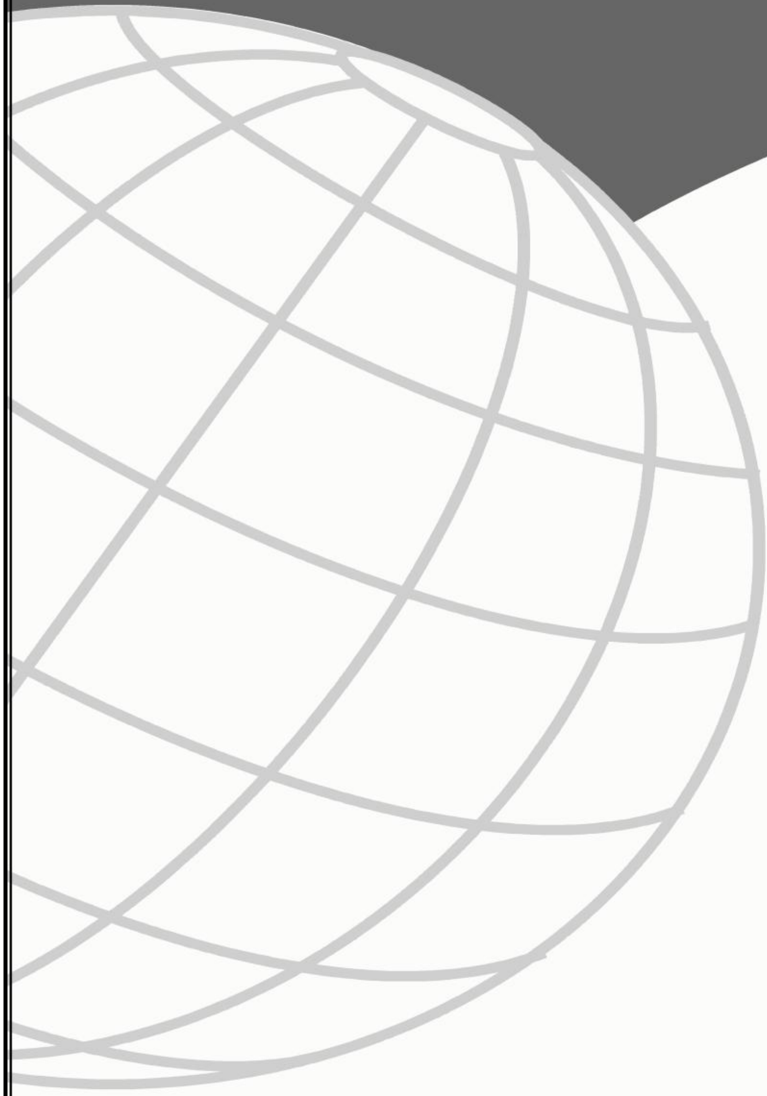


Enabling Service Excellence

Monitoring the Sun Java Web Server

eG Enterprise v6

Restricted Rights Legend

The information contained in this document is confidential and subject to change without notice. No part of this document may be reproduced or disclosed to others without the prior permission of eG Innovations Inc. eG Innovations Inc. makes no warranty of any kind with regard to the software and documentation, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Trademarks

Microsoft Windows, Windows NT, Windows 2000, Windows 2003 and Windows 2008 are either registered trademarks or trademarks of Microsoft Corporation in United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Copyright

Table of Contents

Table of Figures

Chapter

8

Monitoring the Sun Java Web Server

Sun Java System Web Server 7.0 is a multi-process, multi-threaded, secure web server built on industry standards for medium to large enterprises.

The web server incorporates a modular architecture that integrates seamlessly with all products in the Sun Java System family of servers. In addition, Web Server supports a variety of APIs and programming technologies that enable you to do the following:

- Generate dynamic content in response to client requests
- Modify and extend the behavior of the server
- Modify the content that is stored in the server

It is this flexibility that makes the Sun Java web server one of the most popular web servers when it comes to supporting dynamic, mission-critical web site services. Since any issue in the functioning of this web server can cause prolonged service slowdowns or can even deny users access to the service, it is imperative that the wheels of this server roll 24 x 7 so as to ensure high service uptime and to maximize the user satistaction with the quality of the service. The need of the hour is hence a monitoring solution that can monitor the web server continuously, detect performance issues before it affects service quality, and alert administrators to these issues, so that the problem can be resolved much before users notice them.

eG Enterprise provides a 100%, web-based *Sun Java Web Server* monitoring model that scans the web server bottom-up – from the operating system to the virtual servers to the server instances to the web applications deployed on the server - so that potential server overloads/slowdowns, errors encountered in real-time by the server, etc., can be quickly captured and proactively reported to administrators.

Each layer of Figure 1.1 is mapped to tests that report a wide variety of metrics related to the web server. These metrics provide accurate answers to the following performance queries:

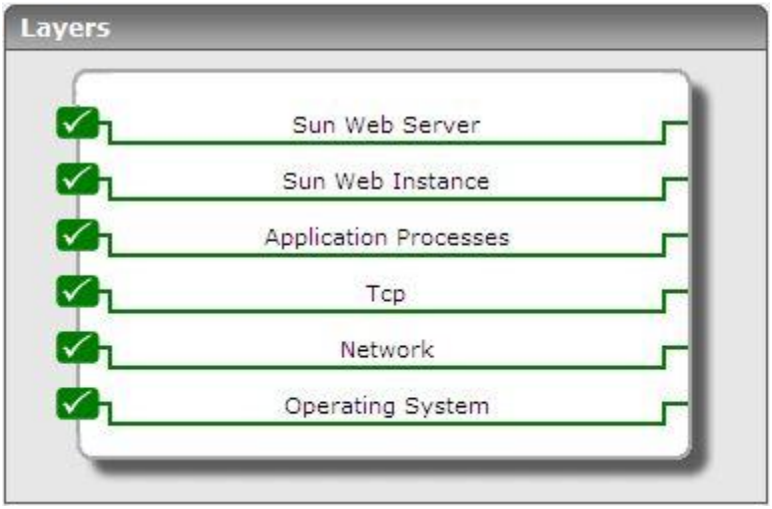


Figure 1.1: The layer model of the Sun Java Web Server

To enable the eG agent to extract these useful metrics from the web server, the pre-requisites documented in Section 1.1 below should be fulfilled.

1.1 Pre-requisites for Monitoring the Sun Java Web Server

eG Enterprise provides agent-based monitoring support for Sun Java Web Server 7 on Windows and Solaris platforms. To enable the eG agent to collect statistics of interest from the Sun Java web server, you need to ensure the following:

- The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server, in the **bin** directory within the install directory of the Sun Java web server. Then, you need to configure the eG tests with the name of that file;
- If this administration server is SSL-enabled, then you need to trust the SSL certificate of the server to enable the eG agent to communicate with that server;
- While monitoring the Sun Java Web server on Windows, make sure that the eGurkhaAgent service on the server runs using the privileges of the user who trusted the SSL certificate of the administration server;
- Configure the Sun Java web server to allow metrics collection.

The steps to fulfill each of these pre-requisites have beend detailed in the sections that will follow.

1.1.1 Creating a Password File

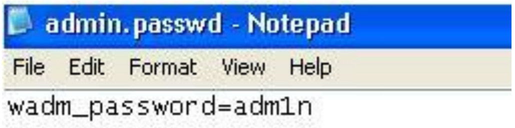
This section explains how to create a password file on a web server operating on a Windows and a Solaris platform.

1.1.1.1 Creating a Password File on Windows

A password file named *admin.passwd* has to be created in the **bin** directory of the Sun Java Web Server's install directory. Specify the following in that file:

wadm_password=<Password_of_admin_server's_administrator>

Figure 1.2 depicts how this is done:



```
wadm_password=adm1n
```

Figure 1.2: The password file containing the administrator's password

1.1.1.2 Creating a Password File on Solaris

Issue the following commands from the **bin** directory of the Sun Java web server's install directory to create the password file named *admin.passwd* with the admin server's administrator password:

vi admin.passwd

"admin.passwd" [New file]

wadm_password=adm1n

1.1.2 Trusting the SSL Certificate of the Administration Server

Follow the instructions provided in this section to trust the SSL Certificate of the administration server to which the target Sun Java web server is registered.

1.1.2.1 Trusting the SSL Certificate of the Administration Server of a Sun Java Web Server on Windows

To achieve the above, follow the steps discussed hereunder:

1. Make sure that the administration server to which the target Sun Java web server is registered is running.
2. Then, login to the target Sun Java web server and go to its command prompt.
3. At the prompt, issue the following command:

   wadm list-virtual-servers --user=<userName> --password-file=admin.passwd --host=<serverHostName> --port=<serverAdminPort> --config=<configName>

   Here,

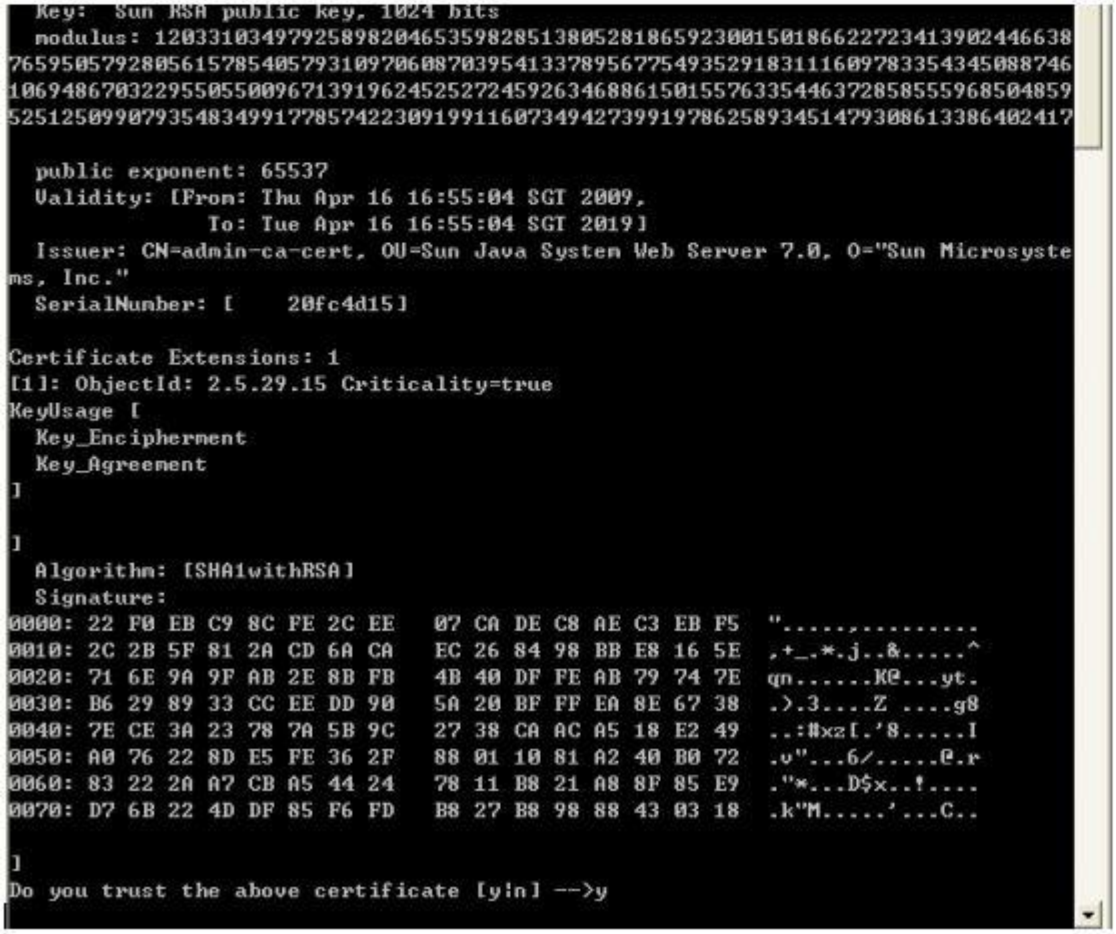
   userName is the name of the administrator to the administration server

   serverHostName is the IP address of the administration server

   serverAdminPort is the port of the administration server; the default is 8989

configName is the name of any configuration in the administration server

4. Executing the aforesaid command displays the SSL certificate of the administration server; at the end of the display, you will be prompted to trust the certificate (see Figure 8.2).



```
  Key:  Sun RSA public key, 1024 bits
  modulus: 120331034979258982046535982851380528186592300150186622723413902446638
76595057928056157854057931097060870395413378956775493529183111609783354345088746
10694867032295505500967139196245252724592634688615015576335446372858555968504859
52512509907935483499177857422309199116073494273991978625893451479308613386402417

  public exponent: 65537
  Validity: [From: Thu Apr 16 16:55:04 SGT 2009,
               To: Tue Apr 16 16:55:04 SGT 2019]
  Issuer: CN=admin-ca-cert, OU=Sun Java System Web Server 7.0, O="Sun Microsyste
ms, Inc."
  SerialNumber: [    20fc4d15]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  Key_Encipherment
  Key_Agreement
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 22 F0 EB C9 8C FE 2C EE   07 CA DE C8 AE C3 EB F5  "...............
0010: 2C 2B 5F 81 2A CD 6A CA   EC 26 84 98 BB E8 16 5E  ,+_.*.j..&.....^
0020: 71 6E 9A 9F AB 2E 8B FB   4B 40 DF FE AB 79 74 7E  qn......K@...yt.
0030: B6 29 89 33 CC EE DD 90   5A 20 BF FF EA 8E 67 38  .).3....Z ....g8
0040: 7E CE 3A 23 78 7A 5B 9C   27 38 CA AC A5 18 E2 49  ..:#xz[.'8.....I
0050: A0 76 22 8D E5 FE 36 2F   88 01 10 81 A2 40 B0 72  .v"...6/.....@.r
0060: 83 22 2A A7 CB A5 44 24   78 11 B8 21 A8 8F 85 E9  ."*...D$x..!....
0070: D7 6B 22 4D DF 85 F6 FD   B8 27 B8 98 88 43 03 18  .k"M.....'...C..

]
Do you trust the above certificate [y|n] -->y
```

Figure 1 3: Trusting the SSL certificate of the administration server of a Sun Java web server on Windows

5. Type **y** at the prompt to trust the certificate.

1.1.2.2 Trusting the SSL Certificate of the Administration Server of a Sun Java Web Server on Solaris

In this case, follow the steps given below:

1. Login to the Sun Java web server as the eG install user.
2. Ensure that the administration server to which the target Sun Java web server is registered is running.
3. Then, at the prompt, switch to the **bin** directory of the Sun Java web server's install directory, and then issue the following command:

./wadm list-virtual-servers --user=<userName> --password-file=admin.passwd --host=<serverHostName> --port=<serverAdminPort> --config=<configName>

Here,

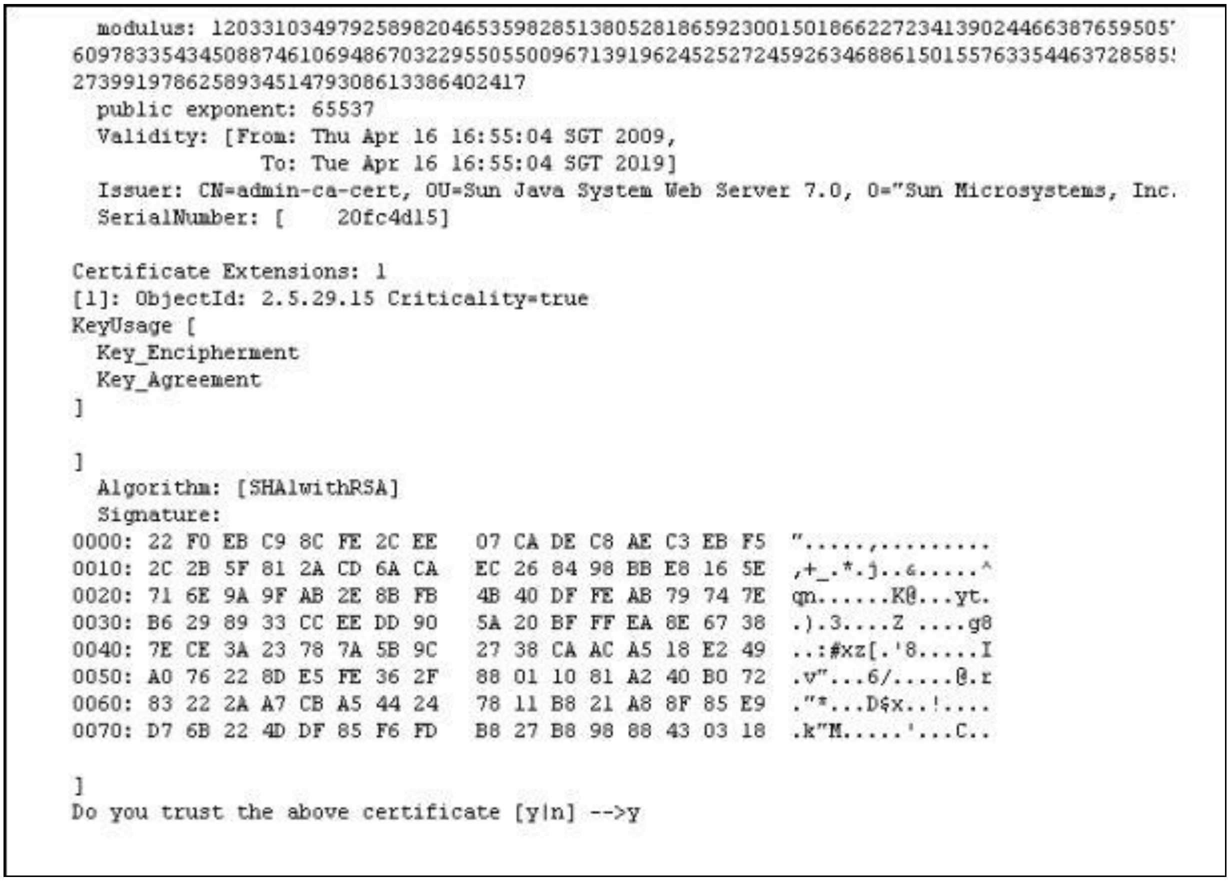
userName is the name of the administrator to the administration server

serverHostName is the IP address of the administration server

serverAdminPort is the port of the administration server; the default is 8989

configName is the name of any configuration in the administration server

4. Executing the aforesaid command displays the SSL certificate of the administration server; at the end of the display, you will be prompted to trust the certificate (see Figure 8.2).



```
  modulus: 1203310349792589820465359828513805281865923001501866227234139024466387659505'
609783354345088746106948670322955055009671391962452527245926346886150155763354463728585!
273991978625893451479308613386402417
  public exponent: 65537
  Validity: [From: Thu Apr 16 16:55:04 SGT 2009,
               To: Tue Apr 16 16:55:04 SGT 2019]
  Issuer: CN=admin-ca-cert, OU=Sun Java System Web Server 7.0, O="Sun Microsystems, Inc.
  SerialNumber: [    20fc4d15]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  Key_Encipherment
  Key_Agreement
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 22 F0 EB C9 8C FE 2C EE   07 CA DE C8 AE C3 EB F5  ".....,.........
0010: 2C 2B 5F 81 2A CD 6A CA   EC 26 84 98 BB E8 16 5E  ,+_.*.j..&.....^
0020: 71 6E 9A 9F AB 2E 8B FB   4B 40 DF FE AB 79 74 7E  qn......K@...yt.
0030: B6 29 89 33 CC EE DD 90   5A 20 BF FF EA 8E 67 38  .).3....Z ....g8
0040: 7E CE 3A 23 78 7A 5B 9C   27 38 CA AC A5 18 E2 49  ..:#xz[.'8.....I
0050: A0 76 22 8D E5 FE 36 2F   88 01 10 81 A2 40 B0 72  .v"...6/.....@.r
0060: 83 22 2A A7 CB A5 44 24   78 11 B8 21 A8 8F 85 E9  ."*...D$x..!....
0070: D7 6B 22 4D DF 85 F6 FD   B8 27 B8 98 88 43 03 18  .k"M.....'...C..

]
Do you trust the above certificate [y|n] -->y
```

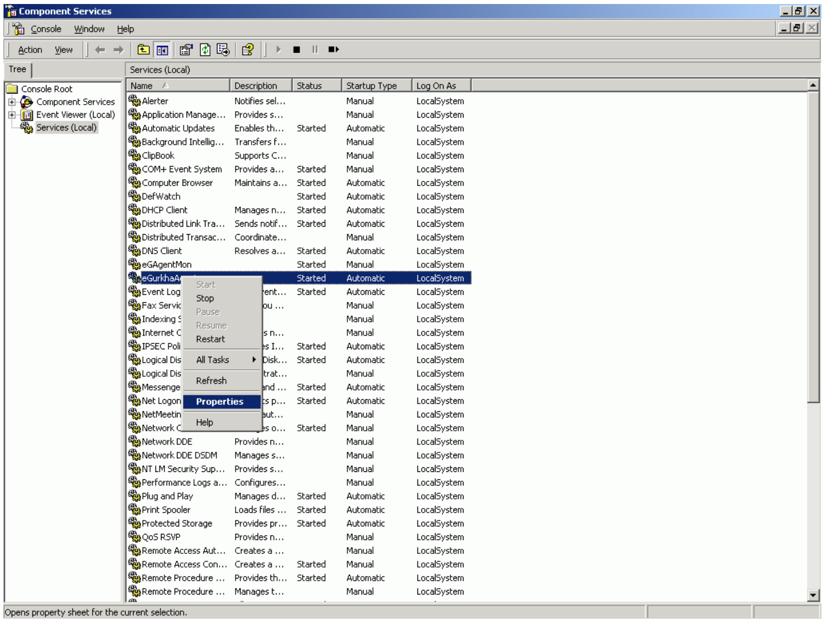
Figure 1.4: Trusting the SSL certificate of the administration server of a Sun Java web server on Solaris

5. Type **y** at the prompt to trust the certificate.

1.1.3 Changing the LogOn Privileges of the eGurkhaAgent Service of the eG Agent that Monitors the Sun Java Web Server on Windows

After trusting the SSL Certificate of the administration server to which a Sun Java web server on Windows is registered, you need to configure the eGurkhaAgent service of the eG agent on that web server to run using the privileges of the user who trusted the certificate. For this purpose, do the following:

1. Login to the target Sun Java web server.
2. Open the **Component Services** window to view the **eGurkhaAgent** service. Right-click on the service, and pick **Properties** from the shortcut menu that appears.



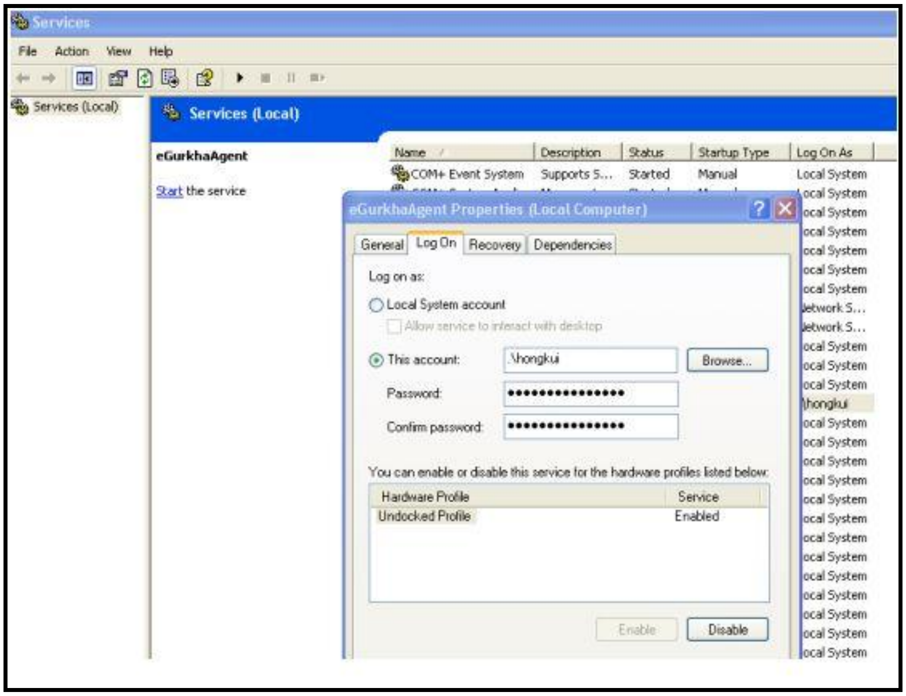
Services (Local)

Name	Description	Status	Startup Type	Log On As
Alerter	Notifies sel...		Manual	LocalSystem
Application Manage...	Provides s...		Manual	LocalSystem
Automatic Updates	Enables th...	Started	Automatic	LocalSystem
Background Intellig...	Transfers f...		Manual	LocalSystem
ClipBook	Supports C...		Manual	LocalSystem
COM+ Event System	Provides a...	Started	Manual	LocalSystem
Computer Browser	Maintains a...	Started	Automatic	LocalSystem
DefWatch		Started	Automatic	LocalSystem
DHCP Client	Manages n...	Started	Automatic	LocalSystem
Distributed Link Tra...	Sends notif...	Started	Automatic	LocalSystem
Distributed Transac...	Coordinate...		Manual	LocalSystem
DNS Client	Resolves a...	Started	Automatic	LocalSystem
eGAgentMon		Started	Manual	LocalSystem
eGurkhaA		Started	Automatic	LocalSystem
Event Log	ent...	Started	Automatic	LocalSystem
Fax Servic	ou ...		Manual	LocalSystem
Indexing S			Manual	LocalSystem
Internet C	s n...		Manual	LocalSystem
IPSEC Poli	s I...	Started	Automatic	LocalSystem
Logical Dis	Disk...	Started	Automatic	LocalSystem
Logical Dis	trat...		Manual	LocalSystem
Messenge	nd ...	Started	Automatic	LocalSystem
Net Logon	s p...	Started	Automatic	LocalSystem
NetMeetin	ut...		Manual	LocalSystem
Network C	s o...	Started	Manual	LocalSystem
Network DDE	Provides n...		Manual	LocalSystem
Network DDE DSDM	Manages s...		Manual	LocalSystem
NT LM Security Sup...	Provides s...		Manual	LocalSystem
Performance Logs a...	Configures...		Manual	LocalSystem
Plug and Play	Manages d...	Started	Automatic	LocalSystem
Print Spooler	Loads files ...	Started	Automatic	LocalSystem
Protected Storage	Provides pr...	Started	Automatic	LocalSystem
QoS RSVP	Provides n...		Manual	LocalSystem
Remote Access Aut...	Creates a ...		Manual	LocalSystem
Remote Access Con...	Creates a ...	Started	Manual	LocalSystem
Remote Procedure ...	Provides th...	Started	Automatic	LocalSystem
Remote Procedure ...	Manages t...		Manual	LocalSystem

Opens property sheet for the current selection.

Figure 1.5: Selecting the Properties option of the eGurkhaAgent service

3. Select the **LogOn** tab from the **Properties** dialog box that appears (see Figure 1.6). Then, choose the **This account** option from the **Log on as** section of Figure 1.7, and provide the *Domainname\Username* of the user using whose privileges the eGurkhaAgent service should run. Provide the **Password** of the domain administrator, and confirm the password by retyping it in the **Confirm Password** text box.



Name	Description	Status	Startup Type	Log On As
COM+ Event System	Supports S...	Started	Manual	Local System

eGurkhaAgent Properties (Local Computer)

Log on as:

○ Local System account

□ Allow service to interact with desktop

● This account: .\hongkui

Password: ••••••••••••••••

Confirm password: ••••••••••••••••

You can enable or disable this service for the hardware profiles listed below:

Hardware Profile	Service
Undocked Profile	Enabled

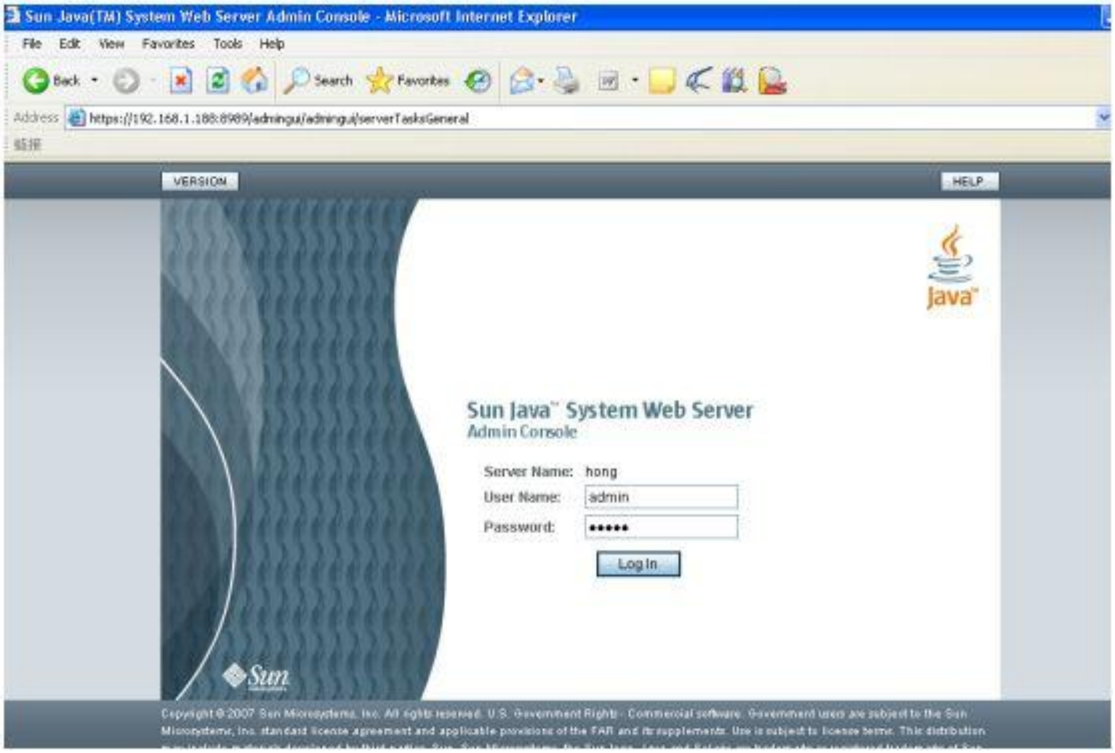
Figure 1.6: Specifying the credentials of the user using whose privileges the eGurkhaAgent service should run

4. Finally, click the **Apply** button in Figure 1.6, and then the **OK** button to register the changes.

1.1.4 Enabling Metrics Collection on the Sun Java Web Server

To achieve this, follow the steps given below:

1. Login to the administration console by first accessing the URL: **http://<IP/hostname of the administration server>:<port of administration server>**, and providing a valid user ID and password (see Figure 1.7).


Sun Java(TM) System Web Server Admin Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
VERSION
HELP
Java
Sun Java™ System Web Server
Admin Console
Log In
Sun

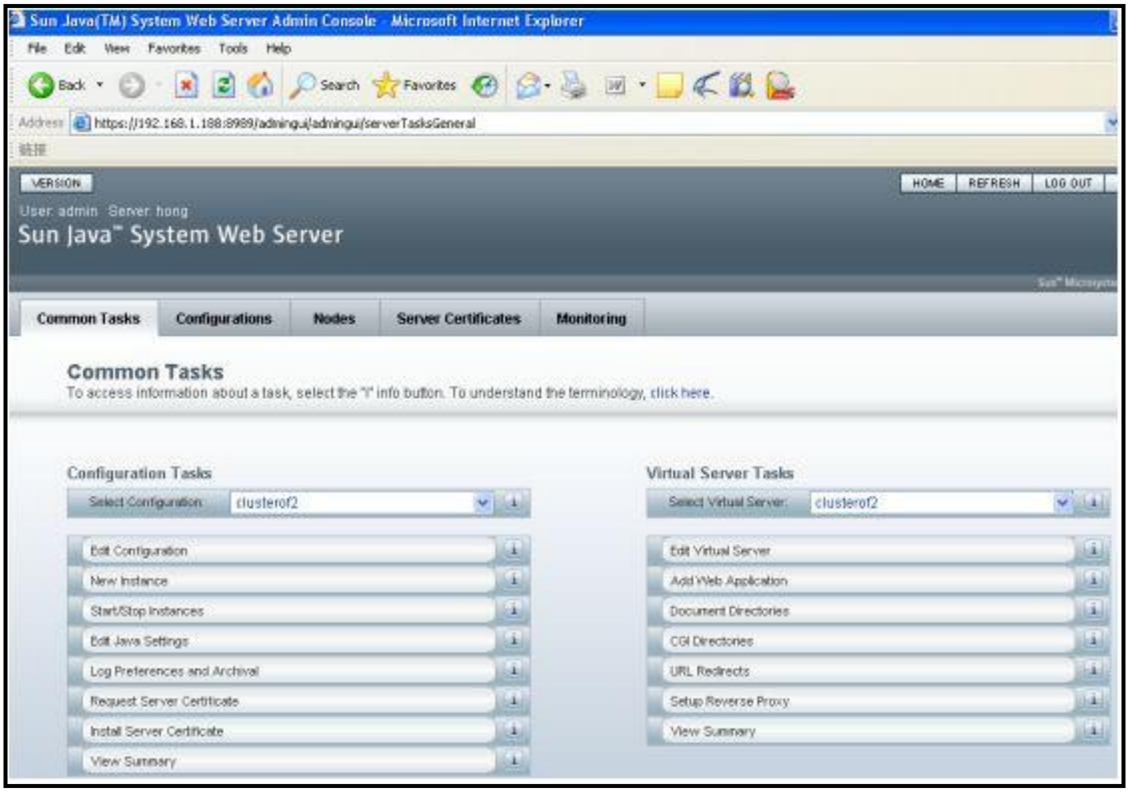
https://192.168.1.188:8989/admingui/admingui/serverTasksGeneral

Server Name: hong
User Name: admin
Password: •••••

Copyright © 2007 Sun Microsystems, Inc. All rights reserved. U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements. Use is subject to license terms. This distribution

Figure 1.7: Logging into the administration server

2. Figure 1.8 then appears. Click on the **Configurations** tab page in Figure 1.8. Figure 1.9 then appears displaying all the server configurations on the administration server. A Configuration in Web Server is an entire server configuration. It can have many virtual servers and many HTTP listeners: a declaration of which IP addresses and ports to use. In Figure 1.9, click on the configuration which the web server instance you want to monitor uses.



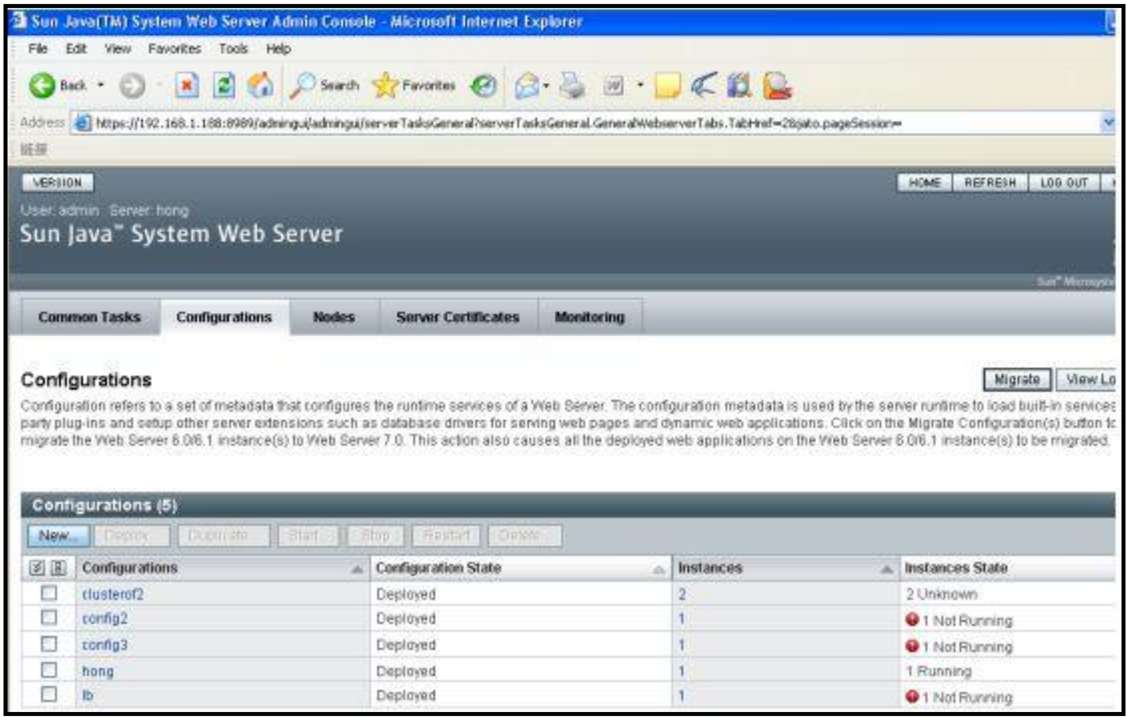
Address: https://192.168.1.188:8989/admingui/admingui/serverTasksGeneral

To access information about a task, select the "i" info button. To understand the terminology, click here.

Select Configuration: clusterof2

Select Virtual Server: clusterof2

Figure 1.8: The administration server console



Address: https://192.168.1.188:8989/admingui/admingui/serverTasksGeneral?serverTasksGeneral.GeneralWebserverTabs.TabHref=2&jato.pageSession=

Configurations

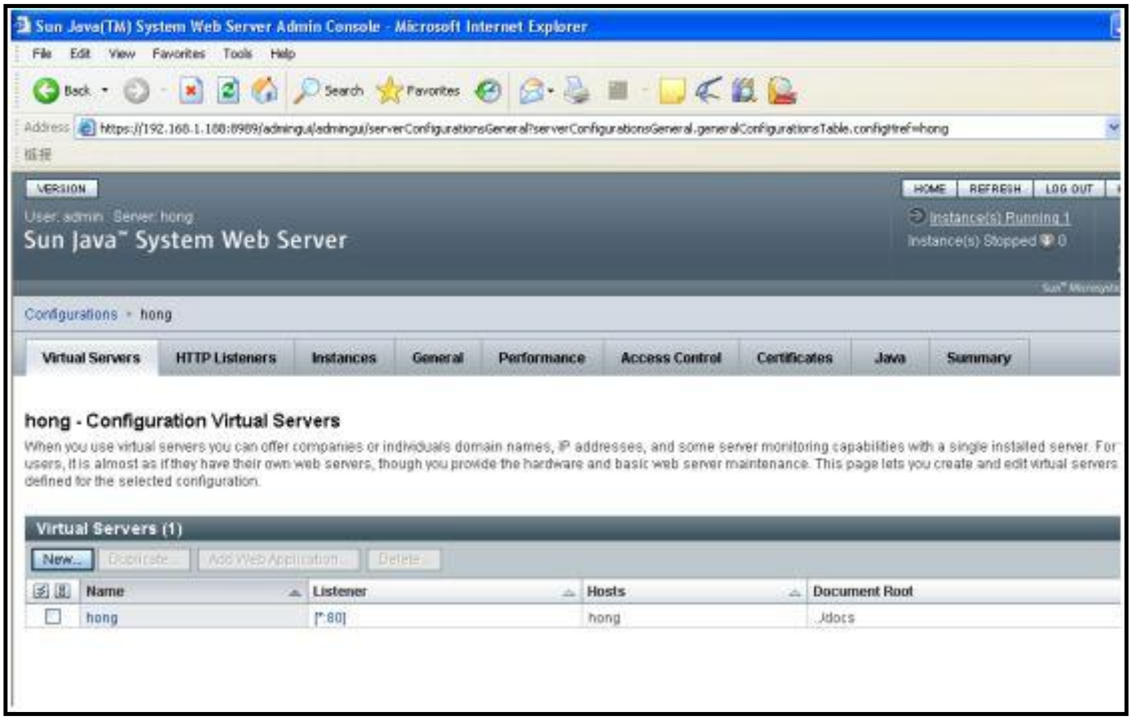
Configuration refers to a set of metadata that configures the runtime services of a Web Server. The configuration metadata is used by the server runtime to load built-in services party plug-ins and setup other server extensions such as database drivers for serving web pages and dynamic web applications. Click on the Migrate Configuration(s) button to migrate the Web Server 6.0/6.1 instance(s) to Web Server 7.0. This action also causes all the deployed web applications on the Web Server 6.0/6.1 instance(s) to be migrated.

Configurations (5)

	Configurations	Configuration State	Instances	Instances State
□	clusterof2	Deployed	2	2 Unknown
□	config2	Deployed	1	● 1 Not Running
□	config3	Deployed	1	● 1 Not Running
□	hong	Deployed	1	1 Running
□	lb	Deployed	1	● 1 Not Running

Figure 1.9: The Configurations tab displaying all the configurations on the administration server

3. Doing so brings up Figure 8. that displays the details of the configuration that is clicked on.


Sun Java(TM) System Web Server Admin Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address https://192.168.1.188:8989/admingui/admingui/serverConfigurationsGeneral?serverConfigurationsGeneral.generalConfigurationsTable.configHref=hong
VERSION
HOME REFRESH LOG OUT
User: admin Server: hong
Sun Java™ System Web Server
Instance(s) Running 1
Instance(s) Stopped 0
Configurations › hong
Virtual Servers
HTTP Listeners
Instances
General
Performance
Access Control
Certificates
Java
Summary
New...
Duplicate...
Add Web Application...
Delete...

hong - Configuration Virtual Servers

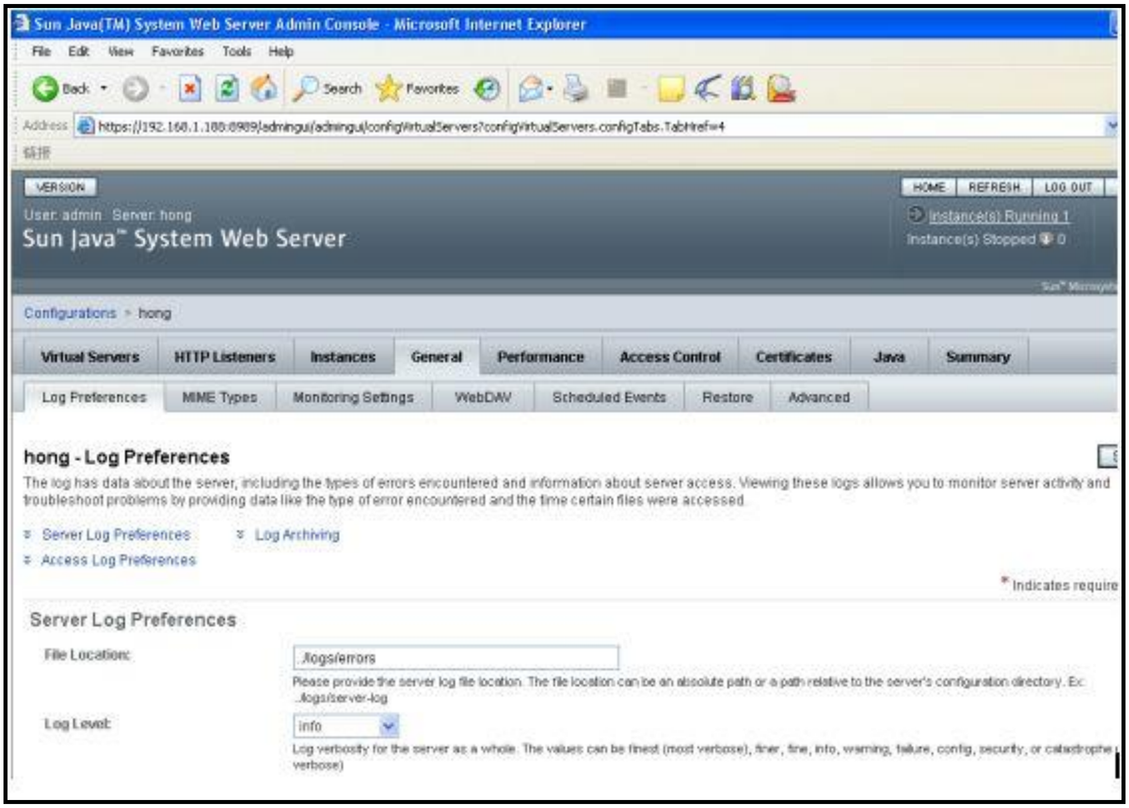
When you use virtual servers you can offer companies or individuals domain names, IP addresses, and some server monitoring capabilities with a single installed server. For users, it is almost as if they have their own web servers, though you provide the hardware and basic web server maintenance. This page lets you create and edit virtual servers defined for the selected configuration.

Virtual Servers (1)

	Name	Listener	Hosts	Document Root
□	hong	[*:80]	hong	./docs

Figure 1.10: The details of the configuration that corresponds to the web server instance to be monitored

4. Click on the **General** tab page in Figure 1.10. Figure 1.11 will then appear. Click on the **Monitor Settings** tab page within the **General** tab page.



hong - Log Preferences

The log has data about the server, including the types of errors encountered and information about server access. Viewing these logs allows you to monitor server activity and troubleshoot problems by providing data like the type of error encountered and the time certain files were accessed.

- Server Log Preferences
- Log Archiving
- Access Log Preferences

* Indicates require

Server Log Preferences

File Location: ../logs/errors

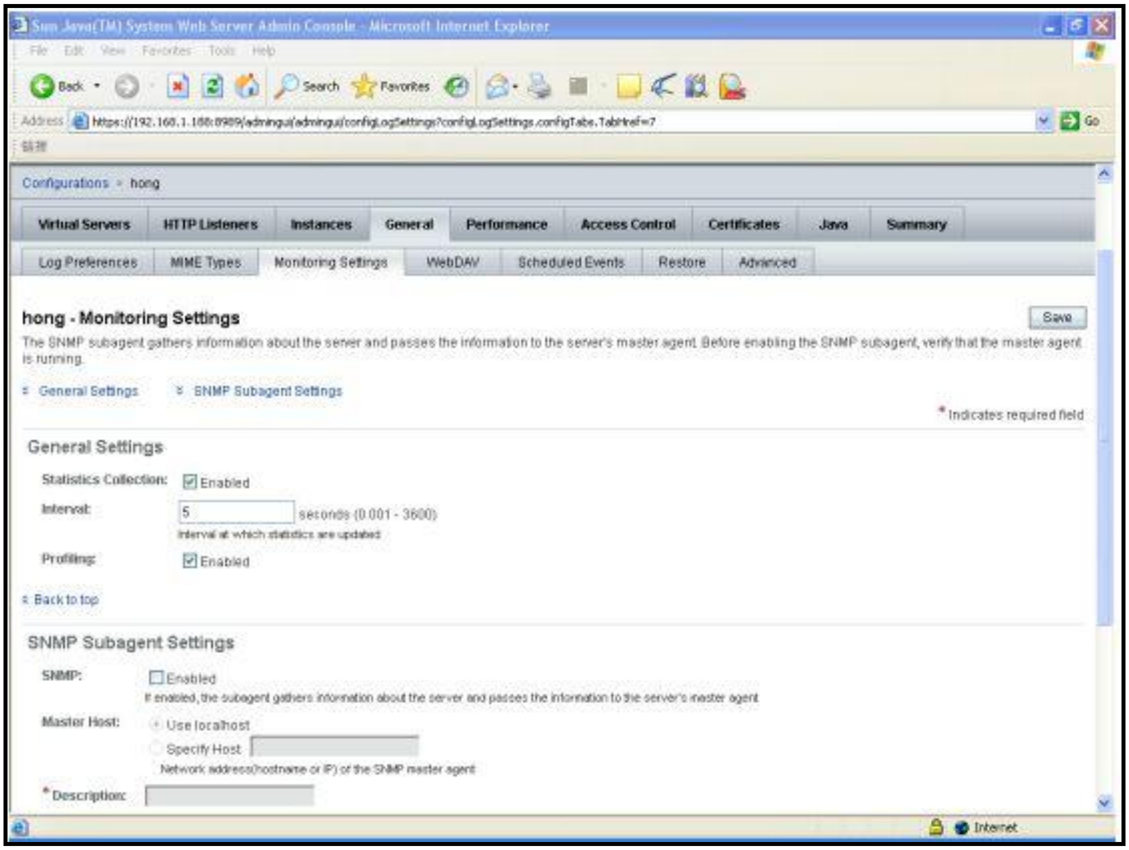
Please provide the server log file location. The file location can be an absolute path or a path relative to the server's configuration directory. Ex: ../logs/server-log

Log Level: info

Log verbosity for the server as a whole. The values can be finest (most verbose), finer, fine, info, warning, failure, config, security, or catastrophe (least verbose)

Figure 1.11: The General tab page

5. This will invoke Figure 1.12. Select the **Enabled** check box against **Statistics collection** to enable metrics collection. Similarly, enable profiling by clicking on the check box against **Profiling.** Then, click the **Save** button in Figure 1.12.



hong - Monitoring Settings

The SNMP subagent gathers information about the server and passes the information to the server's master agent. Before enabling the SNMP subagent, verify that the master agent is running.

* Indicates required field

General Settings

Statistics Collection: ☑ Enabled

Interval: 5 seconds (0.001 - 3600)
Interval at which statistics are updated

Profiling: ☑ Enabled

SNMP Subagent Settings

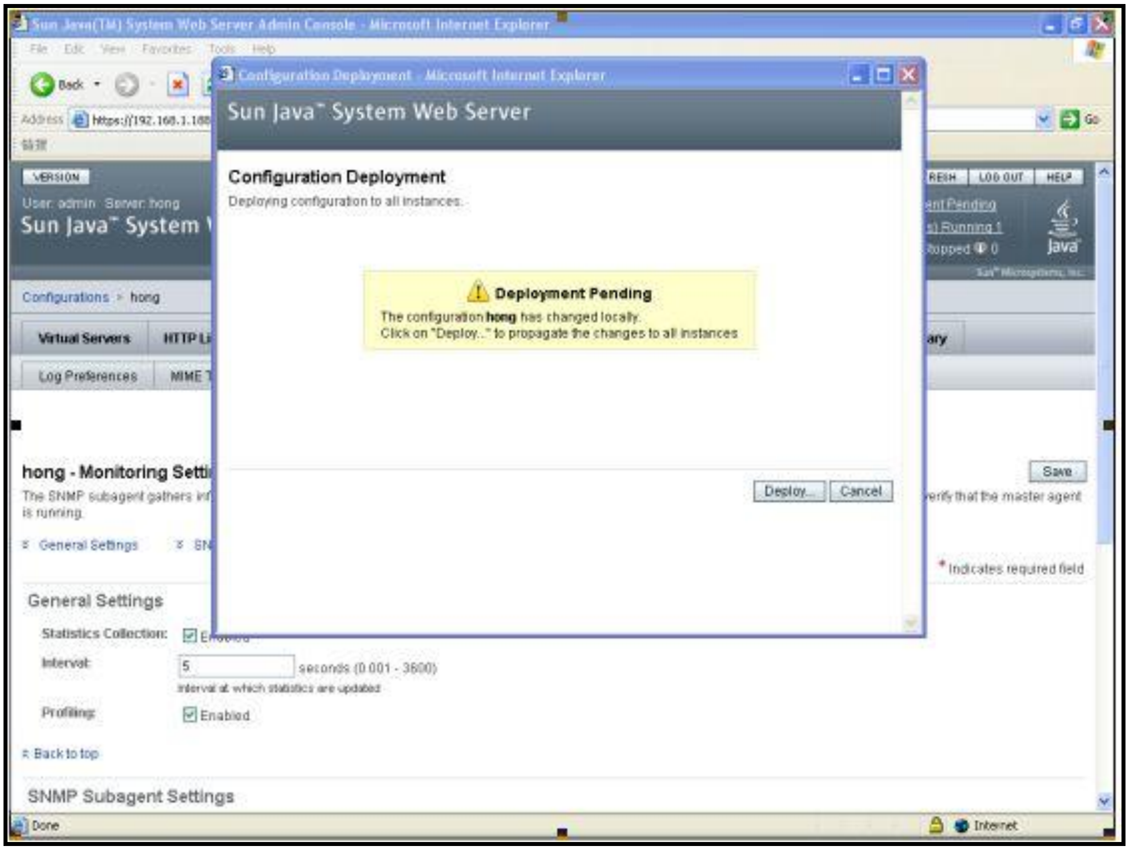
SNMP: □ Enabled
If enabled, the subagent gathers information about the server and passes the information to the server's master agent

Master Host: ● Use localhost
○ Specify Host
Network address(hostname or IP) of the SNMP master agent

* Description:

Figure 1.12: The Monitoring Settings tab page

6. Once the settings are saved, a **Deployment Pending** alert appears as depicted by Figure 1.13. Click on the **Deploy** button therein to deploy the new configuration setting.



Configuration Deployment

Deploying configuration to all instances.

Deployment Pending

The configuration **hong** has changed locally.
Click on "Deploy..." to propagate the changes to all instances

hong - Monitoring Setti…

The SNMP subagent gathers inf… verify that the master agent is running.

General Settings

Statistics Collection: ☑ Enabled

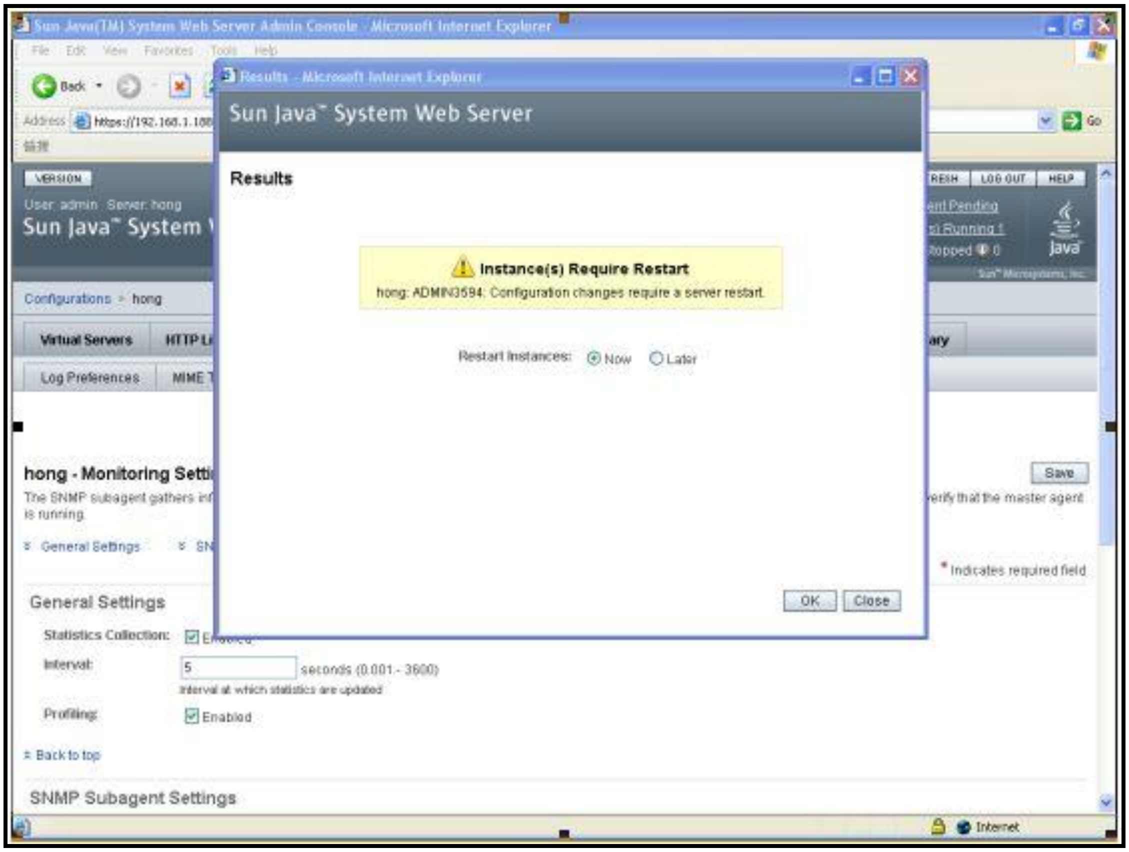
Interval: 5 seconds (0.001 - 3600)
Interval at which statistics are updated

Profiling: ☑ Enabled

SNMP Subagent Settings

Figure 1.13: The Deployment Pending alert

7. This will open Figure 1.14, which will prompt you to restart the web server instance that is affected by the configuration change. With the **Now** option selected click on the **OK** button to restart the web server instance immediately.


Sun Java(TM) System Web Server Admin Console - Microsoft Internet Explorer
Results - Microsoft Internet Explorer
Sun Java™ System Web Server
VERSION
User: admin Server: hong
Configurations > hong
Virtual Servers
Log Preferences
REFRESH
LOG OUT
HELP
Save
OK
Close
Internet

Results

⚠ Instance(s) Require Restart

hong: ADMIN3594: Configuration changes require a server restart.

Restart Instances: ● Now ○ Later

hong - Monitoring Setti

The SNMP subagent gathers inf ... verify that the master agent is running.

* Indicates required field

General Settings

Statistics Collection: ☑ Enabled

Interval: 5 seconds (0.001 - 3600)

Interval at which statistics are updated

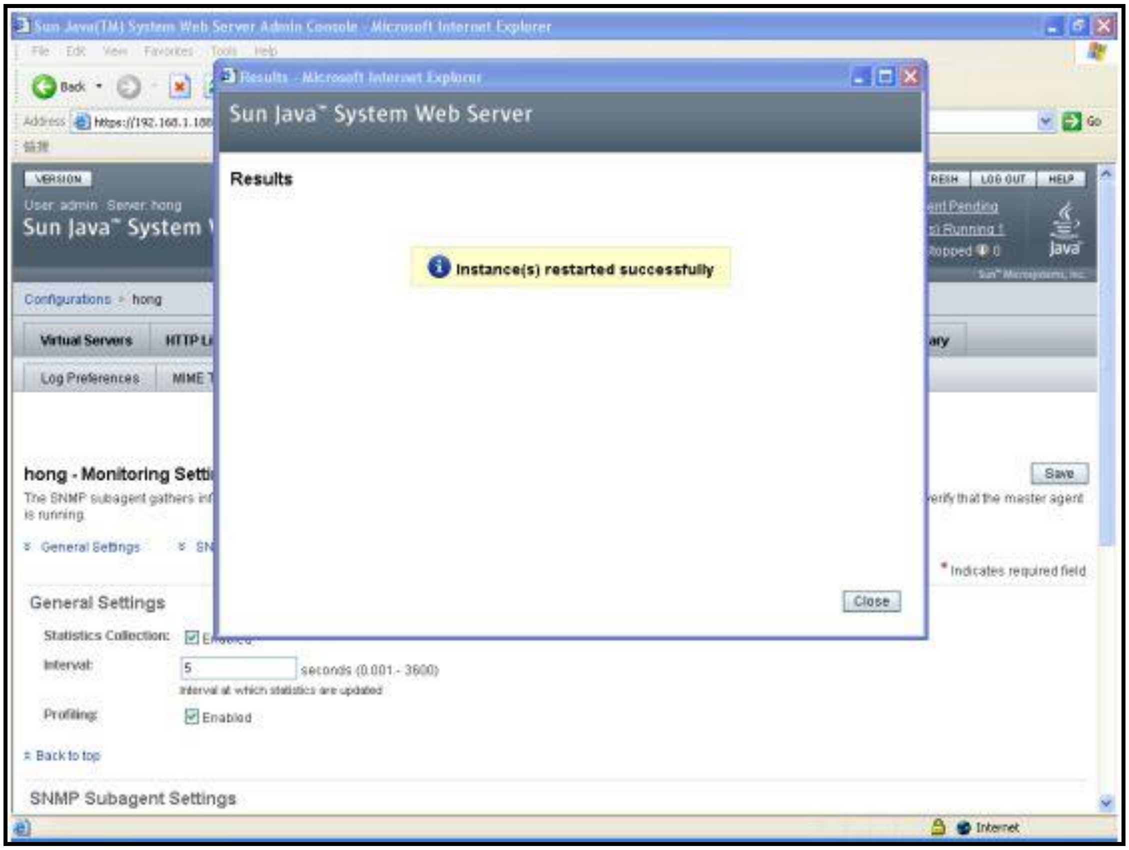
Profiling: ☑ Enabled

Back to top

SNMP Subagent Settings

Figure 1.14: Restarting the web server instance

8. Upon successful restart, Figure 1.15 will appear indicating the same. Click on the **Close** button to complete the operation.



Instance(s) restarted successfully

The SNMP subagent gathers inf … verify that the master agent is running.

Statistics Collection: ☑ Enabled

Interval: 5 seconds (0.001 - 3600)

Interval at which statistics are updated

Profiling: ☑ Enabled

Figure 1.15: Finishing the configuration change

Once all the pre-requisites are fulfilled, the eG agent starts extracting critical performance metrics from the server and reports the same to the eG manager. The manager then presents these metrics to the users using the monitoring model indicated by Figure 1.1.

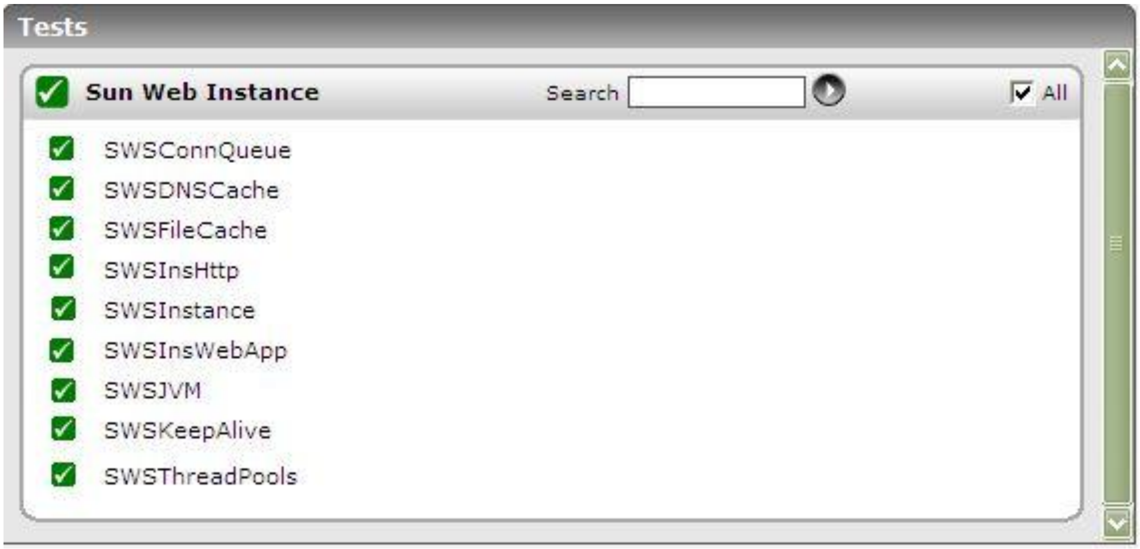
The sections that will follow will discuss each layer of this model.

1.2 The Sun Web Instance Layer

Using the tests mapped to this layer, you can measure the following for the configuration on the single node on which the target web server instance is running:

- The quality of the HTTP requests to and from the web server instance;
- The session activity on the web applications deployed on the virtual servers for the configuration;
- The responsiveness of and the traffic handled by the web server instance;
- The memory usage and thread usage of the web server instance;
- The current size and usage of the connection queues of the web server instance;
- How effectively the DNS cache and file cache were utilized by the server instance;
- The overall health of the JVM of the web server instance;

- The request processing ability of the web server instance measured on the basis of thread pool usage;



☑ Sun Web Instance

☑ All

☑ SWSConnQueue

☑ SWSDNSCache

☑ SWSFileCache

☑ SWSInsHttp

☑ SWSInstance

☑ SWSInsWebApp

☑ SWSJVM

☑ SWSKeepAlive

☑ SWSThreadPools

Figure 1.16: The tests pertaining to the Sun Web Instance layer

1.2.1 SWSConnQueue Test

In Web Server, acceptor threads on a listen socket accept connections and put them into a connection queue. Request processing threads in a thread pool then pick up connections from the queue and service the requests.

If the connection queue becomes full, the server will be forced to drop new connections, thus denying users access to the server. To avoid such an unpleasant outcome, it is imperative that the connection queues are monitored periodically.

This test keeps track of the changes to the size of the connection queues, so that administrators are proactively alerted to any abnormal increase in the queue size or an unusual delay while connections are being picked up for processing.

Purpose	Keeps track of the changes to the size of the connection queues, so that administrators are proactively alerted to any abnormal increase in the queue size or an unusual delay while connections are being picked up for processing
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **Test period** - How often should the test be executed 2. **Host -** The host for which the test is to be configured. 3. **Port –** Refers to the port used by the web server. 4. **user** – Specify the name of the *administrator* of the administration server to which the target web server is registered. 5. **passwdfile** – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the **passwdfile** text box; by default, this is *admin.passwd*. 6. **adminserver** – Specify the IP address of the administration server to which the target web server is registered. 7. **adminport** – Indicate the port at which the administration server listens. 8. **config** – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a **Configuration**. To enable the test to locate the web server instance to be monitored, you need to mention the **Configuration** that applies to the instance. In the **config** text box therefore, specify the name of the **Configuration** which the target web server instance uses. 9. **node** - **Node** is a network resource, such as a server or a host. **Instance** refers to the environment of a web server daemon on a given node, including its configuration, log files and other runtime artifacts such as lock databases, caches and temporary files. To enable the test to identify the exact web server instance to be monitored, you need to specify the name of the **node** on which the instance executes. The test will thus monitor only that web server instance that operates on the given **node** and which uses the specified **config**. 10. **ssl** – Set this flag to **Yes** if the administration server is SSL-enabled; if not, set it to **No**. 11. **webserverdir** – Here, specify the Sun Java web server installation directory		
Outputs of the test	One set of results for the Sun Java Web Server being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Present Conns Queued: Indicates the number of connections currently in the queue.	Number	

	No. of connections queued: Indicates the total number of times a connection was queued during the last measurement period.	Number	This number includes newly-accepted connections and connections from the keep-alive system.
	No. of connections overflowed: Indicates the number of connections that overflowed during the last measurement period.	Number	If the connection queue is full, then the acceptor threads will drop subsequent connections. The dropped connections represent the overflowed connections. Ideally, the value of this measure should be 0.
	Maximum queue size: Indicates the maximum size of the connection queue.	Number	This number is: Maximum Queue Size = Thread Pool Queue Size +Maximum Threads+Keep-Alive Queue Size If the value of this measure is close to the limit, you can increase the maximum connection queue size to avoid dropping connections under heavy load.
	Ticks spent: Indicates the amount of time that connections spent in the connection queue during last measurement period.	Secs	
	Average queuing delay: Indicates the average amount of time a connection spent in the connection queue during last measurement period.	Secs	This represents the delay between when a request connection is accepted by the server and when a request processing thread begins servicing the request. If the value of this measure is high, it could indicate a processing bottleneck on the server.
	Peak queue size: Indicates the largest number of connections that have been in the queue simultaneously.	Number	

1.2.2 SWSDNSCache Test

The DNS cache caches IP addresses and DNS names. The Web Server uses DNS caching for logging and for access control by IP address.

Using the SWSDNSCache test, you can determine whether the DNS cache is enabled or not, and if so, how effectively the web server instance has been using it.

Purpose	Determine whether the DNS cache is enabled or not, and if so, how effectively the web server instance has been using it
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **Test period** - How often should the test be executed 2. **Host -** The host for which the test is to be configured. 3. **Port –** Refers to the port used by the web server. 4. **user** – Specify the name of the *administrator* of the administration server to which the target web server is registered. 5. **passwdfile** – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the **passwdfile** text box; by default, this is *admin.passwd*. 6. **adminserver** – Specify the IP address of the administration server to which the target web server is registered. 7. **adminport** – Indicate the port at which the administration server listens. 8. **config** – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a **Configuration**. To enable the test to locate the web server instance to be monitored, you need to mention the **Configuration** that applies to the instance. In the **config** text box therefore, specify the name of the **Configuration** which the target web server instance uses. 9. **node** - **Node** is a network resource, such as a server or a host. **Instance** refers to the environment of a web server daemon on a given node, including its configuration, log files and other runtime artifacts such as lock databases, caches and temporary files. To enable the test to identify the exact web server instance to be monitored, you need to specify the name of the **node** on which the instance executes. The test will thus monitor only that web server instance that operates on the given **node** and which uses the specified **config**. 10. **ssl** – Set this flag to **Yes** if the administration server is SSL-enabled; if not, set it to **No**. 11. **webserverdir** – Here, specify the Sun Java web server installation directory		
Outputs of the test	One set of results for the Sun Java web server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	No of cache hits: Indicates the number of times the DNS cache was successfully looked up for an IP address.	Number	

	No. of cache misses: Indicates the number of times an IP address was not found in the cache.	Number	This number includes newly-accepted connections and connections from the keep-alive system.
	Hit ratio: Indicates the number of cache hits compared to the number of cache lookups.	Percent	Ideally, the value of this measure should be high. A low value is indicative of too many cache misses. This could be owing to a badly sized cache. Typically, the cache should be as large as the maximum number of clients that access your web site concurrently. Note that setting the cache size too high wastes memory and degrades performance.
	Async lookup enabled: Indicates whether the server uses its own asynchronous DNS resolver instead of the operating system's synchronous resolver.	Boolean	
	Lookups in progress: Indicates the current number of lookups in progress.	Number	
	Async name lookups: Indicates the number of times during the last measurement period DNS names were looked up in the cache, so that the corresponding IP addresses could be resolved.	Number	

	Async address lookups: Indicates the number of times IP addresses were looked up in the cache, so that the corresponding DNS name could be identified.	Number	
	DNS cache enabled: Indicates whether the cache is enabled or not.	Boolean	

1.2.3 SWSFileCache Test

The file cache caches static content so that the server handles requests for static content quickly. The file cache contains information about files and static file content. The file cache also caches information that is used to speed up processing of server-parsed HTML.

By periodically monitoring the usage of the file cache, you will not only be introduced to ineffective cache usage patterns, but also irregularities in the cache size, both of which may contribute to a slowdown in server operations.

Using the SWSFileCache test, you can monitor the usage of the file cache and detect sizing inadequacies.

Purpose	Monitors the usage of the file cache and reveals sizing inadequacies
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **Test period** - How often should the test be executed 2. **Host -** The host for which the test is to be configured. 3. **Port –** Refers to the port used by the web server. 4. **user** – Specify the name of the *administrator* of the administration server to which the target web server is registered. 5. **passwdfile** – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the **passwdfile** text box; by default, this is *admin.passwd*. 6. **adminserver** – Specify the IP address of the administration server to which the target web server is registered. 7. **adminport** – Indicate the port at which the administration server listens. 8. **config** – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a **Configuration**. To enable the test to locate the web server instance to be monitored, you need to mention the **Configuration** that applies to the instance. In the **config** text box therefore, specify the name of the **Configuration** which the target web server instance uses. 9. **node** - **Node** is a network resource, such as a server or a host. **Instance** refers to the environment of a web server daemon on a given node, including its configuration, log files and other runtime artifacts such as lock databases, caches and temporary files. To enable the test to identify the exact web server instance to be monitored, you need to specify the name of the **node** on which the instance executes. The test will thus monitor only that web server instance that operates on the given **node** and which uses the specified **config**. 10. **ssl** – Set this flag to **Yes** if the administration server is SSL-enabled; if not, set it to **No**. 11. **webserverdir** – Here, specify the Sun Java web server installation directory		
Outputs of the test	One set of results for the Sun Java web server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	No of cache hits: Indicates the number of times during the last measurement period the file cache was looked.	Number	Ideally, this value should be high.

	No. of cache misses: Indicates the number of times during the last measurement period the file cache could not service requests for file information or static content.	Number	Ideally, this value should be low. A high value indicates an ineffective cache – in other words, the cache may not have adequate entries to service requests. This could be owing to the small cache heap size. You may want to consider resizing the cache heap, so that the file cache is able to accommodate more entries, and thus service more number of requests.
	No. of cache content hits: Indicates the number of times during the last measurement period the cache was looked up for static content.	Number	Ideally, this value should be high.
	Cache content misses: Indicates the number of times during the last measurement period the file cache could not service requests for static content.	Number	Ideally, this value should be low. A high value indicates an ineffective cache – in other words, enough content may not have been cached by the server, thus causing many requests to go unserviced. This could be owing to the small cache heap size. You may want to consider resizing the cache heap, so that the file cache is able to accommodate more content, and thus service more number of requests.
	No. of file info lookups: Indicates the number of times during the last measurement period the file cache was looked up for file information.	Number	Ideally, this value should be high.
	No of FI lookup failures: Indicates the number of times during the last measurement period the file cache could not service requests for file information.	Number	Ideally, this value should be low. A high value indicates an ineffective cache – in other words, enough file information may not have been cached by the server, thus causing many requests for file info to go unserviced. This could be owing to the small cache heap size. You may want to consider resizing the cache heap, so that the file cache is able to accommodate more file information, and thus service more number of requests.

	No of entries: Indicates the current number of cache entries.	Number	
	Maximum cache size: Indicates the maximum number of entries that can be available in the file cache.	Number	The optimal file cache size is dependent on whether you are running a 32-bit or a 64-bit server. The available address space for a 32–bit server is limited to 4GB. The max-entries for file cache is based on the amount of Java heap, the number of threads (as specified by thread-pool/max-threads), and the connection queue size. It is recommended to cache small, frequently accessed static files in the file cache and use perfdump to ensure that the file cache hit ratio is close to 100%. To achieve this, you may increase file cache size and fine tune the max-entries for optimal performance. In a 64-bit server, due to the increased availability of address space, you can cache more static content in the file cache.
	No of open file entries: Indicates the current number of open file entries.	Number	For performance reasons, the web server caches the open file descriptors of large files, to avoid opening and closing such files frequently. The value of this measure represents the number of open file descriptors that are currently in the cache.
T	**Max open files allowed:** Indicates the maximum number of open files allowed.	Number	
	Heap size: Indicates the current cache heap size.	MB	
	Maximum heap cache size: Indicates the maximum cache heap size.	MB	The optimal cache heap size depends upon how much system memory is free. A larger heap size means that the Web Server can cache more content and therefore obtain a better hit ratio. However, the heap size should not be so large that the operating system starts paging cached files.

	Size of Mem mapped file: Indicates the current size of the memory mapped file content.	KB	For performance reasons, the web server caches the content of small files in memory. The value of this measure represents the number of small files that in cache.
	Max mem mapped file size: Indicates the maximum size of the memory mapped file.	KB	
	Maximum age of entries: Indicates the maximum age of a valid cache entry.	Secs	This measure controls how long cached information is used after a file has been cached. An entry older than the maximum age is replaced by a new entry for the same file.
	File cache enabled: Indicates whether the file cache has been enabled or not.	Boolean	

1.2.4 SWSInsHttp Test

When you use virtual servers you can offer companies or individuals domain names, IP addresses, and some server monitoring capabilities with a single installed server.

All virtual servers have an HTTP Listener specified. When a new request comes in, the web server determines which virtual server to send it to based on the configured HTTP Listener.

This test monitors the HTTP requests to and responses from each virtual server configured on a web server instance, and thus reveals the quality of the HTTP service on the virtual servers.

Purpose	Monitors the HTTP requests to and responses from each virtual server configured on a web server instance, and thus reveals the quality of the HTTP service on the virtual servers
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent

<table>
<tr><td><b>Configurable parameters for the test</b></td><td colspan="3">1. <b>Test period</b> - How often should the test be executed
2. <b>Host -</b> The host for which the test is to be configured.
3. <b>Port –</b> Refers to the port used by the web server.
4. <b>user</b> – Specify the name of the <i>administrator</i> of the administration server to which the target web server is registered.
5. <b>passwdfile</b> – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the <b>passwdfile</b> text box; by default, this is <i>admin.passwd</i>.
6. <b>adminserver</b> – Specify the IP address of the administration server to which the target web server is registered.
7. <b>adminport</b> – Indicate the port at which the administration server listens.
8. <b>config</b> – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a <b>Configuration</b>. To enable the test to locate the web server instance to be monitored, you need to mention the <b>Configuration</b> that applies to the instance. In the <b>config</b> text box therefore, specify the name of the <b>Configuration</b> which the target web server instance uses.
9. <b>node</b> - <b>Node</b> is a network resource, such as a server or a host. <b>Instance</b> refers to the environment of a web server daemon on a given node, including its configuration, log files and other runtime artifacts such as lock databases, caches and temporary files. To enable the test to identify the exact web server instance to be monitored, you need to specify the name of the <b>node</b> on which the instance executes. The test will thus monitor only that web server instance that operates on the given <b>node</b> and which uses the specified <b>config</b>.
10. <b>ssl</b> – Set this flag to <b>Yes</b> if the administration server is SSL-enabled; if not, set it to <b>No</b>.
11. <b>webserverdir</b> – Here, specify the Sun Java web server installation directory</td></tr>
<tr><td><b>Outputs of the test</b></td><td colspan="3">One set of results for the each virtual server configured on the Sun Java web server instance being monitored</td></tr>
<tr><td rowspan="2"><b>Measurements made by the test</b></td><td><b>Measurement</b></td><td><b>Measurement Unit</b></td><td><b>Interpretation</b></td></tr>
<tr><td><b>Requests:</b>
Indicates the rate of requests to this virtual server.</td><td>Reqs/Sec</td><td></td></tr>
</table>

	No. of requests: Indicates the number of requests to this virtual server during the last measurement period.	Number	
	No. of errors: Indicates the number of errors logged on this virtual server during the last measurement period.	Number	Ideally, this value should be 0.
	No. of open connections: Indicates the number of connections currently open on this virtual server .	Number	
	Data received: Indicates the rate of data received by this virtual server.	KB/Sec	
	Data transmitted: Indicates the rate of data transmitted by this virtual server.	KB/Sec	
	No of 200 responses: Indicates the number of HTTP responses sent out by this virtual server during the last measurement period, with the 200 code.	Number	The 200 response is the standard response for successful HTTP requests.
	Successful responses: Indicates the number of successful responses sent out by this virtual server during the last measurement period.	Number	The value of this measure typically represents the number of responses with the 2xx code. This class of status codes indicates the action requested by the client was received, understood, accepted and processed successfully.

	No of 302 responses: Indicates the number of responses sent out by this virtual server during the last measurement period, with the 302 code.	Number	The 302 response indicates that the requested resource has been found under a different URI but the client should continue to use the original URI.
	No. of 304 responses: Indicates the number of responses sent out by this virtual server during the last measurement period, with the 304 code.	Number	The 304 response indicates that the resource has not been modified since the last request.
	Redirection: Indicates the number of responses with the 3xx code that were sent out by this virtual server during the last measurement period.	Number	This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request. The action required *may* be carried out by the user agent without interaction with the user if and only if the method used in the second request is GET or HEAD. A user agent *should not* automatically redirect a request more than five times, since such redirections usually indicate an infinite loop.
	No of 400 responses: Indicates the number of responses with the 400 code that were sent out by this virtual server during the last measurement period.	Number	This type of response indicates that the request contains bad syntax or cannot be fulfilled.
	No of 401 responses: Indicates the number of responses with the 401 code that were sent out by this virtual server during the last measurement period.	Number	This type of response indicates that the request needs user authentication.
	No of 403 responses: Indicates the number of responses with the 403 code that were sent out by this virtual server during the last measurement period.	Number	This type of response indicates that the the request was a legal request, but the server is refusing to respond to it.

	No of 404 responses: Indicates the number of responses with the 404 code that were sent out by this virtual server during the last measurement period.	Number	This type of response indicates that the document/file requested by the client was not found.
	Client errors responses: Indicates the number of responses with the 4xx code that were sent out by this virtual server during the last measurement period.	Number	Normally, value of the measure should be 0. A nonzero value indicates there are some problems with the requested content.
	No of 503 responses: Indicates the number of responses with the 503 code that were sent out by this virtual server during the last measurement period.	Number	The 503 code indicates that the request was unsuccessful due to the server being down or overloaded.
	Server errors responses: Indicates the number of responses with the 5xx code that were sent out by this virtual server during the last measurement period.	Number	Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has encountered an error or is otherwise incapable of performing the request. Normally, the value of the measure should be 0. A nonzero value indicates there are some errors in the web server.

1.2.5 SWSInstance Test

Instance refers to the environment of a web server daemon on a given node, including its configuration, log files and other runtime artifacts such as lock databases, caches and temporary files.

This test reports useful performance statistics related to a web server instance, which indicate the following:

- How long has the instance been up and running?
- How quickly does the instance respond to requests? Is there a processing bottheleck on the server instance?
- Have any been errors been captured on the instance?
- How is the data traffic to and from the instance?

Purpose	Reports useful performance statistics related to a web server instance
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **Test period** - How often should the test be executed 2. **Host -** The host for which the test is to be configured. 3. **Port –** Refers to the port used by the web server. 4. **user** – Specify the name of the *administrator* of the administration server to which the target web server is registered. 5. **passwdfile** – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the **passwdfile** text box; by default, this is *admin.passwd*. 6. **adminserver** – Specify the IP address of the administration server to which the target web server is registered. 7. **adminport** – Indicate the port at which the administration server listens. 8. **config** – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a **Configuration**. To enable the test to locate the web server instance to be monitored, you need to mention the **Configuration** that applies to the instance. In the **config** text box therefore, specify the name of the **Configuration** which the target web server instance uses. 9. **node** - **Node** is a network resource, such as a server or a host. **Instance** refers to the environment of a web server daemon on a given node, including its configuration, log files and other runtime artifacts such as lock databases, caches and temporary files. To enable the test to identify the exact web server instance to be monitored, you need to specify the name of the **node** on which the instance executes. The test will thus monitor only that web server instance that operates on the given **node** and which uses the specified **config**. 10. **ssl** – Set this flag to **Yes** if the administration server is SSL-enabled; if not, set it to **No**. 11. **webserverdir** – Here, specify the Sun Java web server installation directory

	12. **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 enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: ○ The eG manager license should allow the detailed diagnosis capability ○ Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.		
Outputs of the test	One set of results for the Sun Java web server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Instance running time: Indicates the total time for which the instance has been up and running.	Secs	By observing the variations in this measure over time, you can determine whether there were frequent breaks in the availability of the instance. The detailed diagnosis of this measure (if enabled) displays starting time of the instance.
	Requests: Indicates the rate of requests to all the virtual servers in the instance.	Reqs/Sec	
	No. of errors: Indicates the number of errors logged on all the virtual servers in the instance during the last measurement period.	Number	Ideally, this value should be 0.
	Maximum response time: Indicates the maximum response time of all virtual servers since the instance was up.	Secs	A low response time is typically desired. A high value of this measure could indicate a processing bottleneck on the web server instance or a network congestion.

	Avg res time in Last5min: Average response time of all virtual servers in the instance during last 5 minutes.	Secs	
	Data transmitted: Indicates the rate of data transmitted by this instance.	KB/Sec	
	Data received: Indicates the rate of data received by this instance.	KB/Sec	

1.2.6 SWSInsWebApp

This test auto-discovers the web applications deployed on each virtual server configured on the web server instance being monitored, and monitors the uptime of and session activity on each web application.

Purpose	Auto-discovers the web applications deployed on each virtual server configured on the web server instance being monitored, and monitors the uptime of and session activity on each web application
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **Test period** - How often should the test be executed 2. **Host -** The host for which the test is to be configured. 3. **Port –** Refers to the port used by the web server. 4. **user** – Specify the name of the *administrator* of the administration server to which the target web server is registered. 5. **passwdfile** – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the **passwdfile** text box; by default, this is *admin.passwd*. 6. **adminserver** – Specify the IP address of the administration server to which the target web server is registered. 7. **adminport** – Indicate the port at which the administration server listens. 8. **config** – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a **Configuration**. To enable the test to locate the web server instance to be monitored, you need to mention the **Configuration** that applies to the instance. In the **config** text box therefore, specify the name of the **Configuration** which the target web server instance uses. 9. **node** - **Node** is a network resource, such as a server or a host. **Instance** refers to the environment of a web server daemon on a given node, including its configuration, log files and other runtime artifacts such as lock databases, caches and temporary files. To enable the test to identify the exact web server instance to be monitored, you need to specify the name of the **node** on which the instance executes. The test will thus monitor only that web server instance that operates on the given **node** and which uses the specified **config**. 10. **ssl** – Set this flag to **Yes** if the administration server is SSL-enabled; if not, set it to **No**. 11. **webserverdir** – Here, specify the Sun Java web server installation directory		
Outputs of the test	One set of results for each *virtualserver_webapplication* pair auto-discovered on the web server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	No of JSPs loaded: Indicates the number of JSPs that were loaded on to this application during the last measurement period.	Number	

	No of JSPs reloaded: Indicates the number of JSPs reloaded during the last measurement period.	Number	
	No. of sessions serviced: Indicates the number of sessions that were created on the web application during the last measurement period.	Number	This is a good indicator of the session load on the web application.
	No of sessions active: Indicates the number of currently active sessions on this web application.	Number	
	Peak sessions: Indicates the peak active sessions.	Number	
	No of sessions rejected: Indicates the number of sessions rejected by this web application during the last measurement period.	Number	A very high value of this measure could indicate a problem; further investigation is recommended.
	No of sessions expired: Indicates the number of sessions that expired during the last measurement period.	Number	
	Average alive time: Indicates the average lifetime of all sessions that have expired.	Secs	
	Peak alive time: Indicates the longest lifetime of all sessions which have expired.	Secs	

1.2.7 SWSJVM Test

This test reports the overall health of the JVM of the web server instance being monitored by measuring the following:

- How well the JVM manages the memory heap;
- The level of activity on the JVM in terms of the number of threads that are currently in use;

Purpose	Reports the overall health of the JVM of the web server instance being monitored
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent
Configurable parameters for the test	1. **Test period** - How often should the test be executed 2. **Host -** The host for which the test is to be configured. 3. **Port –** Refers to the port used by the web server. 4. **user** – Specify the name of the *administrator* of the administration server to which the target web server is registered. 5. **passwdfile** – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the **passwdfile** text box; by default, this is *admin.passwd*. 6. **adminserver** – Specify the IP address of the administration server to which the target web server is registered. 7. **adminport** – Indicate the port at which the administration server listens. 8. **config** – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a **Configuration**. To enable the test to locate the web server instance to be monitored, you need to mention the **Configuration** that applies to the instance. In the **config** text box therefore, specify the name of the **Configuration** which the target web server instance uses. 9. **node** - **Node** is a network resource, such as a server or a host. **Instance** refers to the environment of a web server daemon on a given node, including its configuration, log files and other runtime artifacts such as lock databases, caches and temporary files. To enable the test to identify the exact web server instance to be monitored, you need to specify the name of the **node** on which the instance executes. The test will thus monitor only that web server instance that operates on the given **node** and which uses the specified **config**. 10. **ssl** – Set this flag to **Yes** if the administration server is SSL-enabled; if not, set it to **No**. 11. **webserverdir** – Here, specify the Sun Java web server installation directory

Outputs of the test	One set of results for the web server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Heap memory size: Indicates the current heap memory size.	MB	
	Elapsed GC Time: Indicates the time taken for garbage collection.	Secs	If adequate memory is not allotted to the JVM, then the value of this measure would be very high. This is not a good sign, as GC, during its execution, has the tendency of suspending an application, and a high value of this measure would only adversely impact the server’s performance. To avoid this, it is recommended that you allot sufficient memory to the JVM.
	No. of GC occurred: Indicates the number of garbage collections that occurred during the last measurement period.	Number	If adequate memory is not allotted to the JVM, then the value of this measure would be very high. A high value of this measure is indicative of a high frequency of GC. This is not a good sign, as GC, during its execution, has the tendency of suspending an application, and a high frequency of GC would only adversely impact the server's performance. To avoid this, it is recommended that you allot sufficient memory to the JVM.
	Present classes loaded: Indicates the number of classes currently loaded in the JVM.	Number	
	No of classes loaded: Indicates the number of classes that were loaded during the last measurement period.	Number	

	No of classes unloaded: Indicates the number of classes unloaded during the last measurement period.	Number	
	No of live threads: Indicates the number of live daemon and non-daemon threads.	Number	A large number of threads is indicative of high JVM activity.
	No of started threads: Indicates the number of threads created and/or started during last measurement period.	Number	
	Peak Live Thread Count: Indicates the peak live thread count since the JVM started or the peak was reset.	Number	

1.2.8 SWSKeepAlive Test

Http operates on what is called a request-response paradigm. This means that a client generates a request for information, and passes it to the server, which answers it. In the original implementation of HTTP, each request created a new socket connection to the server, sent the request, then read from that connection to get the response.

While this approach was simple to understand and implement, it was also slow. So, keep-alive connections were invented for HTTP. Under HTTP 1.0, if the browser supports keep-alive, it adds an additional header to the request:

Connection: Keep-Alive

Then, when the server receives this request and generates a response, it also adds a header to the response:

Connection: Keep-Alive

Following this, the connection is NOT dropped, but is instead kept open. When the client sends another request, it uses the same connection. This will continue until either the client or the server decides that the conversation is over, and one of them drops the connection.

Under HTTP 1.1, the official keepalive method is different. **All** connections are kept alive, unless stated otherwise with the following header:

Connection: close

A web server can receive hundreds of new HTTP requests per second. If every request was allowed to keep the connection open indefinitely, the server could become overloaded with connections. Using the SWSKeepAlive test you can continuously monitor the inflow of keep-alive HTTP connections to the target web server instance, so that you can be proactively alerted to a potential connection overload, and avert it before any irreparable damage occurs.

Purpose	Continuously monitor the inflow of keep-alive HTTP connections to the target web server instance, so that you can be proactively alerted to a potential connection overload, and avert it before any irreparable damage occurs
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent

<table>
<tr><td rowspan="1"><b>Configurable parameters for the test</b></td><td colspan="3">1. <b>Test period</b> - How often should the test be executed
2. <b>Host -</b> The host for which the test is to be configured.
3. <b>Port –</b> Refers to the port used by the web server.
4. <b>user</b> – Specify the name of the <i>administrator</i> of the administration server to which the target web server is registered.
5. <b>passwdfile</b> – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the <b>passwdfile</b> text box; by default, this is <i>admin.passwd</i>.
6. <b>adminserver</b> – Specify the IP address of the administration server to which the target web server is registered.
7. <b>adminport</b> – Indicate the port at which the administration server listens.
8. <b>config</b> – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a <b>Configuration</b>. To enable the test to locate the web server instance to be monitored, you need to mention the <b>Configuration</b> that applies to the instance. In the <b>config</b> text box therefore, specify the name of the <b>Configuration</b> which the target web server instance uses.
9. <b>node</b> - <b>Node</b> is a network resource, such as a server or a host. <b>Instance</b> refers to the environment of a web server daemon on a given node, including its configuration, log files and other runtime artifacts such as lock databases, caches and temporary files. To enable the test to identify the exact web server instance to be monitored, you need to specify the name of the <b>node</b> on which the instance executes. The test will thus monitor only that web server instance that operates on the given <b>node</b> and which uses the specified <b>config</b>.
10. <b>ssl</b> – Set this flag to <b>Yes</b> if the administration server is SSL-enabled; if not, set it to <b>No</b>.
11. <b>webserverdir</b> – Here, specify the Sun Java web server installation directory</td></tr>
<tr><td><b>Outputs of the test</b></td><td colspan="3">One set of results for the web server instance being monitored</td></tr>
<tr><td rowspan="2"><b>Measurements made by the test</b></td><td><b>Measurement</b></td><td><b>Measurement Unit</b></td><td><b>Interpretation</b></td></tr>
<tr><td><b>Keep alive hits:</b>
Indicates the number of times a request was successfully received from a connection that was kept alive during the last measurement period.</td><td>Number</td><td></td></tr>
</table>

	Number of connections added: Indicates the current number of connections in keep-alive mode.	Number	If the *Number of connections added* measure registers an abnormally high value, or is found to increase consistently, then, it is indicative of a probable connection overload. On UNIX and Linux systems, this could lead to a file table overflow very easily. To deal with this problem, the server maintains a counter for the maximum number of waiting keep-alive connections. The *Maximum connection size* measure reports the value of this counter only. A waiting keep-alive connection is one that has fully completed processing the previous request, and is now waiting for a new request to arrive on the same connection. If the server has more than the maximum waiting connections open when a new connection waits for a keep-alive request, the server closes the oldest connection. This algorithm keeps an upper bound on the number of open waiting keep-alive connections that the server can maintain.
	Maximum connection size: Indicates the maximum number of connections allowed in keep-alive mode simultaneously.	Number	
	No of connections flushed: Indicates the number of times during the last measurement period the server had to close a connection because the total number of connections added exceeded the keep-alive maximum connections setting.	Number	The server does not always close existing connections when the keep-alive count exceeds the maximum connection size. Instead, new keep-alive connections are refused and the number of connections refused count is incremented.
	No of connections refused: Indicates the number of times the server could not complete the connection to a keep-alive thread during the last measurement period.	Number	This could happen due to too many persistent connections (or when the total number of connections added exceeds the keep-alive maximum connections setting).

	Keep alive timeouts: Indicates the number of times during the last measurement period the server closed idle keep-alive connections because client connections timed out without any activity.	Number	
	Connection timeout: Indicates the time before idle keep-alive connections are closed.	Secs	The default timeout period is 30 seconds, meaning the connection times out if idle for more than 30 seconds. The maximum is 3600 seconds (60 minutes). Since idle connections are a resource-drain, you may want to fine-tune the timeout settings for your keep-alive connections to prevent the wastage of resources.

1.2.9 SWSThreadPools Test

For web server instances that are configured with the default settings, threads from the default thread pool process requests to the instance. In addition, custom thread pools can also be created and used to run custom NSAPI functions. By default, the web server creates one additional pool, named *NativePool*.

This test auto-discovers the native and custom thread pools used by a web server instance, and reports how well each pool is used.

Purpose	Auto-discovers the native and custom thread pools used by a web server instance, and reports how well each pool is used
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **Test period** - How often should the test be executed 2. **Host -** The host for which the test is to be configured. 3. **Port –** Refers to the port used by the web server. 4. **user** – Specify the name of the *administrator* of the administration server to which the target web server is registered. 5. **passwdfile** – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the **passwdfile** text box; by default, this is *admin.passwd*. 6. **adminserver** – Specify the IP address of the administration server to which the target web server is registered. 7. **adminport** – Indicate the port at which the administration server listens. 8. **config** – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a **Configuration**. To enable the test to locate the web server instance to be monitored, you need to mention the **Configuration** that applies to the instance. In the **config** text box therefore, specify the name of the **Configuration** which the target web server instance uses. 9. **node** - **Node** is a network resource, such as a server or a host. **Instance** refers to the environment of a web server daemon on a given node, including its configuration, log files and other runtime artifacts such as lock databases, caches and temporary files. To enable the test to identify the exact web server instance to be monitored, you need to specify the name of the **node** on which the instance executes. The test will thus monitor only that web server instance that operates on the given **node** and which uses the specified **config**. 10. **ssl** – Set this flag to **Yes** if the administration server is SSL-enabled; if not, set it to **No**. 11. **webserverdir** – Here, specify the Sun Java web server installation directory		
Outputs of the test	One set of results for each thread pool used by the web server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	No of idle threads: Indicates the number of threads in this pool that are currently idle.	Number	A high value for this measure indicates that there are adequate free threads in the pool for servicing additional requests to the web server instance.
	No of threads: Indicates the number of threads in this pool.	Number	

	No of requests queued: Indicates the number of requests currently waiting for a native thread from this pool.	Number	An abnormally high value for this measure could indicate the absence of free threads in the pool to service requests.
	Peak requests queued: Indicates the highest number of requests that were ever queued up simultaneously for the use of a native thread from this pool since the server was started.	Number	This value can be viewed as the maximum concurrency for requests requiring a native thread.

1.2.10 SWSProcess Test

This test reports useful statistics for the process of the web server instance.

Purpose	Reports useful statistics for the process of the web server instance./
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent

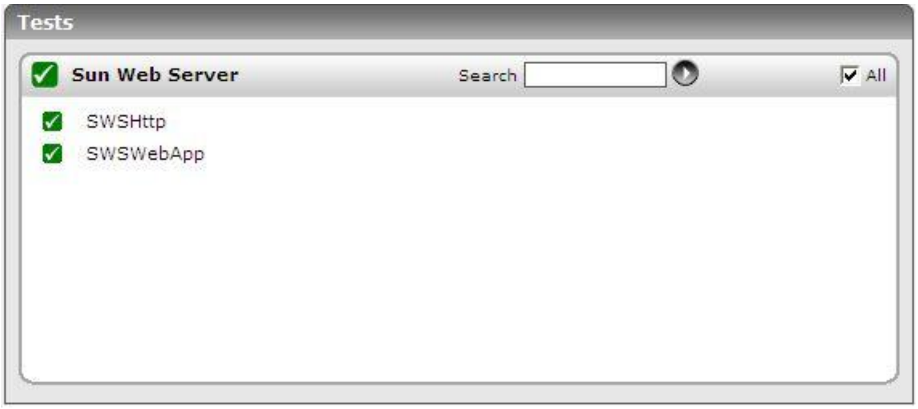
Configurable parameters for the test	1. **Test period** - How often should the test be executed 2. **Host -** The host for which the test is to be configured. 3. **Port –** Refers to the port used by the web server. 4. **user** – Specify the name of the *administrator* of the administration server to which the target web server is registered. 5. **passwdfile** – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the **passwdfile** text box; by default, this is *admin.passwd*. 6. **adminserver** – Specify the IP address of the administration server to which the target web server is registered. 7. **adminport** – Indicate the port at which the administration server listens. 8. **config** – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a **Configuration**. To enable the test to locate the web server instance to be monitored, you need to mention the **Configuration** that applies to the instance. In the **config** text box therefore, specify the name of the **Configuration** which the target web server instance uses. 9. **node** - **Node** is a network resource, such as a server or a host. **Instance** refers to the environment of a web server daemon on a given node, including its configuration, log files and other runtime artifacts such as lock databases, caches and temporary files. To enable the test to identify the exact web server instance to be monitored, you need to specify the name of the **node** on which the instance executes. The test will thus monitor only that web server instance that operates on the given **node** and which uses the specified **config**. 10. **ssl** – Set this flag to **Yes** if the administration server is SSL-enabled; if not, set it to **No**. 11. **webserverdir** – Here, specify the Sun Java web server installation directory		
Outputs of the test	One set of results for the Sun Java web server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Process ID: Indicates the process identifier.	Number	
	Resident memory size: Indicates the size of resident memory.	MB	

	Virtual memory size: Indicates the size of virtual memory.	MB	
	System memory usage: Indicates the fraction of system memory used.	MB	A large value is indicative of excessive memory usage.
	Thread pools: Indicates the number of thread pools.	Number	
	Threads: Indicates the number of threads.	Number	
	Idle threads: Indicates the number of idle threads.	Number	
	JDBC pools: Indicates the number of JDBC pools.	Number	
	Connection queues: Indicates the number of connection queues.	Number	
	Configurations: Indicates the number of configurations loaded.	Number	

1.3 The Sun Web Server Layer

Using the tests mapped to this layer, you can measure the following:

- The quality of the HTTP requests to and from the virtual server across all the nodes where a specified configuration has been deployed;
- The session activity on the web applications deployed on the virtual server across all the nodes where a specified configuration has been deployed



Tests

☑ Sun Web Server

Search

☑ All

☑ SWSHttp

☑ SWSWebApp

Figure 1.17: The tests mapped to the Sun Web Server layer

1.3.1 SWSHttp Test

When you use virtual servers you can offer companies or individuals domain names, IP addresses, and some server monitoring capabilities with a single installed server.

All virtual servers have an HTTP Listener specified. When a new request comes in, the web server determines which virtual server to send it to based on the configured HTTP Listener.

This test monitors the HTTP requests to and responses from each virtual server across all the nodes where a specific configuration applies, and thus reveals the quality of the HTTP service on the virtual servers.

Purpose	Monitors the HTTP requests to and responses from each virtual server across all the nodes where a specific configuration has been deployed
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **Test period** - How often should the test be executed 2. **Host -** The host for which the test is to be configured. 3. **Port –** Refers to the port used by the web server. 4. **user** – Specify the name of the *administrator* of the administration server to which the target web server is registered. 5. **passwdfile** – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the **passwdfile** text box; by default, this is *admin.passwd*. 6. **adminserver** – Specify the IP address of the administration server to which the target web server is registered. 7. **adminport** – Indicate the port at which the administration server listens. 8. **config** – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a **Configuration**. This test will monitor those virtual servers across all nodes where the the configuration specified against **config** is deployed. In the **config** text box therefore, specify the name of the **Configuration** that applies to the virtual servers of interest to you. 9. **ssl** – Set this flag to **Yes** if the administration server is SSL-enabled; if not, set it to **No**. 10. **webserverdir** – Here, specify the Sun Java web server installation directory		
Outputs of the test	One set of results for the each virtual server configured on the Sun Java web server instance being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	Requests: Indicates the rate of requests to this virtual server.	Reqs/Sec	
	No. of requests: Indicates the number of requests to this virtual server during the last measurement period.	Number	

	No. of errors: Indicates the number of errors logged on this virtual server during the last measurement period.	Number	Ideally, this value should be 0.
	No. of open connections: Indicates the number of connections currently open on this virtual server .	Number	
	Data received: Indicates the rate of data received by this virtual server.	KB/Sec	
	Data transmitted: Indicates the rate of data transmitted by this virtual server.	KB/Sec	
	No of 200 responses: Indicates the number of HTTP responses sent out by this virtual server during the last measurement period, with the 200 code.	Number	The 200 response is the standard response for successful HTTP requests.
	Successful responses: Indicates the number of successful responses sent out by this virtual server during the last measurement period.	Number	The value of this measure typically represents the number of responses with the 2xx code. This class of status codes indicates the action requested by the client was received, understood, accepted and processed successfully.
	No of 302 responses: Indicates the number of responses sent out by this virtual server during the last measurement period, with the 302 code.	Number	The 302 response indicates that the requested resource has been found under a different URI but the client should continue to use the original URI.

	No. of 304 responses: Indicates the number of responses sent out by this virtual server during the last measurement period, with the 304 code.	Number	The 304 response indicates that the resource has not been modified since the last request.
	Redirection: Indicates the number of responses with the 3xx code that were sent out by this virtual server during the last measurement period.	Number	This class of status code indicates that further action needs to be taken by the user agent in order to fulfil the request. The action required *may* be carried out by the user agent without interaction with the user if and only if the method used in the second request is GET or HEAD. A user agent *should not* automatically redirect a request more than five times, since such redirections usually indicate an infinite loop.
	No of 400 responses: Indicates the number of responses with the 400 code that were sent out by this virtual server during the last measurement period.	Number	This type of response indicates that the request contains bad syntax or cannot be fulfilled.
	No of 401 responses: Indicates the number of responses with the 401 code that were sent out by this virtual server during the last measurement period.	Number	This type of response indicates that the request needs user authentication.
	No of 403 responses: Indicates the number of responses with the 403 code that were sent out by this virtual server during the last measurement period.	Number	This type of response indicates that the the request was a legal request, but the server is refusing to respond to it.
	No of 404 responses: Indicates the number of responses with the 404 code that were sent out by this virtual server during the last measurement period.	Number	This type of response indicates that the document/file requested by the client was not found.

	Client errors responses: Indicates the number of responses with the 4xx code that were sent out by this virtual server during the last measurement period.	Number	Normally, value of the measure should be 0. A nonzero value indicates there are some problems with the requested content.
	No of 503 responses: Indicates the number of responses with the 503 code that were sent out by this virtual server during the last measurement period.	Number	The 503 code indicates that the request was unsuccessful due to the server being down or overloaded.
	Server errors responses: Indicates the number of responses with the 5xx code that were sent out by this virtual server during the last measurement period.	Number	Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has encountered an error or is otherwise incapable of performing the request. Normally, the value of the measure should be 0. A nonzero value indicates there are some errors in the web server.

1.3.2 SWSWebApp Test

This test auto-discovers the web applications deployed on the virtual servers across all the nodes where a specific configuration has been deployed, and monitors the uptime of and session activity on each web application.

Purpose	Auto-discovers the web applications deployed on the virtual servers across all the nodes where a specific configuration has been deployed, and monitors the uptime of and session activity on each web application
Target of the test	A Sun Java Web Server
Agent deploying the test	An internal agent

Configurable parameters for the test	1. **Test period** - How often should the test be executed 2. **Host -** The host for which the test is to be configured. 3. **Port –** Refers to the port used by the web server. 4. **user** – Specify the name of the *administrator* of the administration server to which the target web server is registered. 5. **passwdfile** – The eG agent connects to the administration server to which the target Sun Java web server is registered to extract the required metrics. To enable the eG agent to communicate with this administration server, you first need to create a password file containing the administrator password for accessing the server; this password file should be created in the install directory of the Sun Java web server. Specify the name of this file in the **passwdfile** text box; by default, this is *admin.passwd*. 6. **adminserver** – Specify the IP address of the administration server to which the target web server is registered. 7. **adminport** – Indicate the port at which the administration server listens. 8. **config** – In Web Server 7.0, all configurable elements of a server instance like web applications, configuration files, and search collection indexes are logically grouped and termed as a **Configuration**. This test will monitor those virtual servers across all nodes where the the configuration specified against **config** is deployed. In the **config** text box therefore, specify the name of the **Configuration** that applies to the virtual servers of interest to you. 9. **ssl** – Set this flag to **Yes** if the administration server is SSL-enabled; if not, set it to **No**. 10. **webserverdir** – Here, specify the Sun Java web server installation directory		
Outputs of the test	One set of results for each *virtualserver_webapplication* pair auto-discovered from the server configuration being monitored		
Measurements made by the test	**Measurement**	**Measurement Unit**	**Interpretation**
	No of JSPs loaded: Indicates the number of JSPs that were loaded on to this application during the last measurement period.	Number	
	No of JSPs reloaded: Indicates the number of JSPs reloaded during the last measurement period.	Number	

	No. of sessions serviced: Indicates the number of sessions that were created on the web application during the last measurement period.	Number	This is a good indicator of the session load on the web application.
	No of sessions active: Indicates the number of currently active sessions on this web application.	Number	
	Peak sessions: Indicates the peak active sessions.	Number	
	No of sessions rejected: Indicates the number of sessions rejected by this web application during the last measurement period.	Number	A very high value of this measure could indicate a problem; further investigation is recommended.
	No of sessions expired: Indicates the number of sessions that expired during the last measurement period.	Number	
	Average alive time: Indicates the average lifetime of all sessions that have expired.	Secs	
	Peak alive time: Indicates the longest lifetime of all sessions which have expired.	Secs	

Chapter

2

Conclusion

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

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