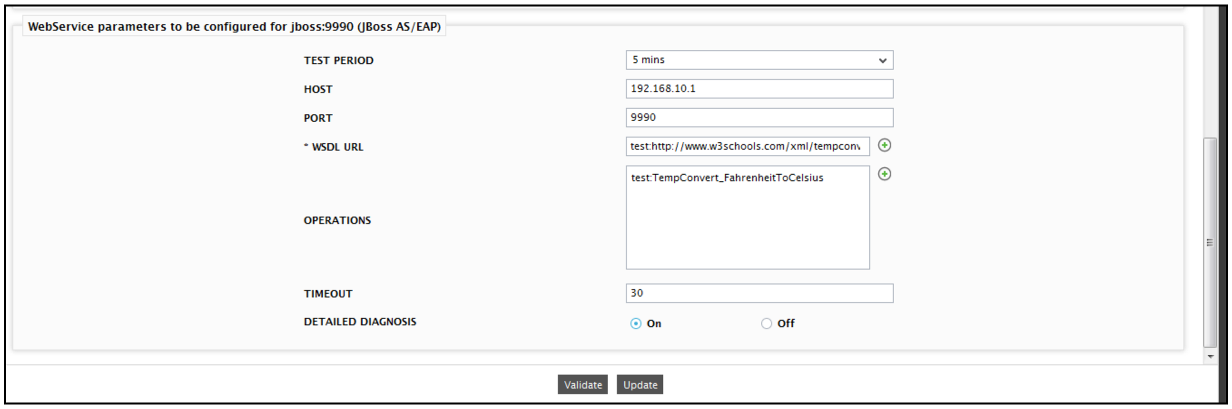
Measurement	Description	Measurement Unit	Interpretation
			misconfigured. If the URL accessed involves the generation of dynamic content by the service, backend problems (e.g., an overload at the application server or a database failure) can also result in an increase in response time.
Port status	Indicates whether/not the port of the web server is reachable.		The values reported by this measure and the corresponding numeric equivalents are listed in the table below: **Measure Values** | **Numeric Values** Yes | 1 No | 0 **Note:** By default, this measure reports the above-mentioned **Measure Value**s to indicate whether the server has been rebooted or not. In the graph of this measure however, the Measure Values are represented using the numeric equivalents only.
TCP connection availability	Indicates whether the test managed to establish a TCP connection to the server.	Percent	Failure to establish a TCP connection may imply that either the web server process is not up, or that the process is not operating correctly. In some cases of extreme overload, the failure to establish a TCP connection may be a transient condition. As the load subsides, the server may start functioning properly again.
TCP connect time	This measure quantifies the time for establishing a TCP connection to the web server host.	Secs	Typically, the TCP connection establishment must be very small (of the order of a few milliseconds). Since

Measurement	Description	Measurement Unit	Interpretation
			TCP connection establishment is handled at the OS-level, rather than by the application, an increase in this value signifies a system-level bottleneck on the host that supports the web server.
Server response time	Indicates the time period between when the connection was established and when the web server sent back a response header to the client.	Secs	While the total response time may depend on several factors, this measure is typically, a very good indicator of a server bottleneck (e.g., because all the available server threads or processes are in use).
Response code	The response code returned by the web server for the simulated request.	Number	A value between 200 and 300 indicates a good response. A 4xx value indicates a problem with the requested content (eg., page not found). A 5xx value indicates a server error.
Service availability	Indicates whether/not the web service is available.	Percent	A value of 100 indicates that the web service is available and a value of 0 indicates that the web service is not available.
Operation status	Indicates whether/not the configured operation is present in the web service.		This measure will not report metrics if the Operation parameter in the test configuration page is none in the test configuration page.
Operation Content length	Indicates the response code returned by the server for the simulated request.	Number	A value between 200 and 300 indicates a good response. A 4xx value indicates a problem with the requested content (e.g., page not found). A 5xx value indicates a server error. This measure will not report metrics if the Operation parameter in the test configuration page is none or if an invalid Value is specified or if the Value is not specified in the HTML View tab while configuring the operation for

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

3.2.2.1 Configuring Multiple WSDL URLs for Monitoring

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

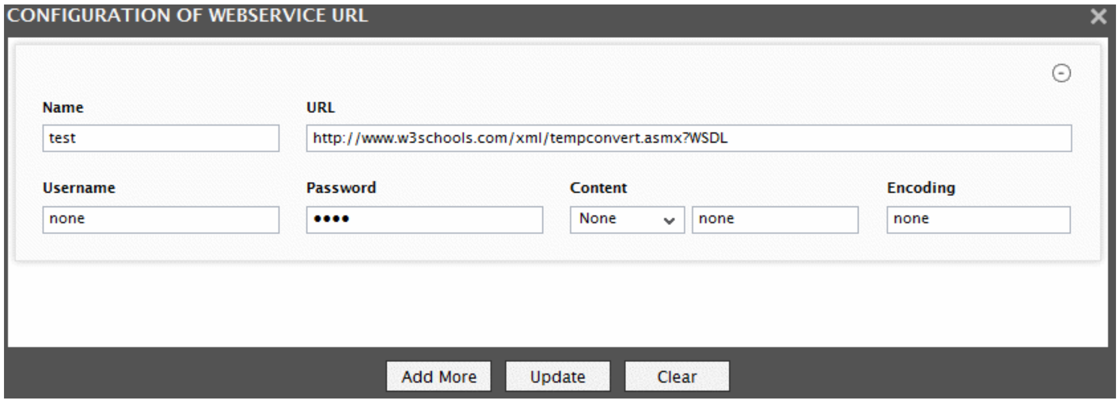


WebService parameters to be configured for jboss:9990 (JBoss AS/EAP)

TEST PERIOD	5 mins
HOST	192.168.10.1
PORT	9990
* WSDL URL	test:http://www.w3schools.com/xml/tempconv
OPERATIONS	test:TempConvert_FahrenheitToCelsius
TIMEOUT	30
DETAILED DIAGNOSIS	[✓ On] Off

Figure 3.4: Configuring the WebService test

1. Click on the encircled '+' button alongside the WSDL URL text box. Figure 3.5 will then appear.



CONFIGURATION OF WEBSERVICE URL

Name	URL
test	http://www.w3schools.com/xml/tempconvert.asmx?WSDL

Username	Password	Content		Encoding
none	••••	None	none	none

Figure 3.5: The WebService URL Configuration page

2. Specify the following in Figure 3.5:

   - **Name:** Specify a unique name by which the WSDL URL you will be specifiying shortly will be referred to across the eG user interface. This is the name that will appear as the descriptor of this test.
   - **URL:** Enter the WSDL URL of the web service that this test should access.

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

  - A valid **Username** and **Password** for the configured WSDL URL
  - *none* in both the **Username** and **Password** text boxes of the configured WSDL URL, if no user authorization is required
  - Some servers however, support NTLM (Integrated Windows) authentication, where valid login credentials are mandatory. In other words, a none specification will not be supported by such servers. Therefore, in this case, against each configured **WSDL URL**, you will have to provide a valid **Username** in the format: domainname\username, followed by a valid **Password**.
  - Please be sure to check if your web service requires HTTP authentication while configuring this parameter. HTTP authentication typically involves a separate pop-up window when you try to access the page. Many services use HTTP POST for obtaining the user name and password and validating the user login. In such cases, the username and password have to be provided as part of the POST information and NOT as part of the **CREDENTIALS** specification for the WebService test.

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

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

3. Similarly, you can add multiple URL specifications by clicking the **Add More** button. To remove a WSDL URL specification, click on the encircled '-' button corresponding to it. To clear all WSDL

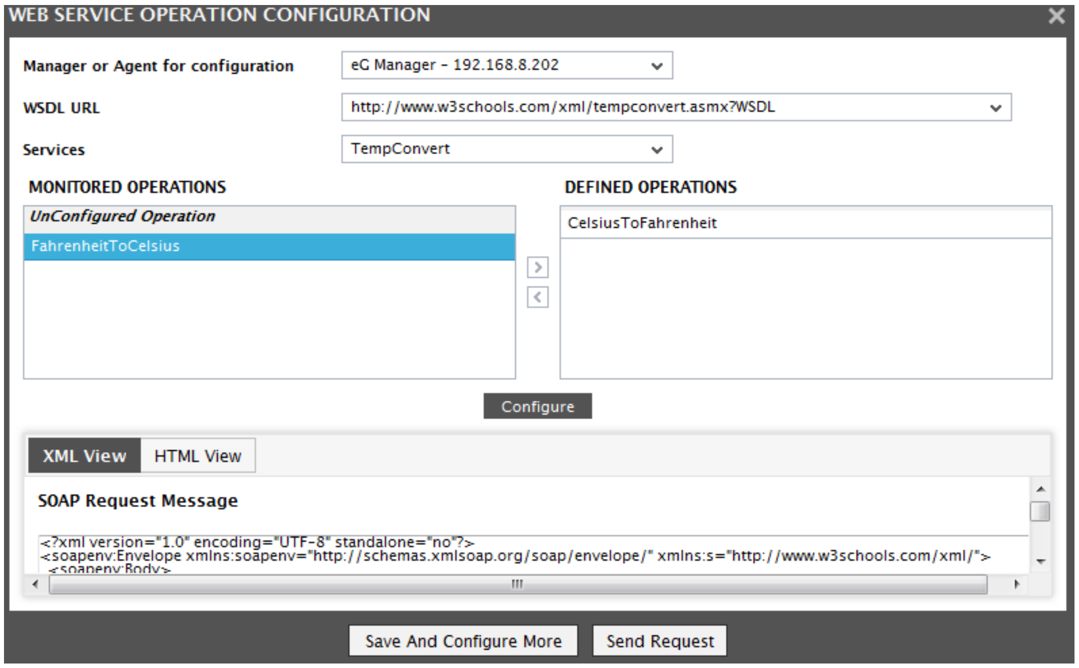
URL specifications, click the **Clear** button. To update all the changes you made, click the **Update** button.

4. Once **Update** is clicked, you will return to the test configuration page as shown in Figure 3.4. The WSDL URL text box in the test configuration page will display just the **Name**s - i.e., the unique display names - that you may have configured for the multiple WSDL URLs, as a comma-separated list. To view the complete WSDL URL specification, click the encircled '+' button alongside the WSDL URL text box, once again.

3.2.2.2 Configuring Multiple Operations for Monitoring - WebServiceTest

By default, the **Web Service Test** test will be configured with the WSDL URLs that offer the web services that are to be monitored. To configure the operations that are offered by the WSDL URLs, do the following:

1. Click on the encircled '+' button alongside the Operations text box as shown in Figure 3.4. Figure 3.6 will then appear.



Manager or Agent for configuration: eG Manager - 192.168.8.202
WSDL URL: http://www.w3schools.com/xml/tempconvert.asmx?WSDL
Services: TempConvert

MONITORED OPERATIONS: *UnConfigured Operation*, FahrenheitToCelsius
DEFINED OPERATIONS: CelsiusToFahrenheit

SOAP Request Message

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:s="http://www.w3schools.com/xml/">
  <soapenv:Body>
```

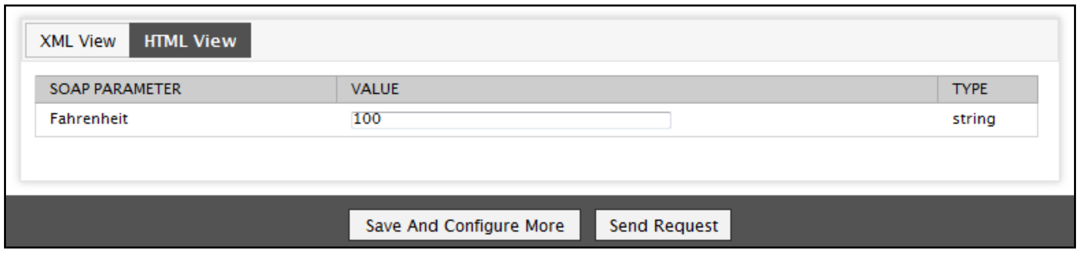
Figure 3.6: Configuring the Web Service Operation

2. Specify the following in Figure 3.6:

   ➢ **Manager/Agent for accessing WSDL URL**: Select the eG agent or the eG Manager that is authorized to access the configured WSDL URL from this list.

- **WSDL URL:** Once the eG agent/eG Manager is chosen from the *Manager/Agent for accessing WSDL URL* list, this list will be populated automatically with all the WSDL URLs specified in the **WSDL URL** text box (See Figure 3.4). Select the **WSDL URL** of your choice from this list.
- **Services:** The web services offered by the chosen WSDL URL will then be populated in this list. Select a service of your choice from this list.

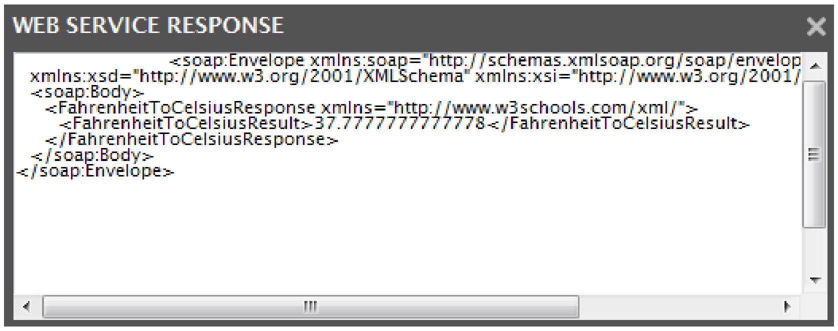
  - The operations that are offered by the chosen service will then be populated in the **DEFINED OPERATIONS** list. To monitor a chosen operation, select the operation and click the **<** button. This will move the chosen operation to the **MONITORED OPERATIONS** list.
  - Click the *Configure* button to save the changes.
  - The eG agent uses SOAP requests to obtain the necessary metrics from the web service. Once the operation is configured, the XML View of the SOAP Request corresponding to the chosen operation will be generated and listed in the **XML View** tab. Likewise, the **HTML View** tab lists the **SOAP Parameter** that is passed to collect the required metrics for the chosen operation.
  - To obtain operation-level statistics, it is important to specify a valid value in the *VALUE* text box of the *HTML View* tab as shown in Figure 3.6. Each time the test is executed, this value will be provided as an input to the chosen operation.



SOAP PARAMETER	VALUE	TYPE
Fahrenheit	100	string

Figure 3.7: Specifying the value for the chosen operation

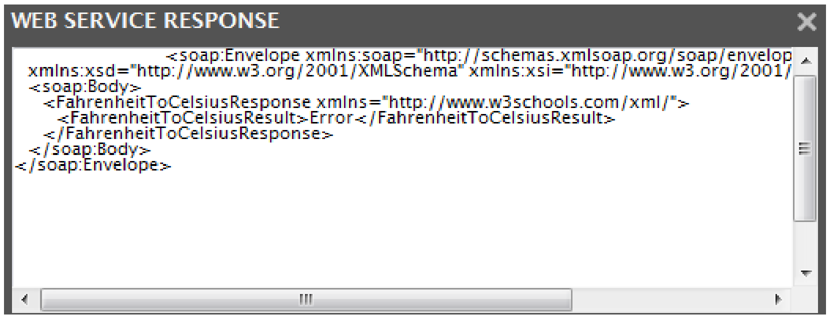
  - Click the *Save and Cofigure More* button to save the changes made.
  - If you wish to verify if the *VALUE* specified in the *HTML View* tab is valid, then you can do so by clicking the **Send Request** button. Figure 3.8 will then appear. If the value specified in the *VALUE* text box is indeed valid, then the operation will be performed on the value and the result will be specified. For example, if your chosen operation is FahrenheittoCelsius , the *SOAP Parameter* is Farenheit and the value that you wish to convert is 100, the result will be specified in the **WEB SERVICE RESPONSE** pop up window as below: *<FahrenheitToCelsiusResult>37.7777777777778</FahrenheitToCelsiusResult>*



```
                    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelop
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
  <soap:Body>
    <FahrenheitToCelsiusResponse xmlns="http://www.w3schools.com/xml/">
      <FahrenheitToCelsiusResult>37.777777777778</FahrenheitToCelsiusResult>
    </FahrenheitToCelsiusResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 3.8: The value that appears when the operation is performed successfully

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



```
                    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelop
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
  <soap:Body>
    <FahrenheitToCelsiusResponse xmlns="http://www.w3schools.com/xml/">
      <FahrenheitToCelsiusResult>Error</FahrenheitToCelsiusResult>
    </FahrenheitToCelsiusResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 3.9: An Error appearing during value conversion

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

3. Similarly, you can configure multiple Operations by clicking the **Configure** button in Figure 3.6. To remove an operation, select the operation from the **MONITORED OPERATION** list and click the > button.
4. Once **Save and Configure More** button is clicked, you will return to the test configuration page (see Figure 3.4). The **OPERATIONS** text box in the test configuration page will display just the operations, as a comma-separated list. To view the complete operation specification, click the encircled '+' button alongside the **OPERATIONS** text box, once again.

3.3 The SunONE DB Layer

The size of the JDBC connection pools governs the constant availability of connections to the database. The metrics reported by the SunONE DB layer assist you in keeping a close watch on the fluctuations in the pool size, so that additional connection requirements can be foreseen and allocated to the pool.

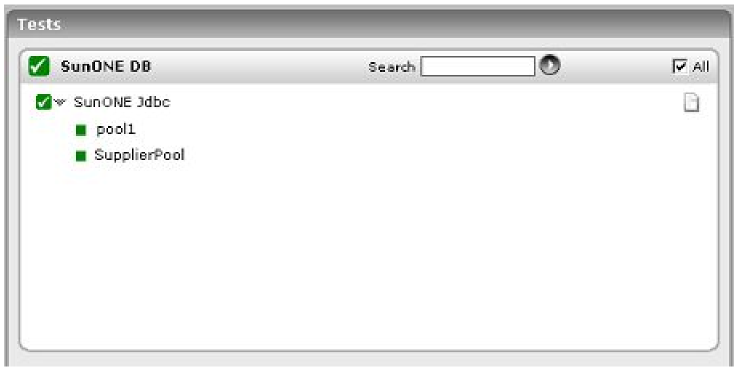


Figure 3.10: Tests associated with the SunONE DB layer

3.3.1 SunOne JDBC Test

This test reports the performance metrics related to the JDBC connection pools of a Sun Java System application server.

Target of the test : Any Sun Java System application server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every JDBC connection pool.

Configurable parameters for the test

Parameter	**Description**
Test period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port number on which the Sun Java System application server is running.
User	A valid user name for the Sun Java System server to be monitored.
Password	Password for the Sun Java System server to be monitored.
Admin Port	The port number on which the “asadmin” tool runs.
Server	Name of the Sun Java System server to be monitored.
AppServerDir	The directory in which Sun Java System application server is installed.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Waiting threads	The number of threads that are waiting for a database connection.	Number	Ideally, this measure should have a low value. A high value of this measure indicates that the database is slow or does not have additional connections to allocate to service requests from the web application server.
Connections failed	The total number of times a thread has failed to acquire a JDBC connection.	Number	This value can increase for various reasons – connection failures can happen if the database server is down or if the maximum limit of simultaneous connections possible has been reached. Alternatively, if the database pool is not adequately sized (i.e., too few maximum connections), connection failures can occur.
Connection timeouts	The total number of connection requests that have been timed out.	Number	Ideally, this measure should have a low value. A high value of this measure indicates that the database is slow or does not have additional connections to allocate to service requests from the web application server.

3.4 The SunONE Transactions Layer

The number of transactions that are committed and rolled back by the application server serves as a good indicator of server workload, processing ability of the server, and potential performance issues (if any). The test associated with the **SunONE Transactions** layer (see Figure 3.11) facilitates this assessment.

Figure 3.11: Test associated with the SunONE Transactions layer

3.4.1 SunOne Transactions Test

This test reports measures pertaining to the transactions in a Sun Java System application server.

Target of the test : Any Sun Java System application server

Agent deploying the test : An internal agent

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

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port number on which the Sun Java System application server is running.
User	A valid user name for the Sun Java System server to be monitored.
Password	Password for the Sun Java System server to be monitored.
Admin Port	The port number on which the “asadmin” tool runs.
Server	Name of the Sun Java System server to be monitored.
AppServerDir	The directory in which Sun Java System application server is installed.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Transactions completed	Returns the total number of transactions that have been completed by this server	Number	
Transaction rollbacks	Returns the total number of transactions that have been rolled back by this server	Number	A high rollback rate indicates a problem with specific beans. Possible reasons for this could be problems with the design and implementation of the specific bean or problems with any of the dependent servers of the bean (e.g., database server).
Transactions in flight	Returns the total number of transactions that are currently in progress	Number	A significantly high value may denote a load on the server. This may indicate that specific transactions are taking too long to process requests.

3.5 The Sun ONE EJB Layer

Sun Java System application server caches stateless, entity and message driven beans. Adequately sized caches are essential for quickly servicing client requests. Using the tests associated with this layer (see Figure 3.12), the caching activity can be closely monitored and deficiencies in size can be promptly reported.

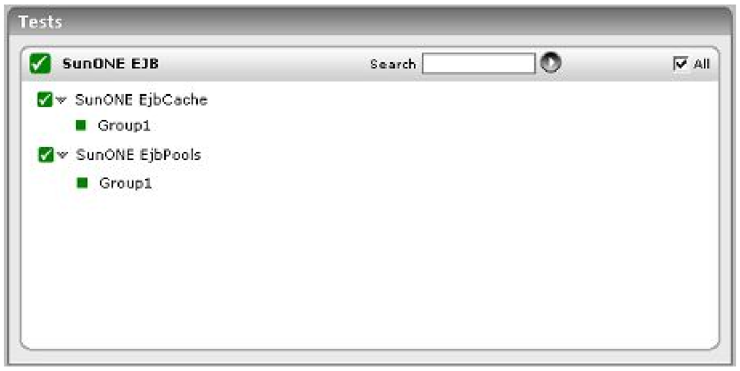


Figure 3.12: Tests associated with the SunONE EJB layer

3.5.1 SunONE EJB Test

This test extracts the caching related metrics pertaining to such EJBs. Use the **Click here** hyperlink in the test configuration page to configure the EJB groups that need to be monitored by the eG Enterprise suite. **By default, the eG Enterprise system will monitor only those EJBs that are part of a group**. The steps for creating the EJBs group has been discussed in the Section **3.5.2**.

Target of the test : Any Sun Java System application server that has one or more stateless, entity, or message driven beans

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every EJB group configured or EJB (as the case may be).

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port number on which the Sun Java System application server is running.
User	A valid user name for the Sun Java System server to be monitored.
Password	Password for the Sun Java System server to be monitored.
Admin Port	The port number on which the “asadmin” tool runs.
Server	Name of the Sun Java System server to be monitored.
AppServerDir	The directory in which Sun Java System application server is installed.
AutoDiscovery	By default, the eG Enterprise suite allows administrators to configure EJB groups using the eG administrative interface, and reports metrics pertaining to every group so created. Accordingly, by default, AutoDiscovery is set to **No**. If you want EJBs to be discovered and monitored automatically, then select the **Yes** option against AutoDiscovery. When this is done, the eG agent automatically discovers all the EJBs on the Sun Java System application server, and reports one set of measures for every EJB hosted on the server.

Measurements made by the test

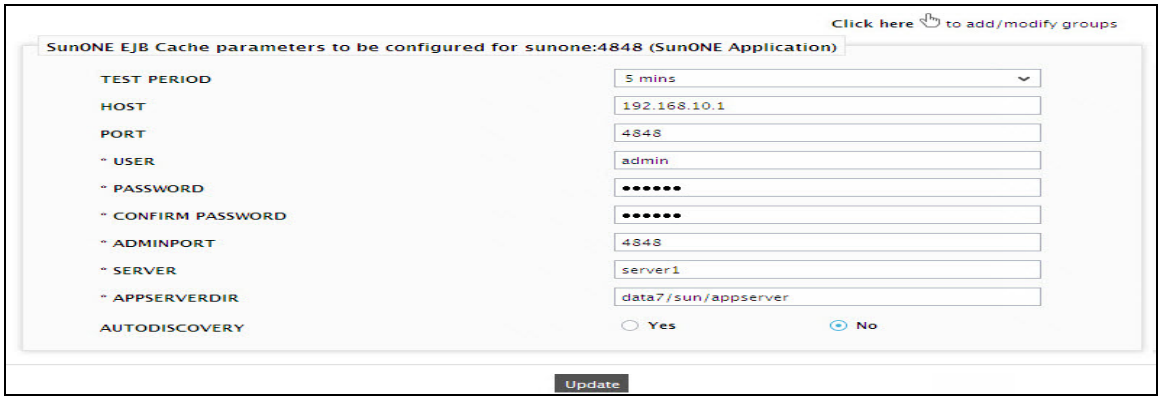
Measurement	Description	Measurement Unit	Interpretation
Resize quantity	The quantity by which the cache size is reduced when the number of beans in the cache equals the maximum cache size (that is, when cache overflow occurs)	Number	This is a configuration parameter for the EJB cache. This measure will not be available for Sun Java Application Server 8.2 and above.
Cache misses	The number of times a user request did not find a bean in the cache during the last measurement period.	Number	Ideally, the percentage of cache misses to cache hits should be a low percentage.
Idle timeouts	Rate at which the cache cleaner thread is scheduled. This cleaner thread examines all beans in the cache and passivates those beans that are not accessed for cache-idle-timeout-in-seconds.	Secs	This is a configuration parameter for the EJB cache. This measure will not be available for Sun Java Application Server 8.2 and above.
Passivations	Number of passivations. Applies only to stateful session beans during the last measurement period.	Number	
Cache hits	The number of times a user request found an entry in the cache during the last measurement period.	Number	
Passivation errors	Number of errors during passivation. Applies only to stateful session beans during the last measurement period.	Number	
Beans in cache	The number of beans in the cache. This is the current	Number	The ratio of beans in cache to the maximum beans in cache is an

Measurement	Description	Measurement Unit	Interpretation
	size of the cache		indicator of the cache occupancy. For maximum performance, the cache occupancy and hit rate must be high.
Expired sessions	The number of expired sessions removed by the cleanup thread during the last measurement period. Applies only to stateful session beans	Number	
Max beans in cache	The maximum number of beans that can be held in the cache beyond which cache overflow occurs	Number	This is a configuration parameter for the EJB cache.
Passivation successes	Number of times passivation completed successfully. Applies only to stateful session beans	Number	

3.5.2 Creating EJB Groups

To create the EJB groups while configuring the SunONE EJB Cache test, do the following;

1. Click on the **Click here** hyperlink provided above the parameters to be configured in the Figure 3.13.

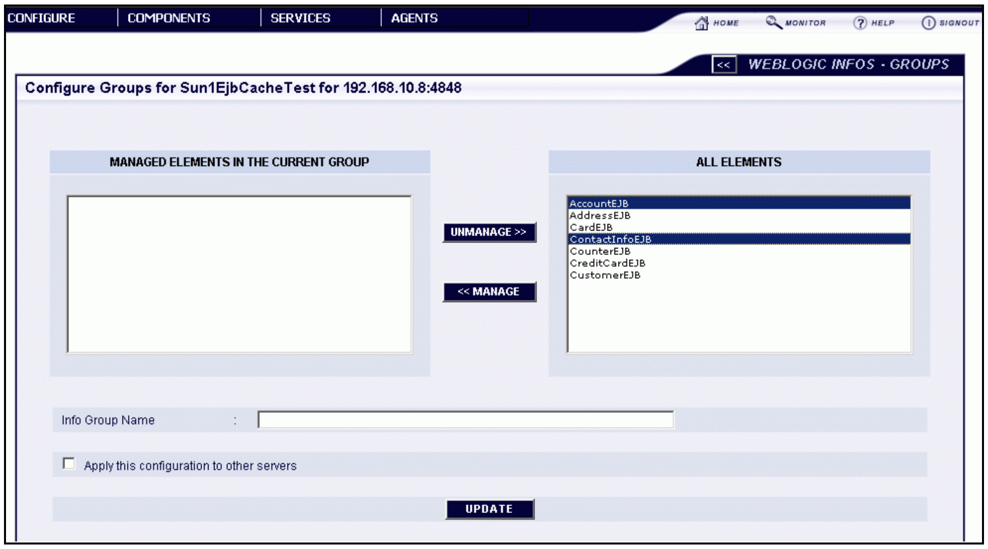

Click here to add/modify groups

SunONE EJB Cache parameters to be configured for sunone:4848 (SunONE Application)

TEST PERIOD	5 mins
HOST	192.168.10.1
PORT	4848
* USER	admin
* PASSWORD	••••••
* CONFIRM PASSWORD	••••••
* ADMINPORT	4848
* SERVER	server1
* APPSERVERDIR	data7/sun/appserver
AUTODISCOVERY	Yes [✓ No]

Figure 3.13: Configuring the Sun Java System EJB Cache test

2. When this is done, the following page will appear (see Figure 3.14).



Configure Groups for Sun1EjbCacheTest for 192.168.10.8:4848

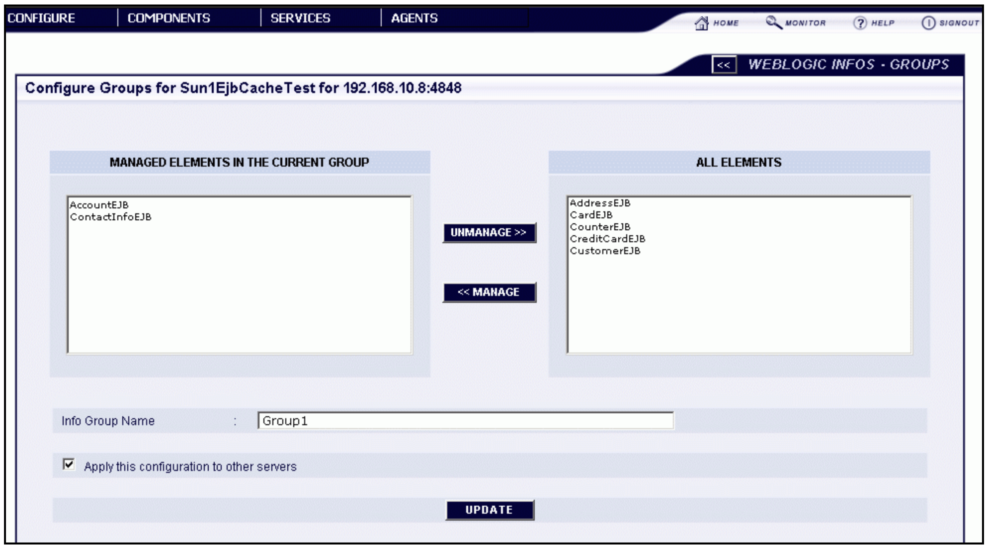
AccountEJB
AddressEJB
CardEJB
ContactInfoEJB
CounterEJB
CreditCardEJB
CustomerEJB

Info Group Name :

Apply this configuration to other servers

Figure 3.14: Selecting the EJBs to be managed

3. In the **ALL ELEMENTS** list of Figure 3.14, the list of EJBs available in the selected Sun Java System application server will be displayed. select the EJBs to be added to the specified group from this list.
4. To manage the selected EJBs, click on the **Manage** button in Figure 3.14. The managed EJBs will now be transferred to the **MANAGED ELEMENTS IN THE CURRENT GROUP** list (see Figure 3.15).



Configure Groups for Sun1EjbCacheTest for 192.168.10.8:4848

AccountEJB
ContactInfoEJB

AddressEJB
CardEJB
CounterEJB
CreditCardEJB
CustomerEJB

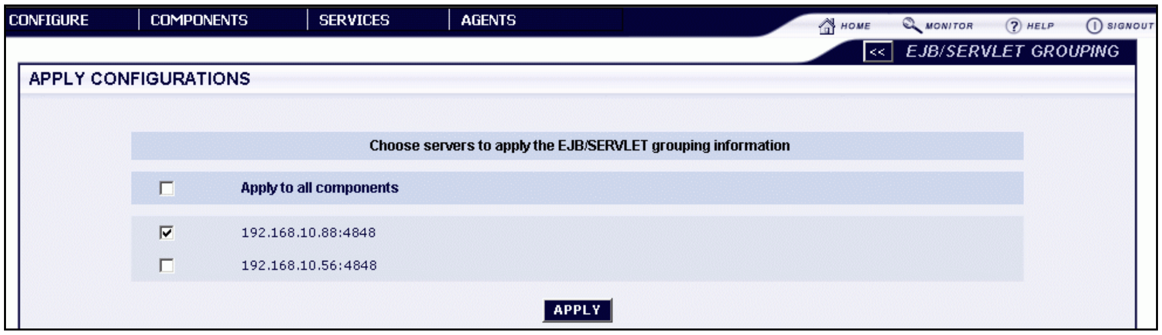
Info Group Name : Group1

☑ Apply this configuration to other servers

Figure 3.15: Managing EJBs

5. Now, specify the name of the EJB group in the **Info Group Name** text box.

6. If the EJB group so created is to be associated with other Sun Java System application servers in the environment, then, select the **Apply this configuration to other servers** check box of Figure 3.15.
7. Finally, click the **Update**button to register the changes.
8. If the **Apply this configuration to other servers** check box is selected, then upon clicking the **UPDATE** button, Figure 3.16 will appear.



Choose servers to apply the EJB/SERVLET grouping information

☐ Apply to all components

☑ 192.168.10.88:4848

☐ 192.168.10.56:4848

Figure 3.16: Associating the EJB group with other Sun Java System servers

9. Upon clicking, the name of the newly created EJB group and the EJBs it contains will be displayed. To modify the EJB group, click on the **MODIFY** button against the group name, and to delete the group, click on the **DELETE**button. If you had earlier associated a group other Sun Java System application servers, then all or a few of these associations can be removed by selecting the **Apply configuration to other servers** check box before clicking on the corresponding **DELETE**button. This will lead you back to Figure 3.16, where you can select the Sun Java System application servers for which the EJB group configuration has to be removed. Clicking on the **APPLY**button after selecting the servers will ensure that the corresponding EJB group configuration does not apply to the chosen servers any longer.

3.5.3 SunOne EJB Pools Test

This test reports statistics pertaining to the EJB pools for stateful session beans and entity beans. Use the **Click here** hyperlink in the test configuration page to configure the EJB groups that need to be monitored by the eG Enterprise suite. **By default, the eG Enterprise system will monitor only those EJBs that are part of a group**.

Target of the test : Any Sun Java System application server with one or more entity beans or stateful session beans

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every EJB group configured or every EJB (as thee cae be).

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port number on which the Sun Java System application server is running.
User	A valid user name for the Sun Java System server to be monitored.
Password	Password for the Sun Java System server to be monitored.
Admin Port	The port number on which the “asadmin” tool runs.
Server	Name of the Sun Java System server to be monitored.
AppServerDir	The directory in which Sun Java System application server is installed.
AutoDiscovery	By default, the eG Enterprise suite allows administrators to configure EJB groups using the eG administrative interface, and reports metrics pertaining to every group so created. Accordingly, by default, AutoDiscovery is set to **No**. If you want EJBs to be discovered and monitored automatically, then select the **Yes** option against AutoDiscovery. When this is done, the eG agent automatically discovers all the EJBs on the Sun Java System application server, and reports one set of measures for every EJB hosted on the server.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Idle timeout	Returns the time out period set for the bean.	Sec	This is a configuration setting. This measure will not be available for Sun Java Application Server 8.2 and above.
Steady pool size	Indicates the average number of beans that a pool contains during the last measurement period.	Number	This measure will not be available for Sun Java Application Server 8.2 and above.
Beans destroyed	The total number of beans in a pool that have been	Number	

Measurement	Description	Measurement Unit	Interpretation
	destroyed by the pool during the last measurement period.		
Waiting threads	Returns the number of threads that are waiting to get an instance of the bean	Number	The value of this measure should be low for optimal performance.
Beans in pool	The number of beans in the bean pool currently	Number	
Max pool size	The maximum number of beans that the pool can contain	Number	This is a configuration setting.
Pool resize quantity	This is the value by which the pool size can be increased or decreased, as per the requirements.	Number	This is a configuration setting. This measure will not be available for Sun Java Application Server 8.2 and above.
Beans created	Indicates the number of beans that have been created for a pool, during the last measurement period	Number	

About eG Innovations

eG Innovations provides intelligent performance management solutions that automate and dramatically accelerate the discovery, diagnosis, and resolution of IT performance issues in on-premises, cloud and hybrid environments. Where traditional monitoring tools often fail to provide insight into the performance drivers of business services and user experience, eG Innovations provides total performance visibility across every layer and every tier of the IT infrastructure that supports the business service chain. From desktops to applications, from servers to network and storage, from virtualization to cloud, eG Innovations helps companies proactively discover, instantly diagnose, and rapidly resolve even the most challenging performance and user experience issues.

eG Innovations is dedicated to helping businesses across the globe transform IT service delivery into a competitive advantage and a center for productivity, growth and profit. Many of the world's largest businesses use eG Enterprise to enhance IT service performance, increase operational efficiency, ensure IT effectiveness and deliver on the ROI promise of transformational IT investments across physical, virtual and cloud environments.

To learn more visit www.eginnovations.com.

Contact Us

For support queries, email support@eginnovations.com.

To contact eG Innovations sales team, email sales@eginnovations.com.