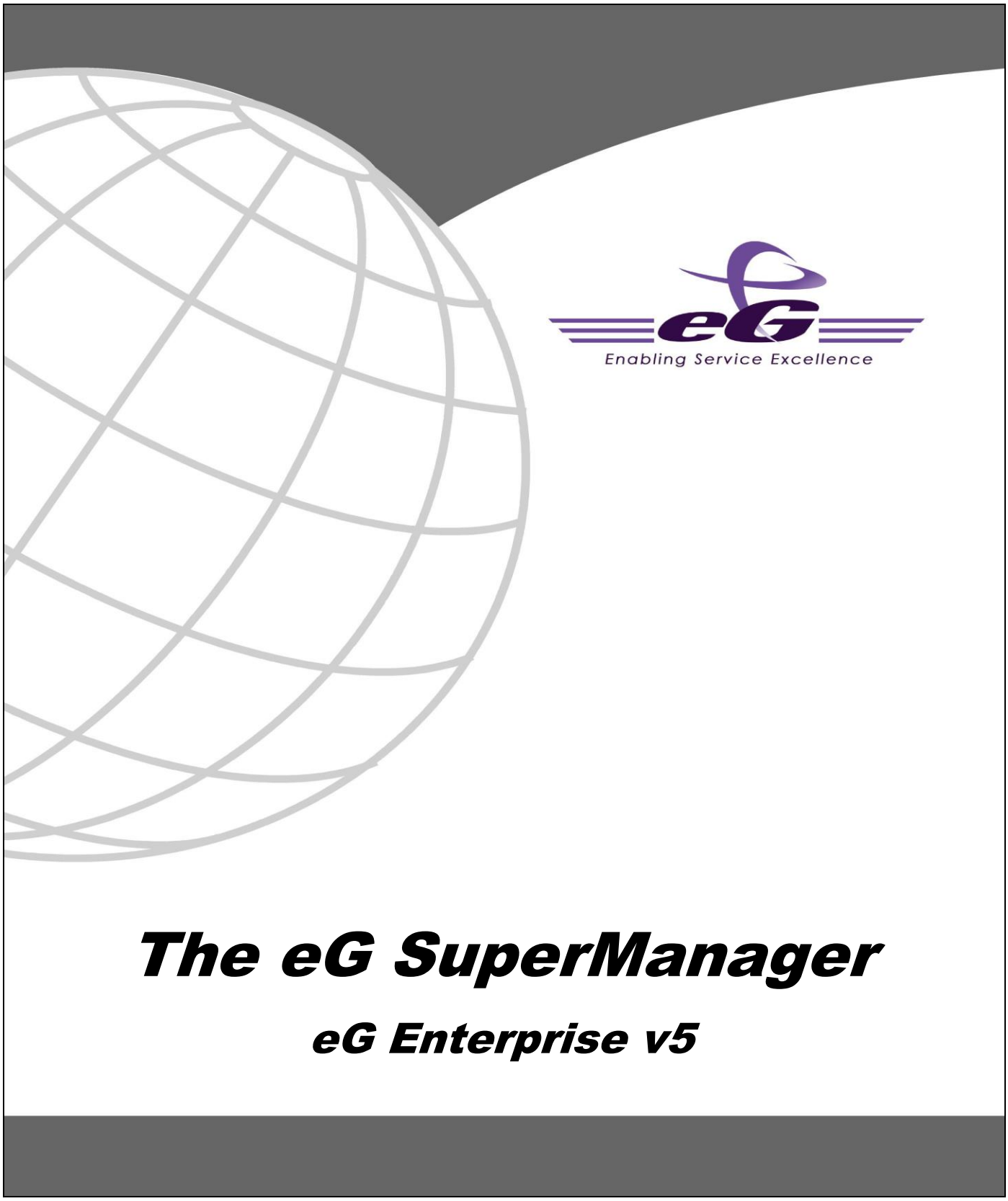


The eG SuperManager

eG Enterprise v5

Restricted Rights Legend

Trademarks

Copyright

Table of Contents

INTRODUCTION 1

INSTALLING AND CONFIGURING THE EG SUPERMANAGER 5

2.1 Installing the eG SuperManager on Linux Environments 6

2.2 Installing the eG SuperManager on Solaris Environments 7

2.3 Configuring the eG SuperManager on Unix Environments 8

2.4 Installing and Configuring the eG SuperManager on Windows Environments 10

2.5 SSL-Enabling the eG SuperManager on Unix 14

2.5.1 SSL-Enabling the eG SuperManager Using a Certificate Signed by an Internal CA 15

2.5.2 SSL-Enabling the eG SuperManager Using a Signed Certificate Obtained from a Valid Certifying Authority 23

2.6 SSL-Enabling the eG SuperManager on Windows 29

2.6.1 SSL-Enabling the eG SuperManager Using a Certificate Signed by an Internal CA 29

2.6.2 SSL-Enabling the eG SuperManager Using a Signed Certificate Obtained from a Valid Certifying Authority 38

2.7 Configuring the eG SuperManager to Work with the Individual eG Managers 46

2.8 Configuring the Individual eG Managers to Work with the eG SuperManager 47

2.9 Configuring an eG SuperManager to Manage the eG Managers in a Redundant Cluster 48

2.10 Starting and Stopping the eG SuperManager 48

2.11 Uninstalling the eG SuperManager 53

WORKING WITH THE EG SUPERMANAGER 55

3.1 An egsm User's View 56

3.2 The View Received by Other Users 61

3.3 How the eG SuperManager Supports Redundant Clusters? 63

3.4 Frequently Asked Questions on eG SuperManager and Redundant Clusters 64

CONCLUSIONS 66

Table of Figures

Figure 1.1: How the eG SuperManager works 2
Figure 1.2: Remote host accessing the SuperManager and individual managers 3
Figure 2.1: Welcome screen of the eG SuperManager Setup program 11
Figure 2.2: License agreement of the eG SuperManager 11
Figure 2.3: Setup enquiring the availability of JDK in the environment 12
Figure 2.4: Message box instructing the user to install JDK 12
Figure 2.5: Specifying the location of the Java home directory 12
Figure 2.6: Hostname and port number of the system on which the eG SuperManager will execute 13
Figure 2.7: Specifying the install directory of the eG SuperManager 14
Figure 2.8: Completion of SuperManager installation 14
Figure 2.9: Requesting for a certificate 17
Figure 2.10: Downloading the certificate 18
Figure 2.11: Requesting for a certificate 32
Figure 2.12: Downloading the certificate 33
Figure 2.13: SSL-enabling the start_sm script 38
Figure 2.14: SSL-enabling the start_sm script 45
Figure 2.15: Add/delete SuperManager 47
Figure 2.16: Providing the IP of the eG SuperManager to which this manager should report 47
Figure 2.17: Modifying/Deleting a SuperManager 48
Figure 2.18: Sample license 49
Figure 2.19: Starting the eG SuperManager 50
Figure 2.20: SSL-enabling the start_sm script 50
Figure 2.21: Stopping the eG SuperManager 51
Figure 2.22: Uninstalling the eG SuperManager 53
Figure 2.23: Selecting the option to remove the SuperManager 54
Figure 2.24: Confirming removal of the SuperManager 54
Figure 3.1: Alarms reported by the New York manager 55
Figure 3.2: Alarms reported by the Ohio manager 55
Figure 3.3: The **SuperManager** login 56
Figure 3.4: Consolidated list of alarms pertaining to the Ohio and New York managers 57
Figure 3.5: Additional alarm details 57
Figure 3.6: Viewing the Critical alarms alone 58
Figure 3.7: Viewing the acknowledgement descriptions submitted by other users 58
Figure 3.8: Acknowledging an alarm 58
Figure 3.9: The egsm user providing acknowledgement description 59
Figure 3.10: The egsm user's acknowledgement added to the history 59
Figure 3.11: The **SuperManager** Home 60
Figure 3.12: Changing the password of the *egsm* user 60
Figure 3.13: The AlarmViewer role 61
Figure 3.14: The details of an alarm of an AlarmViewer along with Feedback and History options 61
Figure 3.15: Alarms pertaining to both the *johns* 62
Figure 3.16: Home page of a user who is registered with only the Newyork manager 62
Figure 3.17: The home page of a 'Reporter-only' user 63
Figure 3.18: The eg_supermanager.ini file 65

1

Introduction

Large enterprises often have thousands of devices, servers, and applications that have to be managed, and a single eG management console may not have the capacity to handle the entire enterprise. To support such enterprises, multiple eG managers may be needed. However, if each of these managers operates independently, they may not provide a common view of the entire enterprise. Hence, it could be very cumbersome to have the IT staff of the enterprise login to different eG management consoles to get a complete view of the status of the target infrastructure.

A SuperManager is a manager of managers that provides a consolidated view of the status of the IT infrastructure that is being handled by different eG managers. The eG suite offers two options for configuring a super manager. The **eG SuperManager** is a 100% web-based component of the eG suite that provides a consolidated view across disparate eG manager. On the other hand, an administrator can also use the Computer Associates Network and System Management (NSM) product as a super manager. The following sections however, discuss the **eG SuperManager** option alone.

To configure an eG SuperManager, you first need to ensure that the eG SuperManager license is enabled for your eG installation.

There are three main reasons why a SuperManager is necessary in an enterprise environment.

- **Scale of operation**: Large enterprises may comprise of thousands of devices, servers, and applications, and a single eG manager may not be sufficient to manage this environment. Hence, a multi-level architecture with individual eG managers reporting to a SuperManager is necessary.
- **Autonomous domains**: Large enterprises often comprise of multiple domains, which are managed autonomously. Each domain may require a separate eG manager, but there could be a common support team that is responsible for first level support for all the domains. The same situation could occur in a managed services environment as well - the manager service provider (MSP) is responsible for supporting different customer infrastructures (possibly from a central location) but each customer may prefer to have their separate management consoles for administration. In this case, the support organization/MSP will need a consolidated view of the status of the infrastructure across the different domains.
- **Geographically distributed environments**: Large enterprises will span multiple geographical locations and multiple eG managers – one per location – may be preferred so as to reduce the bandwidth involved in communicating all the measurements to a central console. In this case as well, a common support organization may be involved, thereby necessitating a management console that consolidates the status across the individual eG managers.

The **eG SuperManager** is a central entity that integrates with multiple eG managers and provides a consolidated view of the infrastructure being monitored by each of the managers. Figure 1.1 depicts how the eG SuperManager works.

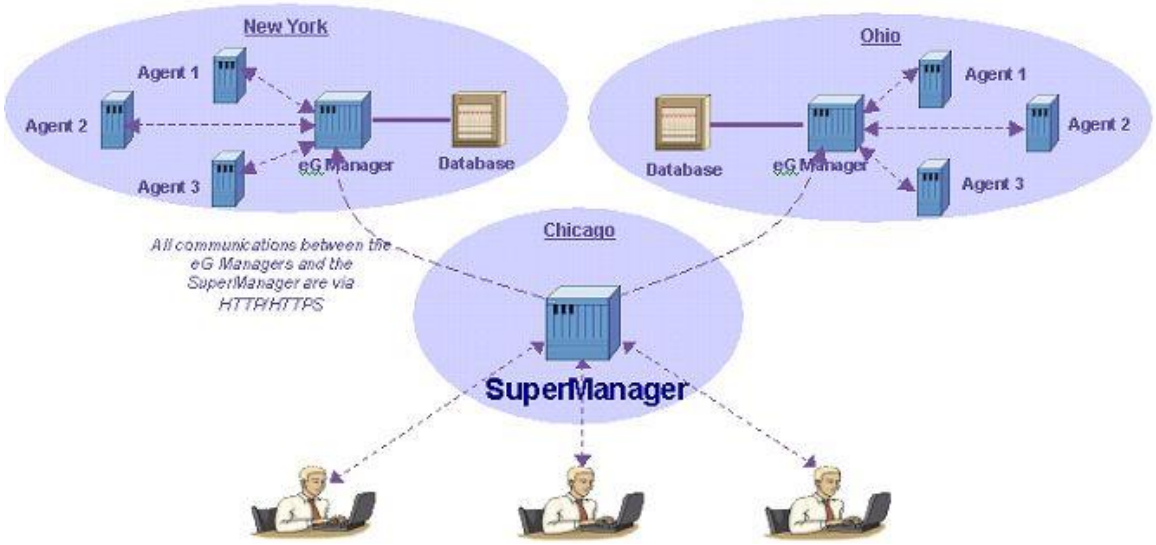


Figure 1.1: How the eG SuperManager works

The salient features of the eG SuperManager are discussed hereunder:

- **HTTP/HTTPS communication between the eG SuperManager and eG Manager:**

  When it is configured, the eG SuperManager is provided details of the individual managers that are to report to it. Periodically, the eG SuperManager polls the individual eG managers at a pre-defined interval, collects status information from them, consolidates the information, and provides it to users. All the communications between the eG SuperManager and the individual managers happen over HTTP/HTTPS.

  The SuperManager initiates all communications with each of eG managers in its control. Therefore, if a firewall exists between the managers and the SuperManager, this firewall should be configured to allow connections initiated by the SuperManager to the individual managers (see Figure 1.2).

  Also, when a user who logs into the eG SuperManager interface clicks on an alarm in the **CURRENT ALARMS** window, or the manager name in the home page, he/she would be redirected to the corresponding manager's console wherein further diagnosis can be performed. To facilitate this, network connectivity should be provided such that all users have direct connectivity not only to the SuperManager but also to the individual managers (see Figure 1.2).

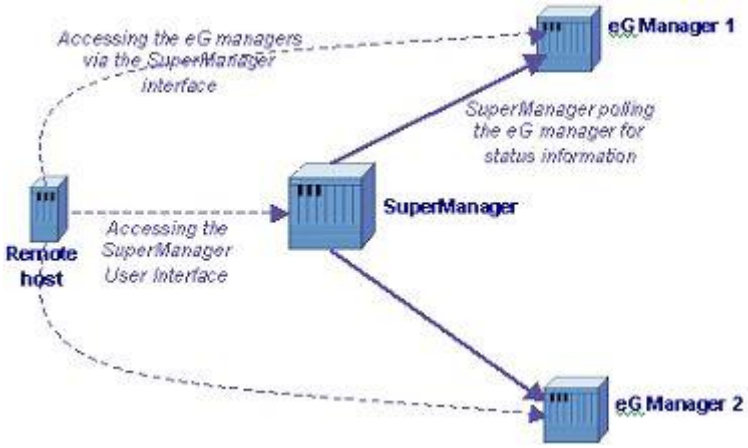


Figure 1.2: Remote host accessing the SuperManager and individual managers

- **Manages redundant clusters:**

  If one of the managers that it operates with is part of a redundant cluster, the eG SuperManager will first try to communicate with the primary manager of the cluster. If the primary manager is not reachable, the eG SuperManager will automatically try to communicate with the secondary manager of the cluster.

- **No local data storage:**

  The eG SuperManager itself does not have a database – hence, all the measurements are stored on the individual managers, not the SuperManager. In a managed services environment, where the SuperManager may be hosted in the MSP premises, but the individual managers are at the customer sites, this architecture provides data security by ensuring that measurement data regarding the customer networks is not stored at the MSP premises.

  Although it does not have a local database, the eG SuperManager can still support user logins. Users can login directly to the eG SuperManager from their web browser. When a user logs in, the SuperManager contacts all the managers reporting to it, to find out which of these managers allows access to the user. A user can login to the SuperManager as long as he/she is a registered user of one or more of the managers reporting to the SuperManager.

  If a user is registered with one of the managers alone, the view that the user sees is consistent with what he/she would see if they were to login to the manager to which they were registered directly. On the other hand, if the user is registered with more than one manager, the SuperManager produces a consolidated view based on the responses from the individual managers.

  Note:

  In its current implementation, the eG SuperManager mainly serves as a central console where users can get a consolidated view of alerts from different managers. Any configuration changes necessary have to be performed directly on the individual managers.

- **Supports eG managers with heterogeneous configurations:**

  The individual managers reporting to a SuperManager can have heterogeneous configurations. For example, one of the managers can be running on Microsoft Windows, whereas another could be running on Sun Solaris. Likewise, the databases of the individual managers can also be different – for example, one of the managers may use Microsoft SQL server as the database, while another may use Oracle as the database.

- **Supports multiple languages:**

  The alarm and dashboard displays in the eG SuperManager console are language-customizable! The language preferences set by the users registered with each of the managers reporting to the SuperManager, will affect the alarm and dashboard displays in the eG SuperManager console as well. For example, say that a user to an eG manager at *Spain* chooses *Spanish* as the language in which he/she would prefer to view performance information reported to the manager. When such a user logs into the SuperManager that manages the *Spain* manager, the alarms and performance details pertaining to that user will be displayed in the eG SuperManager console in the *Spanish* language only. Moreover, if the managers reporting to a SuperManager are double-byte enabled and support double-byte languages such as Chinese or Japanese, then the SuperManager interface too will display data in double-byte.

- **No change in agent architecture:**

  In this architecture, since the agents continue to report to their respective eG managers, there is no change needed in the eG agent architecture.

The eG SuperManager can be applied in the following environments:

- In a large enterprise with thousands of devices, servers, and applications, the eG SuperManager ensures that the eG suite can scale to handle such an environment;
- In a distributed network, eG managers can be installed in each domain of the network and managed autonomously; At the same time, using the eG SuperManager, a single integrated console can be provided for the support team that is responsible for monitoring each of the domains;
- In a managed service environment, each customer network could have an independent eG manager, but with an eG SuperManager installed, the managed service provider could get an integrated view of the status of all the customer networks that they are responsible for managing.

The chapters to come will elaborately discuss how to install and configure the eG SuperManager.

2

Installing and Configuring the eG SuperManager

The procedure for installing the eG SuperManager differs according to the operating system used on the server on which the SuperManager is to be installed. The eG SuperManager is available for Solaris, Red Hat Linux, and Windows operating system environments. This section describes the steps involved in installing and configuring the eG SuperManager on each of the above-mentioned operating system environments.

Prior to installing the eG SuperManager, ensure that the following pre-requisites are in place:

Prerequistes for Unix Installations of the eG SuperManager:

a. Solaris 10 (or higher) or Red Hat Enterprise Linux 5 (or higher)

b. Minimum requirement: JDK 1.3 (and above) Recommended: JDK 1.5 (or its variants)

c. 1 GB disk space with 1 GB RAM

d. A valid eG SuperManager license

Prerequisites for Windows Installations of the eG SuperManager:

a. Minimum requirement: JDK 1.3 (and above) Recommended: JDK 1.5 (or its variants)

b. Windows 2008 server (OR) Windows 2012 server (OR) Windows Vista

c. Only systems with a static IP address (i.e. no DHCP address) should be used for installing the eG SuperManager

d. 1 GB disk space with 1 GB RAM

e. Internet Explorer Version 6 or above (OR) Mozilla Firefox Version 2.0 or above

f. Internet Explorer 10 or higher, Mozilla Firefox v25 or higher, or Chrome as the browser

g. A valid eG SuperManager license

Note:

The eG SuperManager and the eG Manager cannot exist on the same host.

2.1 Installing the eG SuperManager on Linux Environments

For installation on Linux systems, the eG SuperManager is provided as a tar file named **egsm_linux.tar**. The installation process is driven by a shell script named **iSm_linux**.

Note:

The eG manager software has to be installed from a super-user account.

The steps involved in installing the eG SuperManager on a Linux system are as follows:

1. Copy the **egsm_linux.tar** and **iSm_linux** files to a directory, and then execute the **iSm_linux** command to begin the installation process.

2. Next, the installation process attempts to create the eG user account. For this process to continue, specify the user account to be used for executing the eG SuperManager. The default is **egurkha.**

```
This script will install the eG SuperManager.

Enter the name of the eG user [egurkha]: egurkha
```

3. If the user account specified already exists, then a message to that effect will be displayed.

```
User egurkha already exists; continuing ...
```

4. Next, you will be prompted to configure the destination directory of the eG SuperManager.

```
Enter the directory in which the eG SuperManager should be installed
[/opt]:
```

5. Once the install directory is specified, installation will begin.

6. Finally, you will be requested to confirm whether the eG SuperManager needs to be auto-restarted on system boot up. To confirm auto-restart, specify **y**.

```
Would you like the eG SuperManager to auto-restart on system boot-up? y/n
[n] :y
```

7. The following message will appear upon successful installation of the eG SuperManager.

```
************************************************************
```

```
The eG SuperManager has been successfully installed!

Please login as egurkha and run the script
/opt/egurkha/bin/setup_sm to configure the SuperManager.

The licensing terms for eG products are mentioned in the file
/opt/egurkha/license_agreement.
PLEASE READ THIS FILE BEFORE PROCEEDING FURTHER.

Note that the eG SuperManager requires JDK 1.5 or higher.
**************************************************************
```

2.2 Installing the eG SuperManager on Solaris Environments

The eG SuperManager software is provided as a standard Solaris package (named **eGsm**) that can be installed using the **pkgadd** utility. Using the **pkgrm** utility, the eG SuperManager can be uninstalled.

The steps involved in installing the eG SuperManager are as follows:

1. To start the installation process, execute the following command from the command prompt:

   pkgadd -d <path to the eGsm package>

2. The list of available packages will be displayed next. Enter **all** to install all the packages related to the eG SuperManager.

```
1  eGsm     eG SuperManager
                  (sparc) version 4.2.1

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]:
```

3. We recommend that the eG SuperManager be executed by a special user account that is exclusively created for this purpose. Next, the installation process attempts to create the eG user account:

```
Processing package instance <eGsm> from </tmp/eGsm>
eG SuperManager
(sparc) version 4.2.1
eG Innovations, Inc.

Enter the name of the eG user [egurkha]:
```

   Specify the user account to be used for installing the eG SuperManager. The default value is **"egurkha"**.

```
Enter the group to which the eG user is to be associated [egurkha]:
```

   Also, specify the group to which this user account should be associated. The default value taken is **"egurkha".**

Note:

a. An existing user and/or group can be specified during this step.
b. The installation process checks for the existence of the user and/or group, and creates a new user or group only if necessary.

4. Next, the installation process prompts the user to choose the path of the directory in which the eG SuperManager is to reside. If possible, the eG manager should be installed in the **/opt** directory. If space considerations preclude this, the eG manager can be installed in any other directory on the system. At the end of the installation process, a symbolic link is created to link the installation directory to the **/opt/egurkha** directory.

```
Enter the directory where the eG SuperManager should be installed [/opt]:
```

5. `Would you like the eG SuperManager to auto-restart on system boot-up? y/n [n]`

   The install process will now request the user to confirm installation of the auto-restart feature. This feature will enable the eG SuperManager to start automatically every time the system hosting it reboots. Now, press **y** to install the auto-restart feature, or **n** to proceed without installing the same.

6. A message indicating that installing the package requires super user permission appears and now the user needs to inform whether the process can proceed. If the user does not have the super user permissions, he/she needs to login as the super user before proceeding with the installation.

```
This package contains scripts which will be executed with super-user
permission during the process of installing this package.
Do you want to continue with the installation of <eGsm> [y,n,?] y
```

7. Upon successful completion of the installation process, the following message will be displayed:

```
**********************************************************************
The eG SuperManager has been successfully installed!
Please login as <user name> and run the script
                /opt/egurkha/bin/setup_sm
to configure the eG SuperManager.
The licensing terms for eG products are mentioned in the file:
                        /opt/egurkha/license_agreement.
PLEASE READ THIS FILE BEFORE PROCEEDING.

Please note that the eG SuperManager requires JDK 1.3.1
**********************************************************************
Installation of <eGsm> was successful.
```

2.3 Configuring the eG SuperManager on Unix Environments

After the installation, the eG SuperManager needs to be configured for proper functioning. The process of configuring is three-fold:

- Firstly, configure the IP/hostname and port of the eG SuperManager's host;
- Secondly, provide the eG SuperManager with the details of the individual eG managers from which it needs to collect status information;
- Finally, use the eG administrative interface of the individual managers to configure the eG SuperManager to which these managers need to report status information;

The steps involved in configuring the IP and port of the eG SuperManager are the same for Solaris and Linux systems, and are listed below:

1. For configuring the eG SuperManager, first login as the eG user.

8. At the prompt, issue the following command: **/opt/egurkha/bin/setup_sm**. The following message will then appear:

```
************************************************************
Configuring the eG SuperManager ...
The licensing terms for eG products are mentioned in the file
/opt/egurkha/license_agreement. PLEASE READ THIS FILE BEFORE
PROCEEDING FURTHER.
************************************************************
Please indicate if you accept the eG licensing terms y/n [n] y
```

Press **y** for accepting the terms. Once the licensing terms have been accepted, the configuration process prompts the user to enter the full hostname or the IP address of the host on which the eG SuperManager is being configured. Pressing **n** on the other hand, indicates non-acceptance of the licensing terms and terminates the configuration process.

9. The following prompt will appear if **y** is specified at step 2. Provide the IP/host name of the SuperManager host, and the port at which the SuperManager listens.

```
Port configuration for the eG SuperManager
**************************************************
Enter the full hostname (or IP address) of this host: 192.168.10.12
Enter the port number for the eG SuperManager [8088]: 8080
```

10. Next, confirm whether SSL is to be enabled for the SuperManager by specifying **y** or **n**, as the case may be.

```
Do you want the eG SuperManager to be SSL enabled y/n [n]?
```

11. The next step involves configuration of the TCP ports that the eG SuperManager should use. At this step, enter the following information:

    a. the Tomcat server port for the eG SuperManager [default is **8089**]
    b. the JK connector port for the eG SuperManager [default is **8090**]

```
*****************************************************************************
 The eG SuperManager uses a Tomcat server for generating dynamic content.
 The Tomcat server requires two TCP ports for its internal communication.
*****************************************************************************

Please enter the Tomcat server port for the eG SuperManager [8089]:
Please enter the JK connector port for the eG SuperManager [8090]:
```

12. To enable the setup process to configure the eG SuperManager's execution environment to execute Java programs, next specify the Java home directory (say **/opt/java1.3.1_06**).

```
Please enter the location of your Java home directory [/usr/java/jdk1.3.1]:
/usr/java/jdk1.3.1_06
```

13. If the configuration process cannot locate the specified Java home directory, it returns the following error message:

```
The setup process failed to locate /opt/java1.3.1_06
```

Then, the configuration process looks for the Java library (lib) subdirectory of the Java home directory. Within this directory, the setup process checks for a specific file called **tools.jar**. For example, if the user specifies **/opt/java1.3** as the Java home directory, the configuration process will look for the file **/opt/java1.3/lib/tools.jar**. If the process fails to locate the Java lib directory, it will explicitly ask the user to specify the Java lib directory and also display the following message:

```
The setup process failed to locate /opt/java1.3.1_06/lib/tools.jar...
```

14. Once the Java lib directory has been located, the setup process will look for a Java fonts directory within the Java home directory. For example, if the Java home directory is set to **/opt/java1.3.1_06**, the setup process will check for the fonts directory in the location **/opt/java1.3.1_06/jre/lib/fonts**. If this directory is not found, the following error message appears and the setup process will prompt the user to specify the Java fonts directory.

```
The setup process failed to locate /opt/java1.3.1_06/jre/lib/fonts...
```

15. Upon completion of the configuration, the following message will appear:

```
*****************************************************************************
If there were no errors, the eG SuperManager has been configured.

Please use the commands /opt/egurkha/bin/start_sm and
/opt/egurkha/bin/stop_sm to start and stop the eG SuperManager.

You will need a valid license to start the eG SuperManager.
Please contact support@eginnovations.com to request for a license.
*****************************************************************************
```

2.4 Installing and Configuring the eG SuperManager on Windows Environments

1. To start the installation process on a Windows 2008/2012 server, run the **eGSuperManager2008.exe** file. The Welcome screen (see Figure 2.1) of the eG SuperManager Setup program appears. Clicking on the **Next >** button at the bottom of this screen takes the user to the next step of the setup.

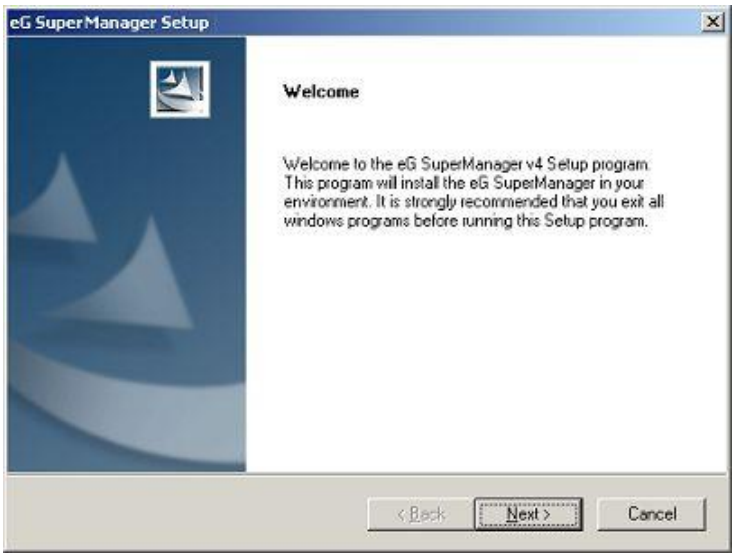


Figure 2.1: Welcome screen of the eG SuperManager Setup program

2. Now, the user can view the eG license agreement (see Figure 2.2). Also, the Setup program seeks the confirmation of the user regarding his/her acceptance of the terms and conditions of the license agreement. It is mandatory that the user accepts the license agreement to proceed with the setup. Click on the **Yes** button at the bottom of the screen to accept and proceed with the setup process.

Figure 2.2: License agreement of the eG SuperManager

3. The Setup program prompts the user to specify if his/her environment contains the required JDK version (see Figure 2.3).

Figure 2.3: Setup enquiring the availability of JDK in the environment

4. If this version of JDK is not available, the following message box appears asking the user to install the required JDK.

Figure 2.4: Message box instructing the user to install JDK

5. If the necessary JDK is already available in the environment, specify the Java home directory to enable the setup process to configure the eG user's execution environment to execute Java programs as in Figure 2.5. The user can also use the **Browse** button to select the location of the Java home directory.

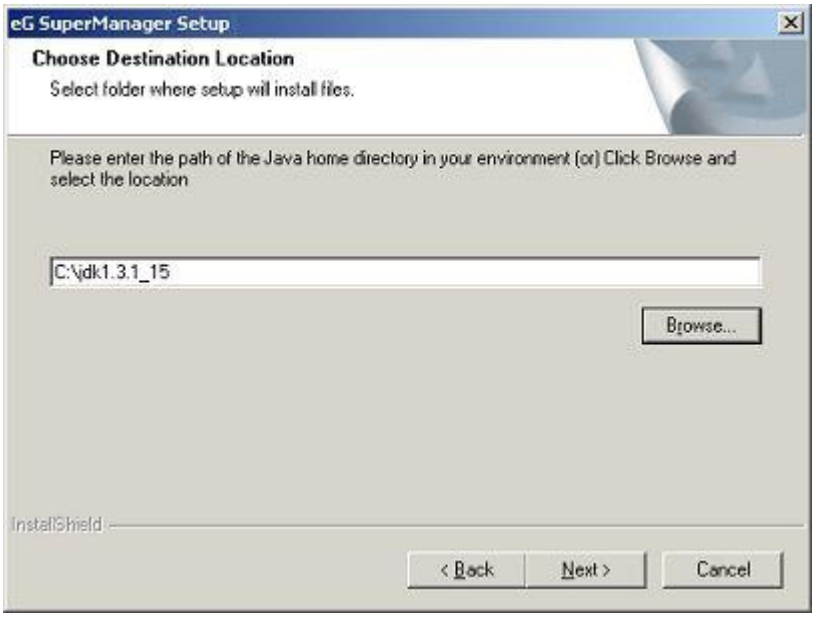


Figure 2.5: Specifying the location of the Java home directory

6. The setup process now requires the hostname and port number of the host on which the eG SuperManager is being configured (see Figure 2.6). The default port is **8088**. If the domain name service is used in the target environment, use the full hostname. Otherwise, specify the IP address.

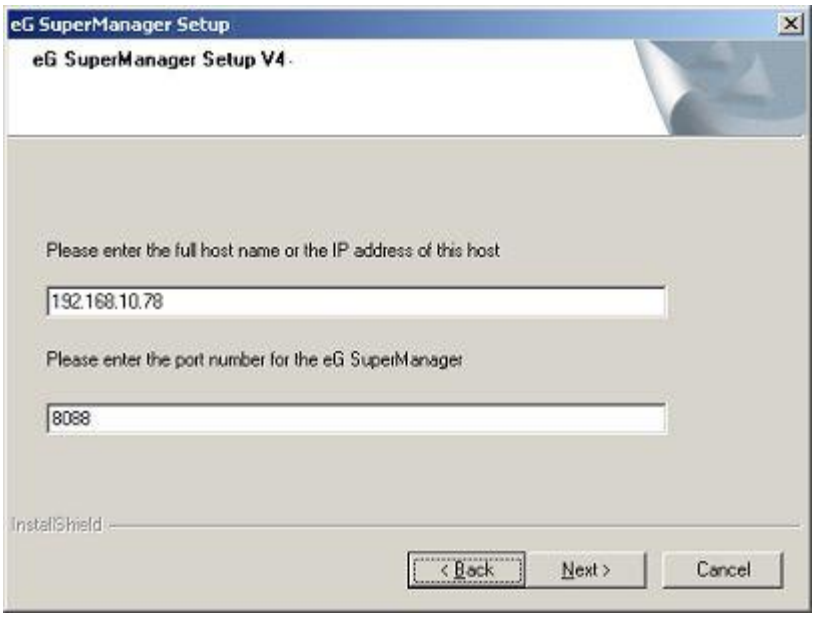


Figure 2.6: Hostname and port number of the system on which the eG SuperManager will execute

Note:

While specifying the host name/IP address of the SuperManager, please take care of the following aspects:

a. If the host name is provided when installing the SuperManager, use this name (and not the IP address) for accessing the user interface via the web browser.

b. If the host name is provided, make sure that forward and reverse lookups for this name are enabled via the DNS service in the target environment.

7. Next, specify the install directory of the eG SuperManager. The default location is **C:** (see Figure 2.7).

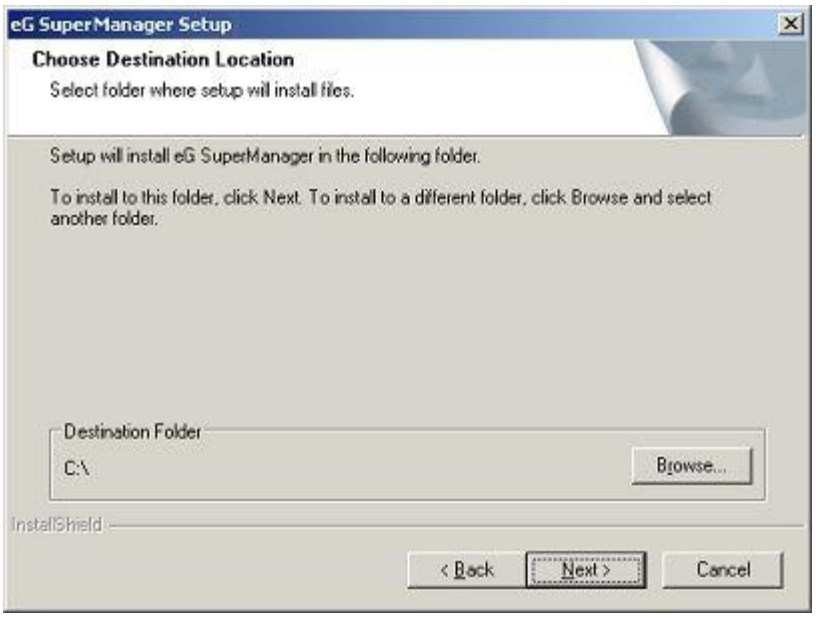


Figure 2.7: Specifying the install directory of the eG SuperManager

8. Once installation completes, Figure 2.8 will appear informing you of the successful installation of the eG SuperManager. It is recommended that you choose to restart your system before proceeding to start the eG SuperManager.

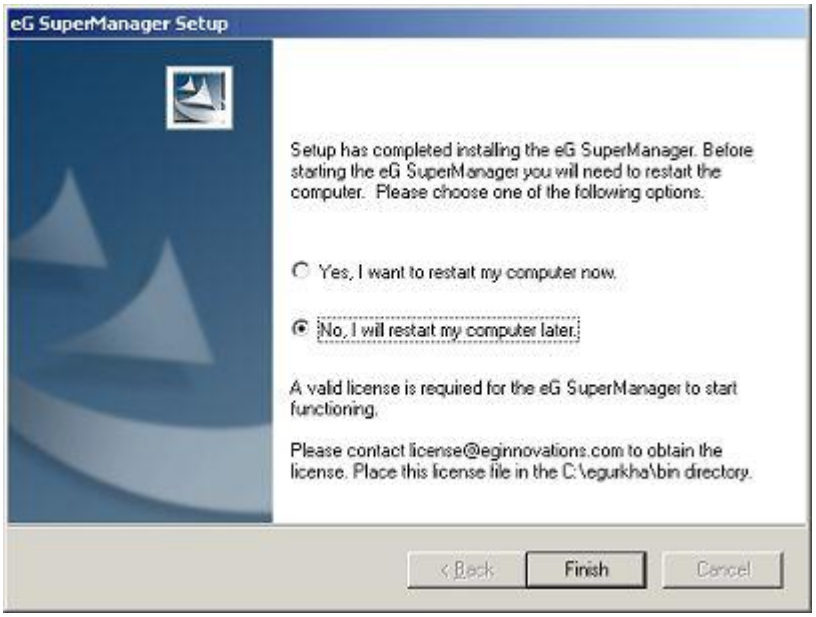


Figure 2.8: Completion of SuperManager installation

9. Finally, click the **Finish** button in Figure 2.8.

2.5 SSL-Enabling the eG SuperManager on Unix

To SSL-enable the eG SuperManager on Unix, you have the following options:

(a) You can obtain a signed SSL certificate from an internal certifying authority (eg., Microsoft Active Directory Certificate Services) and use this certificate to SSL-enable the eG SuperManager, (OR)

(b) You can obtain a signed certificate from a valid, external certifying authority (eg., Verisign) and use this certificate to SSL-enable the eG SuperManager

If you go with option (a), use the procedure detailed in Section 2.5.1. If you pick option (b), use the procedure detailed in 2.5.2.

2.5.1 SSL-Enabling the eG SuperManager Using a Certificate Signed by an Internal CA

For this, follow the steps given below:

- Generate the Keystore file
- Generate a certificate request
- Submit the certificate request to the internal certificate Authority (CA) and obtain a certificate
- Import the certificate into a keystore
- Configure Tomcat for using the keystore file

Each of these steps has been discussed in the sections that follow.

2.5.1.1 Generating the Keystore File

The keystore file stores the details of the **certificates** necessary to make the protocol secure. Certificates contain the information pertaining to the source of the application data, and helps validate the source. To generate the keystore, use the **keytool** command. For this purpose, login to the eG SuperManager and go to the shell prompt. Set the **JAVA_HOME** path if it is not done already. Then, execute the following commands, one after another:

cd $JAVA_HOME/bin

keytool -genkey -alias **egitlab1** *-keyalg* **RSA -***keypass* **mykey** *-keystore* **<Filename>***.keystore -storepass* **mykey** *-keysize* **2048** *-validity* **1095**

The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

- **-alias** : an alias name for the certificate being generated
- -**keypass** : a password used to protect the key that is generated; **ensure that you provide the same values for -keypass and -storepass.**
- **-keyalg** : specifies the algorithm that is used to generate the keys. The options are as follows:
  - **DSA** : Digital Signature Algorithm
  - **RSA :** An algorithm used for publick-key cryptography

- **-keystore** : the *keytool* command stores the generated key in a *.keystore* file; provide a name for this file as input to the **-keystore** command
- **-keysize** : the size of the key that is generated; the default key size is 1024 bits - the key size must be in the range 512 bits - 1024 bits
- **-validity** : indicates the number of days for which the key/certicate will be valid - 1095 days refer to 3 years.

The command, upon execution, will request the following inputs:

```
What is your first and last name?
[Unknown]: <Type the eG SuperManager's Fully qualified domain name here>
What is the name of your organizational unit?
[Unknown]: United States
What is the name of your organization?
[Unknown]: eG Innovations Inc
What is the name of your City or Locality?
[Unknown]: Bridge Water
What is the name of your State or Province?
[Unknown]: New Jersey
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=eG Innovations Inc, OU=United States, O=eG Innovations Inc, L=Bridge
Water, ST=New Jersey, C=US correct?
[no]: yes
```

When requested for the **first and last name**, indicate the *fully qualified domain name* using which you will be accessing the eG SuperManager. For instance, if the eG SuperManager is to be accessed as *http://egsm.eginnovations.*com, where *egsm.eginnovations.com* is the fully qualified domain name of the eG SuperManager, then specify *egsm.eginnovations.com* here.

Once all the required inputs are provided, a .keystore file will be generated in the **<JAVA_HOME_DIR>/bin** directory with the **<Filename>** you had provided while issuing the command.

2.5.1.2 Generating a Certificate Request

Once a keystore file is generated, proceed to request for a certificate from an internal certifying authority. The procedure for this is as follows:

1. Login to the eG SuperManager and go to the shell prompt.
2. Set the **JAVA_HOME** path if it is not done already.
3. Execute the following commands one after another:

   cd $JAVA_HOME/bin

   keytool -certreq -alias **egitlab1** *-keyalg* **RSA** *-file* **<Name_of_the_text_file>** *-keypass* **mykey** *-keystore* **<filename>.keystore** *–storepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name that you provided while generating the keystore file (see Section 2.5.1.1 of this document).**

- **-keyalg** : specifies the algorithm that was used to generate the keys; this can be **RSA** or **DSA**, depending upon **which algorithm was used for key generation in the procedure detailed in Section 2.5.1.1**
- **-file** : Provide a name for the text file to which the certificate request will be saved.
- -**keypass** : the password used to protect the key that was generated; **make sure that you provide the same password that you provided while generating the keystore file (see Section 2.5.1.1 of this document).** Also, note that **-storepass** and **-keypass** should be the same.
- **-keystore** : Provide the name of the *keystore* file in which the key has been stored; **specify the same file name that you used to store the key (see Section 2.5.1.1 of this document)**.

4. If this command executes successfully, then a certificate request will be generated and automatically stored in the text file you specified in step 2 above.

2.5.1.3 Obtaining the Certificate from the Internal CA

1. The first step towards obtaining a certificate is to submit the certificate request to the internal CA. For this connect to the Certificate server of the internal CA and select the option to submit the certificate. For instance, if you are using Microsoft Active Directory Certificate Services to request for a self-signed certificate, then, you need to connect to **http://<YourWebServerName>/certsrv**, and then pick the option to submit the certificate. Figure 2.9 will then appear.

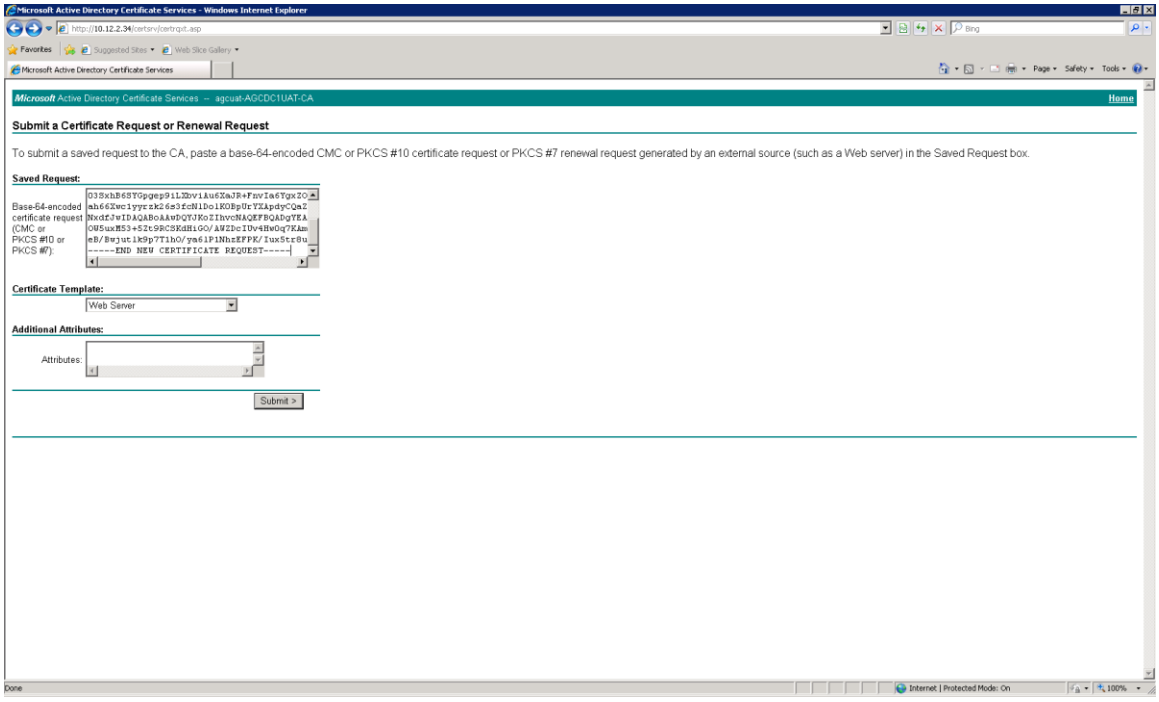


Figure 2.9: Requesting for a certificate

2. Open the text file containing the certificate request (which was created using the procedure detailed in Section 2.6.1.2 above), copy the contents of the file, and then paste it to the text area of the **Base 64-encoded certificate request** text box of Figure 2.9. Then, click the **Submit** button.

3. The certificate will thus be generated. Download the certificate.

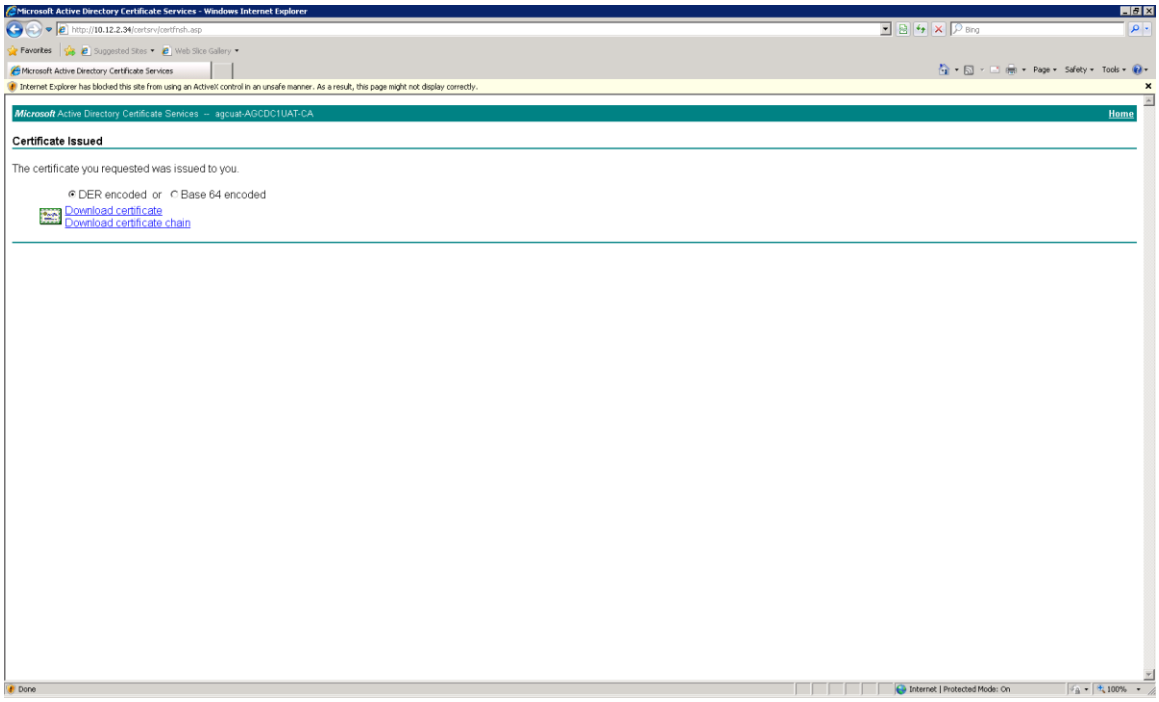


Figure 2.10: Downloading the certificate

2.5.1.4 Importing the Certificates into the Keystore File

The downloaded certificate can be in one of the following forms:

- Can be a single, combined certificate
- Can be accompanied by a certificate chain
- Can be in a PEM format

The procedure for importing certificates differs based on the format of the downloaded certificate. These procedures have been detailed in the sub-sections below.

Importing a Combined Certificate into the Keystore File

In this case, follow the steps below to import the certificate into the keystore file:

1. Set the **JAVA_HOME** path if it is not done already.
2. At the command prompt, execute the following commands, one after another:

   cd $JAVA_HOME/bin

   keytool -import –trustcacerts -alias **egitlab1** *-file* **<Name_of_the_domain_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

- **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name you provided when generating the keystore (see Section 2.5.1.1) .**
- **-file**: the name of the domain certificate that you want to import
- **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.5.1.1 above.

Importing a Signed Certificate and the Certificate Chain into the Keystore File

Digital certificates are verified using a chain of trust. The trust anchor for the digital certificate is the Root Certificate Authority (CA). The Certificate Hierarchy is a structure of certificates that allows individuals to verify the validity of a certificate's issuer. Certificates are issued and signed by certificates that reside higher in the certificate hierarchy, so the validity and trustworthiness of a given certificate is determined by the corresponding validity of the certificate that signed it.

The Chain of Trust of a Certificate Chain is an ordered list of certificates, containing an end-user subscriber certificate and intermediate certificates (that represents the Intermediate CA), that enables the receiver to verify that the sender and all intermediate certificates are trustworthy.

A certificate chain will therefore consist of multiple certificates. Before importing each of these certificates, **you will have to understand the hierarchy of the certificates**. To know which is the root and which is the intermediate certificate, refer to the web site of the certificate authority. Then, set the **JAVA_HOME** path if it is not done already. Next, follow the steps below:

1. First, import the Root certificate. For this, execute the following commands, one after another in the command prompt:

   cd $JAVA_HOME/bin

   keytool -import –trustcacerts -alias **rootcert** *-file* **<Name_of_the_root_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore** *–keypass* **mykey** *–storepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide a unique alias name for the root certificate.**
   - **-file**: the name of the root certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.5.1.1 above.
   - **-keypass** and **–storepass** : Provide the same **keypass** and **storepass** that you specified when generating the keystore; refer to Section 2.5.1.1 above for details.

2. Next, import each of the intermediate certificates, one after another, using the following command:

keytool -import –trustcacerts -alias **intercert1** *-file* **<Name_of_the_intermediate_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore** *–keypass* **mykey** *–storepass* **mykey**

The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

- **-alias** : the alias name of the certificate being requested; **make sure that you provide a unique alias name for every intermediate certificate.**
- **-file**: the name of the intermediate certificate that you want to import
- **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.5.1.1 above.
- **-keypass** and **–storepass** : Provide the same **keypass** and **storepass** that you specified when generating the keystore; refer to Section 2.5.1.1 above for details.

3. Finally, import the entity/domain certificate into the keystore by issuing the following command:

   keytool -import –trustcacerts -alias **egitlab1** *-file* **<Name_of_the_domain_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name you provided when generating the keystore (see** Section 2.5.1.1) .
   - **-file**: the name of the domain certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.5.1.1 above.

If the domain certificate import command throws an error for any reason, it could be because, all related certificates may not have been imported. Check the web site of the CA for more details.

Importing a Certificate that is in the PEM Format

PEM is a container format that may include just the public certificate (such as with Apache installs, and CA certificate files */etc/ssl/certs*), or may include an entire certificate chain including public key, private key, and root certificates, or may only contain a certificate and a private key.

If the certificate you downloaded is in the PEM format and includes only a certificate file and a private key file, then follow the steps below to import that certificate into a keystore file.

1. Run the following command from the command prompt to export the certificate and private key file into the pkcs12 format:

   openssl pkcs12 -export -in **certificate.crt** *-inkey* **private.key** *-certfile* **certificate.crt** *-name* **'My certificate"** *-out* **keystore***.p12*

The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

- **-in** : the name of the certificate that is included in the PEM container
- **-inkey**: the name of the private key file the PEM container consists of
- **-certfile** : the name of the certificate that is included in the PEM container
- **-name** : Provide a **unique name for the certificate file** that is being exported.
- -**out** : Specify the name of the keystore file to which the certificate and private key are to be exported. **The keystore file can have any name of your choice.**

2. Next, you need to convert the keystore file, which is currently in the pkcs12 format, into the Java keystore (i.e., JKS) format. For this, issue the following command at the command prompt:

   keytool -importkeystore –alias **egitlab1** *-deststorepass* **mykey** *-destkeypass* **mykey** *-destkeystore* **keystore,jks** *-srckeystore* **keystore.pk12** *-srcstoretype PKCS12 -srcstorepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the aliasname of the certificate being requested; **make sure that you provide the same alias name that you specified in** Section 2.5.1.1 **of this document.**
   - **-deststorepass** : this refers to the **storepass** of the destination keystore file – i.e., the keystore file in the JKS format. **The storepass of the destination keystore should be the same as the storepass of the source keystore.**
   - **-destkeypass** : this refers to the **keypass** of the destination keystore file - i.e., the keystore file in the JKS format. **The storepass and keypass of the destination keystore file should be the same.**
   - **-destkeystore**: the name of the destination keystore file – i.e., the keystore file in the JKS format.
   - **-srckeystore** : the name of the destination keystore file – i.e., the keystore file in the PKCS12 format.
   - **-srcstorepass** : The **storepass** of the source keystore file – i.e., the keystore file in the PKCS12 firmat. **make sure that you provide the same storepass you specified in Section** 2.5.1.1 **of this document**

2.5.1.5 Configuring Tomcat for Using the Keystore File

The eG SuperManager on Unix uses Tomcat as the web server. Therefore, to SSL-enable the eG SuperManager, you need to configure the **server.xml** file of Tomcat with the name and full path to the keystore file which was created earlier.

For this purpose, do the following:

1. Stop the eG SuperManager.
2. Edit the **server.xml** file in the **<CATALINA_HOME>/conf** directory.
3. In the file, search for the XML block where the SSL Coyote HTTP connector on port 8443 is defined. If

this block is commented, it indicates that the eG SuperManager is not SSL-enabled and is hence listening on an HTTP port only. To SSL-enable the eG SuperManager, first uncomment this block as indicated below:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
    <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
               port=8443 minProcessors="5" maxProcessors="75"
               enableLookups="true"
          acceptCount="10" debug="0" scheme="https" secure="true"
               useURIValidationHack="false">
      <Factory
className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
               clientAuth="false" protocol="TLS" />
    </Connector>
```

4. Then, proceed to make the changes indicated in **Bold** below in the SSL XML block:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->

<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
port="<eG_SuperManager_Port>" minProcessors="5" maxProcessors="75"
enableLookups="true"
 acceptCount="10" debug="0" scheme="https" secure="true"
               useURIValidationHack="false">
     <Factory
className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
               clientAuth="false" protocol="TLS"
keystoreFile="/opt/egurkha/manager/tomcat/conf/<filename>.keystore"  keystorePass="mykey"/>
    </Connector>
```

Set the *port* parameter in the XML block to reflect the port number that you have configured for the eG SuperManager. Also, note that two new parameters, namely - **keystoreFile** and **keystorePass** - have been inserted into the SSL block. While the **keystoreFile** parameter has to be set to the full path to the **.keystore** file that you generated earlier, the **keystorePass** parameter should be set to the keystore password that you specified while issuing the **keytool** command.

5. With that change, the eG SuperManager on Linux has acquired the capability to listen on two ports - the SSL port and the non-SSL port. To configure the eG SuperManager to listen only on the SSL port, simply comment that section of the **server.xml** file where the non-SSL Coyote HTTP connector on port 8081 has been defined, as indicated below:

```
<!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8081 -->
    <!-- <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
                    port="<eG_SuperManager_Port>" minProcessors="32"
maxProcessors="256"
               enableLookups="true" redirectPort="8443"
               acceptCount="10" debug="0" connectionTimeout="20000"
               useURIValidationHack="false"  URIEncoding="UTF-8"/> -->
```

6. Save the file.
7. Finally, start the eG SuperManager.

2.5.2 SSL-Enabling the eG SuperManager Using a Signed Certificate Obtained from a Valid Certifying Authority

In highly secure environments, especially where the eG SuperManager is to be frequently accessed via the public internet, you can you can obtain a valid certificate from a certificate authority and use that certificate to SSL-enable the eG SuperManager.

The broad steps to be followed to achieve this are as follows:

1. Generating the keystore file
2. Generating a certificate request
3. Submitting the certificate request to the Certificate Authority (CA) and obtaining a certificate
4. Importing the certificate into a keystore
5. Configuring Tomcat for using the keystore file

The sub-sections below elaborate on each of these steps.

2.5.2.1 Generating a Keystore File

The keystore file stores the details of the **certificates** necessary to make the protocol secure. Certificates contain the information pertaining to the source of the application data, and helps validate the source. To generate the keystore, use the **keytool** command. For this purpose, login to the eG SuperManager and go to the shell prompt. Set the **JAVA_HOME** path if it is not done already. Then, execute the following commands, one after another:

cd $JAVA_HOME/bin

keytool -genkey -alias **egitlab1** *-keyalg* **RSA -***keypass* **mykey** *-keystore* **<Filename>***.keystore -storepass* **mykey** *-keysize* **2048** *-validity* **1095**

The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

- **-alias** : an alias name for the certificate being generated
- -**keypass** : a password used to protect the key that is generated; **ensure that you provide the same values for -keypass and -storepass.**
- **-keyalg** : specifies the algorithm that is used to generate the keys. The options are as follows:
    - **DSA** : Digital Signature Algorithm
    - **RSA :** An algorithm used for publick-key cryptography
- **-keystore** : the *keytool* command stores the generated key in a *.keystore* file; provide a name for this file as input to the **-keystore** command
- **-keysize** : the size of the key that is generated; the default key size is 1024 bits - the key size must be in the range 512 bits - 1024 bits
- **-validity** : indicates the number of days for which the key/certicate will be valid - 1095 days refer to 3 years.

The command, upon execution, will request the following inputs:

```
What is your first and last name?
[Unknown]: <Type the eG SuperManager's fully qualified domain name here>
What is the name of your organizational unit?
[Unknown]: United States
What is the name of your organization?
[Unknown]: eG Innovations Inc
What is the name of your City or Locality?
[Unknown]: Bridge Water
What is the name of your State or Province?
[Unknown]: New Jersey
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=eG Innovations Inc, OU=United States, O=eG Innovations Inc, L=Bridge
Water, ST=New Jersey, C=US correct?
[no]: yes
```

When requested for the **first and last name**, indicate the *fully qualified domain name* using which you will be accessing the eG SuperManager. For instance, if the eG SuperManager is to be accessed as *http://egsm.eginnovations*.com, where *egsm.eginnovation.com* is the fully qualified domain name of the eG SuperManager, then specify *egsm.eginnovations.com* here.

Once all the required inputs are provided, a .keystore file will be generated in the **<JAVA_HOME_DIR>\bin** directory with the **<Filename>** you had provided while issuing the command.

2.5.2.2 Generating a Certificate Request

Once a keystore file is generated, proceed to request for a certificate from a valid certifying authority. The procedure for this is as follows:

1. Login to the eG SuperManager and go to the command prompt.
2. Set the **JAVA_HOME** path if it is not done already.
3. Execute the following commands one after another:

   cd $JAVA_HOME/bin

   keytool -certreq -alias **egitlab1** *-keyalg* **RSA** *-file* **<Name_of_the_text_file>** *-keypass* **mykey** *-keystore* **<filename>.keystore** *–storepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name that you provided while generating the keystore file (see Section 2.5.2.1 of this document).**
   - **-keyalg** : specifies the algorithm that was used to generate the keys; this can be **RSA** or **DSA**, depending upon **which algorithm was used for key generation in the procedure detailed in Section 2.5.2.1**
   - **-file** : Provide a name for the text file to which the certificate request will be saved.
   - -**keypass** : the password used to protect the key that was generated; **make sure that you provide the same password that you provided while generating the keystore file (see Section 2.5.2.1 of this document).** Also, note that **-storepass** and **-keypass** should be the same.

- **-keystore** : Provide the name of the *keystore* file in which the key has been stored; **specify the same file name that you used to store the key (see Section 2.5.2.1 of this document)**.

4. If this command executes successfully, then a certificate request will be generated and automatically stored in the text file you specified in step 2 above.

2.5.2.3 Obtaining the Certificate from the CA

1. The first step towards obtaining a certificate is to submit the certificate request to the CA. For this connect to the Certificate server of the CA and submit the certificate. The procedure for request submission will differ from one CA to another.
2. The certificate will thus be generated. Download the certificate.

2.5.2.4 Importing the Certificates into the Keystore File

The downloaded certificate can be in one of the following forms:

- Can be a single, combined certificate
- Can be accompanied by a certificate chain
- Can be in a PEM format

The procedure for importing certificates differs based on the format of the downloaded certificate. These procedures have been detailed in the sub-sections below.

Importing a Combined Certificate into the Keystore File

In this case, follow the steps below to import the certificate into the keystore file:

1. Set the **JAVA_HOME** path if it is not done already.
2. Execute the following commands, one after another:

   cd $JAVA_HOME/bin

   keytool -import –trustcacerts -alias **egitlab1** *-file* **<Name_of_the_domain_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name you provided when generating the keystore (see Section 2.5.2.1) .**
   - **-file**: the name of the domain certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.5.2.1 above.

Importing a Signed Certificate and the Certificate Chain into the Keystore File

Digital certificates are verified using a chain of trust. The trust anchor for the digital certificate is the Root Certificate Authority (CA). The Certificate Hierarchy is a structure of certificates that allows individuals to verify the validity of a certificate's issuer. Certificates are issued and signed by certificates that reside

higher in the certificate hierarchy, so the validity and trustworthiness of a given certificate is determined by the corresponding validity of the certificate that signed it.

The Chain of Trust of a Certificate Chain is an ordered list of certificates, containing an end-user subscriber certificate and intermediate certificates (that represents the Intermediate CA), that enables the receiver to verify that the sender and all intermediate certificates are trustworthy.

A certificate chain will therefore consist of multiple certificates. Before importing each of these certificates, **you will have to understand the hierarchy of the certificates**. To know which is the root and which is the intermediate certificate, refer to the web site of the certificate authority. For instance, if Comodo is the Certificate Authority that has issued the SSL certificate, then connect to the following URL, https://support.comodo.com/index.php?/Default/Knowledgebase/Article/View/620/1/, to gain clarity.

Then, follow the steps below:

1. Set the **JAVA_HOME** path if it is not done already.
2. Then, import the Root certificate. For this, execute the following commands, one after another in the command prompt:

   cd $JAVA_HOME/bin

   keytool -import –trustcacerts -alias **rootcert** *-file* **<Name_of_the_root_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore** *–keypass* **mykey** *–storepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide a unique alias name for the root certificate.**
   - **-file**: the name of the root certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.5.2.1 above.
   - **-keypass** and **–storepass** : Provide the same **keypass** and **storepass** that you specified when generating the keystore; refer to Section 2.5.2.1 above for details.
3. Next, import each of the intermediate certificates, one after another, using the following command:

   keytool -import –trustcacerts -alias **intercert1** *-file* **<Name_of_the_intermediate_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore** *–keypass* **mykey** *–storepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide a unique alias name for every intermediate certificate.**
   - **-file**: the name of the intermediate certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.5.2.1 above.
   - **-keypass** and **–storepass** : Provide the same **keypass** and **storepass** that you specified when generating the keystore; refer to Section 2.5.2.1 above for details.

4. Finally, import the entity/domain certificate into the keystore by issuing the following command:

   keytool -import –trustcacerts -alias **egitlab1** *-file* **<Name_of_the_domain_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name you provided when generating the keystore (see Section 2.5.2.1) .**
   - **-file**: the name of the domain certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.5.2.1 above.

If the domain certificate import command throws an error for any reason, it could be because, all related certificates may not have been imported. Check the web site of the CA for more details.

Importing a Certificate that is in the PEM Format

PEM is a container format that may include just the public certificate (such as with Apache installs, and CA certificate files */etc/ssl/certs*), or may include an entire certificate chain including public key, private key, and root certificates, or may only contain a certificate and a private key.

If the certificate you downloaded is in the PEM format and includes only a certificate file and a private key file, then follow the steps below to import that certificate into a keystore file.

1. Run the following command from the command prompt to export the certificate and private key file into the pkcs12 format:

   openssl pkcs12 -export -in **certificate.crt** *-inkey* **private.key** *-certfile* **certificate.crt** *-name* **'My certificate"** *-out* **keystore***.p12*

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-in** : the name of the certificate that is included in the PEM container
   - **-inkey**: the name of the private key file the PEM container consists of
   - **-certfile** : the name of the certificate that is included in the PEM container
   - **-name** : Provide a **unique name for the certificate file** that is being exported.
   - -**out** : Specify the name of the keystore file to which the certificate and private key are to be exported. **The keystore file can have any name of your choice.**

2. Next, you need to convert the keystore file, which is currently in the pkcs12 format, into the Java keystore (i.e., JKS) format. For this, issue the following command at the command prompt:

keytool -importkeystore –alias **egitlab1** *-deststorepass* **mykey** *-destkeypass* **mykey** *-destkeystore* **keystore,jks** *-srckeystore* **keystore.pk12** *-srcstoretype PKCS12 -srcstorepass* **mykey**

The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

- **-alias** : the aliasname of the certificate being requested; **make sure that you provide the same alias name that you specified in Section 2.5.2.1 of this document.**
- **-deststorepass** : this refers to the **storepass** of the destination keystore file – i.e., the keystore file in the JKS format. **The storepass of the destination keystore should be the same as the storepass of the source keystore.**
- **-destkeypass** : this refers to the **keypass** of the destination keystore file - i.e., the keystore file in the JKS format. **The storepass and keypass of the destination keystore file should be the same.**
- **-destkeystore**: the name of the destination keystore file – i.e., the keystore file in the JKS format.
- **-srckeystore** : the name of the destination keystore file – i.e., the keystore file in the PKCS12 format.
- **-srcstorepass** : The **storepass** of the source keystore file – i.e., the keystore file in the PKCS12 firmat. **make sure that you provide the same storepass you specified in Section 2.5.2.1 of this document**

2.5.2.5 Configuring Tomcat for Using the Keystore File

The eG SuperManager on Unix uses Tomcat as the web server. Therefore, to SSL-enable the eG SuperManager, you need to configure the **server.xml** file of Tomcat with the name and full path to the keystore file which was created earlier.

For this purpose, do the following:

1. Stop the eG SuperManager.
2. Edit the **server.xml** file in the **<CATALINA_HOME>/conf** directory.
3. In the file, search for the XML block where the SSL Coyote HTTP connector on port 8443 is defined. If this block is commented, it indicates that the eG SuperManager is not SSL-enabled and is hence listening on an HTTP port only. To SSL-enable the eG SuperManager, first uncomment this block as indicated below:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
    <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
               port=8443 minProcessors="5" maxProcessors="75"
               enableLookups="true"
          acceptCount="10" debug="0" scheme="https" secure="true"
               useURIValidationHack="false">
      <Factory
className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
               clientAuth="false" protocol="TLS" />
    </Connector>
```

4. Then, proceed to make the changes indicated in **Bold** below in the SSL XML block:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->

<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
port="<eG_SuperManager_Port>" minProcessors="5" maxProcessors="75"
enableLookups="true"
```

```
 acceptCount="10" debug="0" scheme="https" secure="true"
                useURIValidationHack="false">
     <Factory
className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
                clientAuth="false" protocol="TLS"
keystoreFile="/opt/egurkha/manager/tomcat/conf/<filename>.keystore"  keystorePass="mykey"/>
    </Connector>
```

Set the *port* parameter in the XML block to reflect the port number that you have configured for the eG SuperManager. Also, note that two new parameters, namely - **keystoreFile** and **keystorePass** - have been inserted into the SSL block. While the **keystoreFile** parameter has to be set to the full path to the **.keystore** file that you generated earlier, the **keystorePass** parameter should be set to the keystore password that you specified while issuing the **keytool** command.

5. With that change, the eG SuperManager on Linux has acquired the capability to listen on two ports - the SSL port and the non-SSL port. To configure the eG SuperManager to listen only on the SSL port, simply comment that section of the **server.xml** file where the non-SSL Coyote HTTP connector on port 8081 has been defined, as indicated below:

```
<!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8081 -->
    <!-- <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
                   port="<eG_SuperManager_Port>" minProcessors="32"
maxProcessors="256"
                enableLookups="true" redirectPort="8443"
                acceptCount="10" debug="0" connectionTimeout="20000"
                useURIValidationHack="false"  URIEncoding="UTF-8"/> -->
```

6. Save the file.
7. Finally, start the eG SuperManager.

2.6 SSL-Enabling the eG SuperManager on Windows

To SSL-enable an eG SuperManager on Windows, you have the following options:

- You can obtain a signed certificate from an internal certifying authority (eg., Microsoft Active Directory Certificate Services) and use this certificate to SSL-enable the eG SuperManager, (OR)
- You can obtain a signed certificate from a valid, external certifying authority (eg., Verisign) and use this certificate to SSL-enable the eG SuperManager

If you go with option (a), use the procedure detailed in Section 2.6.1. If you pick option (b), use the procedure detailed in Section 2.6.2.

2.6.1 SSL-Enabling the eG SuperManager Using a Certificate Signed by an Internal CA

You can obtain a signed certificate from an internal certificate authority and use that certificate for SSL-enabling the eG SuperManager.

For this, follow the steps given below:

- Generate the Keystore file

- Generate a certificate request
- Submit the certificate request to the internal certificate Authority (CA) and obtain a certificate
- Import the certificate into a keystore
- Configure Tomcat for using the keystore file

Each of these steps has been discussed in the sections that follow.

2.6.1.1 Generating the Keystore File

The keystore file stores the details of the **certificates** necessary to make the protocol secure. Certificates contain the information pertaining to the source of the application data, and helps validate the source. To generate the keystore, use the **keytool** command. For this purpose, login to the Windows eG SuperManager and go to the command prompt. Set the **JAVA_HOME** path if it is not done already. Then, execute the following commands, one after another:

cd %JAVA_HOME%\bin

keytool -genkey -alias **egitlab1** *-keyalg* **RSA -***keypass* **mykey** *-keystore* **<Filename>***.keystore -storepass* **mykey** *-keysize* **2048** *-validity* **1095**

The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

- **-alias** : an alias name for the certificate being generated
- -**keypass** : a password used to protect the key that is generated; **ensure that you provide the same values for -keypass and -storepass.**
- **-keyalg** : specifies the algorithm that is used to generate the keys. The options are as follows:
    - **DSA** : Digital Signature Algorithm
    - **RSA :** An algorithm used for public-key cryptography
- **-keystore** : the *keytool* command stores the generated key in a *.keystore* file; provide a name for this file as input to the **-keystore** command
- **-keysize** : the size of the key that is generated; the default key size is 1024 bits - the key size must be in the range 512 bits - 1024 bits
- **-validity** : indicates the number of days for which the key/certicate will be valid - 1095 days refer to 3 years.

The command, upon execution, will request the following inputs:

```
What is your first and last name?
[Unknown]: <Type the eG SuperManager's fully qualified domain name here>
What is the name of your organizational unit?
[Unknown]: United States
What is the name of your organization?
[Unknown]: eG Innovations Inc
What is the name of your City or Locality?
[Unknown]: Bridgewater
What is the name of your State or Province?
[Unknown]: New Jersey
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=eG Innovations Inc, OU=United States, O=eG Innovations Inc, L=Bridge
Water, ST=New Jersey, C=US correct?
[no]: yes
```

When requested for the **first and last name**, indicate the *fully qualified domain name* using which you will be accessing the eG SuperManager. For instance, if the eG SuperManager is to be accessed as *http://egsm.eginnovations.*com, where *egsm.eginnovation.com* is the fully qualified domain name of the eG eG SuperManager, then specify *egsm.eginnovations.com* here.

Once all the required inputs are provided, a .keystore file will be generated in the **<JAVA_HOME_DIR>\bin** directory with the **<Filename>** you had provided while issuing the command.

2.6.1.2 Generating a Certificate Request

Once a keystore file is generated, proceed to request for a certificate from an internal certifying authority. The procedure for this is as follows:

1. Login to the eG SuperManager and go to the Windows command prompt.
2. Set the **JAVA_HOME** path if it is not done already.
3. Execute the following commands one after another:

   cd %JAVA_HOME%\bin

   keytool -certreq -alias **egitlab1** *-keyalg* **RSA** *-file* **<Name_of_the_text_file>** *-keypass* **mykey** *-keystore* **<filename>.keystore** *–storepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name that you provided while generating the keystore file (see Section 2.6.1.1 of this document).**
   - **-keyalg** : specifies the algorithm that was used to generate the keys; this can be **RSA** or **DSA**, depending upon **which algorithm was used for key generation in the procedure detailed in Section 2.6.1.1**
   - **-file** : Provide a name for the text file to which the certificate request will be saved.
   - -**keypass** : the password used to protect the key that was generated; **make sure that you provide the same password that you provided while generating the keystore file (see Section 2.6.1.1 of this document).** Also, note that **-storepass** and **-keypass** should be the same.

- **-keystore** : Provide the name of the *keystore* file in which the key has been stored; **specify the same file name that you used to store the key (see Section 2.6.1.1 of this document)**.

4. If this command executes successfully, then a certificate request will be generated and automatically stored in the text file you specified in step 2 above.

2.6.1.3 Obtaining the Certificate from the Internal CA

1. The first step towards obtaining a certificate is to submit the certificate request to the internal CA. For this connect to the Certificate server of the internal CA and select the option to submit the certificate. For instance, if you are using Microsoft Active Directory Certificate Services to request for a self-signed certificate, then, you need to connect to **http://<YourWebServerName>/certsrv**, and then pick the option to submit the certificate. Figure 2.11 will then appear.

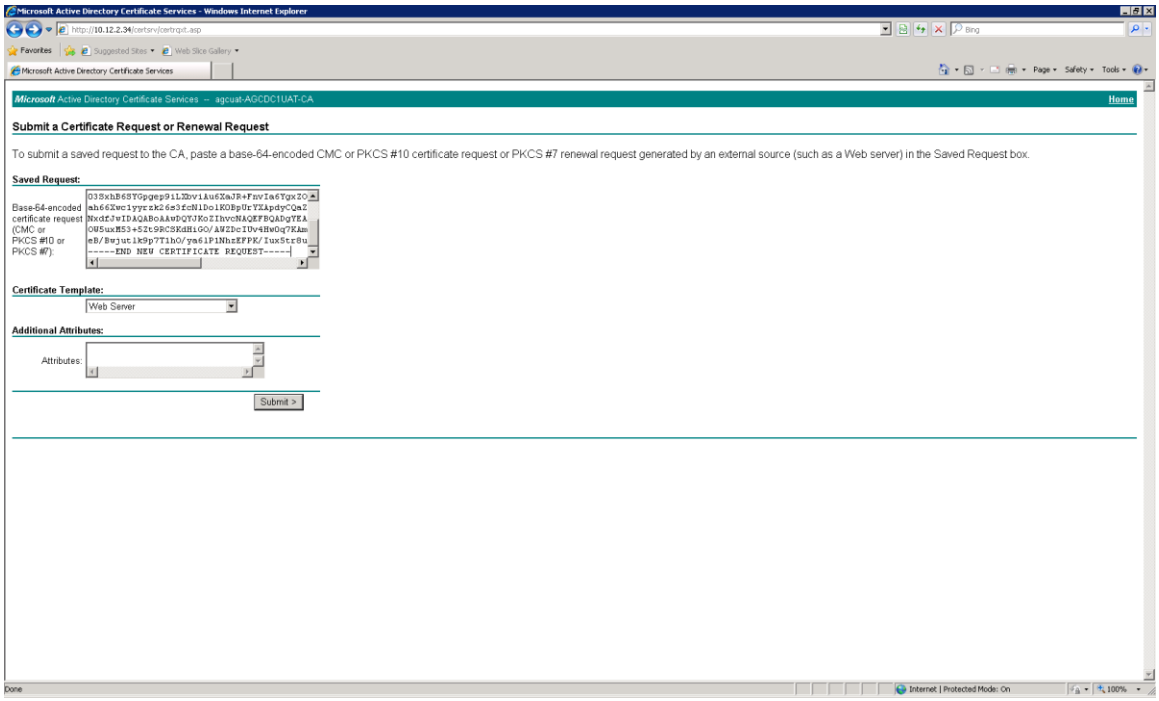


Figure 2.11: Requesting for a certificate

2. Open the text file containing the certificate request (which was created using the procedure detailed in Section 2.6.1.2 above), copy the contents of the file, and then paste it to the text area of the **Base 64-encoded certificate request** text box of Figure 2.11. Then, click the **Submit** button.

3. The certificate will thus be generated. Download the certificate.

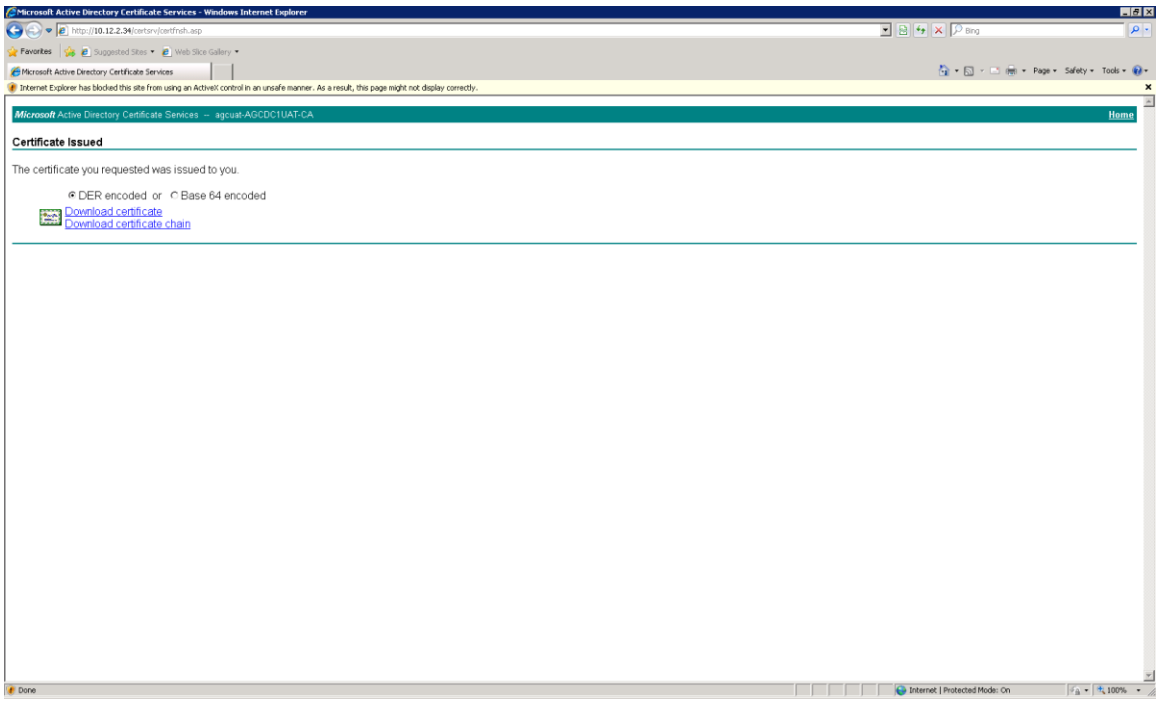


Figure 2.12: Downloading the certificate

2.6.1.4 Importing the Certificates into the Keystore File

The downloaded certificate can be in one of the following forms:

- Can be a single, combined certificate
- Can be accompanied by a certificate chain
- Can be in a PEM format

The procedure for importing certificates differs based on the format of the downloaded certificate. These procedures have been detailed in the sub-sections below.

Importing a Combined Certificate into the Keystore File

In this case, follow the steps below to import the certificate into the keystore file:

1. Set the **JAVA_HOME** path if it is not done already.
2. At the command prompt, execute the following commands, one after another:

   cd %JAVA_HOME%\bin

   keytool -import –trustcacerts -alias **egitlab1** *-file* **<Name_of_the_domain_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name you provided when generating the keystore (see Section 2.6.1.1) .**
   - **-file**: the name of the domain certificate that you want to import

- **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.6.1.1 above.

Importing a Signed Certificate and the Certificate Chain into the Keystore File

Digital certificates are verified using a chain of trust. The trust anchor for the digital certificate is the Root Certificate Authority (CA). The Certificate Hierarchy is a structure of certificates that allows individuals to verify the validity of a certificate's issuer. Certificates are issued and signed by certificates that reside higher in the certificate hierarchy, so the validity and trustworthiness of a given certificate is determined by the corresponding validity of the certificate that signed it.

The Chain of Trust of a Certificate Chain is an ordered list of certificates, containing an end-user subscriber certificate and intermediate certificates (that represents the Intermediate CA), that enables the receiver to verify that the sender and all intermediate certificates are trustworthy.

A certificate chain will therefore consist of multiple certificates. Before importing each of these certificates, **you will have to understand the hierarchy of the certificates**. To know which is the root and which is the intermediate certificate, refer to the web site of the certificate authority. Then, set the **JAVA_HOME** path if it is not done already. Next, follow the steps below:

1. First, import the Root certificate. For this, execute the following commands, one after another in the command prompt:

   cd %JAVA_HOME%\bin

   keytool -import –trustcacerts -alias **rootcert** *-file* **<Name_of_the_root_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore** *–keypass* **mykey** *–storepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide a unique alias name for the root certificate**.
   - **-file**: the name of the root certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.6.1.1 above.
   - **-keypass** and **–storepass** : Provide the same **keypass** and **storepass** that you specified when generating the keystore; refer to Section 2.6.1.1 above for details.

2. Next, import each of the intermediate certificates, one after another, using the following command:

   keytool -import –trustcacerts -alias **intercert1** *-file* **<Name_of_the_intermediate_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore** *–keypass* **mykey** *–storepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide a unique alias name for every intermediate certificate**.
   - **-file**: the name of the intermediate certificate that you want to import

- **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.6.1.1 above.
- **-keypass** and **–storepass** : Provide the same **keypass** and **storepass** that you specified when generating the keystore; refer to Section 2.6.1.1 above for details.

3. Finally, import the entity/domain certificate into the keystore by issuing the following command:

   keytool -import –trustcacerts -alias **egitlab1** *-file* **<Name_of_the_domain_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name you provided when generating the keystore (see** Section 2.6.1.1) .
   - **-file**: the name of the domain certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.6.1.1 above.

If the domain certificate import command throws an error for any reason, it could be because, all related certificates may not have been imported. Check the web site of the CA for more details.

Importing a Certificate that is in the PEM Format

PEM is a container format that may include just the public certificate (such as with Apache installs, and CA certificate files */etc/ssl/certs*), or may include an entire certificate chain including public key, private key, and root certificates, or may only contain a certificate and a private key.

If the certificate you downloaded is in the PEM format and includes only a certificate file and a private key file, then follow the steps below to import that certificate into a keystore file.

1. Run the following command from the command prompt to export the certificate and private key file into the pkcs12 format:

   openssl pkcs12 -export -in **certificate.crt** *-inkey* **private.key** *-certfile* **certificate.crt** *-name* **'My certificate"** *-out* **keystore***.p12*

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-in** : the name of the certificate that is included in the PEM container
   - **-inkey**: the name of the private key file the PEM container consists of
   - **-certfile** : the name of the certificate that is included in the PEM container

- **-name** : Provide a **unique name for the certificate file** that is being exported.
- -**out :** Specify the name of the keystore file to which the certificate and private key are to be exported. **The keystore file can have any name of your choice.**

2. Next, you need to convert the keystore file, which is currently in the pkcs12 format, into the Java keystore (i.e., JKS) format. For this, issue the following command at the command prompt:

   keytool -importkeystore –alias **egitlab1** *-deststorepass* **mykey** *-destkeypass* **mykey** *-destkeystore* **keystore,jks** *-srckeystore* **keystore.pk12** *-srcstoretype PKCS12 -srcstorepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the aliasname of the certificate being requested; **make sure that you provide the same alias name that you specified in** Section 2.6.1.1 **of this document.**
   - **-deststorepass** : this refers to the **storepass** of the destination keystore file – i.e., the keystore file in the JKS format. **The storepass of the destination keystore should be the same as the storepass of the source keystore.**
   - **-destkeypass** : this refers to the **keypass** of the destination keystore file - i.e., the keystore file in the JKS format. **The storepass and keypass of the destination keystore file should be the same.**
   - **-destkeystore**: the name of the destination keystore file – i.e., the keystore file in the JKS format.
   - **-srckeystore** : the name of the destination keystore file – i.e., the keystore file in the PKCS12 format.
   - **-srcstorepass** : The **storepass** of the source keystore file – i.e., the keystore file in the PKCS12 firmat. **make sure that you provide the same storepass you specified in Section 2.6.1.1 of this document**

2.6.1.5 Configuring Tomcat for Using the Keystore File

The eG SuperManager on Windows uses Tomcat as the web server. Therefore, to SSL-enable the eG SuperManager, you need to configure the **server.xml** file of Tomcat with the name and full path to the keystore file which was created earlier.

1. Edit the **server.xml** file in the **<CATALINA_HOME>\conf** directory.
2. In the file, search for the XML block where the SSL Coyote HTTP connector on port 8443 is defined. If this block is commented, it indicates that the eG SuperManager is not SSL-enabled and is hence listening on an HTTP port only. To SSL-enable the eG SuperManager, first uncomment this block as indicated below:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
    <Connector protocol="HTTP/1.1"
               port="8443" minSpareThreads="32" maxThreads="512"
        enableLookups="false" acceptCount="10"  connectionTimeout="20000"

        useURIValidationHack="false" URIEncoding="UTF-8" compression="on"
compressionMinSize="2048" noCompressionUserAgents="gozilla, traviata"
compressableMimeType="text/html,text/xml,text/plain,application/x-java-
applet,application/octet-
stream,application/xml,text/javascript,text/css,image/png,image/jpeg,image/
gif,application/pdf,application/x-javascript,application/javascript"
        SSLEnabled="true" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS"
keystoreFile="webapps/eGmanager.bin" keystorePass="eginnovations"/>
```

3. Then, proceed to make the changes indicated in **Bold** below in the SSL XML block:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
    <Connector protocol="HTTP/1.1"
               port="7077" minSpareThreads="32" maxThreads="512"
        enableLookups="false" acceptCount="10"  connectionTimeout="20000"
        useURIValidationHack="false" URIEncoding="UTF-8" compression="on"
compressionMinSize="2048" noCompressionUserAgents="gozilla, traviata"
compressableMimeType="text/html,text/xml,text/plain,application/x-java-
applet,application/octet-
stream,application/xml,text/javascript,text/css,image/png,image/jpeg,image/
gif,application/pdf,application/x-javascript,application/javascript"
        SSLEnabled="true" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS" keyAlias="egitlab1"
keystoreFile="<The_full_path_to_the_keystore_file>" keystorePass="mykey" />
```

Set the *port* parameter in the XML block to reflect the SSL port number that you have configured for the eG SuperManager. Also, note that three new parameters, namely - **keyAlias**, **keystoreFile** and **keystorePass** - have been inserted into the SSL block. While the **keystoreFile** parameter has to be set to the full path to the **.keystore** file that you generated earlier, the **keystorePass** parameter should be set to the keystore password that you specified while issuing the **keytool** command. Likewise, the **keyAlias** parameter is to be set to the **alias name** that you provided for the certificate file, when you generated it in Section 2.6.1.1 above.

4. With that change, the eG SuperManager on Windows has acquired the capability to listen on two ports - the SSL port and the non-SSL port. To configure the eG SuperManager to listen only on the SSL port, simply comment that section of the **server.xml** file where the non-SSL Coyote HTTP connector on port 8081 has been defined, as indicated below:

```
<!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8081 -->
    <!-- <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
                   Port7077" minSpareThreads="32" maxThreads="512"
               enableLookups="true" redirectPort="8443"
               acceptCount="10" debug="0" connectionTimeout="20000"
               useURIValidationHack="false"  URIEncoding="UTF-8"/> -->
```

5. Save the file.

6. Then, SSL-enable the **start_sm.bat** script. For this, first open the **start_sm.bat** file (see Figure 2.13) residing in the **<EG_INSTALL_DIR>\lib** directory. Change the URL **http://<eGsupermanagermanagerIp>:<eGsupermanagerdefaultTCPPort>/final/servlet/StartSM** present in the last line of the batch file to **https://<eGmanagerIp>:<eGmanagerPort>/final/servlet/StartSM**(see

Figure 3.76).

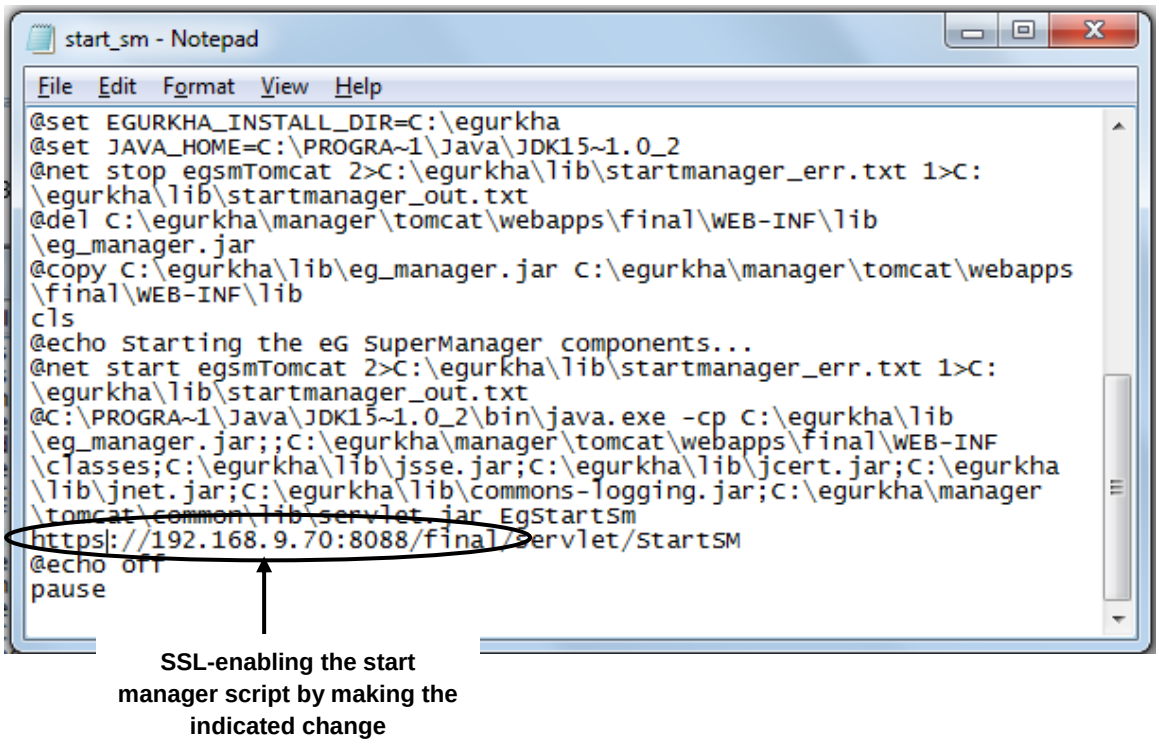


Figure 2.13: SSL-enabling the start_sm script

7. Finally, start the eG SuperManager.

2.6.2 SSL-Enabling the eG SuperManager Using a Signed Certificate Obtained from a Valid Certifying Authority

Self-signed certificates are useful in environments where 'security' is not a priority. In highly secure environments, especially where the eG SuperManager is to be frequently accessed via the public internet, using a self-signed certificate may not be preferred. In such a case, you can obtain a valid certificate from a certificate authority and use that certificate to SSL-enable the eG SuperManager.

The broad steps to be followed to achieve this are as follows:

1. Generating the keystore file
2. Generating a certificate request
3. Submitting the certificate request to the Certificate Authority (CA) and obtaining a certificate
4. Importing the certificate into a keystore
5. Configuring Tomcat for using the keystore file

The sub-sections below elaborate on each of these steps.

2.6.2.1 Generating a Keystore File

The keystore file stores the details of the **certificates** necessary to make the protocol secure. Certificates contain the information pertaining to the source of the application data, and helps validate the source. To generate the keystore, use the **keytool** command. For this purpose, login to the Windows manager and go to the command prompt. Set the **JAVA_HOME** path if it is not done already. Then, execute the following commands, one after another:

cd %JAVA_HOME%\bin

keytool -genkey -alias **egitlab1** *-keyalg* **RSA -***keypass* **mykey** *-keystore* **<Filename>***.keystore -storepass* **mykey** *-keysize* **2048** *-validity* **1095**

The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

- **-alias** : an alias name for the certificate being generated
- -**keypass** : a password used to protect the key that is generated; **ensure that you provide the same values for -keypass and -storepass.**
- **-keyalg** : specifies the algorithm that is used to generate the keys. The options are as follows:
    - **DSA** : Digital Signature Algorithm
    - **RSA :** An algorithm used for publick-key cryptography
- **-keystore** : the *keytool* command stores the generated key in a *.keystore* file; provide a name for this file as input to the **-keystore** command
- **-keysize** : the size of the key that is generated; the default key size is 1024 bits - the key size must be in the range 512 bits - 1024 bits
- **-validity** : indicates the number of days for which the key/certicate will be valid - 1095 days refer to 3 years.

The command, upon execution, will request the following inputs:

```
What is your first and last name?
[Unknown]: <Type the eG SuperManager's fully qualified domain name here>
What is the name of your organizational unit?
[Unknown]: United States
What is the name of your organization?
[Unknown]: eG Innovations Inc
What is the name of your City or Locality?
[Unknown]: Bridgewater
What is the name of your State or Province?
[Unknown]: New Jersey
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=eG Innovations Inc, OU=United States, O=eG Innovations Inc, L=Bridge
Water, ST=New Jersey, C=US correct?
[no]: yes
```

When requested for the **first and last name**, indicate the *fully qualified domain name* using which you will be accessing the eG SuperManager. For instance, if the eG SuperManager is to be accessed as *http://egmanager.eginnovations.*com, where *egmanager.eginnovation.com* is the fully qualified domain name of the eG SuperManager, then specify *egmanager.eginnovations.com* here.

Once all the required inputs are provided, a .keystore file will be generated in the **<JAVA_HOME_DIR>\bin** directory with the **<Filename>** you had provided while issuing the command.

2.6.2.2 Generating a Certificate Request

Once a keystore file is generated, proceed to request for a certificate from a valid certifying authority. The procedure for this is as follows:

1. Login to the eG SuperManager and go to the Windows command prompt.
2. Set the **JAVA_HOME** path if it is not done already.
3. Execute the following commands one after another:

   cd %JAVA_HOME%\bin

   keytool -certreq -alias **egitlab1** *-keyalg* **RSA** *-file* **<Name_of_the_text_file>** *-keypass* **mykey** *-keystore* **<filename>.keystore** *–storepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name that you provided while generating the keystore file (see Section 2.6.2.1 of this document).**
   - **-keyalg** : specifies the algorithm that was used to generate the keys; this can be **RSA** or **DSA**, depending upon **which algorithm was used for key generation in the procedure detailed in Section 2.6.2.1**
   - **-file** : Provide a name for the text file to which the certificate request will be saved.
   - -**keypass** : the password used to protect the key that was generated; **make sure that you provide the same password that you provided while generating the keystore file (see Section 2.6.2.1 of this document).** Also, note that **-storepass** and **-keypass** should be the same.
   - **-keystore** : Provide the name of the *keystore* file in which the key has been stored; **specify the same file name that you used to store the key (see Section 2.6.2.1 of this document)**.

4. If this command executes successfully, then a certificate request will be generated and automatically stored in the text file you specified in step 2 above.

2.6.2.3 Obtaining the Certificate from the CA

1. The first step towards obtaining a certificate is to submit the certificate request to the CA. For this connect to the Certificate server of the CA and submit the certificate. The procedure for request submission will differ from one CA to another.

2. The certificate will thus be generated. Download the certificate.

2.6.2.4 Importing the Certificates into the Keystore File

The downloaded certificate can be in one of the following forms:

- Can be a single, combined certificate
- Can be accompanied by a certificate chain
- Can be in a PEM format

The procedure for importing certificates differs based on the format of the downloaded certificate. These procedures have been detailed in the sub-sections below.

Importing a Combined Certificate into the Keystore File

In this case, follow the steps below to import the certificate into the keystore file:

1. Set the **JAVA_HOME** path if it is not done already.
2. At the command prompt, execute the following commands, one after another:

   cd %JAVA_HOME%\bin

   keytool -import –trustcacerts -alias **egitlab1** *-file* **<Name_of_the_domain_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name you provided when generating the keystore (see Section 2.6.2.1) .**
   - **-file**: the name of the domain certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.6.2.1 above.

Importing a Signed Certificate and the Certificate Chain into the Keystore File

Digital certificates are verified using a chain of trust. The trust anchor for the digital certificate is the Root Certificate Authority (CA). The Certificate Hierarchy is a structure of certificates that allows individuals to verify the validity of a certificate's issuer. Certificates are issued and signed by certificates that reside higher in the certificate hierarchy, so the validity and trustworthiness of a given certificate is determined by the corresponding validity of the certificate that signed it.

The Chain of Trust of a Certificate Chain is an ordered list of certificates, containing an end-user subscriber certificate and intermediate certificates (that represents the Intermediate CA), that enables the receiver to verify that the sender and all intermediate certificates are trustworthy.

A certificate chain will therefore consist of multiple certificates. Before importing each of these certificates, **you will have to understand the hierarchy of the certificates**. To know which is the root and which is the intermediate certificate, refer to the web site of the certificate authority. For instance, if Comodo is the Certificate Authority that has issued the SSL certificate, then connect to the following URL, https://support.comodo.com/index.php?/Default/Knowledgebase/Article/View/620/1/, to gain clarity.

Then, follow the steps below:

1. Set the **JAVA_HOME** path if it is not done already.

2. Then, import the Root certificate. For this, execute the following commands, one after another in the command prompt:

   cd %JAVA_HOME%\bin

   keytool -import –trustcacerts -alias **rootcert** *-file* **<Name_of_the_root_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore** *–keypass* **mykey** *–storepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide a unique alias name for the root certificate.**
   - **-file**: the name of the root certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.6.2.1 above.
   - **-keypass** and **–storepass** : Provide the same **keypass** and **storepass** that you specified when generating the keystore; refer to Section 2.6.2.1 above for details.

3. Next, import each of the intermediate certificates, one after another, using the following command:

   keytool -import –trustcacerts -alias **intercert1** *-file* **<Name_of_the_intermediate_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore** *–keypass* **mykey** *–storepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide a unique alias name for every intermediate certificate.**
   - **-file**: the name of the intermediate certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.6.2.1 above.
   - **-keypass** and **–storepass** : Provide the same **keypass** and **storepass** that you specified when generating the keystore; refer to Section 2.6.2.1 above for details.

4. Finally, import the entity/domain certificate into the keystore by issuing the following command:

   keytool -import –trustcacerts -alias **egitlab1** *-file* **<Name_of_the_domain_certificate>** *-keystore* **<Name_of_the_keystore_file>.keystore**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the alias name of the certificate being requested; **make sure that you provide the same alias name you provided when generating the keystore (see Section 2.6.2.1) .**
   - **-file**: the name of the domain certificate that you want to import
   - **-keystore** : Provide the name of the *keystore* file you generated when you followed the procedure detailed in Section 2.6.2.1 above.

If the domain certificate import command throws an error for any reason, it could be because, all related certificates may not have been imported. Check the web site of the CA for more details.

Importing a Certificate that is in the PEM Format

PEM is a container format that may include just the public certificate (such as with Apache installs, and CA certificate files */etc/ssl/certs*), or may include an entire certificate chain including public key, private key, and root certificates, or may only contain a certificate and a private key.

If the certificate you downloaded is in the PEM format and includes only a certificate file and a private key file, then follow the steps below to import that certificate into a keystore file.

1. Run the following command from the command prompt to export the certificate and private key file into the pkcs12 format:

   openssl pkcs12 -export -in **certificate.crt** *-inkey* **private.key** *-certfile* **certificate.crt** *-name* **'My certificate"** *-out* **keystore**.*p12*

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-in** : the name of the certificate that is included in the PEM container
   - **-inkey**: the name of the private key file the PEM container consists of
   - **-certfile** : the name of the certificate that is included in the PEM container
   - **-name** : Provide a **unique name for the certificate file** that is being exported.
   - -**out** : Specify the name of the keystore file to which the certificate and private key are to be exported. **The keystore file can have any name of your choice.**

2. Next, you need to convert the keystore file, which is currently in the pkcs12 format, into the Java keystore (i.e., JKS) format. For this, issue the following command at the command prompt:

   keytool -importkeystore –alias **egitlab1** *-deststorepass* **mykey** *-destkeypass* **mykey** *-destkeystore* **keystore,jks** *-srckeystore* **keystore.pk12** *-srcstoretype PKCS12 -srcstorepass* **mykey**

   The text in **Bold** in the above command line indicates those inputs that can change according to the requirements of your environment. These inputs have been described below:

   - **-alias** : the aliasname of the certificate being requested; **make sure that you provide the same alias name that you specified in Section 2.6.2.1 of this document.**
   - **-deststorepass** : this refers to the **storepass** of the destination keystore file – i.e., the keystore file in the JKS format. **The storepass of the destination keystore should be the same as the storepass of the source keystore.**
   - **-destkeypass** : this refers to the **keypass** of the destination keystore file - i.e., the keystore file in the JKS format. **The storepass and keypass of the destination keystore file should be the same.**

- **-destkeystore**: the name of the destination keystore file – i.e., the keystore file in the JKS format.
- **-srckeystore** : the name of the destination keystore file – i.e., the keystore file in the PKCS12 format.
- **-srcstorepass** : The **storepass** of the source keystore file – i.e., the keystore file in the PKCS12 firmat. **make sure that you provide the same storepass you specified in Section 2.6.2.1 of this document**

2.6.2.5 Configuring Tomcat for Using the Keystore File

The eG SuperManager on Windows uses Tomcat as the web server. Therefore, to SSL-enable the eG SuperManager, you need to configure the **server.xml** file of Tomcat with the name and full path to the keystore file which was created earlier.

1. Edit the **server.xml** file in the **<CATALINA_HOME>\conf** directory.
2. In the file, search for the XML block where the SSL Coyote HTTP connector on port 8443 is defined. If this block is commented, it indicates that the eG SuperManager is not SSL-enabled and is hence listening on an HTTP port only. To SSL-enable the eG SuperManager, first uncomment this block as indicated below:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
    <Connector protocol="HTTP/1.1"
               port="8443" minSpareThreads="32" maxThreads="512"
        enableLookups="false" acceptCount="10"  connectionTimeout="20000"

        useURIValidationHack="false" URIEncoding="UTF-8" compression="on"
compressionMinSize="2048" noCompressionUserAgents="gozilla, traviata"
compressableMimeType="text/html,text/xml,text/plain,application/x-java-
applet,application/octet-
stream,application/xml,text/javascript,text/css,image/png,image/jpeg,image/
gif,application/pdf,application/x-javascript,application/javascript"
        SSLEnabled="true" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS"
keystoreFile="webapps/eGmanager.bin" keystorePass="eginnovations"/>
```

3. Then, proceed to make the changes indicated in **Bold** below in the SSL XML block:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
    <Connector protocol="HTTP/1.1"
               port="7077" minSpareThreads="32" maxThreads="512"
        enableLookups="false" acceptCount="10"  connectionTimeout="20000"
        useURIValidationHack="false" URIEncoding="UTF-8" compression="on"
compressionMinSize="2048" noCompressionUserAgents="gozilla, traviata"
compressableMimeType="text/html,text/xml,text/plain,application/x-java-
applet,application/octet-
stream,application/xml,text/javascript,text/css,image/png,image/jpeg,image/
gif,application/pdf,application/x-javascript,application/javascript"
        SSLEnabled="true" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS" keyAlias="egitlab1"
keystoreFile="<The_full_path_to_the_keystore_file>" keystorePass="mykey" />
```

Set the *port* parameter in the XML block to reflect the SSL port number that you have configured for the eG SuperManager. Also, note that three new parameters, namely - **keyAlias**, **keystoreFile** and **keystorePass** - have been inserted into the SSL block. While the **keystoreFile** parameter has to be set

to the full path to the **.keystore** file that you generated earlier, the **keystorePass** parameter should be set to the keystore password that you specified while issuing the **keytool** command. Likewise, the **keyAlias** parameter is to be set to the **alias name** that you provided for the certificate file, when you generated it in Section 2.6.2.1 above.

4. With that change, the eG SuperManager on Windows has acquired the capability to listen on two ports - the SSL port and the non-SSL port. To configure the eG SuperManager to listen only on the SSL port, simply comment that section of the **server.xml** file where the non-SSL Coyote HTTP connector on port 8081 has been defined, as indicated below:

```
<!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8081 -->
    <!-- <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
                  Port7077" minSpareThreads="32" maxThreads="512"
              enableLookups="true" redirectPort="8443"
              acceptCount="10" debug="0" connectionTimeout="20000"
              useURIValidationHack="false"  URIEncoding="UTF-8"/> -->
```

5. Save the file.

6. Then, SSL-enable the **start_sm.bat** script. For this, first open the **start_sm.bat** file (see Figure 2.14) residing in the **<EG_SM_INSTALL_DIR>\lib** directory. Change the URL **http://<eGsupermanagermanagerIp>:<eGsupermanagerdefaultTCPPort>/final/servlet/StartSM** present in the last line of the batch file to **https://<eGmanagerIp>:<eGmanagerPort>/final/servlet/StartSM** (see Figure 2.14).

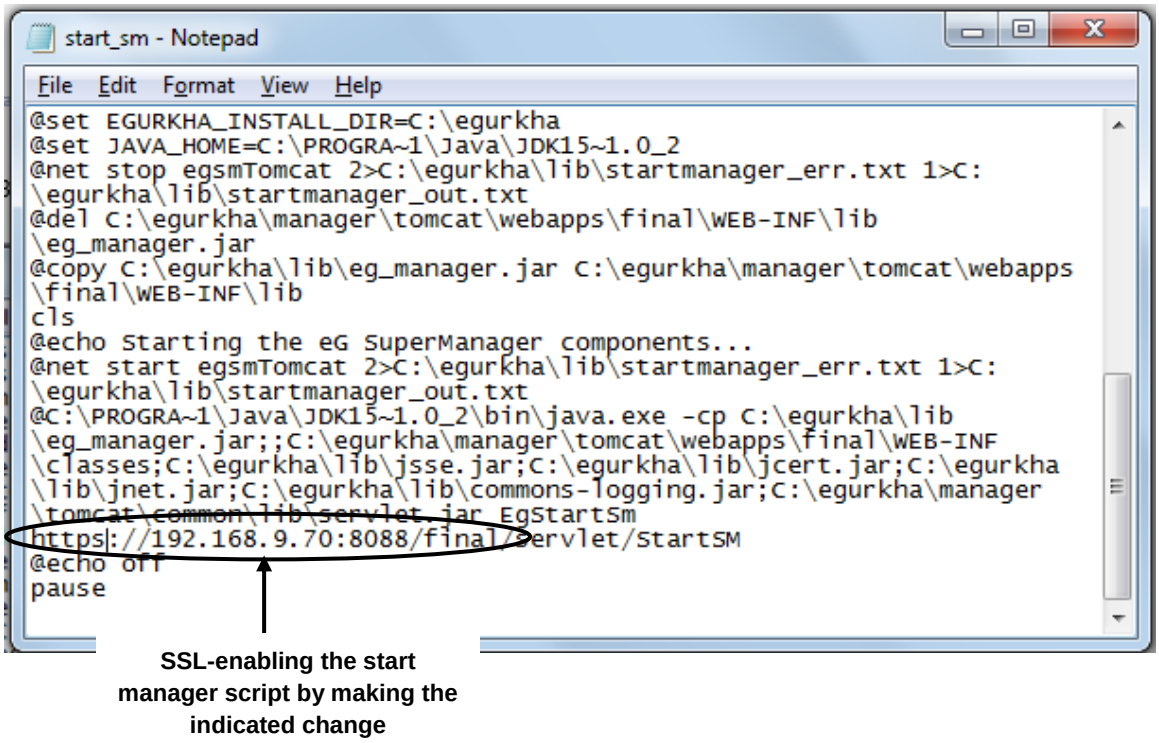


Figure 2.14: SSL-enabling the start_sm script

7. Finally, start the eG SuperManager.

2.7 Configuring the eG SuperManager to Work with the Individual eG Managers

Next, proceed to configure the eG managers that will be managed by the eG SuperManager. To achieve this, do the following:

1. First, login as the eG user to the eG SuperManager host.
2. At the prompt, issue the following command: **/opt/egurkha/bin/configure_sm** (in case of a Unix installation. In a Windows installation, run the batch file: **configure_sm.bat** in the **<EGSM_INSTALL_DIR>\bin** directory). You will first be prompted to specify the Java home directory.

```
Please enter the location of your Java home directory []:
```

3. You will then be prompted to provide the nick name of the eG manager to report to the SuperManager.

```
Specify a nickname for an eG Manager that will report to this SuperManager:
Mgr1
```

The nickname should not contain any white spaces. The only special character that is permissible is the "_" (underscore). All other special characters cannot be used in the nick name.

4. Then, specify the hostname/IP address of this manager, and the port at which it listens.

```
Specify the hostname (IP address) of this eG Manager: 192.168.10.82
Please enter the port for this eG Manager: 7077
```

5. To indicate whether the eG manager being configured for the SuperManager is SSL-enabled, specify **y** when the following question is posed. Otherwise, specify **n**.

```
Please indicate if this eG Manager uses SSL y/n: [n] y
```

6. This completes the configuration of one manager. Next, you will be requested to confirm the addition of a manager.

```
Do you wish to configure another eG Manager for this SuperManager y/n: [n]
n
```

Specifying **y** here will invoke step 2 to step 6 yet again. This time provide the details of another manager. To quit configuration, specify **n**. The following message will then appear.

```
************************************************************
Configuration of this eG SuperManager has been completed!
************************************************************
```

2.8 Configuring the Individual eG Managers to Work with the eG SuperManager

Finally, login to the admin interface of each of the managers configured for the SuperManager and then explicitly add the SuperManager. To achieve this, do the following:

1. Login to the eG administrative interface as *admin* with password *admin*.
2. From the **Miscellaneous** tile of the **Admin** menu, select the **SuperManager Settings** option. Figure 2.15 will then appear.

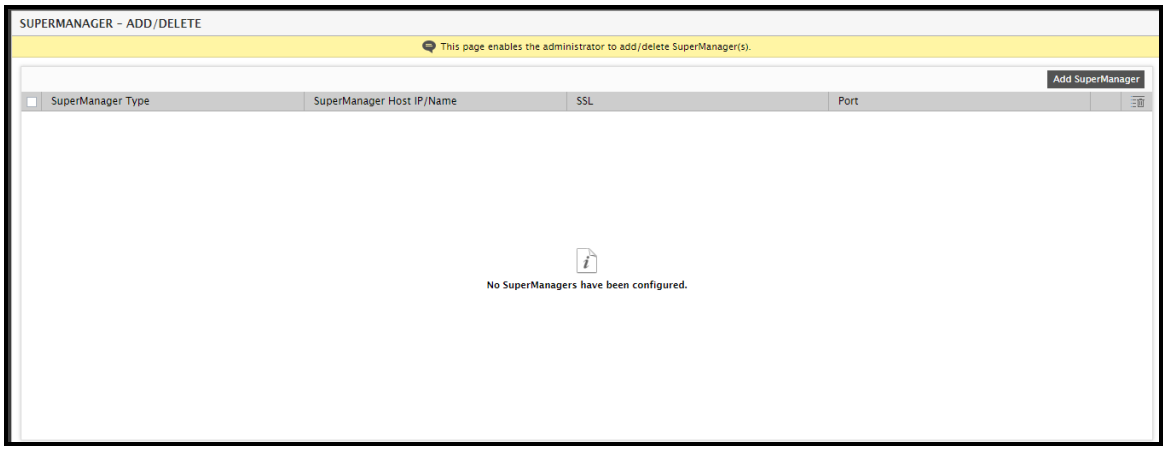


Figure 2.15: Add/delete SuperManager

3. To add an eG SuperManager, first click the **Add SuperManager** button in Figure 2.15.
4. Figure 2.16 will then appear. Here, select **eG SuperManager** as the **SuperManager Type**. Then, provide the **SuperManager Host IP/Name**, indicate whether the supermanager is SSL-enabled or not, and provide the **Port** at which the eG SuperManager listens.

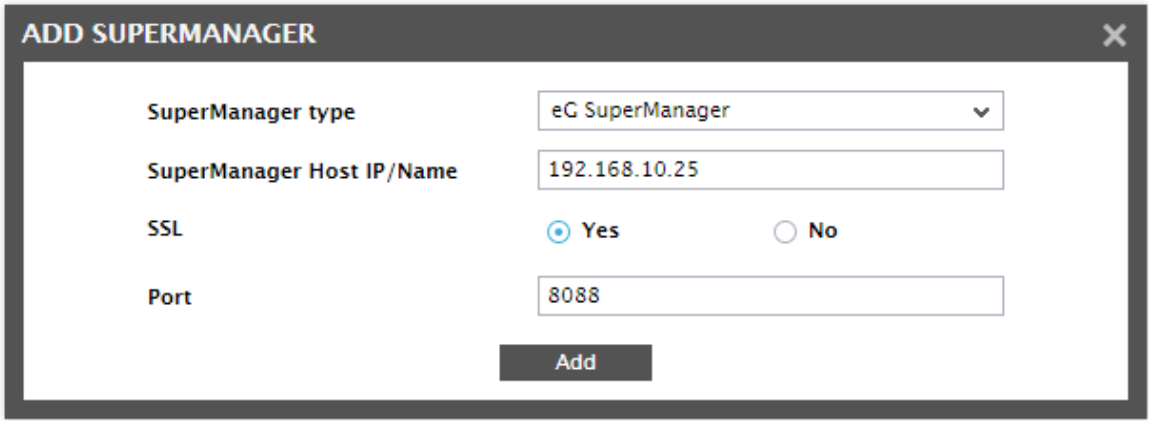


Figure 2.16: Providing the IP of the eG SuperManager to which this manager should report

5. Finally, click the **Add** button in Figure 2.16 to save the changes. Figure 2.17 will then appear displaying the SuperManager you just added.

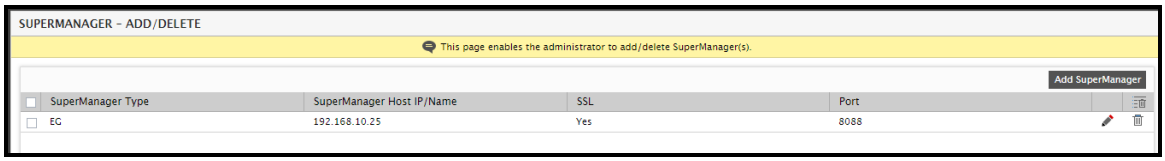


Figure 2.17: Modifying/Deleting a SuperManager

6. To modify the details of any SuperManager listed in Figure 2.17, just click the button corresponding to that SuperManager.

7. Likewise, you can delete any SuperManager by clicking the button corresponding to it, in Figure 2.17.

8. If required, you can delete multiple SuperManagers at one shot. For that, first mark the SuperManagers for deletion by selecting the check box against each SuperManager in Figure 2.17. Then, click the button in Figure 2.17.

2.9 Configuring an eG SuperManager to Manage the eG Managers in a Redundant Cluster

You can ensure that the managers in a redundant setup report to an eG SuperManager, by following the steps given below:

1. Install and configure the eG SuperManager using the procedure discussed in Section 2.1 to Section 2.3 in the case of Unix environments, or Section 2.4 in the case of Windows environments.

2. Next, you would have to ensure that the eG SuperManager is made aware of the individual eG managers that it needs to receive metrics from. In a non-redundant setup, you would typically follow the procedure detailed in Section 2.6 to achieve this. In a redundant setup however, since the primary manager is configured to receive metrics from all the other managers in a cluster, it would suffice to configure a line of communication between the the eG SuperManager and the primary manager alone – in other words, when the **configure_sm.bat** file or the **configure_sm** script (as the case may be) prompts for an IP address and port, you need to provide the IP address and port of the primary manager in the cluster, then assign a nick name to the primary manager, and exit the script/batch file.

3. The next step is to configure each of the eG managers to talk to the eG SuperManager. As you know, in a non-redundant setup, you can achieve this by executing the steps discussed in Section 2.8, on every manager. However, in the case of a redundant setup, since the primary manager shares its configuration information with the secondary managers in the clusters, it would suffice to execute the procedure detailed in Section 2.8 on the primary manager alone.

2.10 Starting and Stopping the eG SuperManager

Before starting a SuperManager, ensure the following:

- Ensure that a proper license file is available in the **/opt/egurkha/bin** directory (or the **<EGSM_INSTALL_DIR>\bin** directory)
- View the license by executing the command: **/opt/egurkha/bin/viewCert <License File Name>**. A sample output of the **viewCert** command has been provided below.

Figure 2.18: Sample license

The **IP address**, if specified, restricts the eG SuperManager to a specific host. The **Host ID**, if specified, restricts the eG SuperManager to a host that has a specific host ID. On Unix systems, run the command **hostid** to determine the host id of the system. The output of the command must match the host ID specified in the license for the eG SuperManager to start. On Windows systems, look for the physical address specification in the output of the **ipconfig /all** command. The host ID specified in the license must match one of the physical addresses of the host (ignore any dashes (-) in the physical address). The **Company Name** indicates the name of the company that is licensed to use the SuperManager, and the **Email Address** is the email ID to which license expiry mails are to be sent. The date on which the license will become invalid is set against **Expiry Date**, and the name of the **Product** and its **Version** is also displayed.

The license also controls the maximum number of individual eG managers that a SuperManager is licensed to handle. For example, in Figure 2.18, the **No. of eGManagers** is set to **5**, which means that the eG system will not allow administrators to configure more than 5 eG managers for this SuperManager.

To start a non-SSL SuperManager installed on Linux /Solaris environments, do the following:

1. From the command prompt, switch to the directory, **/opt/egurkha/bin**.
2. Next, issue the command **./start_sm**. The following messages will then appear:

```
Version : 4.0.0
Starting the eG SuperManager components ...
Please wait ...
```

3. Once the SuperManager successfully starts, the following message will appear:

```
**********************************************
The eG SuperManager started successfully!!!
**********************************************
```

To stop the eG SuperManager installed on Linux/Solaris environments, first go to the command prompt, and switch to the directory: **/opt/egurkha/bin**. Next, issue the command, **./stop_sm**. Once the SuperManager stops, the following message will appear:

```
*****************************************************************
The eG SuperManager has been successfully stopped.
*****************************************************************
```

To start a non-SSL eG SuperManager on a Windows environment, follow the menu sequence depicted by Figure 2.19:

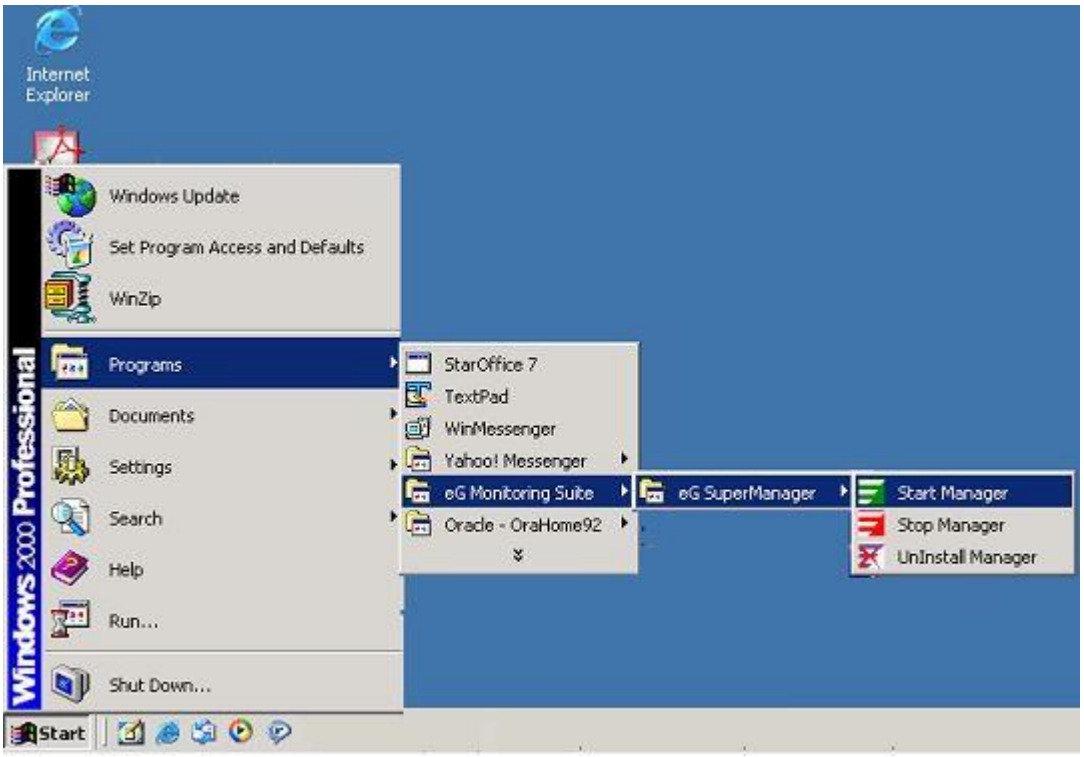


Figure 2.19: Starting the eG SuperManager

The first step towards starting a Windows SuperManager with SSL support is to SSL-enable the **start_sm.bat** script by following the steps below:

1. Open the **start_sm.bat** file (see Figure 2.19) residing in the **<EGSM_INSTALL_DIR>\lib** directory. Change the URL **http://<eGsupermanagermanagerIp>:<eGsupermanagerdefaultTCPPort>/final/servlet/StartSM** present in the last line of the batch file to **https://<eGmanagerIp>:<eGmanagerPort>/final/servlet/StartSM** (see Figure 2.20).

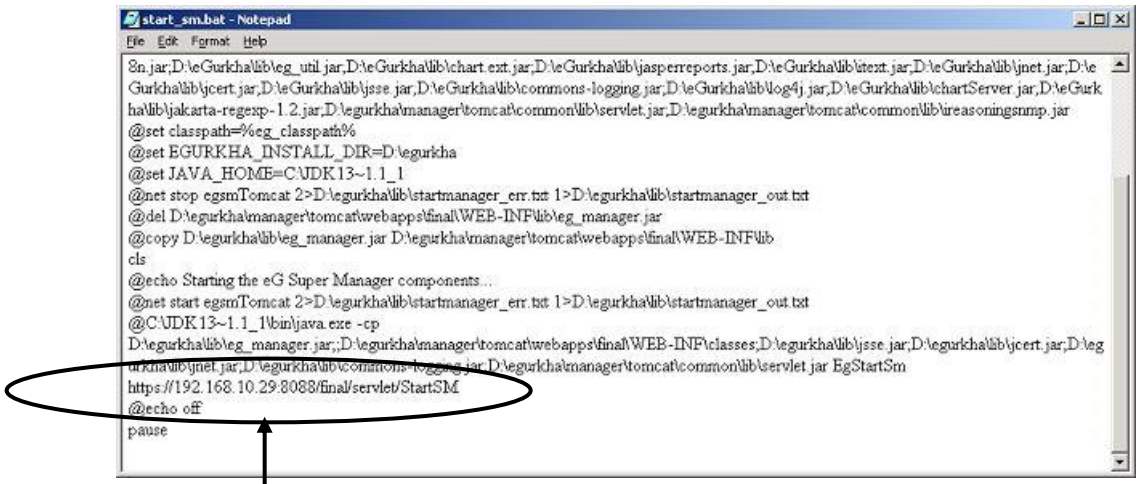


Figure 2.20: SSL-enabling the start_sm script

SSL-enabling the start_sm script by making the indicated change

2. Start the **World Wide Web** (WWW) **Publishing** service using the **Services** window that appears when you follow the menu sequence: **Start -> Programs -> Administrative Tools -> Services**.
3. Then, start the eG manager by following the menu sequence depicted by Figure 2.19.

To stop the SuperManager, follow the menu sequence depicted by Figure 2.21:

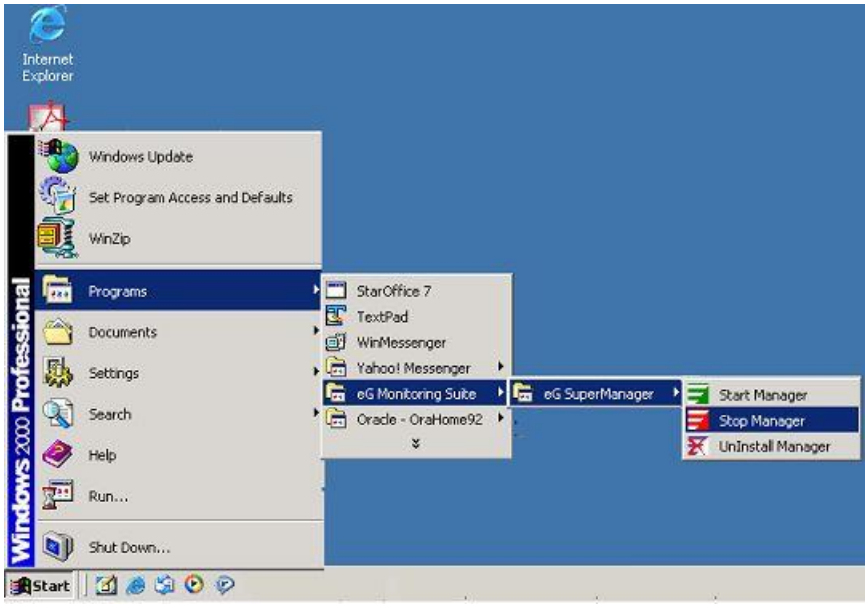


Figure 2.21: Stopping the eG SuperManager

Note:

When the eG SuperManager successfully starts, the following processes will be started up:

- A Tomcat process named **eGsmTomcat**
- A recovery process named **eGMon_sm**

In case of a non-SSL SuperManager, the **eGMon_sm** automatically detects the failure of the **eGsmTomcat** service and restarts it. However, in case of an SSL-enabled SuperManager, the 'auto recovery' capability of the **eGMon_sm** service is governed by the **SSL_RESTART** flag in the **[MANAGER_RESTART]** section of the **<EGSM_INSTALL_DIR>\manager\config\eg_supermanager.ini** file. By default, this flag is set to **false**, indicating that, by default, the **eGMon_sm** service will not automatically start the Tomcat service in the event of the latter's failure. To enable the capability, set the **SSL_RESTART** flag to **true**. However, if SSL has been enabled **with pass phrase**, the **SSL_RESTART** flag **should not be set to true**, since the **eGMon_sm** script does not provide a pass phrase when it restarts the eG SuperManager.

Note:

- Once the SuperManager starts, it polls each of the managers it is configured to receive performance data from, and tries to figure out if that manager is in a redundant cluster. If so, it then downloads the details of the secondary managers in the cluster from this manager and stores the details in the **eg_supermanager.ini** file in the **<SUPERMANAGER_INSTALL_DIR>\manager\config** directory.
- Typically, the eG SuperManager checks each of the configured eG managers for status updates, every 12 hours.

Note:

In Windows environments, executing the SuperManager in the **mgrdebugon** mode automatically triggers error logging. The steps involved in this process are detailed below:

1. Stop the eG SuperManager.
2. Run the **mgrdebugon.bat** file in the **<EGSM_INSTALL_DIR>\lib** directory by double-clicking on it.
3. Finally, restart the eG SuperManager.
4. Upon restarting, a file named **catalina.log** will be automatically created in the **<EGSM _INSTALL_DIR>\manager\tomcat\logs** directory. This is the file to which critical processes involved in the operation of the SuperManager are logged.
5. The errors (if any) will be logged in the **error_log** file that will be automatically created in the **<EGSM_INSTALL _DIR>\manager\logs** directory.

You can 'switch off' error logging if so required, by running the **mgrdebugoff.bat** file in the **<EGSM_INSTALL_DIR>\lib** directory.

2.11 Uninstalling the eG SuperManager

To uninstall the eG SuperManager on Solaris environments, follow the steps given below:

1. Stop the SuperManager.
2. Execute the **pkgrm eGsm** command from the **/opt/egurkha/bin** directory.
3. The details of the **eGsm** package installed on the host will then be displayed:

```
The following package is currently installed:
eGsm            eG SuperManager
                   (sparc) version 4.0
```

4. You will then be requested to confirm removal of the **eGsm** package.

```
Do you want to remove this package? y
```

   Press **y** to confirm the uninstallation of the eG SuperManager package.

5. When the uninstallation ends successfully, the following message will appear:

```
Removal of <eGsm> was successful.
```

On Linux, the **/opt/egurkha** directory has to be manually removed to uninstall the eG SuperManager.

To uninstall the SuperManager installed on a Windows host, do the following:

1. Stop the SuperManager.
2. Begin uninstallation by following the menu sequence depicted by Figure 2.22.

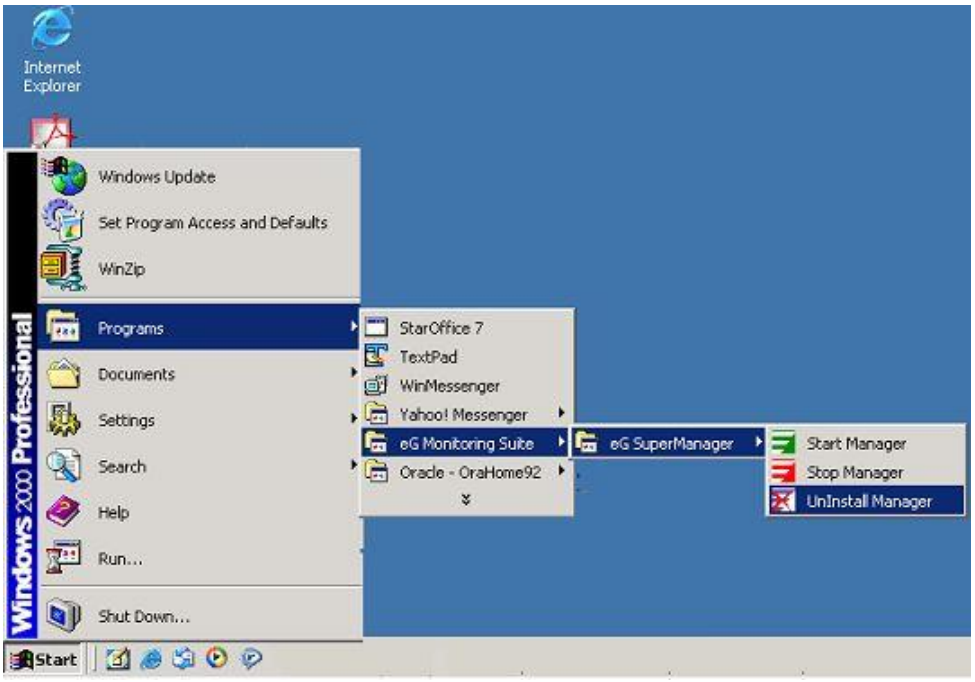


Figure 2.22: Uninstalling the eG SuperManager

3. Figure 2.23 will then appear. Select the **Remove** option from Figure 2.23, and then, click the **Next** button.

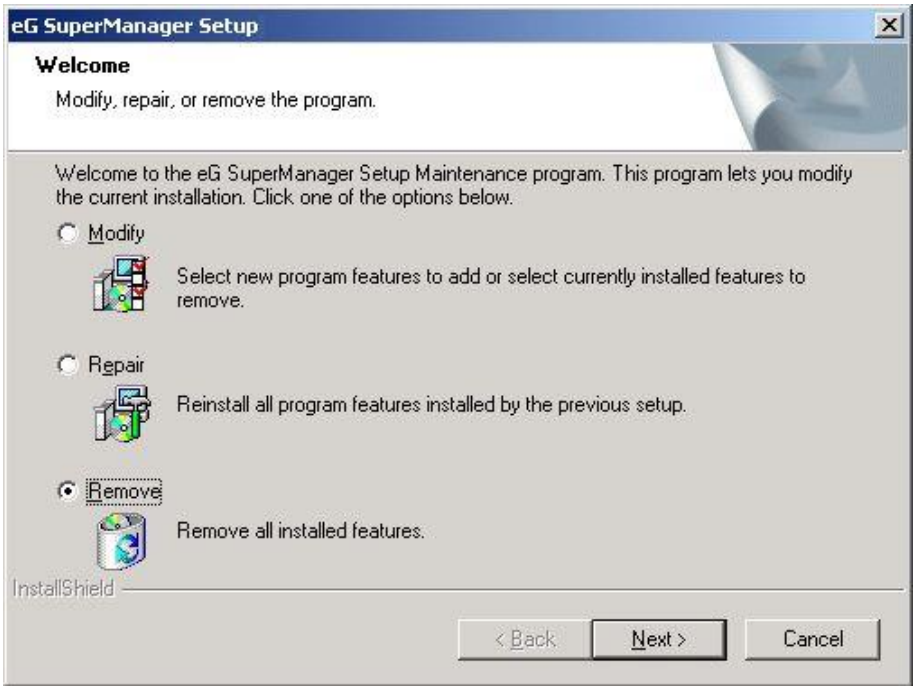


Figure 2.23: Selecting the option to remove the SuperManager

4. Next, click the **OK** button in Figure 2.24 to begin uninstallation.

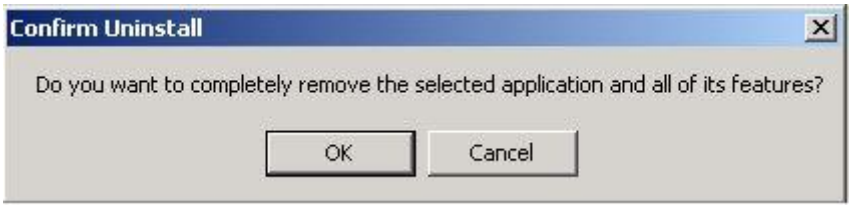


Figure 2.24: Confirming removal of the SuperManager

3

Working with the eG SuperManager

To understand the workings of the eG SuperManager better, let us take the case of an environment that is distributed across New York and Ohio. Individual managers have been installed in New York and Ohio respectively, to monitor the status of IT operations in each of these cities. For an overall status report, an eG SuperManager has been configured at Chicago.

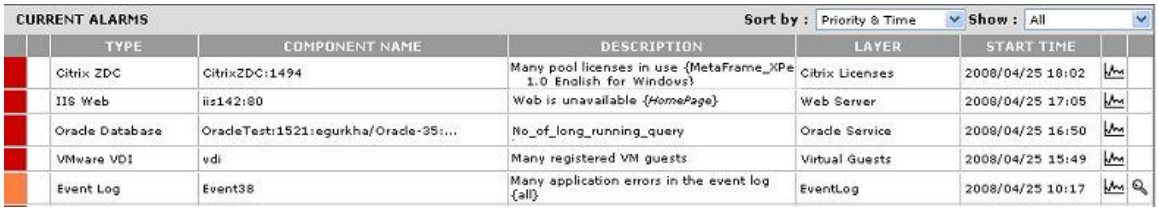
Prior to accessing the SuperManager, let us login to the NewYork and Ohio managers, to view their current status and current alarms. Figure 3.1 depicts the current issues affecting New York's performance.

CURRENT ALARMS — Sort by : Priority & Time — Show : All

TYPE	COMPONENT NAME	DESCRIPTION	LAYER	START TIME
Citrix MF XP	mfxp_egurkha22:1494	Process is not running {ImaServer}	Application Processes	2008/04/25 18:10
Apache Web	web:7077	Web is unavailable {*HomePage*}	Web Server	2008/04/25 17:01
WebLogic	探险家探险家:7011	Process not running {/任□?□}	Application Processes	2008/04/25 16:11
VMware ESX	vm_esx	Virtual guest is powered off {virtual-appliance}	Virtual Guests	2008/04/25 15:49
Microsoft SQL	SQL:1433	Many connections	MS SQL Service	2008/04/25 18:06

Figure 3.1: Alarms reported by the New York manager

Figure 3.2 depicts the issues reported by the Ohio manager.



CURRENT ALARMS — Sort by : Priority & Time — Show : All

TYPE	COMPONENT NAME	DESCRIPTION	LAYER	START TIME
Citrix ZDC	CitrixZDC:1494	Many pool licenses in use {MetaFrame_XPe 1.0 English for Windows}	Citrix Licenses	2008/04/25 18:02
IIS Web	iis142:80	Web is unavailable {*HomePage*}	Web Server	2008/04/25 17:05
Oracle Database	OracleTest:1521:egurkha/Oracle-35:...	No_of_long_running_query	Oracle Service	2008/04/25 16:50
VMware VDI	vdi	Many registered VM guests	Virtual Guests	2008/04/25 15:49
Event Log	Event38	Many application errors in the event log {all}	EventLog	2008/04/25 10:17

Figure 3.2: Alarms reported by the Ohio manager

Next, login to the eG SuperManager at Chicago. To do so, type the URL of the eG SuperManager. Figure 3.3 will then appear.

Figure 3.3: The **SuperManager** login

Note:

- The default language of the eG SuperManager is English. Accordingly, the **defaultLanguage** parameter in the **[DEFAULT_LANGUAGE]** section of the **eg_supermanager.ini** file (in the **<EG_SUPERMANAGER_INSTALL_DIR>\manager\config** directory) is set to **en_US** by default. To override this default setting, you can change the **defaultLanguage** parameter to reflect any other language of your choice. For instance, to set Chinese as the default language, specify **ch_CH** against **defaultLanguage**, where **ch** is the language code and **CH** the country code. Then, save the file and restart the SuperManager. When this is done, then the next time you connect to the SuperManager, you will find that the field labels in the SuperManager login page (i.e, **Error! Reference source not found.**) have changed to Chinese.
- The **[COUNTRY_AND_LANGUAGE_CODE]** section of the **eg_supermanager.ini** file provides a list of languages that are largely preferred by users and their corresponding country and language code notations. While setting a **defaultLanguage**, you can refer to this section for the code of the language you prefer.

3.1 An egsm User's View

The default user to the eG SuperManager is *egsm* with password *egsm*. This user has unrestricted rights to the SuperManager, and can view alarms/status information pertaining to all the individual managers that are being managed by the eG SuperManager.

When you login to the SuperManager as *egsm*, you will be welcomed by the **CURRENT ALARMS** window (see Figure 3.4).

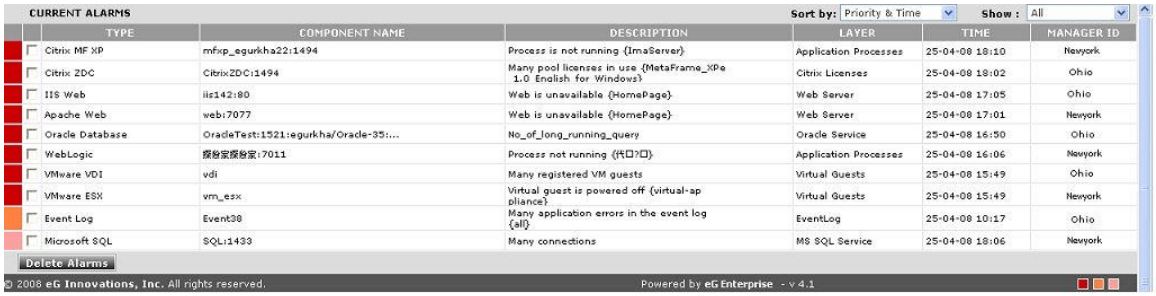


Figure 3.4: Consolidated list of alarms pertaining to the Ohio and New York managers

Figure 3.4 displays the consolidated list of issues encountered by both the New York and Ohio managers. The alarm details include the component type that has suffered a performance degradation, the component name, a brief description of the issue, the problem layer, the problem date and time, and the manager that reported the issue. For additional alarm information, simply move your mouse pointer over the alarm. Additional details such as the problem test and last measure value will appear as depicted by Figure 3.5.

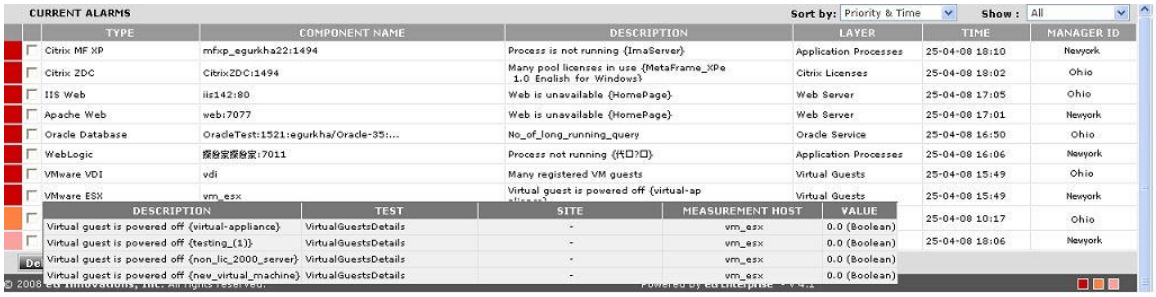


Figure 3.5: Additional alarm details

Note:

If you do not want to move your mouse pointer over an alarm every time you wish to view additional alarm details, then, you can configure the eG SuperManager console to automatically display the additional details as soon as the **CURRENT ALARMS** window opens. To achieve this, you will have to alter the profile of the *egsm* user. For that, click the **Profile** button at the right, top corner of the *eG SuperManager* home page (see Figure 3.11). Then, set the **Mouse over in alarm page** flag in the **User Profile** page that appears to **No**.

By default, the alarms displayed in the **CURRENT ALARMS** window are sorted in the descending order of the problem priority and problem date/time. This is why the **Sort by** list in Figure 3.5 is set to **Priority & Time** by default. Typically, the alarms can be sorted in the ascending order of any of the following options – i.e., **Component Type**, **Component Name**, **Description**, **Layer**, or **Start Time**.

Also, though the **CURRENT ALARMS** window displays all the unresolved issues across managers by default, you can choose to view a specific priority of alarms alone by selecting the alarm priority of interest to you from the **Show** list. The default selection in the **Show** list is **All**. To view only the critical alarms, select the **Critical** option from the list. Figure 3.6 will then appear.

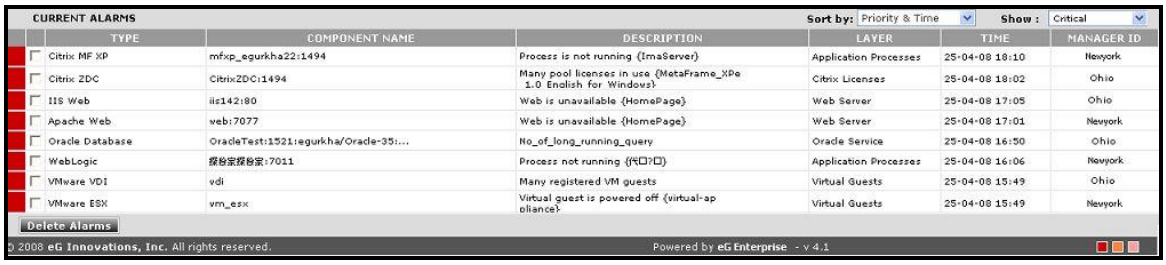


Figure 3.6: Viewing the Critical alarms alone

Likewise, you can choose to view **Critical & Major** alarms, **Major** alarms alone, or simply the **Minor** alarms.

Also, by default, the *egsm* user does not have the right to delete alarms. If the *egsm* user wants to delete alarms, then the user should first reconfigure his/her profile to enable the alarm deletion capability. For that, click on the **Profile** button available at the top of Figure 3.11, and then set the **Allow Delete** flag in the **USER PROFILE** page that appears to **Yes**. Once this is done, then the **CURRENT ALARMS** window will display a **Delete Alarms** button (see Figure 3.6); moreover, every row of alarm information in Figure 3.6 will be preceded by a check box. To delete an alarm, select the check box corresponding to the alarm to be deleted, and then click the **Delete Alarms** button.

Like alarm deletion, the *egsm* user does not possess alarm acknowledgement privileges too by default - in other words, the *egsm* user cannot submit acknowledgement descriptions for any alarm by default. However, the *egsm* can view the acknowledgement descriptions provided by other users with acknowledgement rights. To do this, the *egsm* will only have to move his/her mouse pointer over the symbol that appears just before the alarm of interest in the **CURRENT ALARMS** window. For instance, Figure 3.7 below displays the history of acknowledgement descriptions provided by the *admin* and *supermonitor* users to an alarm generated by the eG manager, 192.168.10.99, on the Web server, *web:80*.

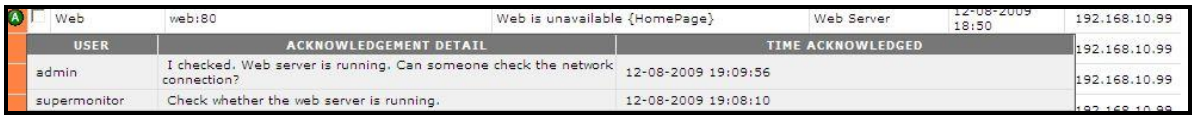


Figure 3.7: Viewing the acknowledgement descriptions submitted by other users

If the alarm acknowledgement capability is enabled for an *egsm* user, then, he/she can post his/her own comments on the same alarm. By acknowledging an alarm, a user can indicate to other users to the supermanager that the issue raised by an alarm is being attended to. In fact, if need be, the user can even propose a course of action using this interface.

To enable the alarm acknowledgement capability of an *egsm* user, login o the eG SuperManager interface as *egsm,* click on the **Profile** button available at the top of Figure 3.11, and then set the **Allow Acknowledgement** flag in the **USER PROFILE** page that appears to **Yes**. Once this is done, an **Acknowledge** button will appear in the **CURRENT ALARMS** window.

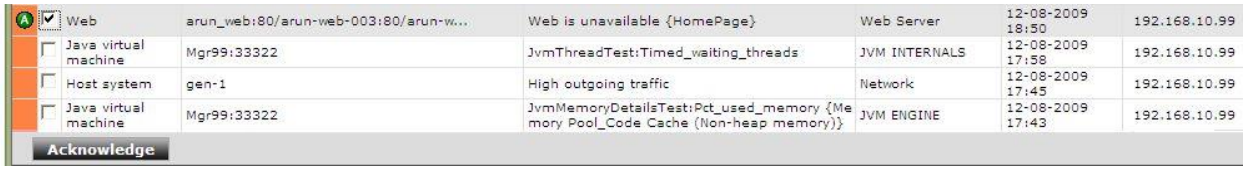


Figure 3.8: Acknowledging an alarm

To acknowledge an alarm, select the check box corresponding to it and click on the **Acknowledge** button. Then, when Figure 3.9 appears, provide an **Acknowledgement** description and click the **Submit** button therein to save the changes.

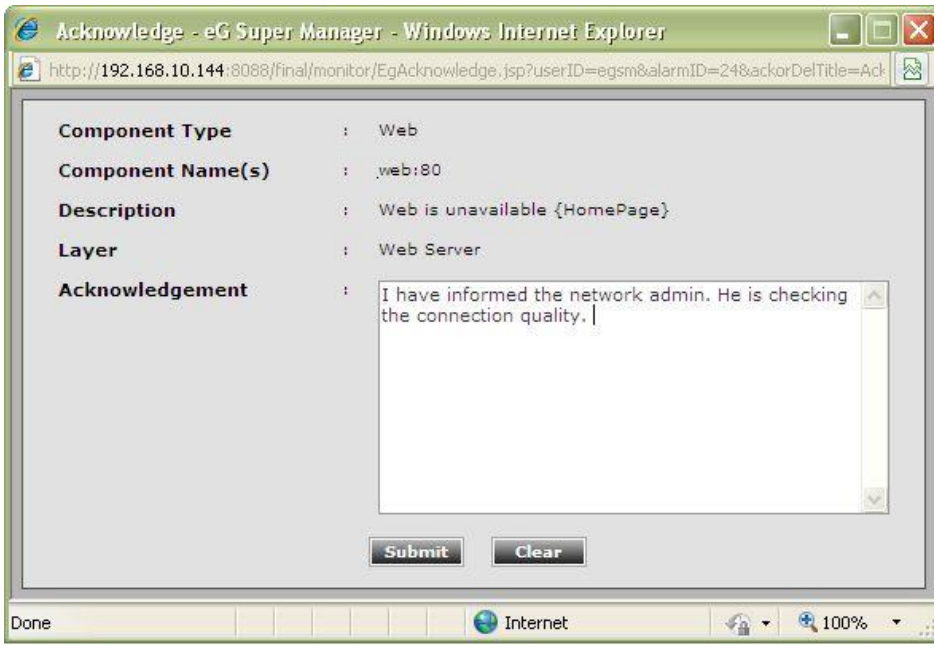


Figure 3.9: The egsm user providing acknowledgement description

If you now return to the **Alarms** window, you will find your own description added to the acknowledgement history that pre-exists for that alarm (see Figure 3.10).

Figure 3.10: The egsm user's acknowledgement added to the history

Similarly, an *egsm* user can remove the acknowledgement descriptions that have been provided by him/her. However, the *egsm* **cannot edit/remove the descriptions submitted by other users**. In fact, any users to the SuperManager can alter/remove their own acknowledgement descriptions only, but not that of other users.

To perform further diagnosis, click on an alarm in Figure 3.6. This will lead you to the problem layer, test, and measurements in the corresponding manager's console.

Once you close the **CURRENT ALARMS** window in the eG SuperManager, a home page appears (see Figure 3.11). For an *egsm* user, this page reveals the current status of the Newyork and Ohio managers in terms of the total number of measurements that have been collected by each of the managers, the percentage of measurements in the Critical, Major, Minor, Unknown, and Normal state, the number of unresolved issues, and the average and maximum durations for which a problem had remained unresolved.

Figure 3.11: The **SuperManager** Home

Clicking on a manager's name will lead you to the monitor home page of the corresponding manager.

Also, using the eG SuperManager console, the *egsm* user can change his/her password. To achieve this, click on the **Profile** button at the right top corner of Figure 3.11. Figure 3.12 that then appears facilitates the password change.

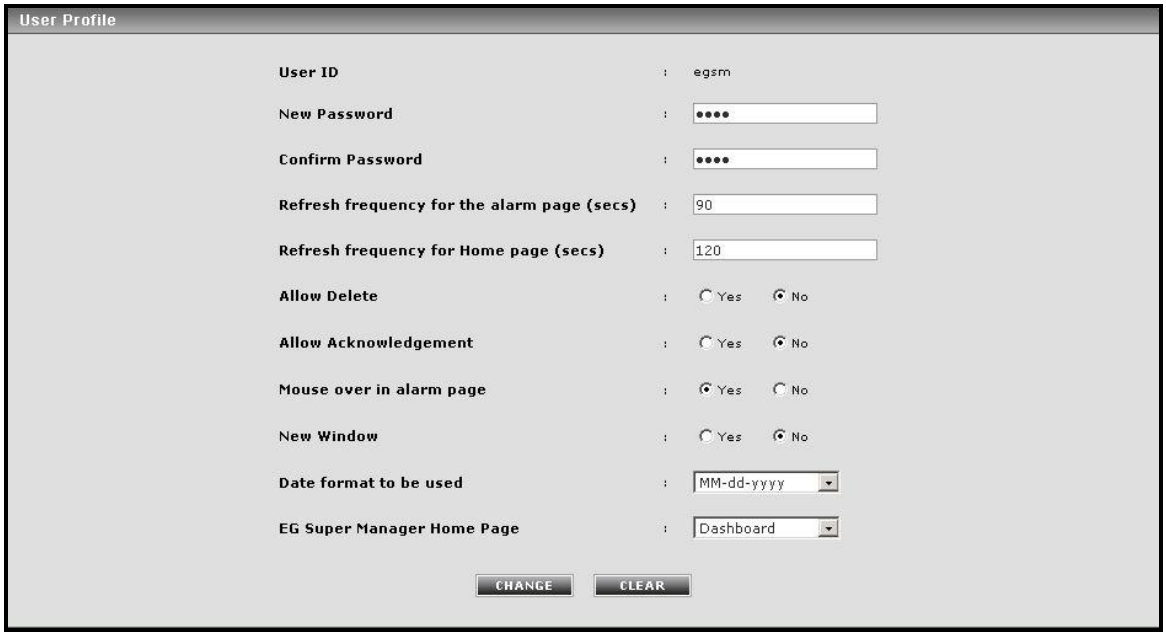


Figure 3.12: Changing the password of the *egsm* user

Moreover, since the *egsm* user is the super-user of the *eG SuperManager*, he/she has the right to not only modify the *egsm* password, but also indicate how the eG SuperManager console should function by default. Towards this end, the *egsm* user can set the following additional parameters using Figure 3.12:

- **Refresh frequency for the alarm page:** The frequency with which the **CURRENT ALARMS** window is to be refreshed
- **Refresh frequency for the home page:** The frequency with which the SuperManager home page is to be refreshed
- **Allow Delete:** Set this flag to **Yes** to enable the *egsm* to delete alarms. By default, this flag is set to **No**.

- **Allow Acknowledgement**: Set this flag to **Yes** to enable the *egsm* to acknowledge alarms. By default, this flag is set to **No**.
- **Mouse over in alarm page**: By default, this flag is set to **Yes**, meaning that additional alarm information will be available to a user only when he/she moves the mouse pointer over an alarm in the **CURRENT ALARMS** window. Setting this flag to **No** ensures that users are able to view additional details of all alarms as soon as the **CURRENT ALARMS** window opens, without having to move the mouse pointer.
- **New Window:** By default, this flag is set to **No**; this indicates that when the name of a manager is clicked on in the SuperManager home page, then, by default, the monitoring console of the corresponding eG manager opens in the same window as the eG SuperManager console. You can set this flag to **Yes**, so that whenever a manager is clicked on, the monitoring console opens in a different window.
- **Date format to be used**: Indicate the date format to be used across the eG SuperManager console.
- **EG SuperManager Home Page**: If need be, you can set the **CURRENT ALARMS** window as the home page of the eG SuperManager, instead of the dashboard, by selecting the **Current Alarms** option from this list. In such a case, an additional **Dashboard** option appears at the right, top corner of the eG SuperManager console. Clicking on this option will take you to the SuperManager dashboard.

3.2 The View Received by Other Users

Besides the default user *egsm*, the eG SuperManager supports all other users registered with each of the individual managers reporting to it. The access rights defined for them in their respective managers will automatically apply. Accordingly, the alarms displayed in this page are specific to the services/segments/components/zones being monitored by the current user. For instance, when a user who is assigned the *AlarmViewer* role logs into the SuperManager interface, he/she will receive a home page that only **lists the alarms** pertaining to those infrastructure elements that are associated with that user (see Figure 3.13).

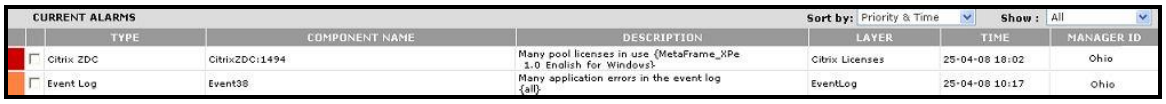


Figure 3.13: The AlarmViewer role

Moving your mouse pointer over an alarm will reveal additional alarm details along with the following options (see Figure 3.14):

- An option to view the history of fixes to the issue indicated by the alarm
- An option to add more fixes

Figure 3.14: The details of an alarm of an AlarmViewer along with Feedback and History options

In case of a *SuperAlarmViewer*, the only difference is that the list of alarms displayed in the home page will pertain to the monitored environment as a whole.

Also, if a particular user has been configured to view only a specific priority of alarms then this window will display alarms pertaining to that priority only. In addition, if the user has set a language preference in the eG manager, then the alarms window will display details in that language only.

Moreover, if the saWme user name and password are registered with multiple managers, then the eG SuperManager alarms window will display the all the alarms pertaining to that user across all the managers to which he/she has access. For example, consider the case of user *john* who has been created both in manager Newyork and manager Ohio, with *Monitor* user privileges. While the Citrix MF XP has been assigned to the user *john* registered with the New York manager, the VMware VDI server has been assigned to the user *john* created in the Ohio manager. However, if the passwords of both the *johns* were the same, then the resulting **CURRENT ALARMS** window will list alarms pertaining to the Citrix server and the VMware VDI server (see Figure 3.15).

Figure 3.15: Alarms pertaining to both the *johns*

However, if user *john* has set a different language preference in the Newyork and Ohio managers, then the eG SuperManager randomly picks one of the languages and displays alarms in that language only.

Upon closing the alarms window, the eG SuperManager home page appears. If a *Monitor* user registered with one of the eG managers logs into the eG SuperManager interface, then the home page will display the IP address and SSL-enabled status of only that manager to which the user has access. Also, the measurement status, the count of unresolved issues, and the duration statistics displayed in the home page will only pertain to the specific segments/components/zones/services (if any) that have been assigned to the login user for monitoring. For instance, if a user registered with the *Newyork* manager logs into the eG SuperManager console, then the home page will be as depicted by Figure 3.16.

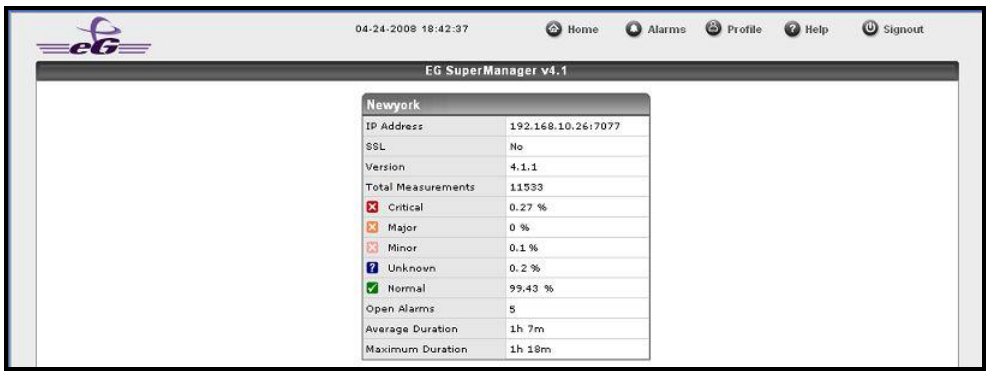


Figure 3.16: Home page of a user who is registered with only the Newyork manager

Now, assume that a user named *repo* (password: *repo*) with 'Reporter-only' rights has been configured in both the NewYork and Ohio managers. When this user logs into the eG SuperManager interface, the **CURRENT ALARMS** window will not appear, as a Reporter user does not have access to the alarm information pertaining to the target environment. However, the home page appears, but lists only the total number of measurements returned by both the managers (see Figure 3.17).

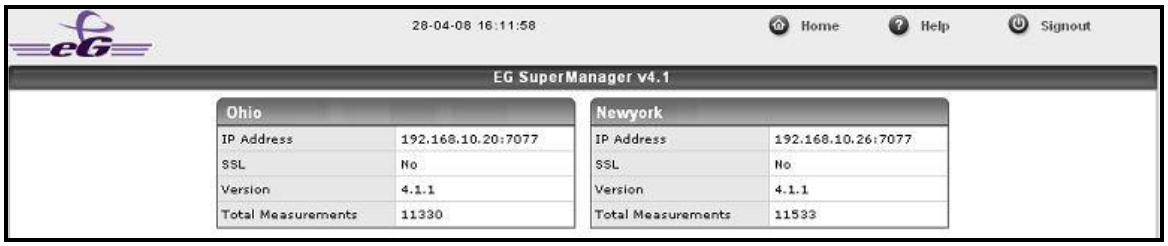


Figure 3.17: The home page of a 'Reporter-only' user

Clicking on the name of a manager in Figure 3.17, will lead you to the eG Reporter interface of that manager.

Similarly, if a user who has been assigned an *Admin* role logs into the eG SuperManager, then clicking on the manager name in the eG SuperManager home page will open the administrative interface of the corresponding manager, and not the monitoring console.

Now, if a component is managed with the same nick name by both the Ohio and NewYork managers, then, in the event of a problem with both the components, the component will appear twice in the **CURRENT ALARMS** window of the SuperManager interface. You can however differentiate between the two components using the **MANAGER ID** provided against every alarm displayed in the **CURRENT ALARMS** windows.

Note:

In order to avoid duplications, it is recommended that you develop standard naming conventions that would be applicable to the users/roles/components across all the managers in your environment

If the same user name and password are registered with multiple managers, then the eG SuperManager home page will display the state of that user's environment in each of the managers to which he/she has access. However, if such a user has set a different language preference in every manager, then the eG SuperManager randomly picks one of the languages and displays details in the SuperManager dashboard in that language only.

If a user who has been assigned the *AlarmViewer* or the *SuperAlarmViewer* role logs into the eG SuperManager interface, then he/she will only be able to view the **CURRENT ALARMS** window only and not the home page.

3.3 How the eG SuperManager Supports Redundant Clusters?

The eG SuperManager is capable of managing the eG managers in a redundant cluster. This ensures that there is no single point of failure of the eG managers, which consequently ensures that the SuperManager provides a consolidated view across managers, 24 x 7.

As long as the redundant cluster is operating normally – i.e., as long as the primary manager is up and running – the eG SuperManager receives status updates from the primary manager. Therefore, if you login as *egsm*, then the eG SuperManager home page will display a section bearing the nick name of the primary manager, which will provide an overview of the performance of the target environment managed by the redundant cluster. The **IP Address** filed in this section will display the IP address of the primary manager.

Similarly, the **CURRENT ALARMS** window for the *egsm* user will display alarms pertaining to all the managed components in the target environment, and the **MANAGER ID** column of each alarm will carry the nick name of the primary manager only.

Now, assume that the primary manager fails. The eG SuperManager automatically detects this failure when it polls the primary manager for status updates, and instantly switches to one of the secondary managers in the cluster for information. This secondary manager will then start feeding the eG SuperManager with performance and problem information related to the target infrastructure, so that the eG SuperManager remains unaffected by the non-availability of the primary manager. Moreover, the secondary manager that takes over from the primary also assumes the nick name that was originally assigned to the primary manager at the time of configuring the primary-supermanager communication. Therefore, if an *egsm* user logs in, he/she will still be able to view the current status of the target environment in the home page, in the section with the nick name of the primary manager. However, the manager **IP Address** displayed in this section will be that of the secondary manager, and not the primary manager. Alarms too will continue to appear in the **CURRENT ALARMS** window of the eG SuperManager carrying the ID of the primary manager.

You can login to the eG SuperManager as any user who is registered with the primary manager. As always, the details displayed in the eG SuperManager will differ according to the monitoring rights of that user.

3.4 Frequently Asked Questions on eG SuperManager and Redundant Clusters

- **I have added a new secondary manager to my cluster. How do I make the eG SuperManager aware of this change?**

  Every 12 hours, the eG SuperManager polls the primary manager in the cluster for status information, performance data, and changes in cluster configuration. If changes exist, the SuperManager then downloads the latest details from the primary manager. This implies that if you have added a new secondary manager to your cluster, then the eG SuperManager is automatically updated with this change the next time it contacts the primary manager for information. Alternatively, you can restart the eG SuperManager to ensure that the changes are downloaded immediately.

- **I have converted a secondary manager in my cluster into a primary manager. How do I ensure that the eG SuperManager now talks to the new primary manager and not the old one?**

  To ensure this, you will have to follow the steps given below:

  - Edit the **eg_supermanager.ini** file in the **<EGSM_INSTALL_DIR>\manager\config** directory.
  - In this file, you will find a section that carries the nick name of the old primary manager as its title. For eg., if the original primary manager in the cluster was nick named *manager1*, then the **eg_supermanager.ini** file will comprise of a section titled *[MANAGER1]*.

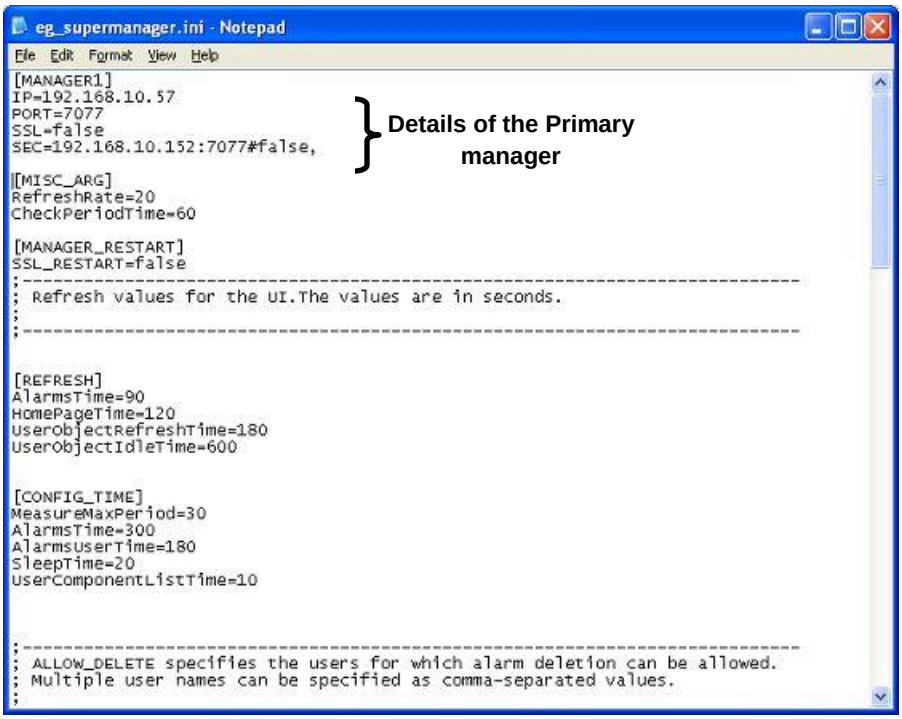


Figure 3.18: The eg_supermanager.ini file

- Details of the primary manager, such as its IP address, port, SSL status, and secondary manager list will be stored in this section. Since these details have changed now because of converting a secondary into the primary, you will have to manually update the **eg_supermanager.ini** file to reflect the changes. Hence, keeping the nick name of the primary manager as is, proceed to change the value of the **IP**, **PORT**, and if need be, the **SSL** parameters in the section.
- Finally, save the file and restart the eG SuperManager.
- When this is done, the SuperManager connects to the new primary manager, downloads details about the additional secondary manager that is now available in the environment, and updates the **SEC** parameter in the **eg_supermanager.ini** file with it.

4

Conclusions

The **eG SuperManager** allows the eG Enterprise suite to efficiently handle large IT infrastructures spanning multiple geographies, different domains that are autonomous, and disparate networks. Besides scalability, the eG SuperManager brings the following additional benefits into the eG Enterprise suite:

- Automatically consolidates the measure data distributed across a wide area, and displays it in a single interface; it thus saves the time, labor, and cost involved in manual data collection and consolidation;
- Does not require any backend support, thereby considerably reducing database maintenance overheads;
- Is a flexible solution that can easily adapt to heterogeneous configurations of eG managers;
- Reduces the bandwidth usage across geographies

For more information about the eG SuperManager and the eG Enterprise suite, contact sales@eginnovations.com.