J2EE Agent Guide

Access Manager 3.2

May 2012

Legal Notice

Contents

About This Guide

This guide describes the J2EE Agents and explains how to install, configure, and manage them:

Audience

This guide is intended for Access Manager administrators. It is assumed that you have knowledge of evolving Internet protocols, such as:

- Extensible Markup Language (XML)
- Simple Object Access Protocol (SOAP)
- Security Assertion Markup Language (SAML)
- Public Key Infrastructure (PKI) digital signature concepts and Internet security
- Secure Socket Layer/Transport Layer Security (SSL/TLS)
- Hypertext Transfer Protocol (HTTP and HTTPS)
- Uniform Resource Identifiers (URIs)
- Domain Name System (DNS)
- Web Services Description Language (WSDL)

Feedback

We want to hear your comments and suggestions about this guide and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation.

Documentation Updates

For the most recent version of the *Access Manager J2EE Agent Guide*, visit the NetIQ Access Manager Documentation Web site (http://www.netiq.com/documentation/novellaccessmanager32).

Additional Documentation

Before proceeding, you should be familiar with the *NetIQ Access Manager 3.2 SP1 Installation Guide* and the *NetIQ Access Manager 3.2 Setup Guide*, which provide information about setting up the Access Manager system.

NOTE: Contact namsdk@novell.com for any query related to Access Manager SDK.

---

1 Installing the J2EE Agents

The J2EE Agents allow you to use roles and other types of policies to restrict access to specific application modules and Enterprise JavaBeans. These agents leverage the Java Authentication and Authorization Service (JAAS) and Java Authorization Contract for Containers (JACC) standards for Access Manager-controlled authentication and authorization to Java Web applications and Enterprise JavaBeans.

NOTE: You cannot upgrade J2EE Agents from version 3.1 to 3.2. You must perform a fresh installation of version 3.2 for J2EE Agents.

Access Manager currently has J2EE agents for JBoss, WebLogic, and WebSphere servers. For information on the supported operating systems, see Table 1-1, “Software Requirements,” on page 10.

This section has the following information:

1.1 Overview of the J2EE Agents

Users of application servers, such as J2EE servers, commonly fall into one of three abstract roles: buyer, seller, or administrator. For example, a rental car company might apply a variety of Enterprise JavaBeans (EJB) components that offer different products and services to clients. One service could be a specific component that enables a Web-based reservation process. In this case, the customer could access a Web site to reserve a rental car. The seller could access a site that provides a list of available cars and prices. Then the administrator could access a site that tracked inventory and maintenance schedules. These components provide the basic business services for the application to function and the tasks they accomplish require a security policy to enforce appropriate use of such services.

Using the deployment descriptors, the application developer can set up a method to protect the components by using abstract security role names. For example, there can be a role called Service Representative, which protects the component that creates a rental agreement. Similarly, there can be a role called Approver, which protects the component that approves the agreement. Although these

roles convey the intent of the application vendor or developer to enforce such security policies, they are not useful unless these abstract role names are mapped to real life principals such as actual users or actual roles.

1.2 Overview of the Sample Payroll Application

NetIQ provides a test application, `PayrollApp.ear`, that is copied to the J2EE server during installation of the J2EE Agents.

This sample payroll application is configured to grant access based on whether the user has an Employee role or a Manager role.

For more information on deploying and using the sample payroll application, see Chapter 7, "Deploying the Sample Payroll Application," on page 93

1.3 Prerequisites

- Make sure that the system on which you want to install J2EE Agent does not have any other Access Manager components installed on it.
- You must have a static IP address.

  If you do not have a static IP address and the address assigned at boot changes, the J2EE Agent and the Administration Console can no longer communicate with each other.

1.4 Software Requirements

Table 1-1 *Software Requirements*

Requirements	JBoss	WebSphere	WebLogic
Application Software	JBoss 4.2.3 The JBoss server package does not ship on the SUSE Linux Enterprise Server (SLES) installation media. To download and install JBoss version 4.2.3, see (http://www.jboss.org/jbossweb/downloads).	WebSphere 6.1 and 7.0	BEA WebLogic 9.2 and WebLogic 10.0 **NOTE** ◆ The 64-bit version is not supported on Solaris. ◆ WebLogic 10.0 is not supported on Solaris.

Requirements	JBoss	WebSphere	WebLogic
Operating System	**Linux:** The following operating systems are supported on Linux: ◆ SUSE Linux Enterprise Server 11 SP1 (or a higher version) 64-bit platforms. ◆ Red Hat 6.2 **Windows:** The following versions of operating systems, with the latest support packs, are supported on Windows: ◆ Windows Server* 2003	**Linux:** The following operating systems are supported on Linux: SUSE Linux Enterprise Server 11 (or a higher version) 64-bit platforms. **Windows:** The following versions of operating systems, with the latest support packs, are supported on Windows: ◆ Windows Server 2003 **AIX:** AIX 5.3 **NOTE**: WebSphere 7.0 on AIX is not tested.	**Linux:** The following operating systems are supported on Linux: SUSE Linux Enterprise Server 11 (or a higher version) 64-bit platforms. **Windows:** The following versions of operating systems, with the latest support packs, are supported on Windows: ◆ Windows Server 2003 **Solaris:** Solaris 10 on SPARC*, X86, 32-bit, and 64-bit platforms. **NOTE**: There is no support for Novell Audit on Solaris for this release.
Java	JRE 1.6 or higher **NOTE**: The JBoss Agent has not been tested with the IBM* JRE.	JRE1.6 or higher	JRE 1.6 or higher
RAM	4 GB	4 GB	4 GB
Hard Disk Space	100 GB	100 GB	100 GB

NOTE: The software versions listed in the table have been tested with the product. Higher Later versions of the software might or might not work.

1.5 Installing the J2EE Agents on JBoss

This section describes the prerequisites and the procedure to install J2EE Agents on a JBoss machine. You must install the J2EE Agents on the same machine as the JBoss server. For specific requirements for J2EE Agents, see Section 1.5.1, “Prerequisites,” on page 11.

1.5.1 Prerequisites

- You must know the path where the JBoss server is installed. For more information, refer to the JBoss documentation.

- You must know the server configuration set you have selected for your JBoss server.
- Verify that the machine meets the minimum requirements. See Section 1.4, "Software Requirements," on page 10.
- If you use the custom configurations for JBoss, complete the steps in Section 1.5.2, "Installing and Configuring the JBoss Web Deployer Service," on page 12, before you proceed with the installation.

1.5.2 Installing and Configuring the JBoss Web Deployer Service

The J2EE Agents depend on the JBoss Web deployer service in order to use a custom JBoss configuration. The JBoss Web deployer service must be already installed before you proceed with the installation of the J2EE Agents.

Verifying if the JBoss Web Deployer Service is Installed

To verify if the JBoss Web deployer service is already installed, browse to the following location and check to see if a folder named `jboss-web.deployer` already exists:

`<path-to-your-custom-configuration>/deploy/`

If the folder does exist, it indicates that the JBoss Web Deployer service is installed. You can proceed with installing the agent. For more information, see Section 1.5.3, "Installing JBoss by Using the Installer," on page 13.

If the folder does not exist, refer to "Installing the JBoss Web Deployer Service" on page 12 to install the JBoss Web Deployer service.

Installing the JBoss Web Deployer Service

Follow the steps given below to install and configure the JBoss Web deployer service for your JBoss server:

1. Enter the following command to copy the JBoss Web deployer:

```
cp -R <jboss-home>/server/default <path-to-your-custom-configuration>/deploy/
```

   Replace <*jboss-home*> with the home directory of JBoss.

   Replace <*path-to-your-custom-configuration*> with the location of the custom configuration.

2. To use the custom JBoss configuration, you must disable the services that JBoss Web deployer service depends on. To disable the services, open the `<path-to-your-custom-configuration>/deploy/jboss-web.deployer/META-INF/jboss-service.xml` file and comment out lines that are within the tags. By default, JBoss depends on the following services: e `<depends></depends>`

   ```
   <depends>jboss:service=TransactionManager</depends>
   ```

   ```
   <depends>jboss.jca:service=CachedConnectionManager</depends>
   ```

3. Open the `<path-to-your-custom-configuration>/deploy/jboss-web.deployer/server.xml` file, delete content within the `<CachedConnectionValve></CachedConnectionValve>` tags.

4. Add the required security services to the `<path-to-your-custom-configuration>/conf/jboss-service.xml` file within the `<mbean></mbean>` tags. For example:

```
<mbean code="org.jboss.security.plugins.SecurityConfig"
name="jboss.security:service=SecurityConfig">
<attribute name="LoginConfig">jboss.security:service=XMLLoginConfig</
attribute>
</mbean>
<mbean code="org.jboss.security.auth.login.XMLLoginConfig"
name="jboss.security:service=XMLLoginConfig">
<attribute name="ConfigResource">login-config.xml</attribute>
</mbean>
<!-- JAAS security manager and realm mapping -->
<mbean code="org.jboss.security.plugins.JaasSecurityManagerService"
name="jboss.security:service=JaasSecurityManager">
<attribute name="ServerMode">true</attribute>
<attribute
name="SecurityManagerClassName">org.jboss.security.plugins.JaasSecurityManager
</attribute>
<attribute name="DefaultUnauthenticatedPrincipal">anonymous</attribute>
<attribute name="DefaultCacheTimeout">1800</attribute>
<attribute name="DefaultCacheResolution">60</attribute>
<attribute name="DeepCopySubjectMode">false</attribute>
</mbean>
```

5 Copy the `login-config.xml` and `standardjboss.xml` files from the `<jboss- home>/server/default/conf` location to the `<path-to-your-custom-configuration>/conf` location.

6 Copy the `ejb-deployer.xml` file from the `<jboss-home>/server/default/deploy/` location to the `<path-to-your-custom-configuration>/deploy` location.

7 Specify the following commands to copy the additional JAR files in sequence:

```
cd default/lib/
```

```
cp jboss.jar jboss-j2ee.jar jbosssx.jar servlet-api.jar
```

```
jsp-api.jar jbossws* el-api.jar jboss-ejb3x.jar <path-to-your-custom-
configuration>/lib
```

8 Restart the JBoss application server.

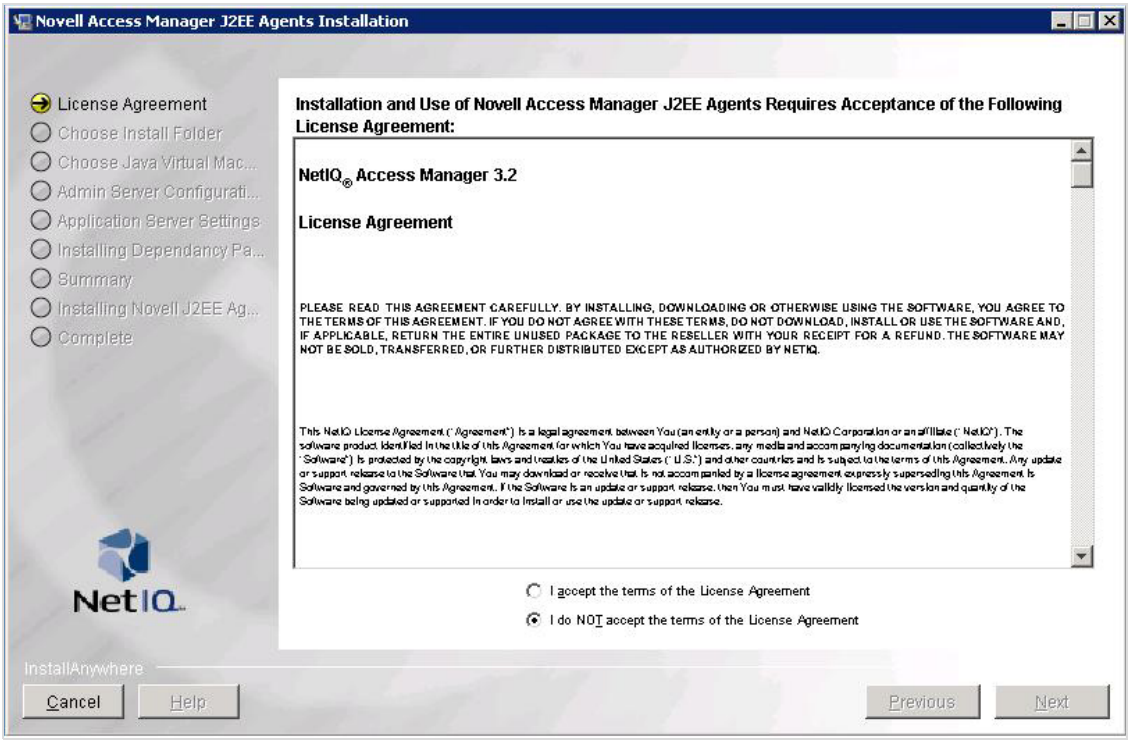
9 Proceed with the installation of the JBoss Agent. For more information, see Section 1.5.3, "Installing JBoss by Using the Installer," on page 13.

1.5.3 Installing JBoss by Using the Installer

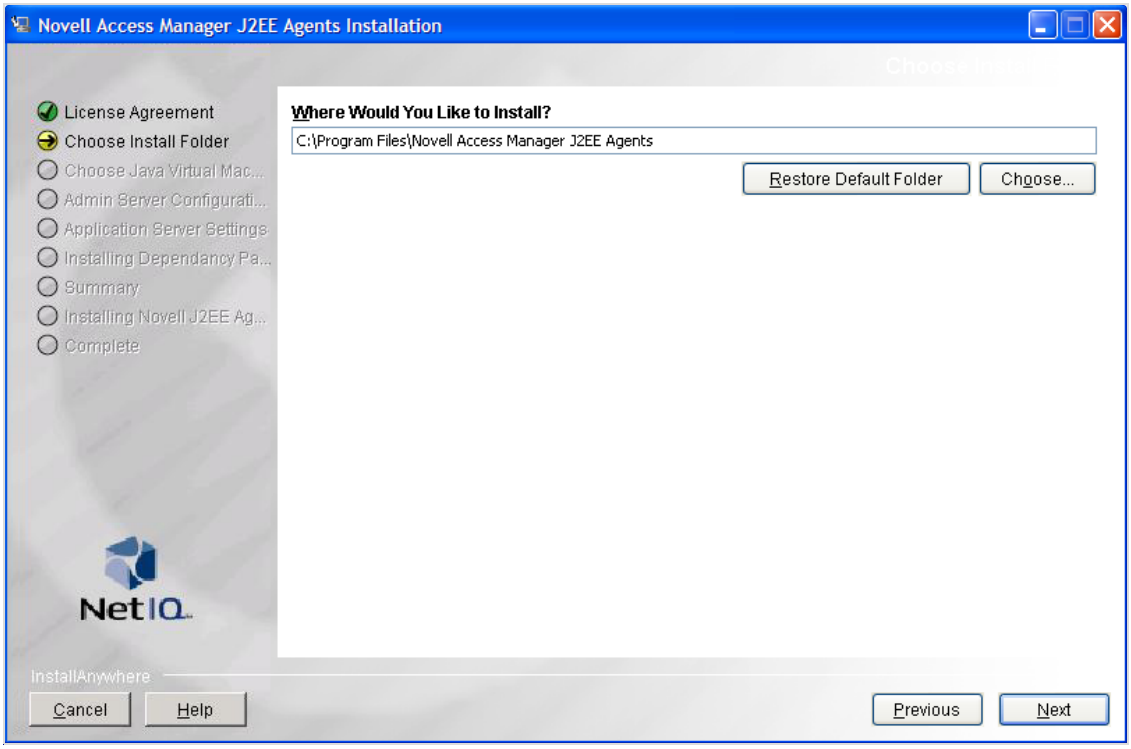
1 If JBoss is running, stop JBoss.

2 Download and execute the agent installer.

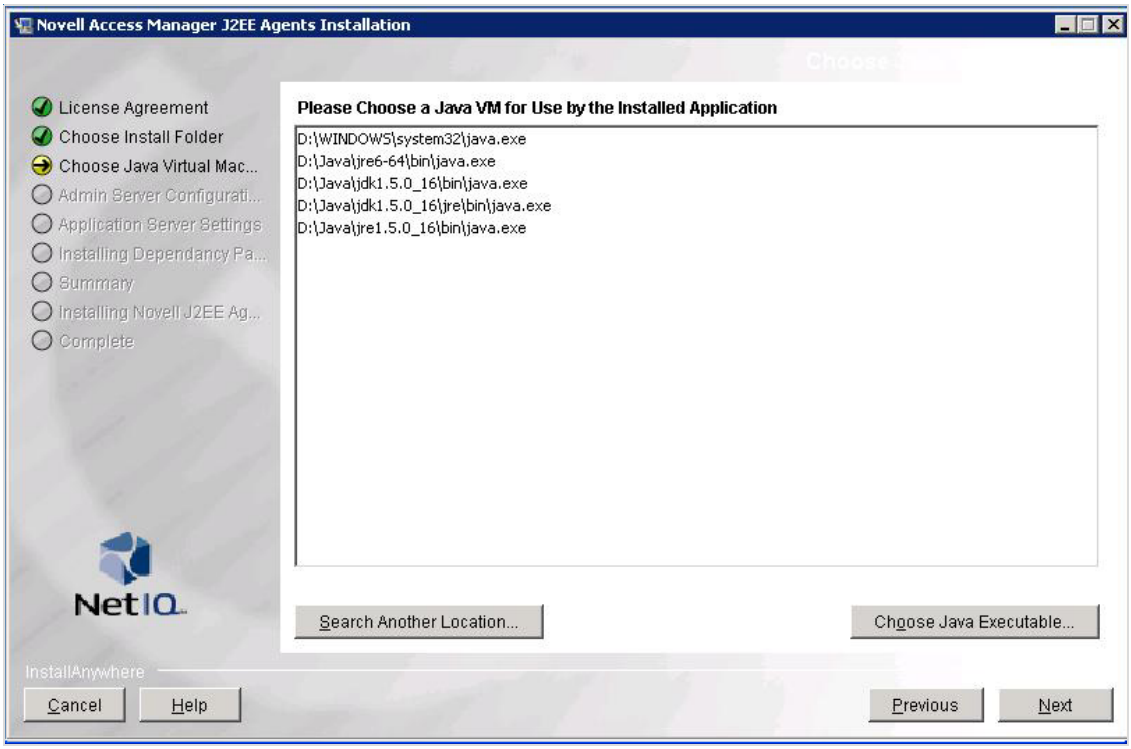
The license agreement page is displayed. For software download instructions, see the Access Manager Readme.



3 Review the License Agreement, accept it, then click *Next*. The installation selection page is displayed.



4 Select a directory to install the J2EE agent components, then click *Next*. The Choose a Java Virtual Machine page is displayed.



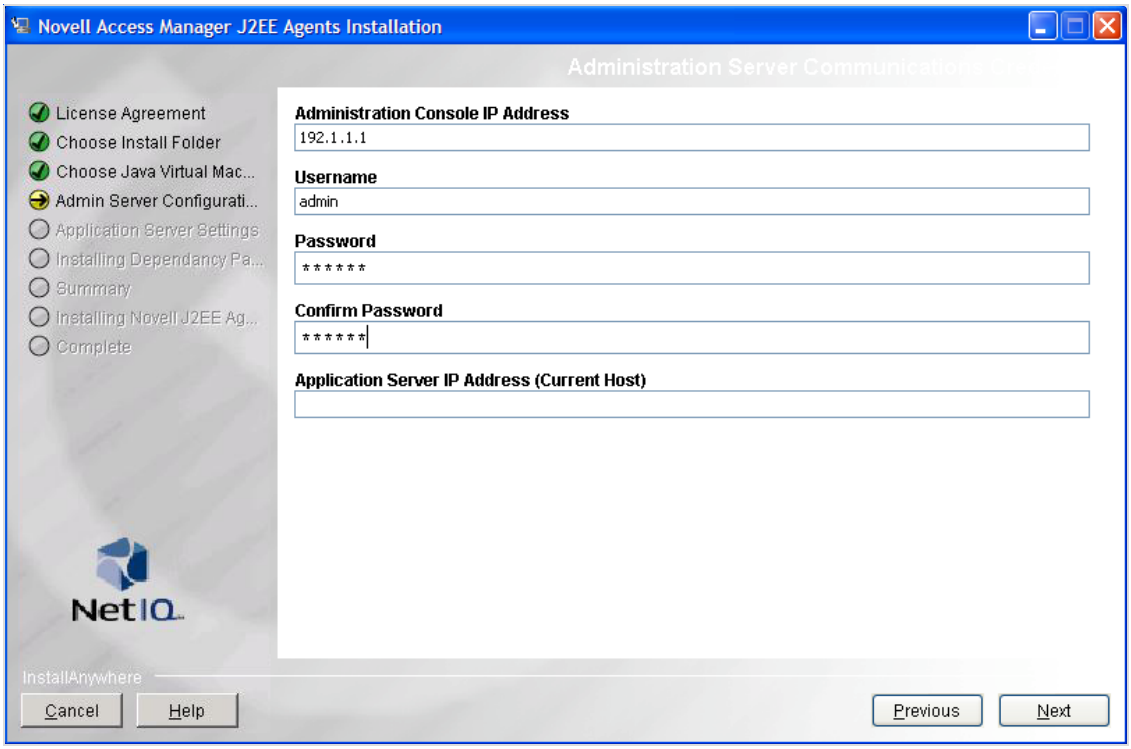
5 Select a Java Virtual Machine (JVM*) to be used by the installed application.

A default JVM is displayed.

If you do not select a JVM here, the installer uses the java.home property value of the Java runtime that is used to run the installer to proceed with the installation

6 (Optional) If you want to select another JVM, click *Choose Another* and browse to select the JVM of your choice. Click *Search for Others* to get a list of available JVMs and select the one you want.

7 Click *Next*. The Administration Server Communication page is displayed.



8 Specify the information required for server communication between the agent and the Administration Console:

Administration Console IP Address: Specify the IP address of your Access Manager Administration Console.

Username: Specify the username of the admin user of the Access Manager Administration Console.

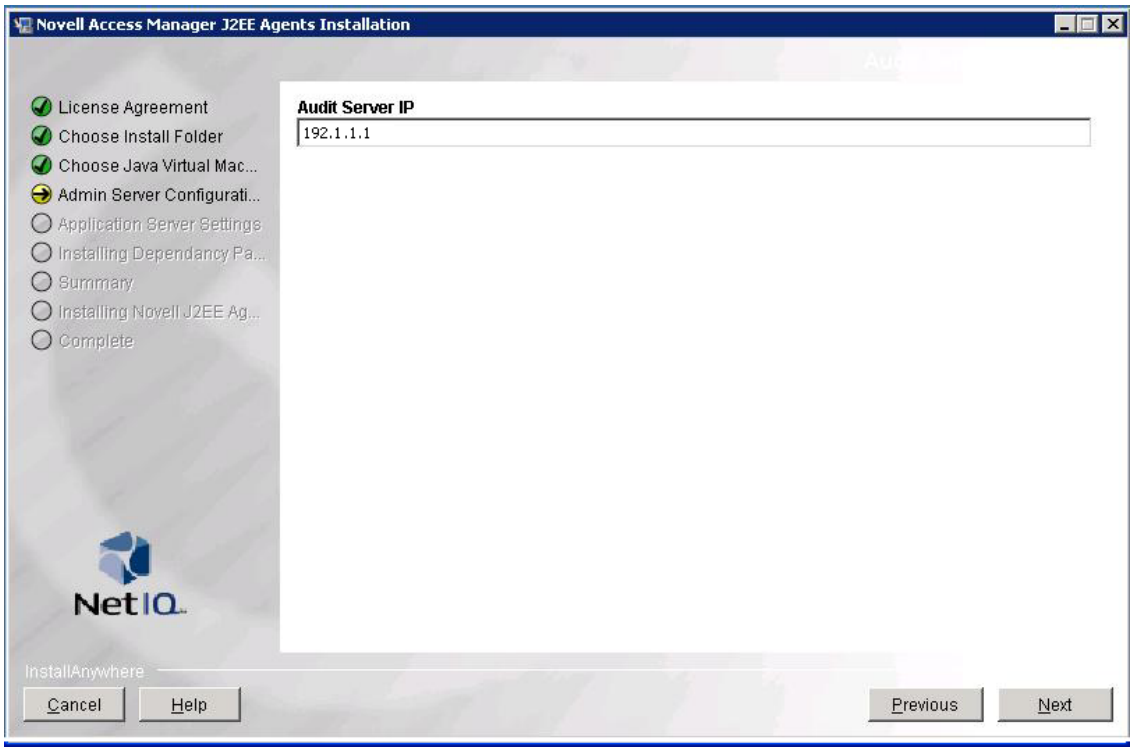
Password: Specify password of the admin user of the Access Manager Administration Console.

Confirm Password: Specify the password again to confirm it.

Application Server IP Address (Current Host): Review the entered address. If your server is configured for more than one IP address, make sure you specify the IP address of the machine from which the Access Manager Administration Console is reachable.

9 Click *Next*.

10 (Conditional) If you do not have the audit server installed, the J2EE installer installs the Audit server for you. Specify the IP address of the Access Manager Administration Console as the *Audit Server IP*.

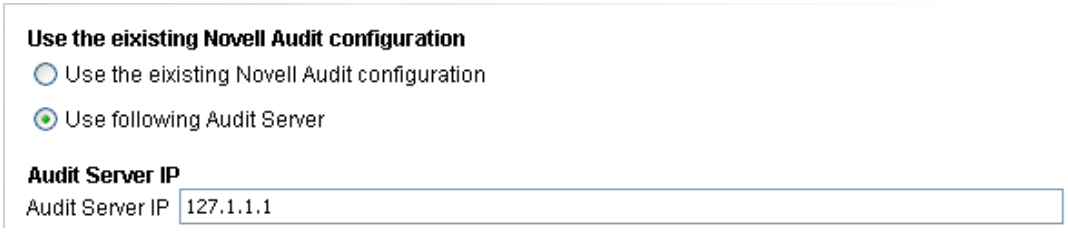


11 (Conditional) If you have the Audit server installed, follow the prompts to continue using the existing Audit server or to replace it:

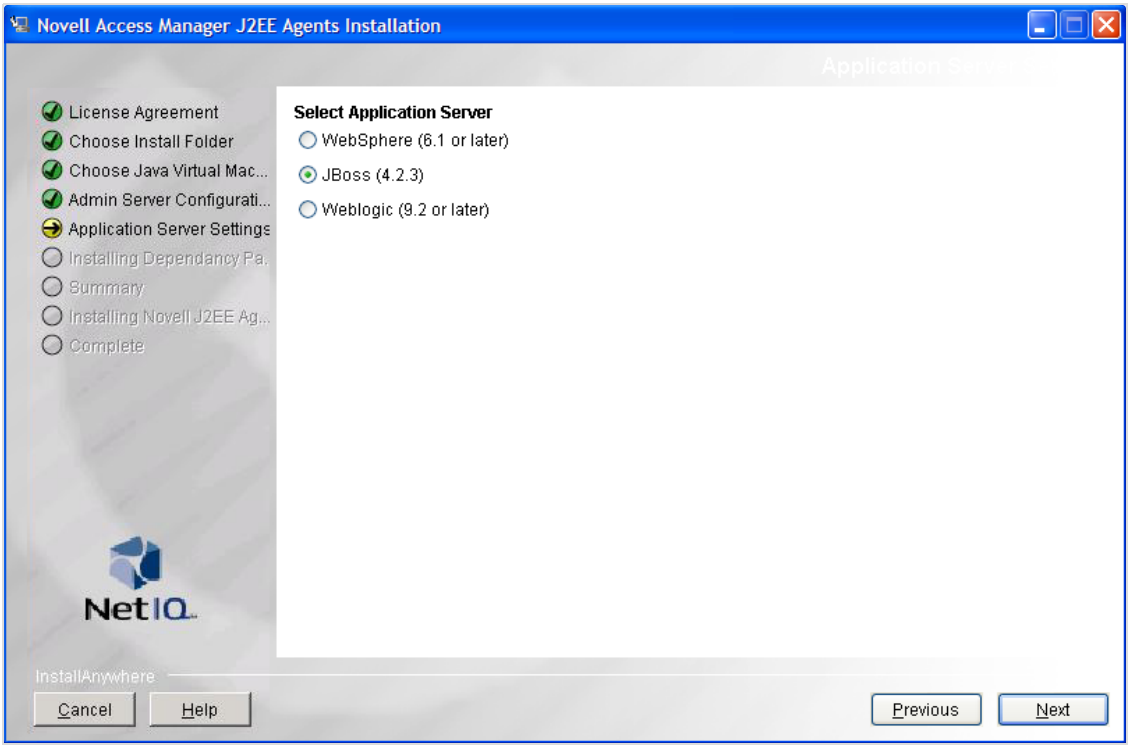
11a (Conditional) To continue using the same server, click *Yes* to display the Audit Server Setting page.

11b Select *Use following Audit Server*, then continue with Step 13

11c (Conditional) To use another server, click *No*, select Use following *Audit Server*, then specify an IP address for the Audit server.



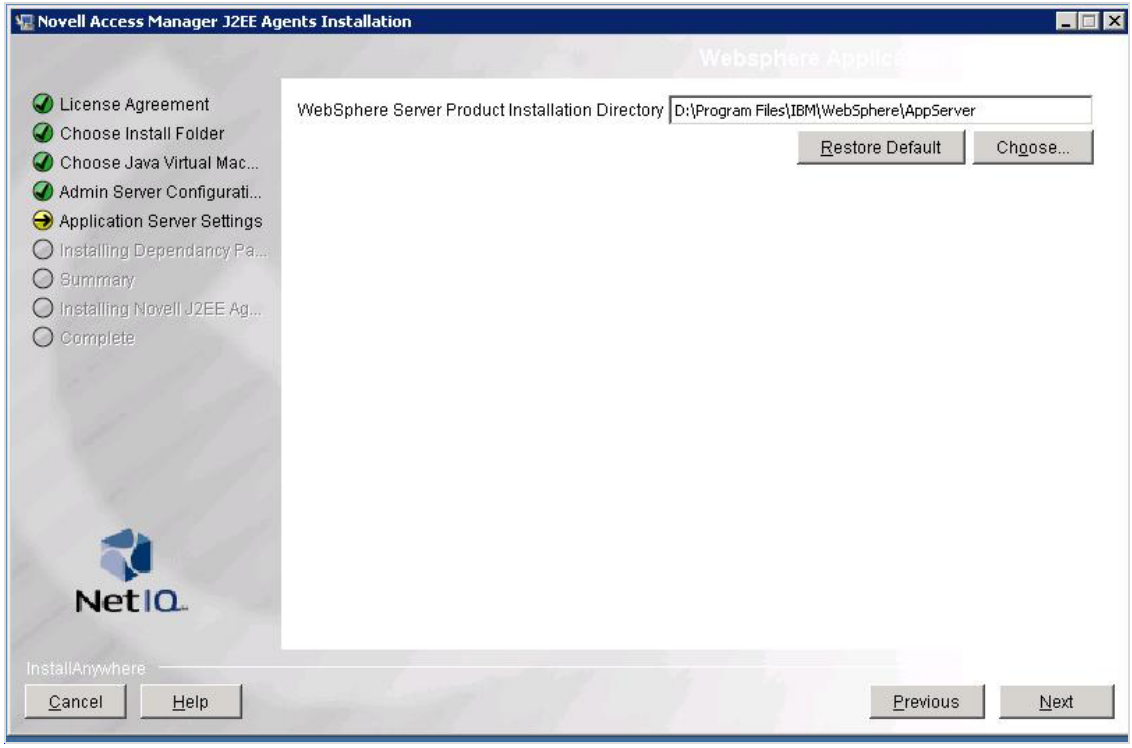
12 Click *Next*. The Select Application Server page is displayed.



13 Click *OK* when the Alert prompt is displayed.

14 Select *JBoss*, then click *Next*.

The following page is displayed.



15 Specify the following information:

WebSphere Server Product Installation Directory: Specify the directory where you have installed the JBoss server.

16 Click *Next*. The JCC Dependencies page is displayed.

17 Click *Install* to install the dependent JCC packages.

18 Review the installation summary, then click *Install* to install the agent.

19 Click *Done* when the installation is complete.

20 Start JBoss.

The agent is not imported into the Administration Console until the JBoss server is running.

21 To verify the installation of the agent, see Section 1.8, "Verifying If a J2EE Agent Is Installed," on page 40.

1.5.4 Installing the JBoss Agent through the Console

1 Download the agent installer. For software download instructions, see the Access Manager Readme.

2 Enter the following command in the command prompt to run the installer on the console:

```
<filename> -i console
```

Replace <*filename*> with the name of the J2EE agent installer.

3 Review the License Agreement, then press `Y` to accept it.

4 Specify an absolute path to install the J2EE agent components, or press Enter to continue with the default installation path.

5 Specify a Java Virtual Machine (JVM) to be used by the installed application.

All the available JVMs are displayed with a number. The default JVM is displayed with an arrow. Press Enter to select the default JVM, or specify the number of one of the listed JVMs.

6 Specify the information required for communication between the agent and the Administration Console:

- Specify the IP address of your Access Manager Administration Console.
- Specify the username and password of the admin user of the Access Manager Administration Console. Confirm the password by re-entering it.
- Specify the IP address of the Application Server. If your server is configured for more than one IP address, make sure you specify the IP address of the machine from which the Access Manager administration console is reachable.

7 (Conditional) If you do not have the Audit server installed, J2EE installer installs the Audit server for you. Specify the IP address of the Access Manager Administration Console as the *Audit Server IP*, then press Enter.

8 (Conditional) If the Audit server is already installed on your machine:

8a To specify if you want to replace the existing Audit server or use the existing server:

- Press 1 to use the existing Audit server.
- Press 2 to replace the existing Audit server, then specify the IP address of the new server.

8b (Conditional) Press 1 to use the existing Novell Audit Configuration.

8c (Conditional) Press 2 to use a different Audit Server and then specify the IP address.

9 For the Web Application Server to be installed, specify 2 for JBoss, then press Enter.

10 Read the alert message and press Enter to continue.

11 Specify the directory where you have installed the JBoss server, then press Enter to continue.

12 Specify the name of the server configuration folder, then press Enter.

13 Review the installation summary, then press Enter to install the agent.

14 To verify the installation of the agent, see Section 1.8, "Verifying If a J2EE Agent Is Installed," on page 40.

1.6 Installing the J2EE Agent on WebSphere

You must install J2EE Agents on the same machine as your WebSphere server, and your WebSphere server needs to be installed on a machine that does not contain any Access Manager components.

The WebSphere agent supports the WebSphere LTPA and SWAM authentication mechanisms. To support this mechanism, the J2EE agent installer modifies the LTPA, LTPA_WEB, SWAM, WEB_INBOUND, and JAAS login configurations in your WebSphere configurations.

1.6.1 Prerequisites

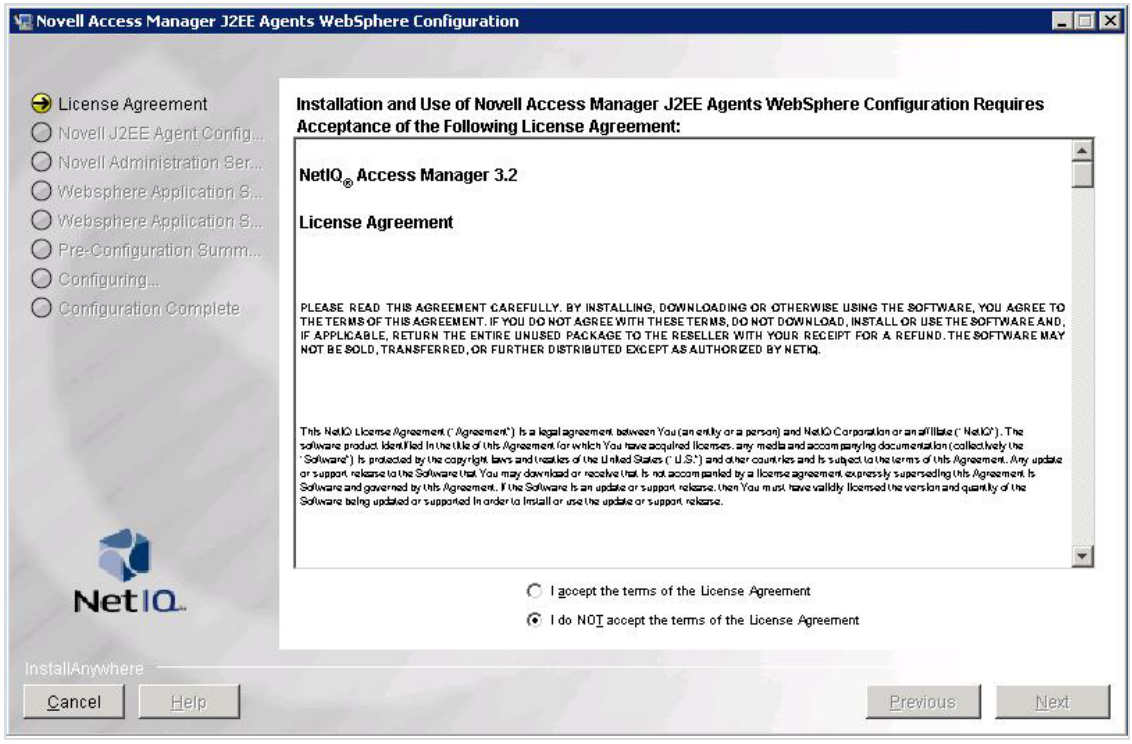
- Know the following about your WebSphere installation:
  - Path to the directory where WebSphere is installed.
  - Username and password of the WebSphere administrator.
- Verify the version of the JVM used by WebSphere, then download and install the JVM of same version. Do not use the JVM provided by WebSphere.
- Verify that the machine meets the minimum requirements. See Section 1.3, "Prerequisites," on page 10.

NOTE: If you have disabled the admin security feature in WebSphere, the installation of J2EE agent will be successful, but you must enable admin security to import the Agents into the administration console.

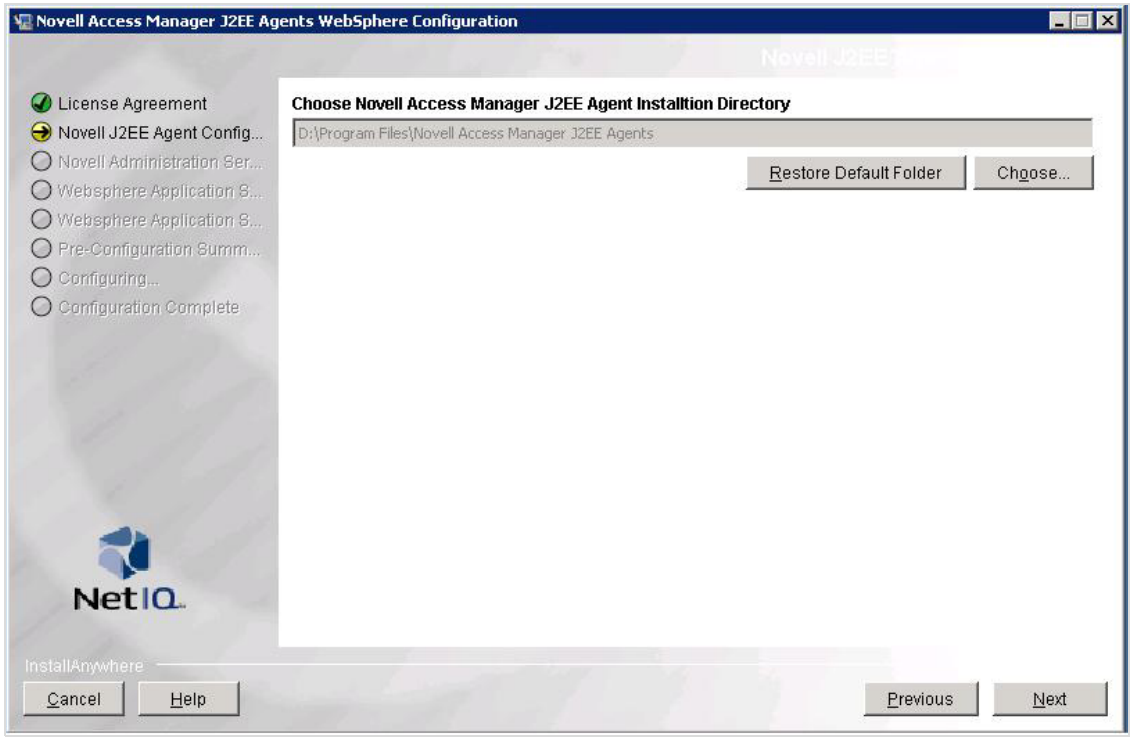
1.6.2 Installing on WebSphere by Using the Installer

1 Download and execute the agents installer. For software download instructions, see the Access Manager Readme.

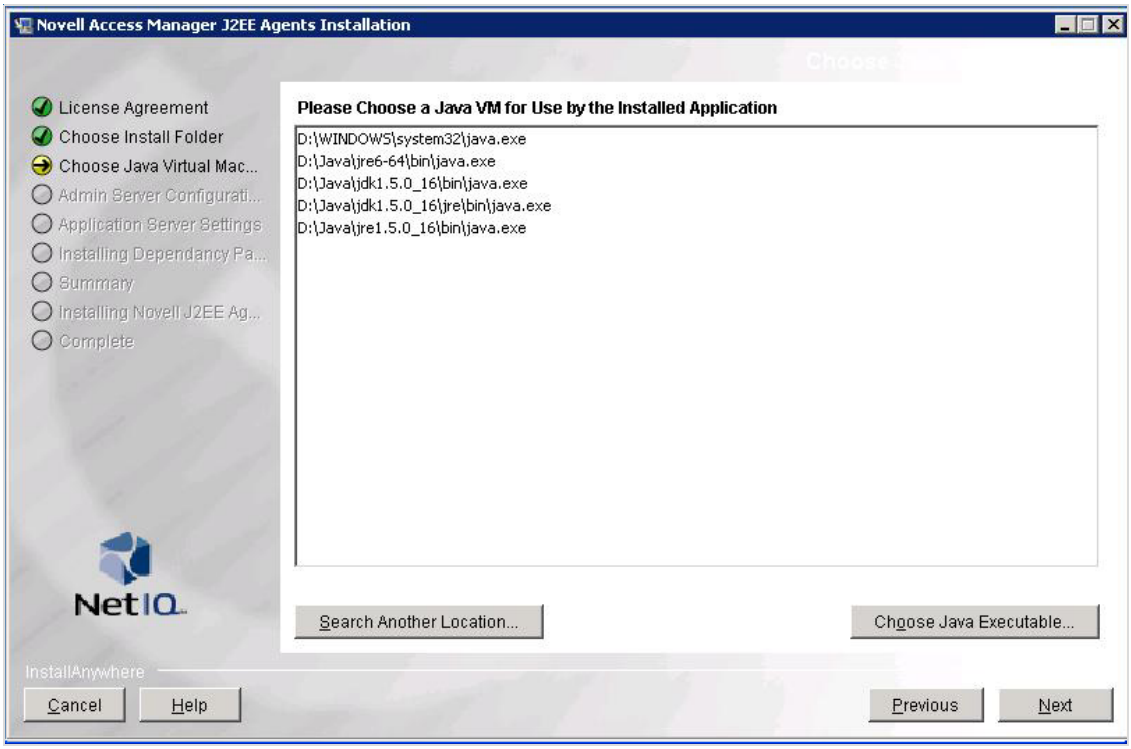
The Licence Agreement page is displayed.



2 Review the License Agreement, accept it, then click *Next*. The installation selection page is displayed.



3 Select a directory to install the J2EE agent components, then click *Next*. The Choose Java Virtual Machine page is displayed.



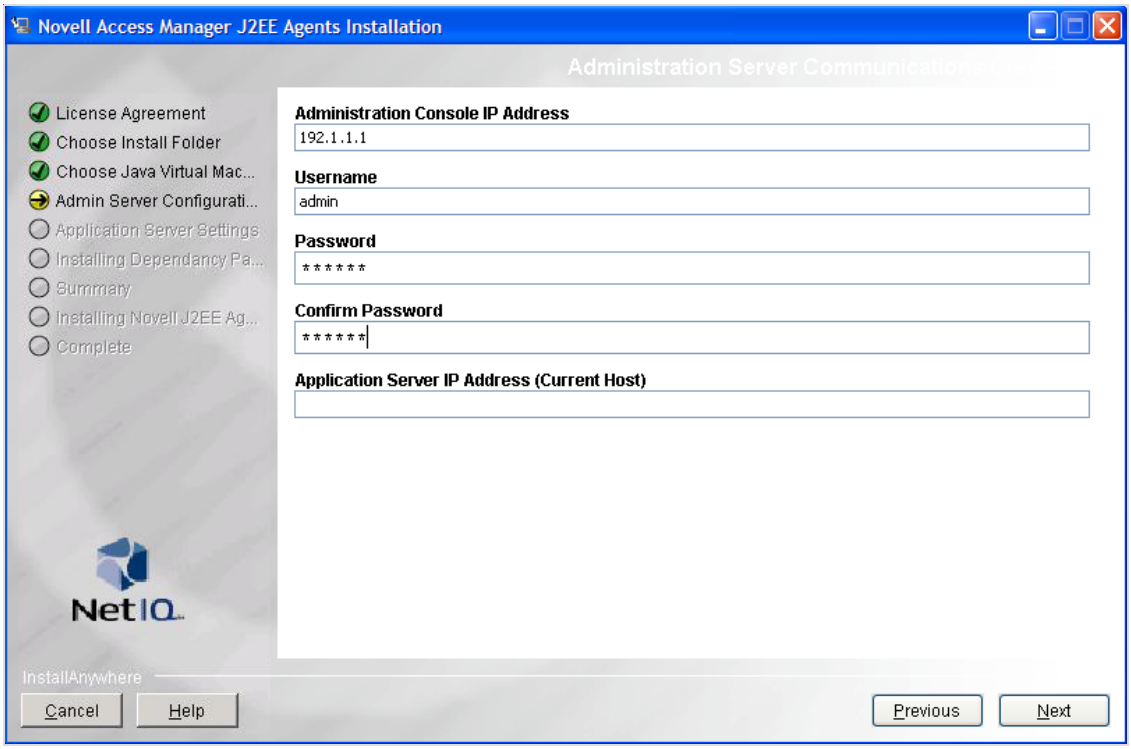
4 Select a Java Virtual Machine (JVM) to be used by the installed application.

A default JVM is displayed.

If you do not select a JVM here, the installer uses the java.home property value of the Java runtime that is used to run the installer to proceed with the installation.

5 (Optional) If you want to select another JVM, click *Choose Another* and browse to select the JVM of your choice. Click *Search for Others* to get a list of available JVMs and select the one you want.

6 Click *Next*. The Administration Server Communication page is displayed.



7 Specify the information required for server communication between the agent and the Administration Console:

Administration Console IP Address: Specify the IP address of your Access Manager Administration Console.

Username: Specify the username of the admin user of the Access Manager Administration Console.

Password: Specify password of the admin user of the Access Manager Administration Console.

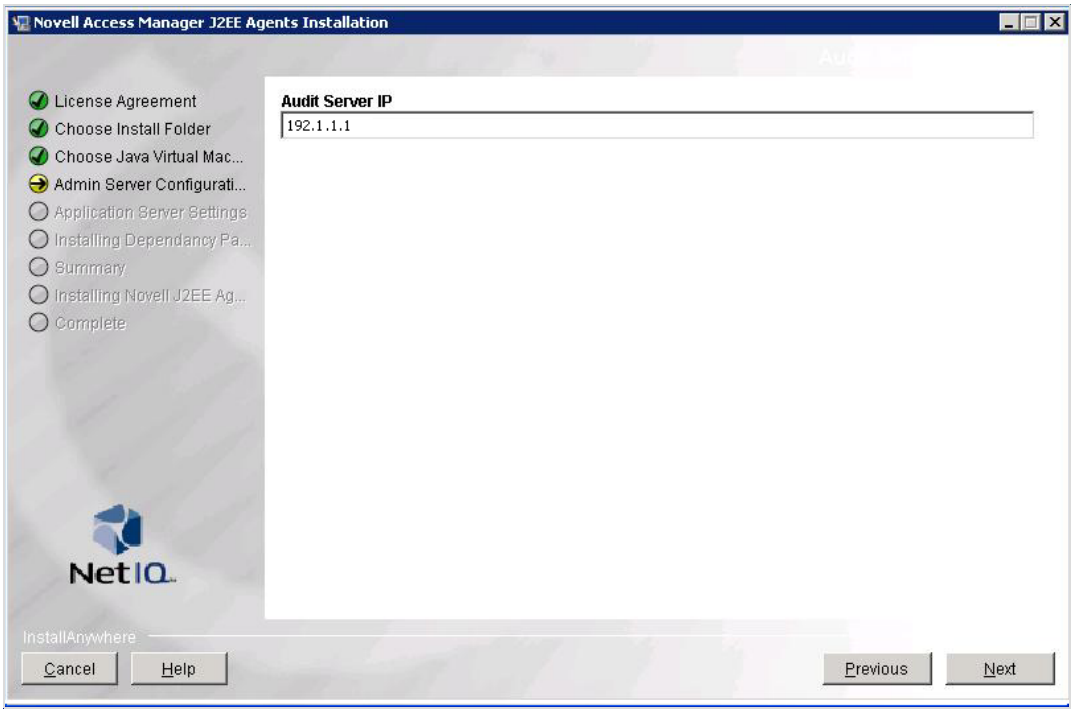
Confirm Password: Specify the password again to confirm it.

Application Server IP Address (Current Host): Review the entered address. If your server is configured for more than one IP address, make sure you specify the IP address of the machine from which the Access Manager administration console is reachable.

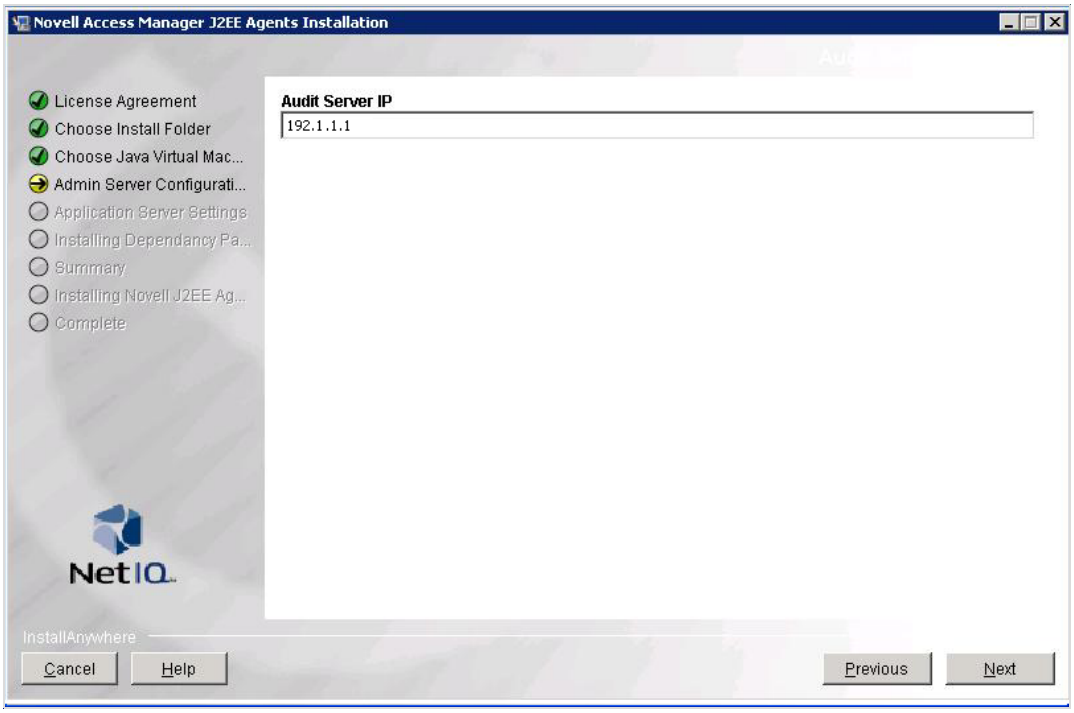
8 Click *Next*. The Audit Server page is displayed.

9 Specify the audit server IP address:

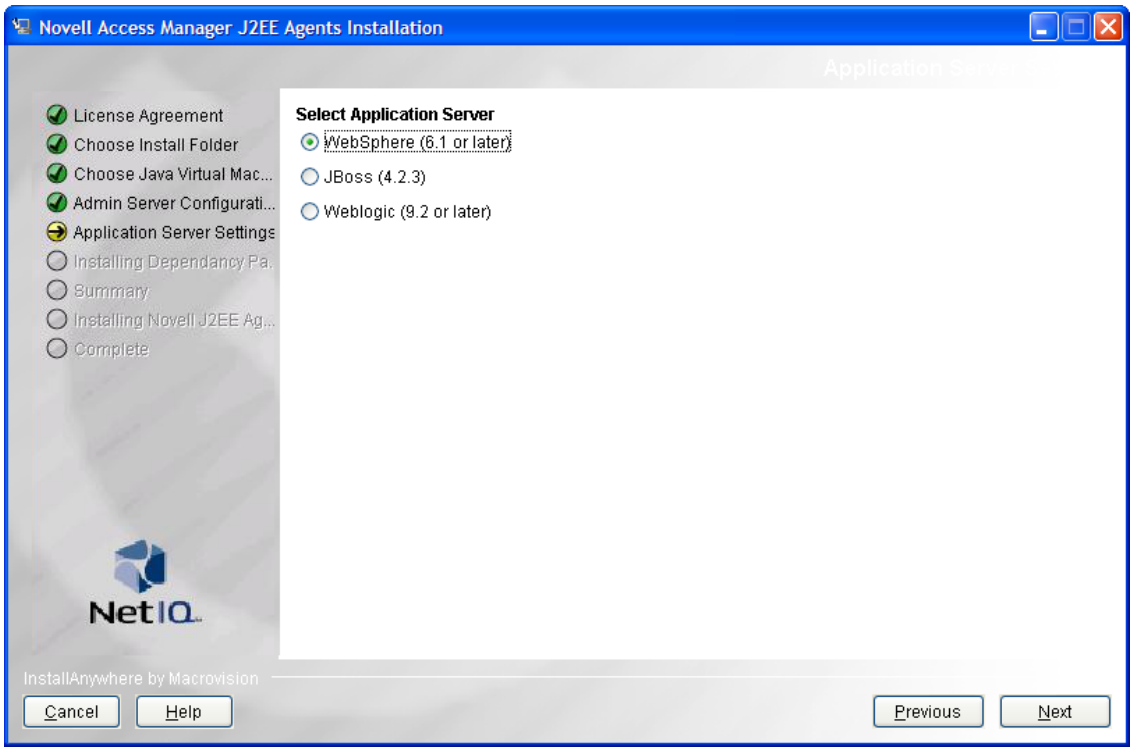
9a Conditional) If you do not have the audit server installed, the J2EE installer installs the Audit server for you. Specify the IP address of the Access Manager Administration Console as the *Audit Server IP*.



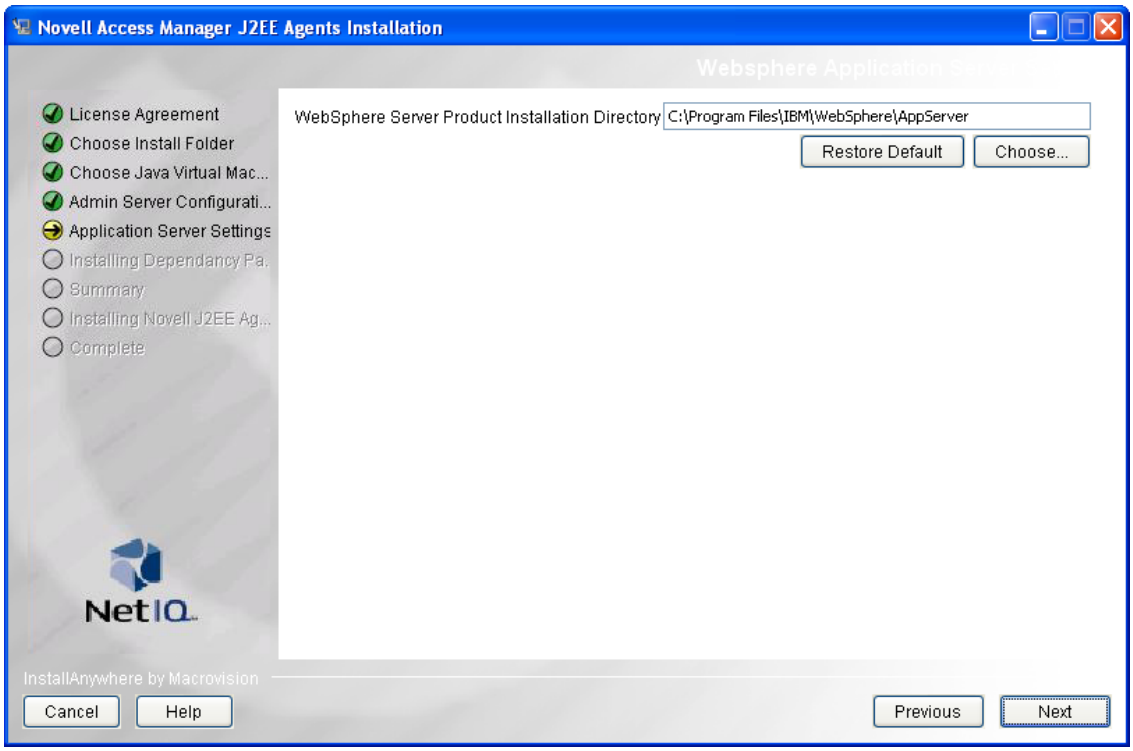
9b Conditional) If you have the Audit server installed, specify if you want to replace the existing audit server or use the existing server.



10 Click *Next*. The Select Application Server page is displayed.



11 Select *WebSphere*, then click *Next*. The WebSphere Application Server Settings page is displayed.



12 Specify the directory where you have installed the WebSphere server and click *Next*. The JCC Dependencies page is displayed.

13 Click *Install* to continue with the Agent installation.

14 Review the installation summary, then click *Install* to install the agent.

When installation is complete, the Configure IBM WebSphere Application Server Instance page is displayed.

15 (Conditional) If you want to complete the configuration now, select the *Configure IBM WebSphere Application Server* option to configure application server instances, then click *Next* to launch the configuration utility.

Complete the configuration procedure in Section 1.6.4, "Configuring WebSphere for J2EE Agents," on page 27.

or

If you want to perform the configuration at a later point of time, click *Next*. The successful installation page is displayed.

16 Click *Done* to quit the installer.

1.6.3 Installing the WebSphere Agent through the Console

1 Download the file and execute it.

For software download instructions, see the Access Manager Readme.

2 Enter the following command in the command prompt to run the installer on the console:

```
<filename> -i console
```

Replace <*filename*> with the name of the J2EE agent installer.

3 Review the License Agreement, then press `Y` to accept it.

4 Specify an absolute path to install the J2EE agent components, or press Enter to continue with the default installation path.

5 Specify a Java Virtual Machine (JVM) to be used by the installed application.

All the available JVMs are displayed with a number. The default JVM is displayed with an arrow. Press Enter to select the default JVM, or specify the number of one of the listed JVMs.

6 Specify the information required for communication between the agent and the Administration Console:

- Specify the IP address of your Access Manager Administration Console.
- Specify the username and password of the admin user of the Access Manager Administration Console. Confirm the password by re-entering it.
- Review the entered address. If your server is configured for more than one IP address, make sure you specify the IP address of the machine from which the Access Manager administration console is reachable.

7 (Conditional) If you do not have the Audit server installed, the J2EE installer installs the Audit server for you. Specify the IP address of the Access Manager Administration Console as the *Audit Server IP*, then press Enter.

8 (Conditional) If the Audit server is already installed on your machine:

8a To specify if you want to replace the existing Audit server or use the existing server:

- Press 1 to use the existing Audit server.
- Press 2 to replace the existing Audit server, then specify the IP address of the new server.

8b (Conditional) Press 1 to use the existing Novell Audit Configuration.

8c (Conditional) Press 2 to use a different Audit Server and then specify the IP address.

9 For the Web Application Server to be installed, specify 2 for JBoss, then press Enter.

10 Read the alert message and press Enter to continue.

11 Specify the directory where you have installed the WebSphere server. Press Enter to continue.

12 Review the installation summary, then press Enter to install the agent.

13 Complete the configuration procedure in Section 1.6.4, "Configuring WebSphere for J2EE Agents," on page 27.

1.6.4 Configuring WebSphere for J2EE Agents

After you install the WebSphere application server, you must use the ConfigureWSAgent utility to configure it for the J2EE Agent.

NOTE: You can run the `configure_websphere_agent.sh` or `configure_websphere_agent.bat` multiple times to configure multiple instances of a WebSphere application server on a single physical machine.

---

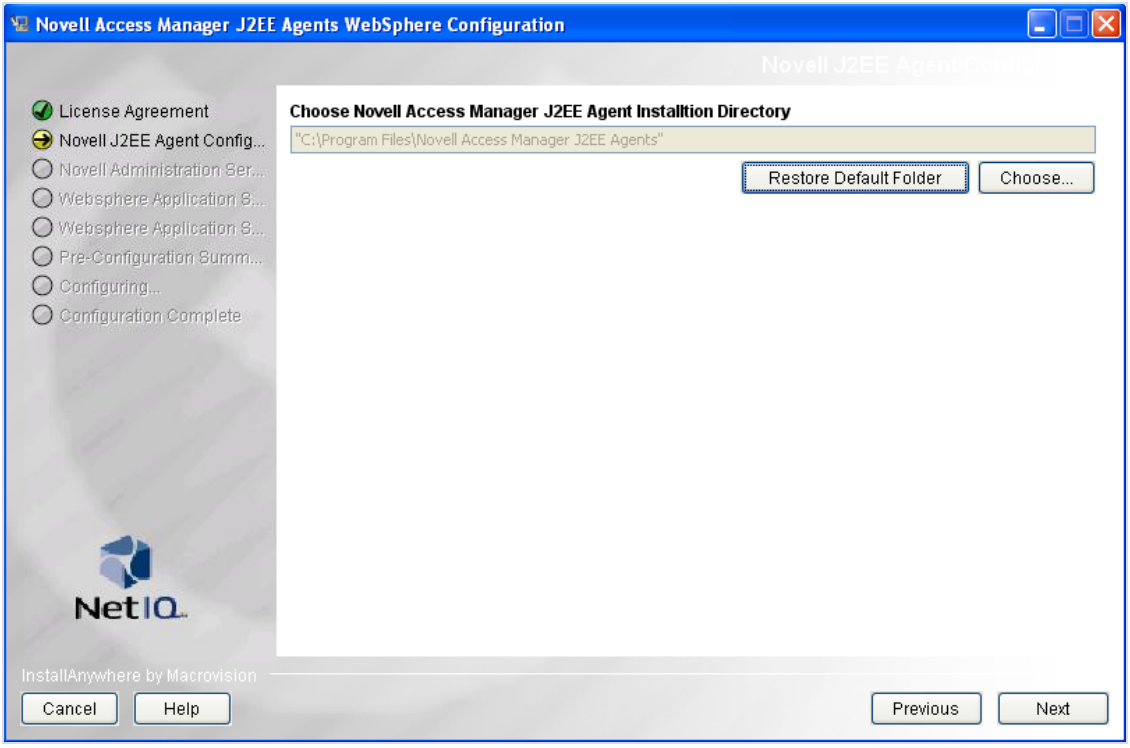
1 Start the utility located at:

Linux/AIX: `/opt/novell/nids-agents/bin/configure_websphere_agent.sh`

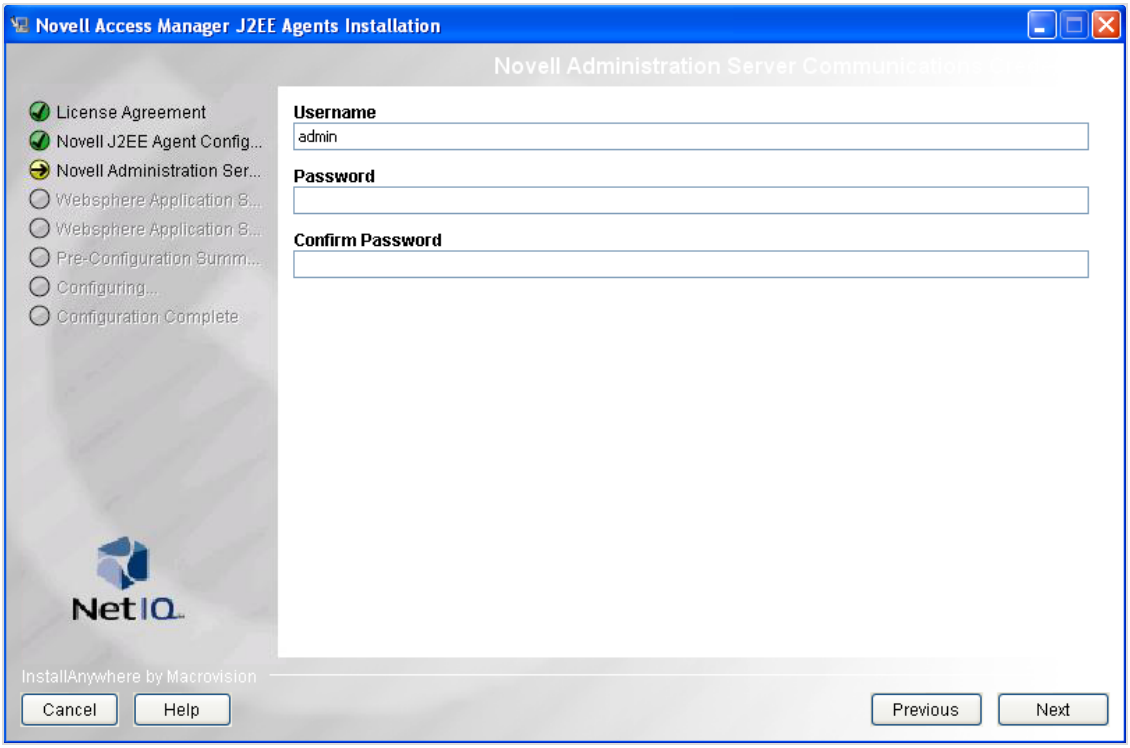
Windows: `<Installation-directory>/nids-agents/bin/configure_websphere_agent.bat`

2 Ensure that WebSphere is running.

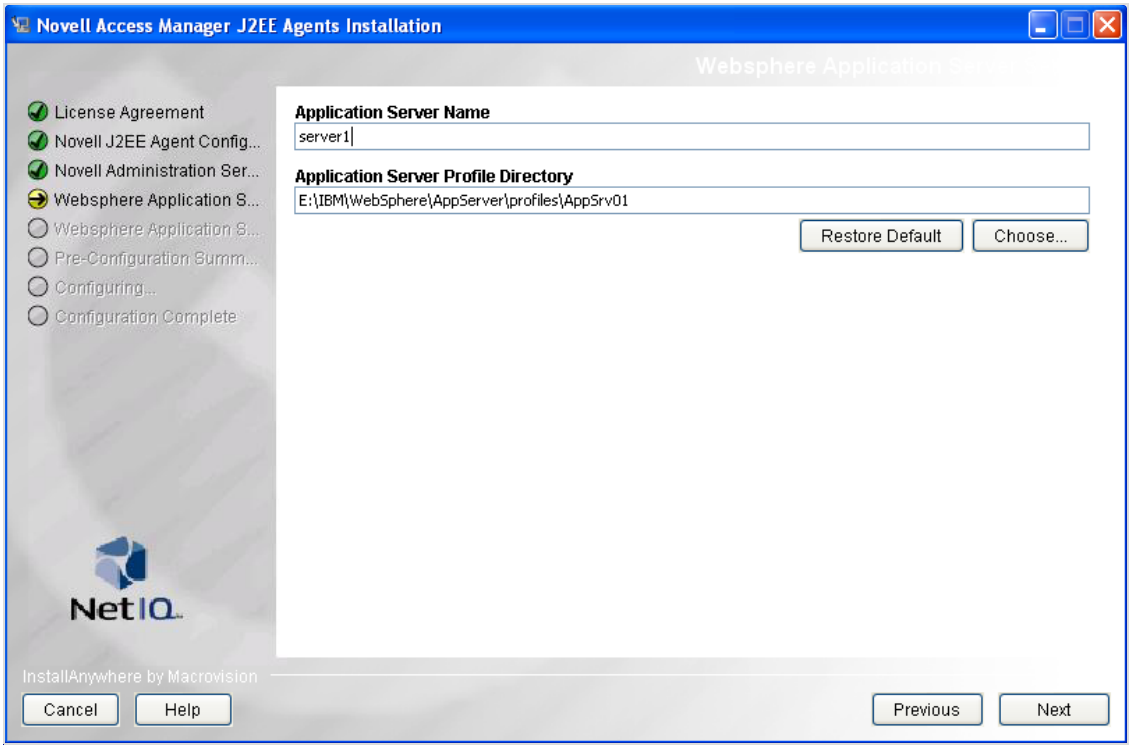
3 Review the License Agreement, accept it, then click *Next*. The J2EE Agent Configuration page is displayed.



4 Select the directory where the J2EE agent is installed and click *Next*. The Administration Server Communications Credentials page is displayed.



5 Specify the administration credentials to contact the Access Manager and click *Next*. The WebSphere Application Server Settings page is displayed.



6 Specify the following:

Application Server Name: Specify a name for the application server.

Application Server Profile Directory: Specify the path to the application server profile.

7 Click *Next*. The WebSphere Application Server Security Settings page is displayed.

8 Specify the following:

Username: Specify the name of the WebSphere administrator.

Password: Specify the password of the WebSphere administrator.

Re-enter Password: Specify the password again to reconfirm.

9 Click *Next*. The Pre-configuration Summary page is displayed.

10 Click *Next* to configure changes required for this application server instance. The Configuration Complete page is displayed.

11 Click *Done* to exit the utility.

12 When the installation completes, restart WebSphere.

The agent is not imported into the Administration Console until the WebSphere server is running.

13 (Conditional) When the application server is restarted, the agent's health in the Administration Console may show the embedded service provider (ESP) is in halted state. Start the ESP manually.

13a In the Administration Console, click *Devices* > *J2EE Agents*.

13b On the Servers page, select the server's check box, then click *Actions* > *Service Provider* > *Start Service Provider* > *OK*.

14 (Conditional) If you are using the WEB_INBOUND login configuration (which is the default), you need to manually move the J2EE agent login module (com.novell.nids.agent.auth.websphere.NidsLTPALoginModule) to the top of the list:

14a Open the IBM administration console.

14b Click *Security* > > *Global Security* > *Secure administration, applications, and infrastructure*

14c Expand the *Java Authentication and Authorization Service* option and click *System Logins*.

14d Select *WEB_INBOUND* > *JAAS login modules*.

14e Change the order of com.novell.nids.agent.auth.websphere.NidsLTPALoginModule so it is first in the list.

14f Save your changes.

15 (Optional) To verify the installation of the agent, see Section 1.8, "Verifying If a J2EE Agent Is Installed," on page 40.

1.7 Installing the J2EE Agent on WebLogic

The agent needs to be installed on the same machine as your WebLogic server. The WebLogic server must be installed on a machine that does not contain any Access Manager components.

1.7.1 Prerequisites

- You must know the following about your WebLogic installation:
  - Path to the directory where WebLogic is installed.
  - Username and password of the WebLogic administrator.
- Make sure that your installation folder name has no spaces. For example, you cannot specify the folder name as `Novell Access Manager J2EE Agents`, but you can specify the name as `Novell_Access_Manager_J2EE_Agents`.
- Verify that the machine meets the minimum requirements. See Section 1.3, "Prerequisites," on page 10.

1.7.2 Installing WebLogic Agent by Using the Installer

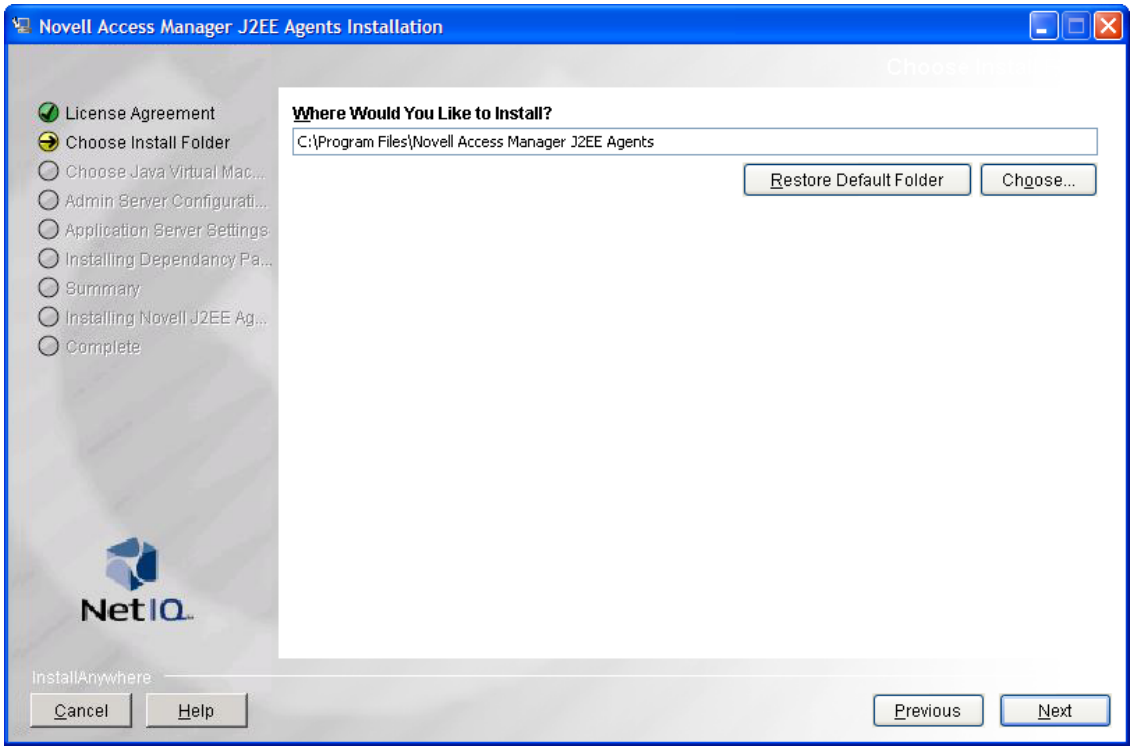
1. Make sure that the WebLogic server is running.

   The WebLogic server must be running if you are performing a single server installation of J2EE Agents. The WebLogin server does not need to be running if you are installing J2EE Agents in a Base or Cluster mode.

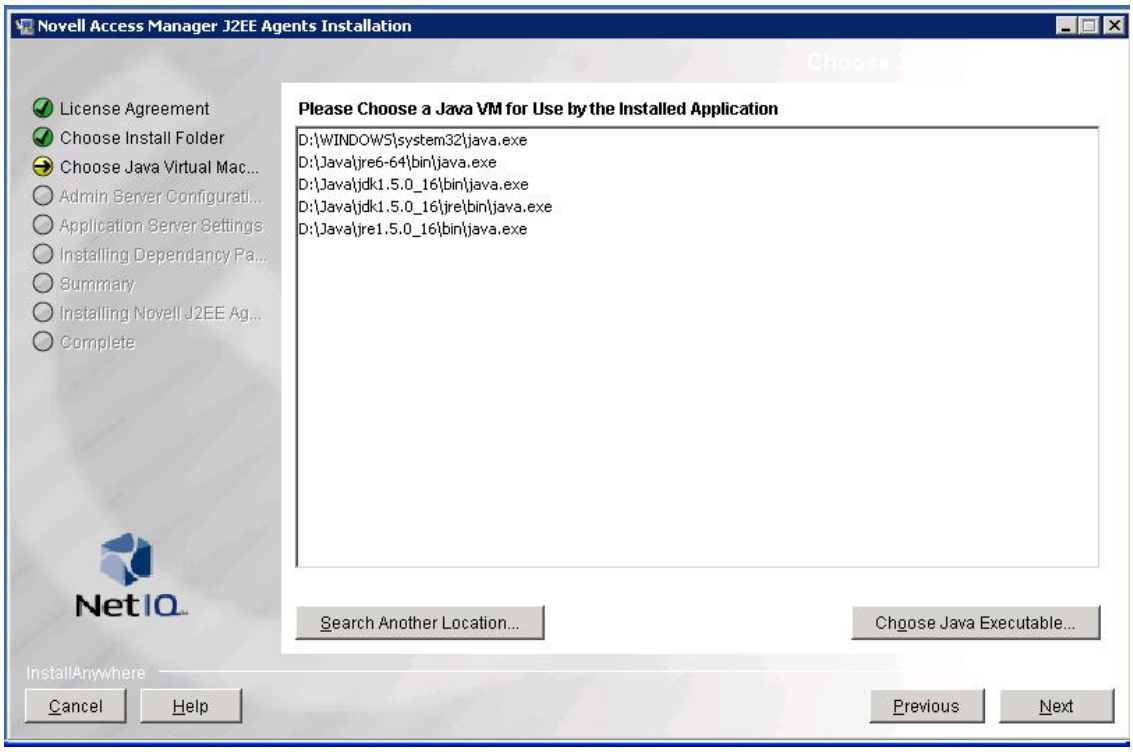
2. Download and execute the agent installer.

   For software download instructions, see the Access Manager Readme.

3. Review the License Agreement, accept it, then click *Next*. The installation selection page is displayed.



4. Select a directory to install the J2EE Agent components, then click *Next*. The Choose a Java Virtual Machine page is displayed.



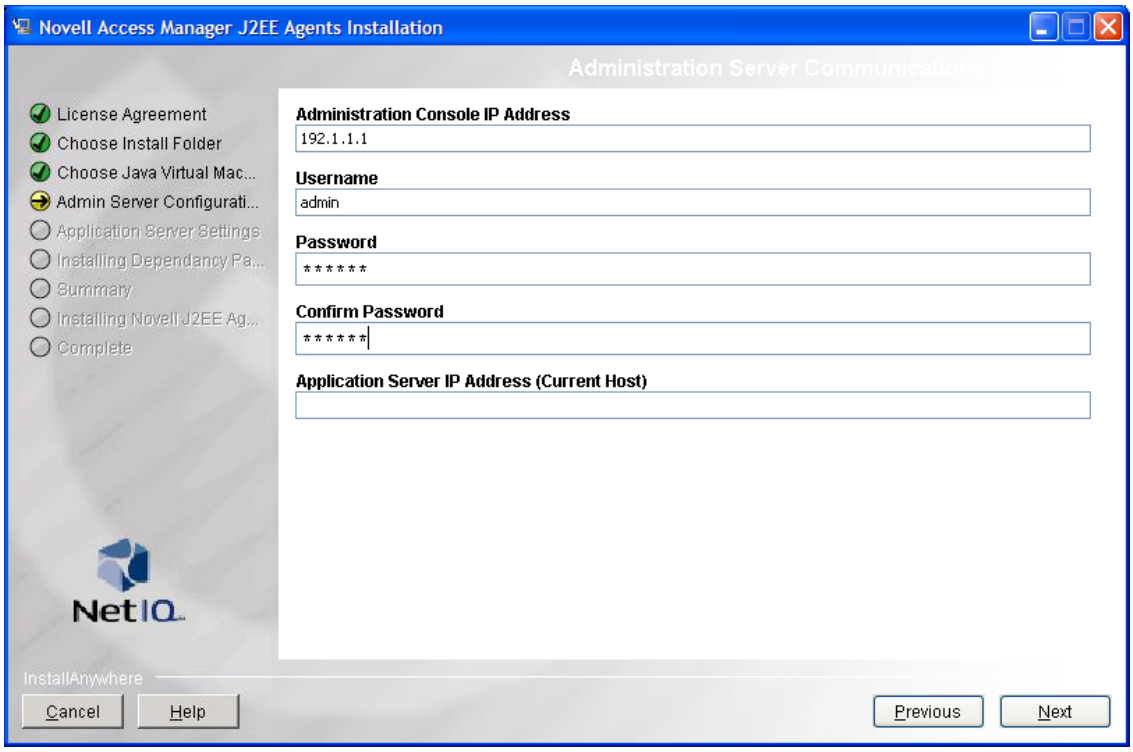
5 Select a Java Virtual Machine (JVM) to be used by the installed application.

A default JVM is displayed.

If you do not select a JVM here, the installer uses the java.home property value of the Java runtime that is used to run the installer to proceed with the installation.

6 (Optional) If you want to select another JVM, click *Choose Another* and browse to select the JVM of your choice. Click *Search for Others* to get a list of available JVMs and select the one you want.

7 Click *Next*. the Administration Server Communication page is displayed.



8 Specify the information required for server communication between the agent and the Administration Console:

Administration Console IP Address: Specify the IP address of your Access Manager Administration Console.

Username: Specify the username of the admin user of the Access Manager Administration Console.

Password: Specify password of the admin user of the Access Manager Administration Console.

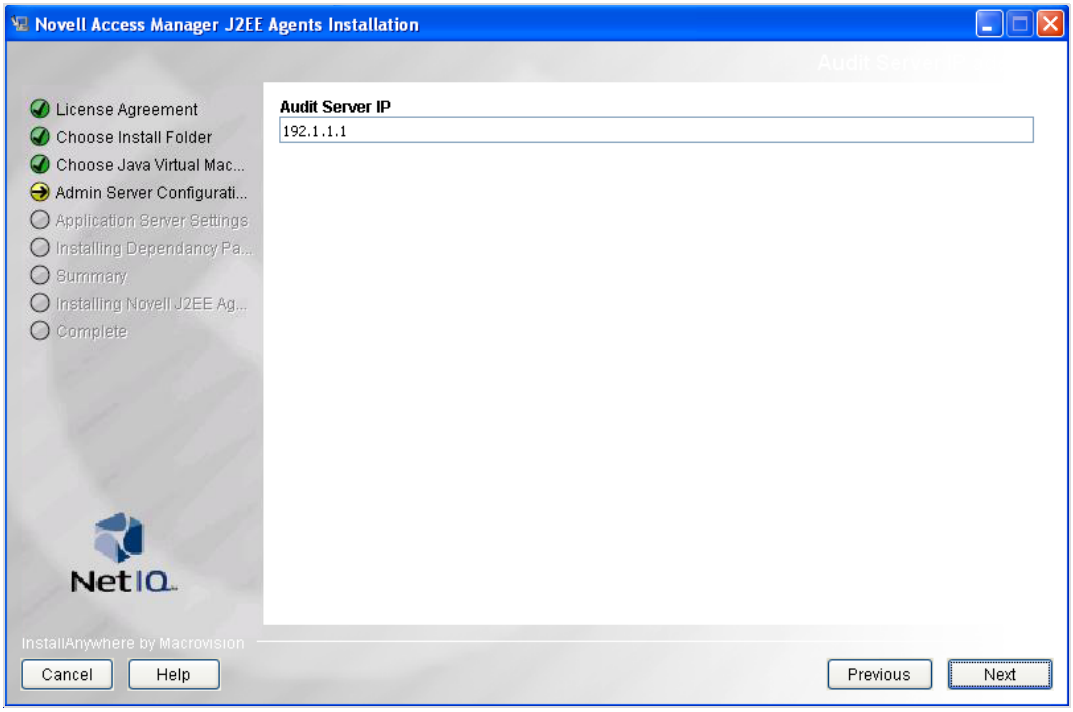
Confirm Password: Specify the password again to confirm it.

Application Server IP Address (Current Host): Review the entered address. If your server is configured for more than one IP address, make sure you specify the IP address of the machine from which the Access Manager Administration Console is reachable.

9 Click *Next*. The Audit Server page is displayed.

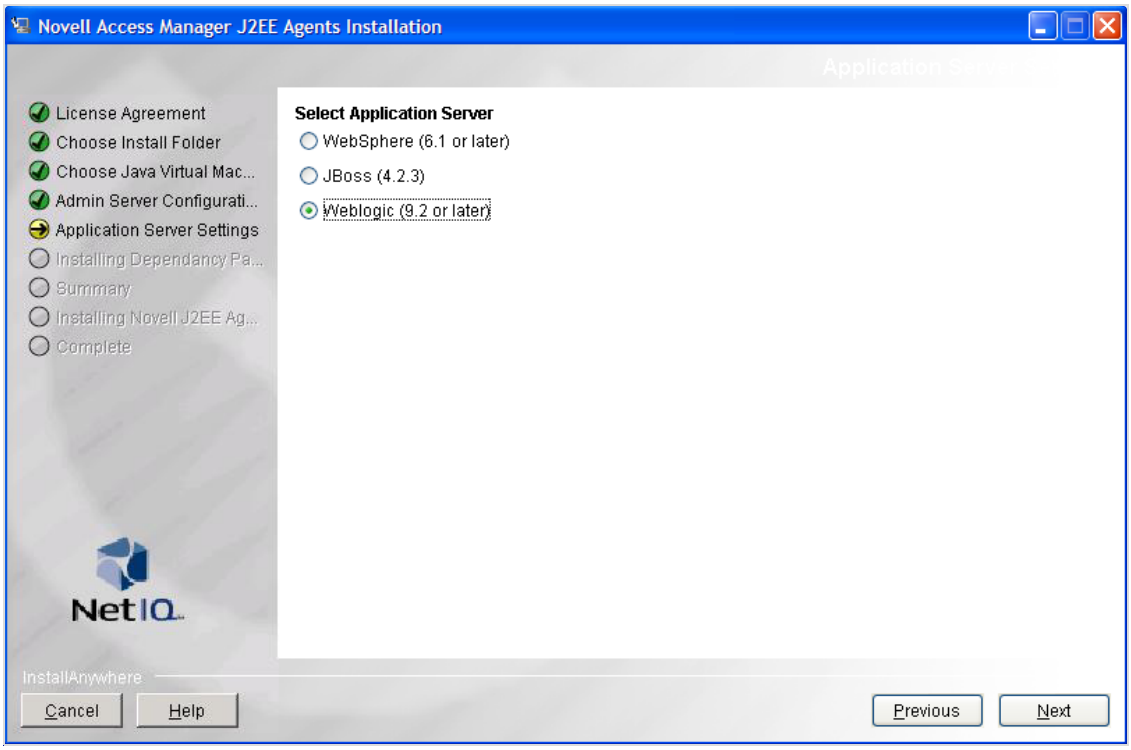
10 Specify the audit server IP address:

10a (Conditional) If you do not have the Audit server installed, the J2EE installer installs the Audit server for you. Specify the IP address of the Access Manager Administration Console as the *Audit Server IP*.

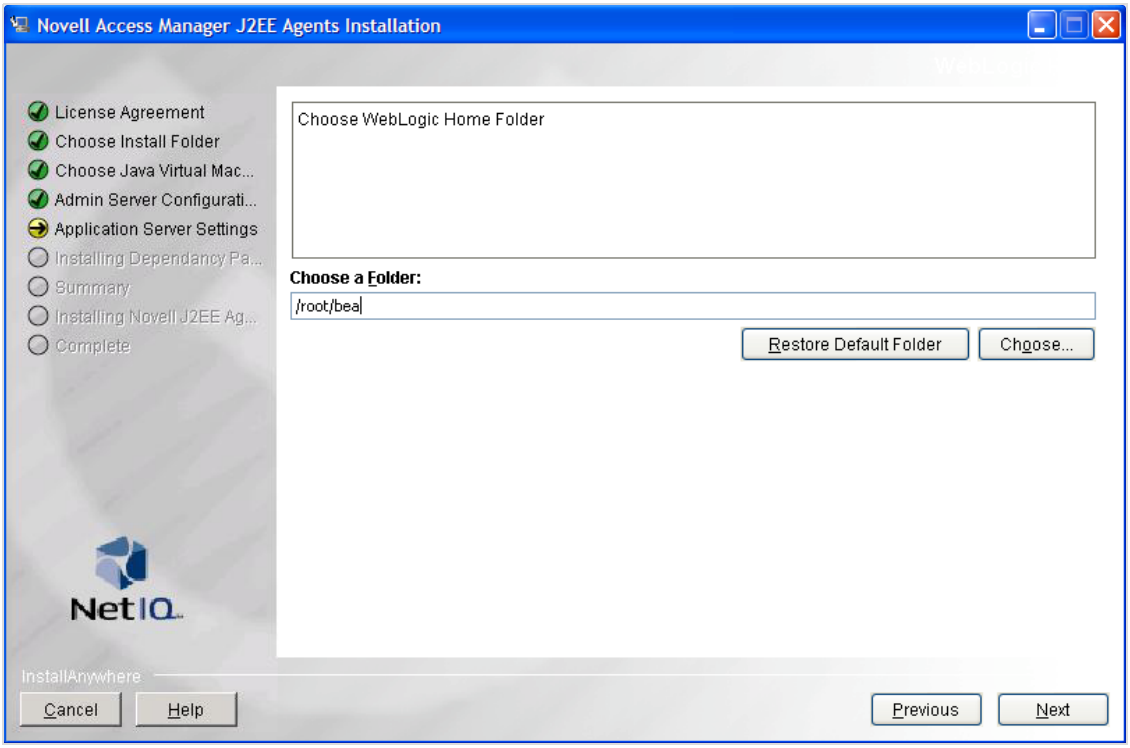


10b (Conditional) If you have the Audit server installed, specify if you want to replace the existing Audit server or use the existing server.

11 Click *Next*. The Select Application Server page is displayed.

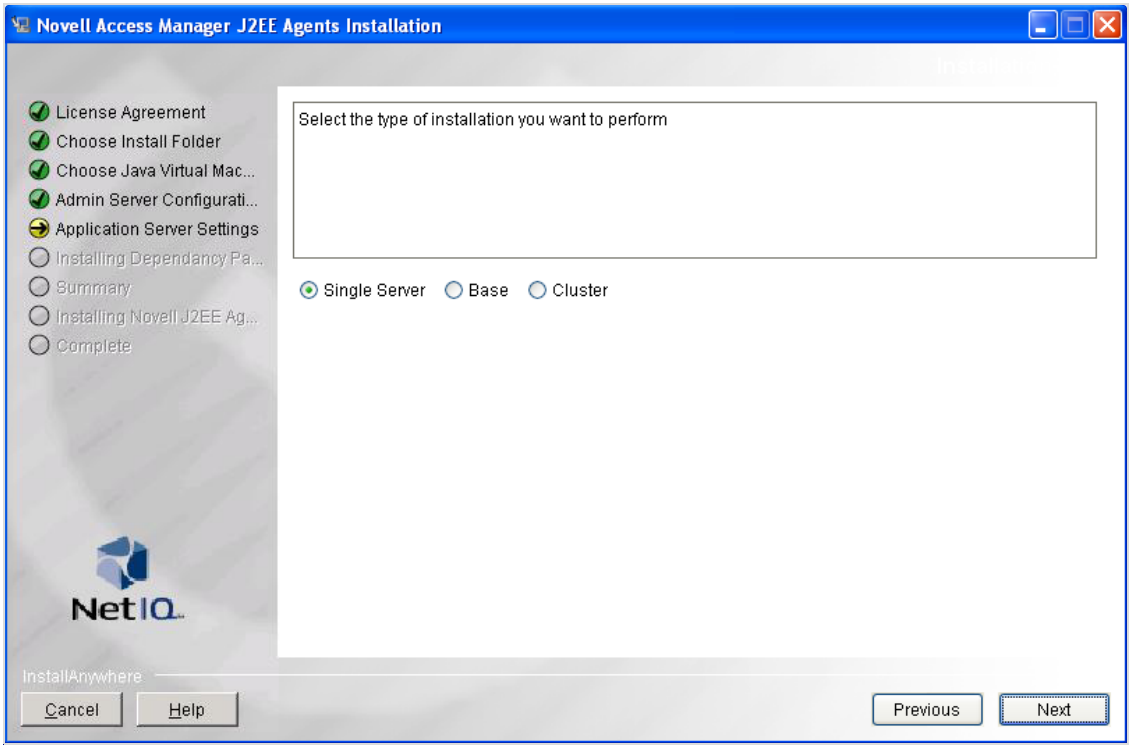


12 Select *WebLogic*, click *Next*. The installation selection page is displayed.



13 Specify the path to the directory where WebLogic is installed, or click *Choose* to select a folder for installation. Click *Restore Default* to restore the default installation location.

14 Click *Next*. The Installation Type page is displayed.



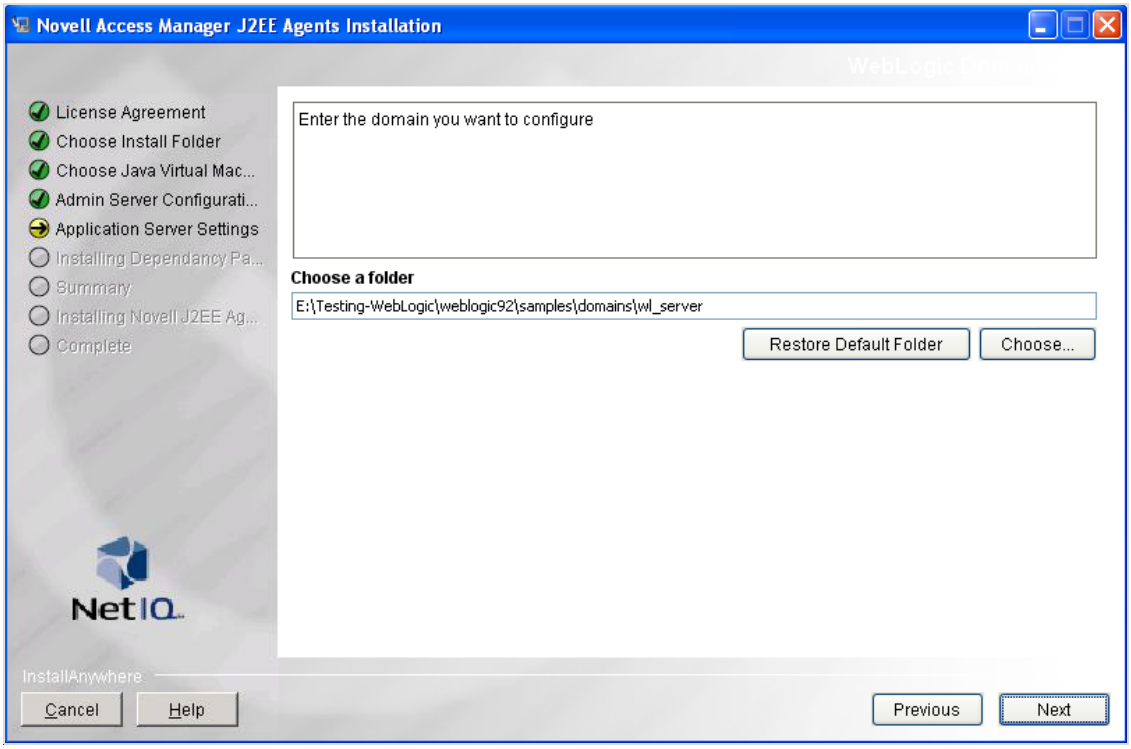
15 Specify any one of the following options, then click *Next*:

Single Server: Select this option to install a single instance of an application server.

Base: Select this option while installing the agent on a machine that acts as a node and is part of a cluster.

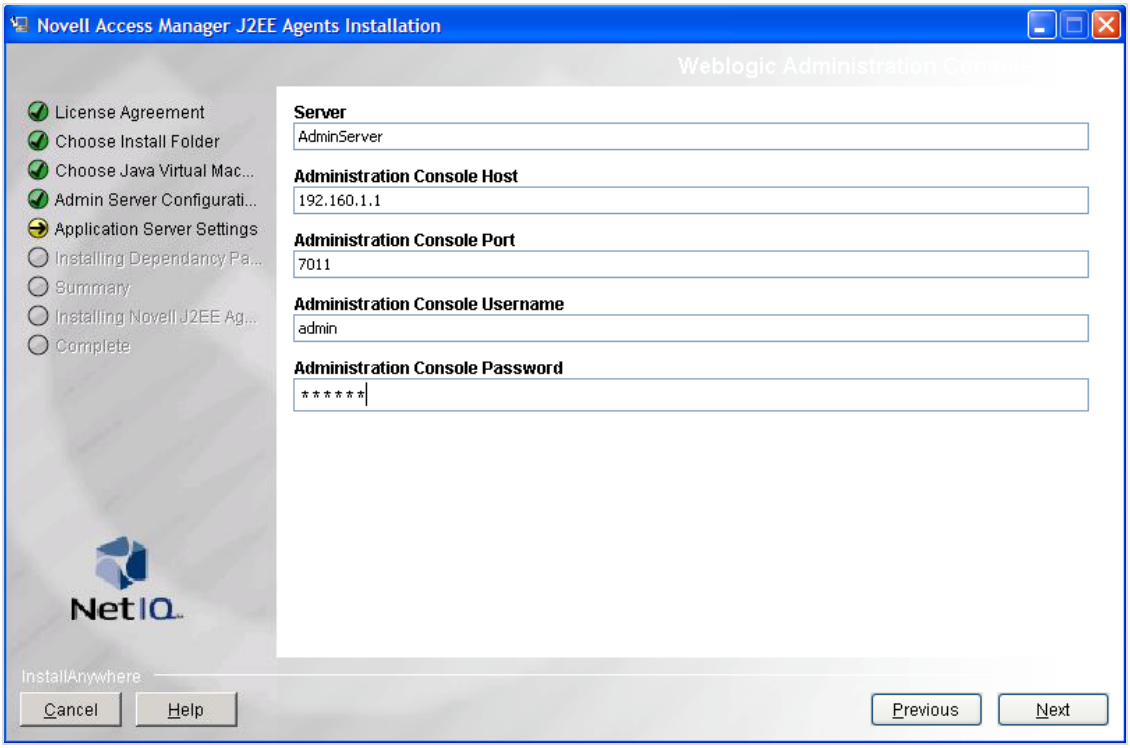
Cluster: Select this option while installing the agent on a machine where the domain is configured.

The WebLogic Domain page is displayed.



16 Specify the WebLogic Domain Home folder. Click *Choose* to select a folder for installation. Click *Restore Default* to restore the default installation location.

17 Click *Next*. The WebLogic Administration Console Details page is displayed.



18 Specify the information required for server communication between the agent and the Administration Console. Fill in the following fields:

Server: Specify the name of the WebLogic Administration Console server.

Administration Console Host: Specify the IP address of the Administration Console.

Administration Console Port: Specify a port number for the Administration Console.

Administration Console Username: Specify the username of the admin user of the Administration Console.

Administration Console Password: Specify the password of the admin user of the Administration Console.

19 Click *Next*. The JCC Dependent Packages Installation page is displayed.

20 Click *Install* to continue with the agent installation.

21 Review the installation summary, then click *Install* to install the agent.

22 Click *Done* when the installation is complete.

23 Complete the procedure in Section 1.7.4, "Configuring WebLogic for J2EE Agents," on page 38.

24 Optional) To verify if the installation of the agent is complete, see Section 1.8, "Verifying If a J2EE Agent Is Installed," on page 40.

25 Stop the WebLogic Server if it is running.

26 (Optional) If you want to deploy the sample Payroll application to test the WebLogic Agent, refer to "Deploying the Sample Payroll Application" on page 93.

1.7.3 Installing a J2EE Agent through the Console

1 Download the agent installer.

For software download instructions, see the Access Manager Readme.

2 Enter the following command in the command prompt to run the installer on the console:

```
<filename> -i console
```

Replace *<filename>* with the name of the J2EE agent installer.

3 Review the License Agreement, then press `Y` to accept it.

4 Specify an absolute path to install the J2EE Agent components, or press Enter to continue with the default installation path.

5 Specify a Java Virtual Machine (JVM) to be used by the installed application.

All the available JVMs are displayed with a number. The default JVM is displayed with an arrow. Press Enter to select the default JVM, or specify the number of one of the listed JVMs.

6 Specify the information required for communication between the agent and the Administration Console:

- Specify the IP address of your Access Manager Administration Console.
- Specify the username and password of the admin user of the Access Manager Administration Console. Confirm the password by re-entering it.
- Review the entered address. If your server is configured for more than one IP address, make sure you specify the IP address of the machine from which the Access Manager Administration Console is reachable.

7 (Conditional) If you do not have the Audit server installed, the J2EE installer installs the Audit server for you. Specify the IP address of the Access Manager Administration Console as the *Audit Server IP*, then press Enter.

8 (Conditional) If the Audit server is already installed on your machine:

8a You are asked to specify if you want to replace the existing Audit server or use the existing server:

- Press 1 to use the existing Audit server.
- Press 2 to replace the existing Audit server.

8b (Conditional) Press 1 to use the existing Novell Audit Configuration.

8c (Conditional) Press 2 to use a different Audit Server and then specify the IP address.

9 Specify 3 for WebLogic, then press Enter.

10 Read the alert message and press Enter to continue.

11 Specify the path to the directory where WebLogic is installed, then press Enter.

12 Specify the WebLogic Domain Home folder, then press Enter.

13 Specify the name of the WebLogic Administration Console server, then press Enter.

14 Specify the IP address of the Administration Console, then press Enter.

15 Specify a port number for the Administration Console, then press Enter.

16 Specify the username of the admin user of the Administration Console, then press Enter.

17 Specify the password of the admin user of the Administration Console, then press Enter.

18 Click *Next*. The JCC Dependent Packages Installation page is displayed.

19 Press Enter.

20 Review the installation summary, press Enter to install the agent, then press Enter again.

21 To verify the installation, see Section 1.8, "Verifying If a J2EE Agent Is Installed," on page 40.

1.7.4 Configuring WebLogic for J2EE Agents

After you install the WebLogic application server, you must configure it for the WebLogic J2EE Agent as follows:

- "Modifying the WebLogic Java Security Policy" on page 38
- "Configuring the Login" on page 38

Modifying the WebLogic Java Security Policy

Java 2 Security uses the `weblogic. policy` file to determine access to resources. You can modify the policy file so that it uses the correct defaults.

1. In a text editor, browse to and open one of the following files, depending on your platform:
   - **Linux:** `<Domain Home>/bin/startWeblogic.sh`
   - **Windows:** `<Domain Home>/bin/startWeblogic.cmd`
2. Remove the following Java parameter:

```
-Djava.security.policy=<filename>
```

3. If you are running WebLogic 9.2, add Java permissions by adding the following lines to the file:

   ```
   grant {
        java.security.AllPermission
   };
   ```

   There appears to be a bug in WebLogic 9.2 that prevents the Administration Console applications from functioning with the default permissions in the weblogic.policy file. This bug also prevents some of the Java 2 permissions for the agent to be explicitly set when the security manager is enabled. The only workaround NetIQ has found is to grant Java 2 permissions to everything. This should not add any more security risk than running WebLogic without the security manager enabled, which is the default configuration for WebLogic.
4. Save and close the file.
5. Continue with "Configuring the Login" on page 38.

After the installation of J2EE Agents, the security policy refers to the `<AGENT_HOME>/weblogic.policy` file.

Configuring the Login

To configure the login, you can use either use a script or the WebLogic Administration Console:

- "Using a Script to Configure Login" on page 38
- "Using the Administration Console to Configure Login" on page 39

Using a Script to Configure Login

1. Start WebLogic.
2. Execute the `weblogic_config.jy` WebLogic scripting tool. Specify the command appropriate for the platform:

   Linux: `WL_HOME/common/bin/wlst.sh`

   Windows: `WL_HOME\common\bin\wlst.cmd`

Use the following parameters to execute the script. Separate each parameter with a space. Running the script without additional parameters prints the required parameters

Parameter	Possible Value	Description
WebLogic administrator username	weblogic	The name of the administrator that you specified when you installed WebLogic.
WebLogic administrator password	password	The password for the specified user.
Domain name	base_domain	Specify the WebLogic domain name.
Server name	AdminServer	By default, WebLogic names the server AdminServer. If you changed this name during installation, specify your name.
Hostname and port	localhost:7001	The host and port are separated with a colon.

Linux Example: `/opt/bea/weblogic92/common/bin/wlst.sh /opt/novell/nids_agents/bin/weblogic_config.jy weblogic password base_domain AdminServer localhost:7001`

Windows Example: `C:\bea\weblogic92\common\bin\wlst.cmd C:\Novell\bin\weblogic_config.jy weblogic password base_domain AdminServer localhost:7001`

3 Restart the WebLogic server.

The agent should import into the Access Manager Administration Console when the WebLogic server starts.

4 (Optional) Verify and test the installation:

- To verify that the agent is installed, see Section 1.8, "Verifying If a J2EE Agent Is Installed," on page 40.
- To test the agent, see Chapter 7, "Deploying the Sample Payroll Application," on page 93.

5 Continue with Chapter 2, "Configuring the Agent for Authentication," on page 43 to configure the J2EE Agent.

Using the Administration Console to Configure Login

In the WebLogic Administration Console, you need to configure the JAAS Login Module:

1 Start WebLogic.

2 In a browser, log in to the WebLogic Administration console:

`http://<weblogic ip>:<Weblogic port>/console`

Replace *<weblogic ip>* with the IP address or DNS name of your WebLogic Administration Console.

Replace *<weblogic port>* with the port number of your Web.

3 In the *Domain Structure* list, click *Security Realms*.

4 Click the default realm (*myrealm*).

5 Click the *Providers* tab.

6 In the top right corner, click *Lock and Edit*.

7 In the *Authentication Providers* list, click *New*.

8 Specify a name in the *name* field, select *NovellAccessManagerAuthenticator* for the *type*, then click *OK*.

9 In the *Authentication Providers* list, click *DefaultAuthenticator* and change the *Control Flag* from *Required* to *Sufficient*.

10 Return to the *Authentication Providers* list.

11 Change the *NovellAccessManagerAuthenticator* > *Control Flag* to *Sufficient*.

12 Click *Activate Changes*.

13 Restart the WebLogic server.

The agent imports into the Access Manager Administration Console when the WebLogic server starts.

14 (Optional) Do the following to verify and test the installation:

- To verify that the agent is installed, see Section 1.8, "Verifying If a J2EE Agent Is Installed," on page 40.
- To test the agent, see Chapter 7, "Deploying the Sample Payroll Application," on page 93.

15 Continue with Chapter 2, "Configuring the Agent for Authentication," on page 43 to configure the J2EE Agent.

1.8 Verifying If a J2EE Agent Is Installed

You can verify the installation of the agent by using the Administration Console.

1 In the Administration Console, click *Devices* > *J2EE Agents*.

If the installation was successful, the IP address of your agent appears in the Server list. The import into Administration Console can take a few minutes, so if your agent does not appear in the list, wait a few minutes, then refresh the screen.

If an agent starts to import into the Administration Console but fails to complete the process, the following message appears:

```
Server agent-<name> is currently importing. If it has been several minutes
after installation, click repair import to fix it.
```

If you have waited at least ten minutes, but the message doesn't disappear and the agent doesn't appear in the list, click the *repair import* link. If the agent isn't in the list and you don't receive a repair import message, verify that you have restarted the J2EE server after installing the agent. The J2EE server must be running for the import process to begin. For additional help, see Section 9.1, "Troubleshooting the J2EE Agent Import," on page 117.

2 The agent must be configured before the Server Status turns green. See Chapter 2, "Configuring the Agent for Authentication," on page 43.

1.9 Uninstalling a J2EE Agent

1 Browse to `<agent Install folder>\Novell Access Manager J2EE Agents\Uninstall_Novell Access Manager J2EE Agents`

2 Double-click the uninstaller.

3 Click *Next* in the Uninstall J2EE Agents page.

4 Select one of the following options in the *Uninstall Options* page:

- **Complete Uninstall:** This removes all the features and files that were installed during the installation.
- **Uninstall Specific Features:** This allows you to select the features that you want to uninstall.

5 (Optional) If you selected the *Uninstall Specific Features* option, select the features that you want to uninstall.

6 Click *Uninstall*.

7 Click *Done* to complete the uninstallation procedure.

2 Configuring the Agent for Authentication

You can configure the Access Manager to interact with your application server.

- You can configure it as an identity provider for the user authentication and user roles. In this configuration, the application server is accessed directly by the user, and the agent is configured to redirect the user to the Identity Server for authentication and user roles. If you need the security of SSL, you need to configure the application server for SSL.
- You can configure it as a protected resource of the Access Gateway. When the agent is configured to be an Access Gateway protected resource, the IP address of the application server is hidden from the user and the user must access it through the Access Gateway. You can configure the Access Gateway to require SSL connections without configuring the application server for SSL.

This section has the following information.

2.1 Prerequisites

- You have set up a basic configuration. See "Setting Up a Basic Access Manager Configuration" in the *NetIQ Access Manager 3.2 Setup Guide*.
- You have a J2EE application server that has an application with security constraints.
- You have configured the Identity Server with policies for the roles required by your application. For the sample payroll application, this is an Employee role and a Manager role.
- You have the agent installed on your J2EE server. See Chapter 1, "Installing the J2EE Agents," on page 9.

2.2 Possible Configurations

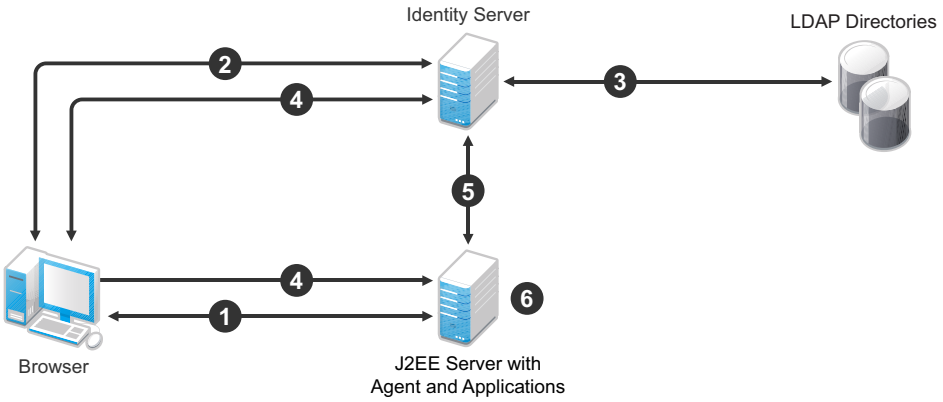
The J2EE server uses the Identity Server for authentication.You can configure your J2EE server in such a way that either the users have direct access to it or the J2EE server is a protected resource of the Access Gateway.

2.2.1 Allowing Direct Access to the J2EE Server

This scenario is most often used when you have users who want to access the application server behind your firewall. This configuration requires an internal DNS server that resolves the DNS name of the application server to its IP address.

When you configure the Identity Server to provide authentication for the applications on the J2EE server, the communication process follows the paths illustrated in Figure 2-1.

Figure 2-1 *JBoss Applications Using the Identity Server*



1. The user requests access to an application on the J2EE server. The user is redirected to the Identity Server.
2. The Identity Server prompts the user for a username and password.
3. The Identity Server verifies the username and password against a user store (an LDAP directory).
4. The Identity Server builds the roles for the user and redirects the user back to the application server.
5. The agent verifies the user's credentials and obtains the user's role information.
6. The application server allows access to the requested application.

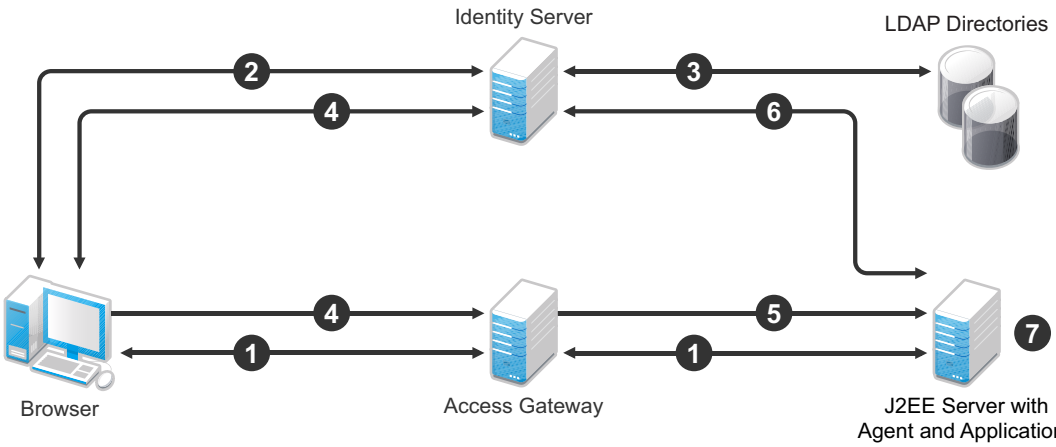
For configuration information, see Chapter 2, "Configuring the Agent for Authentication," on page 43.

2.2.2 Protecting the Application Server with the Access Gateway

In this scenario, the J2EE server with the application is protected by the Access Gateway. The Access Gateway is a reverse proxy server that restricts access to Web-based content, portals, and Web applications that employ authentication and access control policies.

When you configure the Access Gateway to protect the application server, the communication process follows the paths illustrated in Figure 2-2.

Figure 2-2 *The J2EE Server as a Protected Resource*

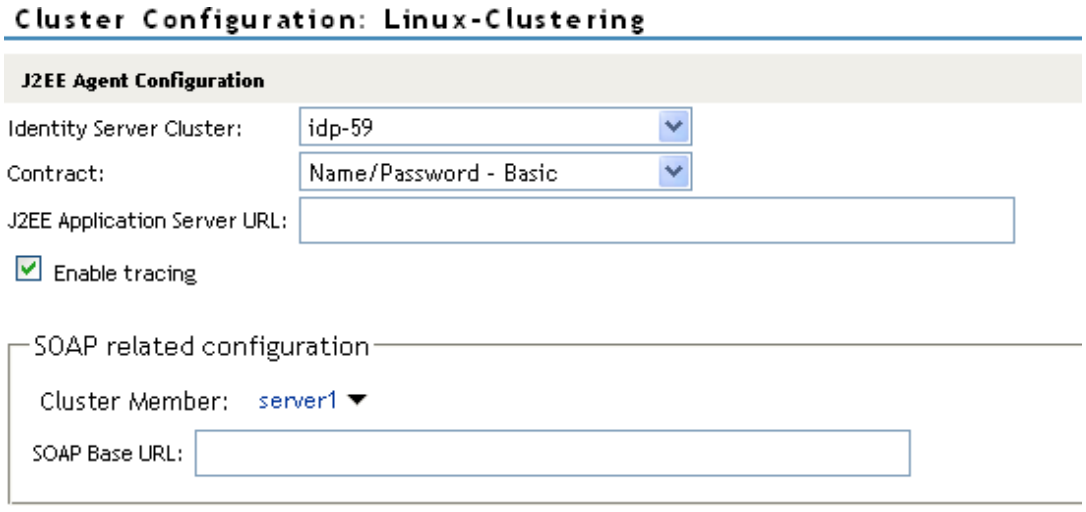


1. The user requests access to the application server by using a published DNS name. The request is sent to the Access Gateway, and the Access Gateway proxies the request to the agent.
2. The agent redirects the request to the Access Gateway, and the Access Gateway redirects the user to the Identity Server, which prompts the user for a username and password.
3. The Identity Server verifies the username and password against a user store (an LDAP directory).
4. The Identity Server builds the roles for the user and redirects the user back to the Access Gateway.
5. The Access Gateway directs the user's request to the application server.
6. The agent verifies the user's credentials and obtains the user's role information.
7. The application server allows the user to access to the requested application.

For configuration information, see Section 2.5, "Protecting the Application Server with the Access Gateway," on page 50.

2.3 Configuring the Agent for Direct Access

1 In the Administration Console, click *Devices* > *J2EE Agents* > *Edit*.



2 Fill in the fields:

Identity Server Cluster: Select the Identity Server you want the agent to trust for authentication by selecting the configuration you have assigned to the Identity Server.

The *[None]* option is used as the default, before you configure the agent.

Contract: Select the type of contract, which determines the information a user must supply for authentication. By default, the Administration Console allows you to select from the following contracts and options when specifying an authentication contract.

- **Name/Password - Basic:** Specifies basic authentication over HTTP, using a standard login pop-up provided by the Web browser.
- **Name/Password - Form:** Specifies a form-based authentication over HTTP, using the Access Manager login form.
- **Secure Name/Password - Basic:** Specifies basic authentication over HTTPS, using a standard login pop-up provided by the Web browser.
- **Secure Name/Password - Form:** Specifies a form-based authentication over HTTPS, using the Access Manager login form.
- **Any Contract:** If the user has authenticated, this option allows any contract defined for the Identity Server to be valid; or if the user has not authenticated, it prompts the user to authenticate by using the default contract assigned to the Identity Server configuration.

You can configure other contract types.

J2EE Application Server URL: Specify the URL to access the application server, including the port. For example, if the DNS name of your J2EE server is j2ee.mycompany.com, enter the following:

```
https://j2ee.mycompany.com:8443
```

SOAP Base URL: Specify the URL used to communicate between the agent components residing in an application server. If you have created a cluster, select each cluster node from the *Cluster Member* drop-down list and specify separate URLs for each node. The SOAP URL must end with `nesp`. For example:

```
https://j2ee.mycompany.com:8443/nesp
```

Both the J2EE application server and SOAP base URL have three parts:

- **Scheme:** For the scheme, specify the scheme you have configured the application server to use for connections (HTTP or HTTPS). See your application server documentation for information on configuring SSL so you can use HTTPS.

  For more information on SSL and the required certificates for the agent, see Section 5.3, "Configuring SSL Certificate Trust," on page 85.
- **Domain:** Specify a DNS name in the URL if you want to configure the application server in such a way that it is accessible internally behind your firewall and externally outside the firewall.
- **Port:** Port 8443 is the standard HTTPS port for an SSL connection to a JBoss server, port 7002 for an SSL connection to a WebLogic server, and port 9443 for an SSL connection to a WebSphere server. The HTTP port is 8080 for JBoss, 7001 for WebLogic, and 9080 for WebSphere. If you have configured a different port, use that port.

3 Click *OK*, then click *Update* > *OK*.

4 To update the Identity Server, click *Identity Servers*, then click *Update* > *OK*.

Whenever you set up a new trusted identity configuration, you need to update the Identity Server configuration.

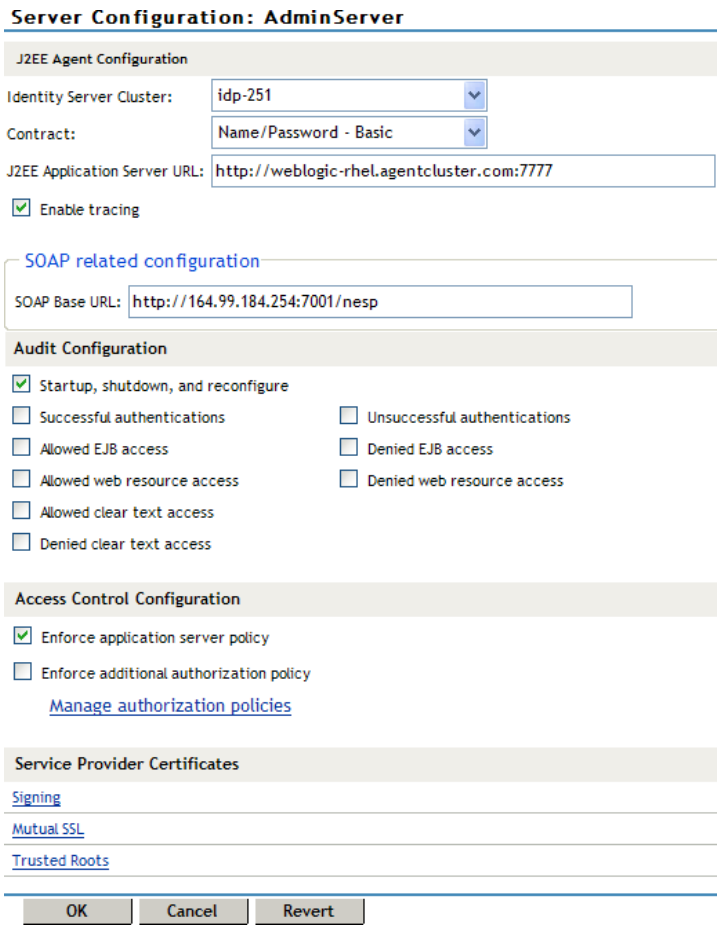
5 Continue with "Preparing the Applications and the J2EE Servers" on page 65.

2.4 Configuring Authentication Contracts

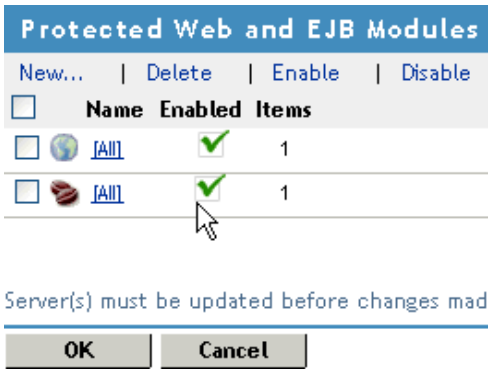
The J2EE Agent now comes with the ability to configure different authentication contracts to protect different applications that reside on the same application server instance. You can also configure additional authentication contracts to applications that require them.

2.4.1 Protecting Different Applications by Using Different Authentication Contracts

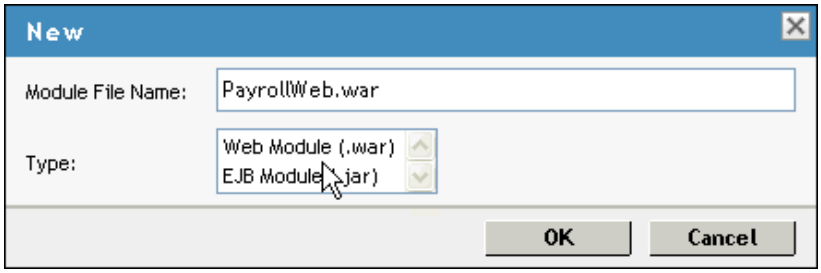
1 In the Administration Console, click *Devices* > *J2EE Agents* > *Edit*. The J2EE Agents Configuration page is displayed.



2 Click *Manage authorization policies* to configure J2EE Agents Policies. The Protected Web and EJB Resource page is displayed.



3 Click *New* to create a new protected Web resource.



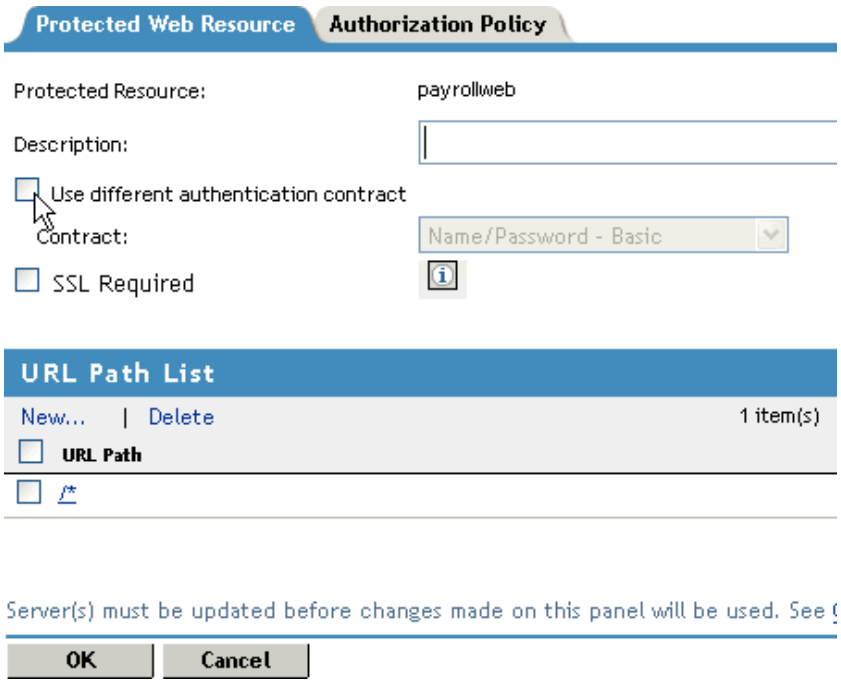
Fill in the following fields:

Module File Name: Specify the name of the file you are protecting, including the file extension (`.jar` or `.war`).

Type: Select *Web Module (.war)* to protect the Web application. You can configure different authentication contracts only for different Web applications.

4 Click *OK*.

5 Click the newly added protected Web resource.



Fill in the following fields:

Protected Resource: Displays the name of the resource you are configuring

Description: (Optional). Provides a field where you can enter a description for this protected resource. You can use it to briefly describe the purpose for protecting this resource.

SSL Required: If this option is selected, the J2EE Agent sets up an SSL connection between the client and the application.

IMPORTANT: If the Web pages that you are now protecting with SSL have been publicly available over HTTP, they remain publicly available over HTTP until you either restart the Web server or reinstall the application. If this is a new application, reinstalling the application might be less disruptive to your network environment than restarting the Web server.

For the JBoss Agent, selecting the *SSL Required* option is only part of the process. On JBoss, you must also either disable the HTTP port and enable the SSL port or configure SSL in the `web.xml` file.

---

6 Click *New* in the *URL Path List* section and add a new URL path, then click *OK*.

For example, to allow access to all the pages in the public directory on the Web server, specify the following path:

```
/public/*
```

To allow access to everything on the Web server, specify the following path:

```
/*
```

To use this protected resource to protect a single page, specify the path and the filename. For example, to protect the `login.html` page in the `/login` directory, specify the following

```
/login/login.html
```

7 Repeat Step 1 to Step 6 for all the applications for which you want to configure different authentication contract.

8 Click *OK*, then click *Update* > *OK*.

9 To update the Identity Server, click *Identity Servers*, then click *Update* > *OK*.

Whenever you set up a new trusted identity configuration, you need to update the Identity Server configuration.

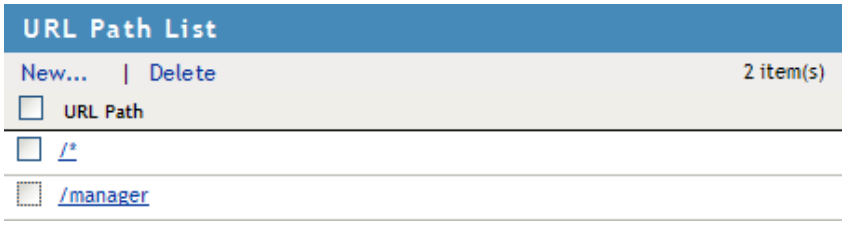
2.4.2 Configuring Additional Authentication for Applications

You might want to configure additional authentication for certain resources. For example, in an organization, certain confidential policies can be viewed only by Managers. In such a scenario, you need to perform additional authentication.

1 Complete the procedure in Section 2.3, "Configuring the Agent for Direct Access," on page 45.

2 Click the protected resource for which you want to add an additional authentication contract.

3 Click *New* in the *URL Path List* section and add a new URL path, then click *OK*.



4 Click *OK*, then click *Update* > *OK*.

5 To update the Identity Server, click *Identity Servers*, then click *Update* > *OK*.

Whenever you set up a new trusted identity configuration, you need to update the Identity Server configuration.

2.5 Protecting the Application Server with the Access Gateway

When you configure the Access Gateway so it can protect your application server, the Access Gateway must be configured to protect multiple resources. The first reverse proxy and proxy service combination of the Access Gateway is assigned to perform authentication. The agent must be set up as a secondary proxy service because the proxy service for an agent cannot be used for authentication.

If the Access Gateway has multiple IP addresses, you can configure the Access Manager so that users access different types of Web resources from each IP address. If the Access Gateway has only one IP address, you still can configure it so users access different types of resources. In this case, you configure the resources to use multi-homing. The following configuration steps assume that you have only one IP address and that you must use multi-homing to access multiple resources, either domain-base or path-based.

With path-based multi-homing, you use one DNS name for the Access Gateway, and have the user specify a path-based URL to access the correct resource. For example:

- You configure the name, www.mytest.com, to resolve to the Access Gateway, and the Access Gateway is configured to proxy the request to a Web server.
- You have users access the application server with the URL www.mytest.com/j2ee. The domain name, www.mytest.com, resolves to the Access Gateway, and the Access Gateway uses the path portion of the URL to proxy the request to the J2EE server.

For more information, see Section 2.5.1, “Setting Up a Path-Based Proxy Service for an Application Server,” on page 50.

With domain-based multi-homing, your Access Gateway uses domain names to access multiple resources. For example:

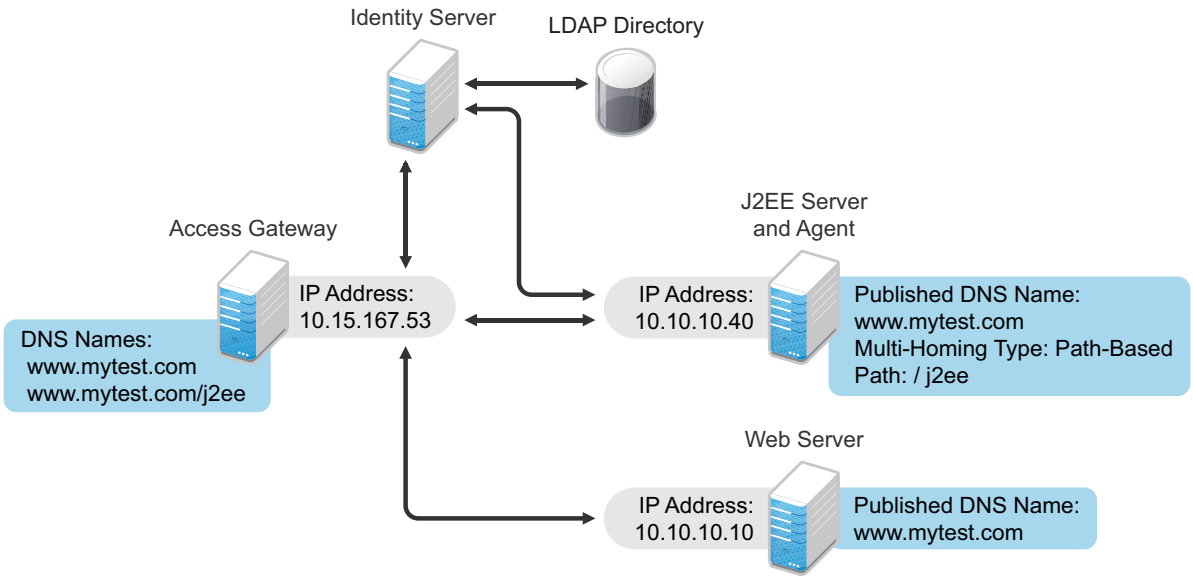
- You configure the name mytest.company.com to resolve to the Access Gateway, and the Access Gateway is configured to proxy the request to a Web server.
- You configure the name j2ee.company.com to resolve to the Access Gateway, and the Access Gateway is configured to proxy it to the application server.

For more information, see Section 2.5.2, “Setting Up a Domain-Based Proxy Service for an Application Server,” on page 54.

2.5.1 Setting Up a Path-Based Proxy Service for an Application Server

Figure 2-3 illustrates the basic configuration for a path-based proxy service. The www.mytest.com name is the published DNS name of the parent proxy service that protects the Web servers. The www.mytest.com/j2ee name resolves to the Access Gateway, and the Access Gateway uses the /j2ee path to proxy the request to the application server.

Figure 2-3 *Protecting the Application Server with Path-Based Multi-Homing*

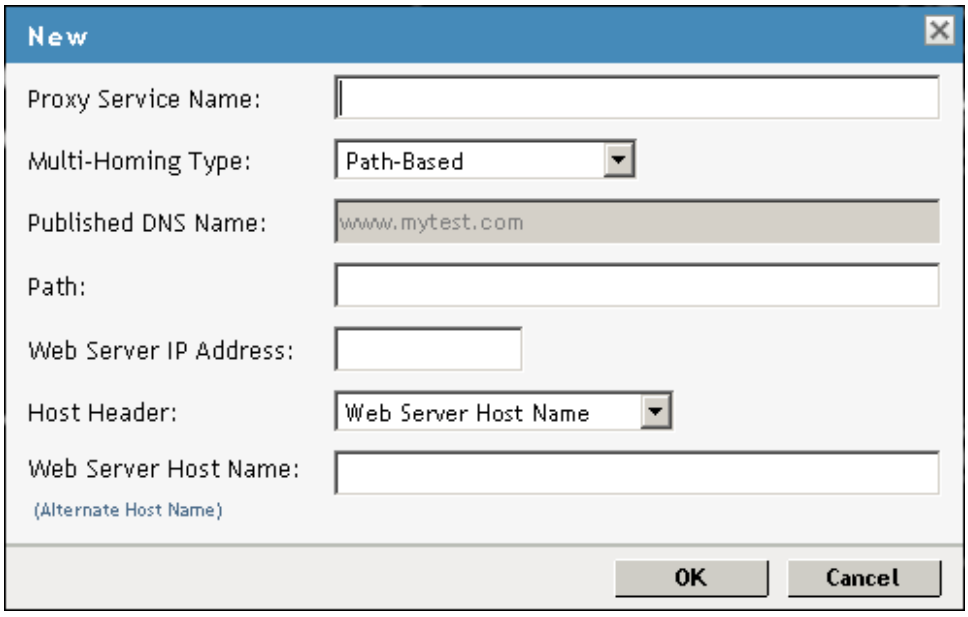


Your DNS server needs to be configured to resolve www.mytest.com and www.mytest.com/j2ee to the Access Gateway.

1 In the Administration Console, click *Devices* > > *Access Gateways* > *Edit* > *[Reverse Proxy Name]*.

The following steps assume that you have already enabled SSL between the Access Gateway and the browsers. If you haven't, see "Configuring SSL Communication with the Browsers and the Identity Server " in the *NetIQ Access Manager 3.2 SP1 Access Gateway Guide*.

2 In the *Proxy Service List* section, click *New*.



3 Fill in the following fields:

Proxy Service Name: Specify a display name for this configuration.

Multi-Homing Type: Select *Path-Based*.

Path. Specify the path for J2EE server. For this example, this is `/j2ee`.

Web Server IP Address: Specify the IP address of the application server. For the configuration in Figure 2-3, enter 10.10.10.40.

Host Header: Select *Web Server Host Name*.

Web Server Host Name: Specify the DNS name of the application server.

4 Click *OK*.

5 To create a protected resource for the application server, select the name of the parent proxy in the *Proxy Service List*.

6 Click *Protected Resources,* then click *New.*

7 Specify a name for the resource, then click *OK*.

Specify a name that allows you to associate this protected resource with your path-based service.

8 Configure the resource for the type of protection you want.

Public Access to the First Page: If you want users to be able to access the first page of the application without authentication, select *None* for the type of contract and accept the default path of `/*` in the *URL Path List*. Click *OK* and continue with Step 9. If you have already created this type of protected resource, you don't need to create another one.

J2EE Agent configuration allows you to set up authentication and access restrictions to the pages in the application.

Authentication Required for the First Page: If you want users to authenticate before they have access to the first page of the application, you need to create two protected resources: one to prompt for authentication and one to allow public access to the nesp application. A path-based service can only have multiple protected resources if the multi-homing path exists on the Web server and the path is not removed when the request is sent to the Web server (see Step 10). To create the multiple resources:

8a For this first protected resource, select *None* for the contract.

8b In the *URL Path List*, specify the path to the nesp application. For this example:

```
/j2ee/nesp
```

8c Click *OK* twice.

8d To add a second protected resource, click *New*, specify a name, then click *OK*.

8e For the contract, select the contract you want to use for authentication.

8f In the *URL Path List*, specify the path to the application. For the sample payroll application, this is the following path:

```
/j2ee/payroll
```

8g Click *OK* three times.

9 In the *Proxy Service List*, select the path-based proxy service.

10 Configure the *Remove Path on Fill* option.

- If the path you specified for the proxy service exists on the Web server and specifies the location of the Web resource, do not select this option.
- If the path you specified for the proxy service does not exist on the Web server, select this option. The *Reinsert Path in "set cookie" Header* option is also selected.

11 In the *Path List* on the Path-Based Multi-Homing page, configure the paths.

- **Remove Path on Fill Service:** If the path is removed before sending the request to the J2EE server, the path specified here must allow public access (no authentication required) to the nesp application. A path is automatically created for you (in this example, `/j2ee`) and a protected resource is assigned. Click the *Protected Resource* link, verify that the contract for this resource is *None* and the path is `/*`, then click *OK*.

  If the wrong type of protected resource is assigned, return to Step 8 and create a protected resource that allows public access.

- **Keep Path on Fill Service:** If you are keeping the path, select the default path and delete it. Click *New*, specify the path to the nesp application (for example, `/j2ee/nesp`), then click *OK*. The protected resource that you created for this path should be automatically assigned to the path.

  Create the path to the application. Click *New*, specify the path to the application (for example, `/j2ee/payroll`), then click *OK*. The protected resource that you created for this path should be automatically assigned to the path.

  If the wrong protected resource is assigned, return to Step 8 and create protected resources with the correct paths.

12 Click the *Web Servers* tab.

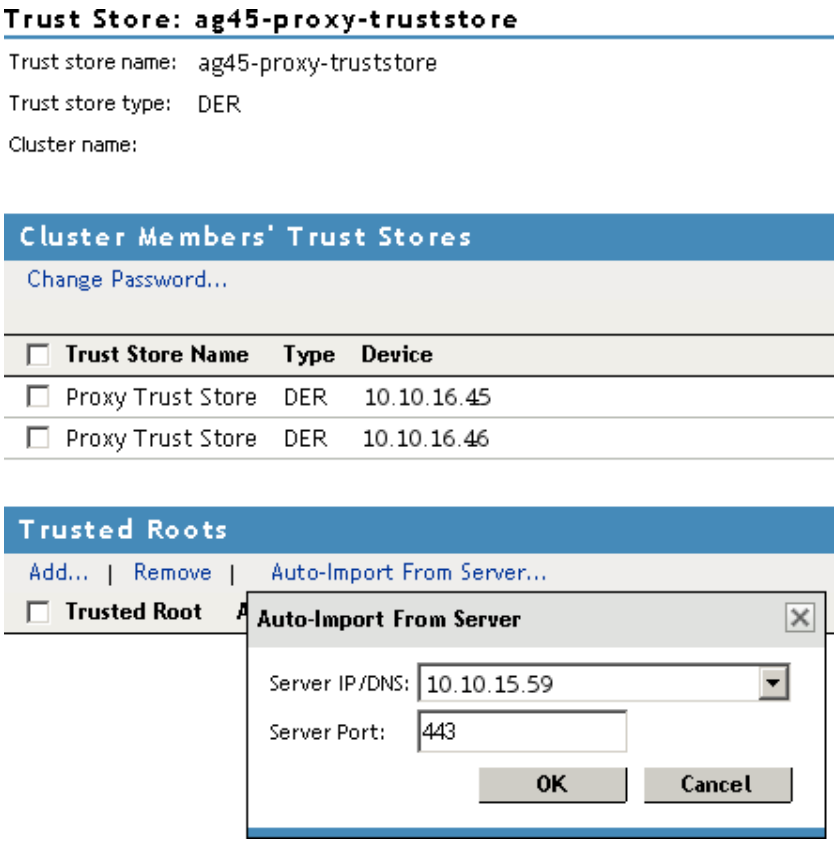
13 To configure SSL, select *Connect Using SSL*.

This option is not available if you have not set up SSL between the browsers and the Access Gateway. See "Configuring SSL Communication with the Browsers and the Identity Server " in the *NetIQ Access Manager 3.2 SP1 Access Gateway Guide* and select the *Enable SSL between Browser and Access Gateway* field.

14 Configure how you want the certificate verified.

- To not verify this certificate, select *Do not verify*.
- To allow the certificate to match any certificate in the trust store, select *Any in Reverse Proxy Trust Store*. Continue with Step 18.
- To add a certificate to the trust store for the application server, click the *Manage Reverse Proxy Trust Store* icon. Continue with Step 15.

The auto import screen appears.



15 Select the IP address of the application server and change the port if the application server is using a different port for SSL.

16 Click *OK*.

The server certificate, the root CA certificate, and any CA certificates from a chain are displayed and selected.

17 Specify an alias, then click *OK*.

18 In the *Connect Port* option, specify the port that your application server uses for SSL connections. For JBoss, the default value is 8443. For WebSphere, the default value is 9443. For WebLogic, the default value is 7002.

19 Click *OK*.

20 Click the *Access Gateways* link.

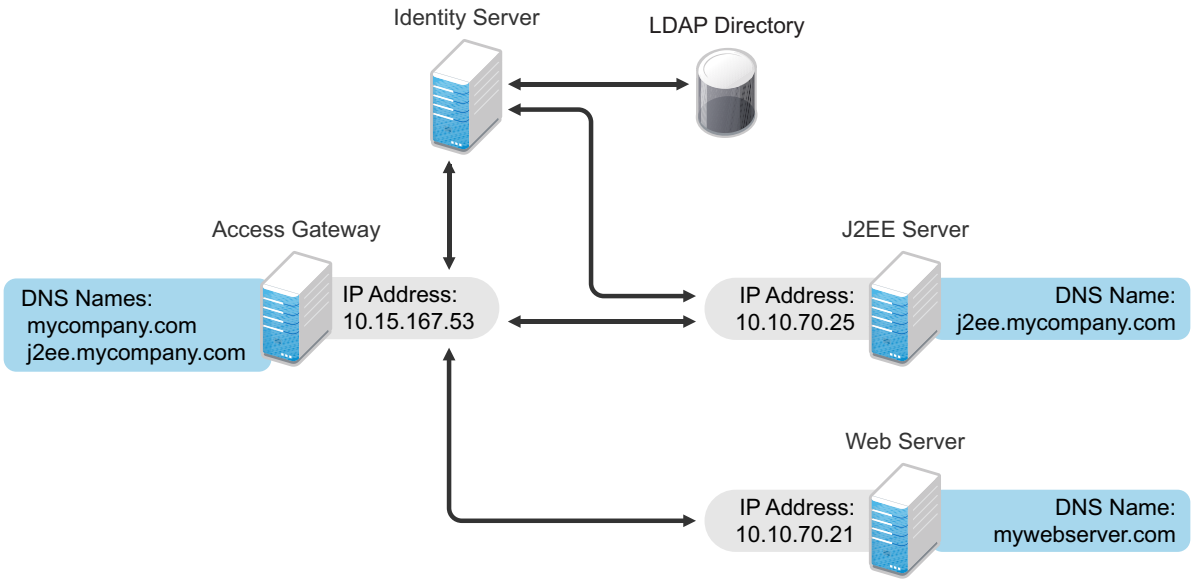
21 On the *Access Gateways* page, click *Update*.

22 Continue with "Configuring a Protected Agent for Access" on page 57.

2.5.2 Setting Up a Domain-Based Proxy Service for an Application Server

Figure 2-4 illustrates the basic configuration for a domain-based proxy service. The mycompany.com name is the published DNS name of parent proxy service that protects the Web server. The j2ee.mycompany.com name is the published DNS name of the proxy service that protects the J2EE server.

Figure 2-4 *J2EE Server as a Domain-Based Protected Resource*

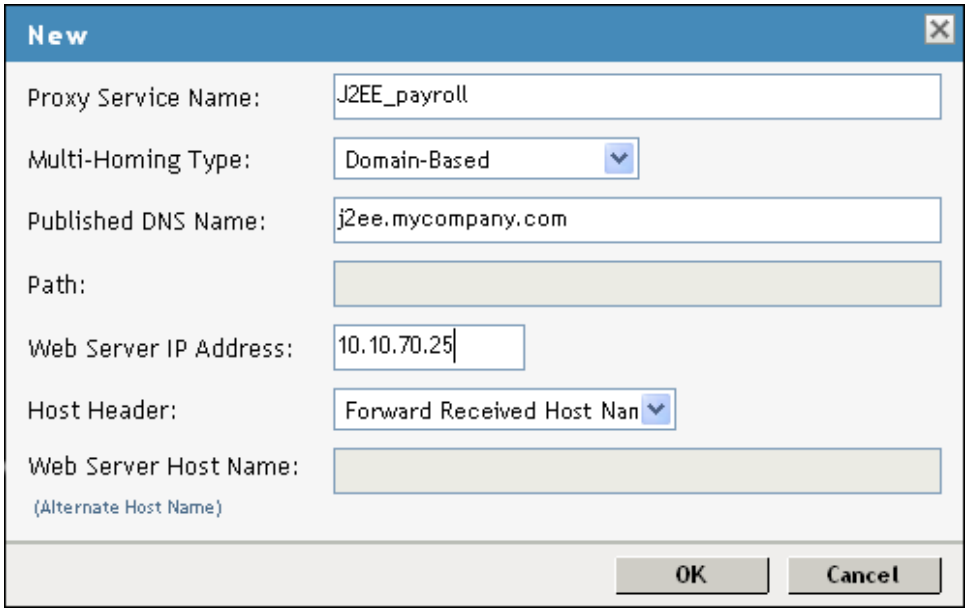


You must set up your DNS configuration so that it resolves mycompany.com and j2ee.mycompany.com to the IP address of your Access Gateway. The Access Gateway URL requests for mycompany.com to the Web server (mywebserver.com) and requests for j2ee.mycompany.com to the application server.

1 In the Administration Console, click *Devices* > *Access Gateways* > *Edit* > *[Reverse Proxy Name]*.

The following steps assume that you have already enabled SSL between the Access Gateway and the browsers. If you haven't, refer to the *NetIQ Access Manager 3.2 SP1 Access Gateway Guide*.

2 In the *Proxy Service List* section, click *New*.



3 Fill in the following fields.

Proxy Service Name: Specify a display name for this configuration.

Multi-Homing Type: Because this configuration example uses a domain name to access the J2EE server, select *Domain-Based*.

Published DNS Name. Specify the domain name for the application server.

Web Server IP Address: Specify the IP address of the application server. For the configuration in Figure 2-4, enter 10.10.70.25.

Host Header: Select either *Forward Received Host Name* or *Web Server Host Name*.

4 Click *OK*.

5 Click the name of the proxy service you just created.

6 Click *Web Servers*.

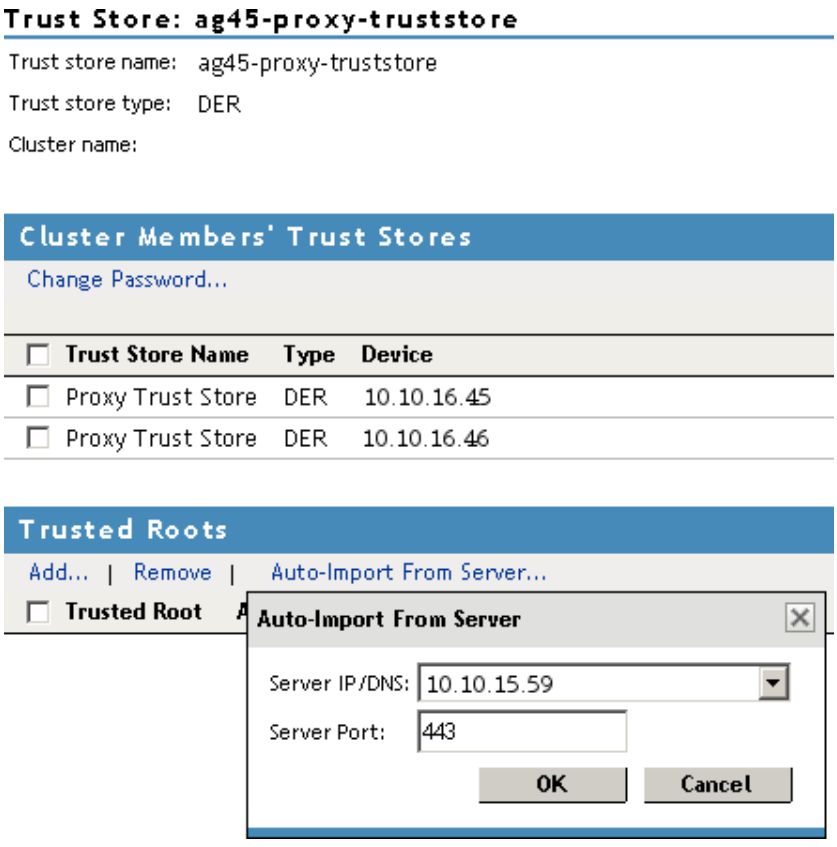
7 To configure SSL, select *Connect Using SSL*.

This option is not available if you have not set up SSL between the browsers and the Access Gateway. See "Configuring the Access Gateway for SSL and Other Security Features" in the *NetIQ Access Manager 3.2 SP1 Access Gateway Guide* and select the *Enable SSL between Browser and Access Gateway* field.

8 Configure how you want the certificate verified.

- To not verify this certificate, select *Do not verify*.
- To allow the certificate to match any certificate in the trust store, select *Any in Reverse Proxy Trust Store*. Continue with Step 12.
- To add a certificate to the trust store for the Web server, click the *Manage Reverse Proxy Trust Store* icon. Continue with Step 9.

The auto import screen appears.



9 Select the IP address of the application server and change the port if the application server is using a different port for SSL.

10 Click *OK*.

The server certificate, the root CA certificate, and any CA certificates from a chain are displayed and selected.

11 Specify an alias, then click *OK*.

12 In the *Connect Port* option, specify the port that your application server uses for SSL connections. For JBoss, the default value is 8443. For WebSphere, the default value is 9443. For WebLogic, the default value is 7002.

13 To create a protected resource for the application server, click *Protected Resources,* then click *New*.

14 Specify a name for the resource, then click *OK*.

15 Configure the resource for the type of protection you want.

Public Access to the First Page: If you want users to be able to access the first page of the application without authentication, select *None* for the type of contract and accept the default path in the *URL Path List*. Click *OK,* then continue with Step 16.

J2EE Agent configuration allows you to set up authentication and access restrictions to the pages in the application.

Authentication Required for the First Page: If you want users to authenticate before they have access to the first page of the application, you need to create two protected resources: one to prompt for authentication and one to allow public access to the nesp application.

15a For this first protected resource, select *None* for the contract.

15b In the *URL Path List*, specify the following path:

```
/nesp
```

15c Click *OK* twice.

15d To add a second protected resource, click *New*, specify a name, then click *OK*.

15e For the contract, select the contract you want to use for authentication.

15f In the *URL Path List*, specify the path to the application. For the sample payroll application, this is the following path:

```
/payroll
```

15g Click *OK* twice.

16 In the *Protected Resource List*, make sure your J2EE protected resources are enabled, then click *OK*.

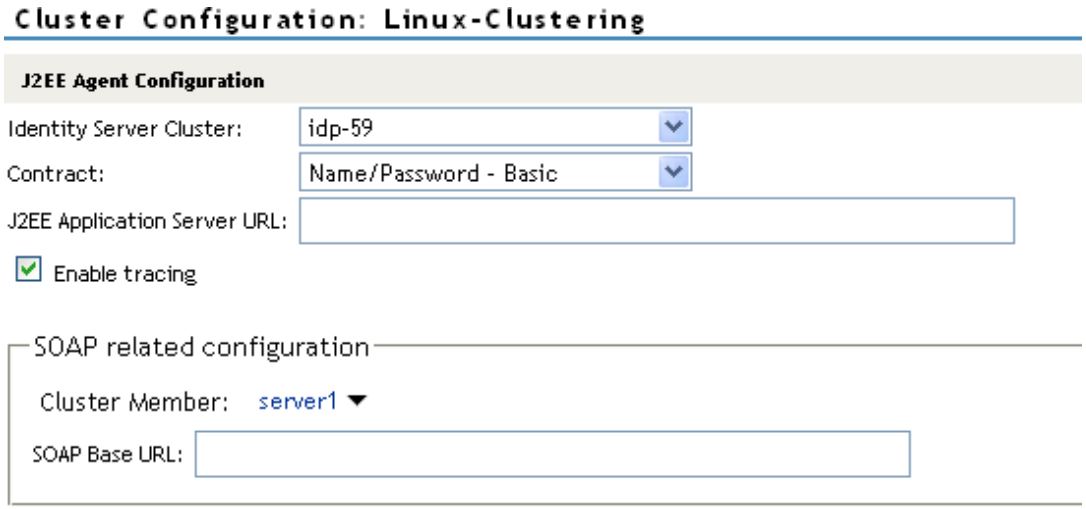
17 Click the *Access Gateways* link.

18 On the *Access Gateways* page, click *Update*.

19 Continue with "Configuring a Protected Agent for Access" on page 57.

2.5.3 Configuring a Protected Agent for Access

1 In the Administration Console, click *Devices* > *J2EE Agents* > *Edit*.



2 Fill in the fields:

Identity Server Cluster: Select the Identity Server you want the agent to trust for authentication by selecting the configuration you have assigned to the Identity Server.

The option is used as the default, before you configure the agent.

Contract: Select the type of contract, which determines the information a user must supply for authentication. By default, the Administration Console allows you to select from the following contracts and options when specifying an authentication contract.

- **Name/Password - Basic:** Specifies basic authentication over HTTP, using a standard login pop-up provided by the Web browser.
- **Name/Password - Form:** Specifies a form-based authentication over HTTP, using the Access Manager login form.
- **Secure Name/Password - Basic:** Specifies basic authentication over HTTPS, using a standard login pop-up provided by the Web browser.
- **Secure Name/Password - Form:** Specifies a form-based authentication over HTTPS, using the Access Manager login form.
- **Any Contract:** If the user has authenticated, allows any contract defined for the Identity Server to be valid; or if the user has not authenticated, prompts the user to authenticate by using the default contract assigned to the Identity Server configuration.

You can configure other contract types.

J2EE Application Server URL: Specify the URL to access the application server, including the port. Select the format based on whether the agent is protected by a path-based or a domain-based proxy service.

- If the agent is protecting a path-based proxy service, specify the published DNS name of the Access Gateway proxy service, including the path. For example:

  ```
  http://j2ee.mycompany.com/j2ee
  ```

- If the agent is protecting a domain-based proxy service, specify the published DNS name of the Access Gateway proxy service. For example:

  ```
  http://j2ee.mycompany.com
  ```

SOAP Base URL: Specify the URL used to communicate between the agent components residing in an application server. If you have created a cluster, select each cluster node from the *Cluster Member* drop list and specify separate URLs for each node. The SOAP URL must end with `nesp`. For example:

```
https://j2ee.mycompany.com:8443/nesp
```

Both J2EE application server and SOAP base URL have three parts:

- **Scheme:** For the scheme, specify the scheme you have configured the Access Gateway to use for connections (http or https). If you have configured the Access Gateway to use SSL, the scheme needs to be https.
- **Domain:** Specify the published DNS name of the Access Gateway proxy service.
- **Path:** (Conditional) If the proxy service is a path-based service, specify the path. For this example, this is `/j2ee`.

3 Click *OK*, then click *Update* > *OK*.

4 To update the Identity Server, click *Identity Servers* > *Update*.

Whenever you set up a new trusted identity configuration, you need to update the Identity Server.

5 Continue with "Preparing the Applications and the J2EE Servers" on page 65.

3 Clustering J2EE Agents

The J2EE Agents can be clustered to provide load balancing and fault tolerance. If the agent where the user's session was established goes down, the user's request is sent to another agent in the cluster. This agent pulls the user's session information from the Identity Server. This allows the user to continue accessing resources, without needing to re-authenticate.

A cluster of J2EE Agents must reside behind a Layer 4 (L4) server. Clients access the virtual IP on the L4, and the L4 alleviates server load by balancing traffic across the cluster of agents. Whenever a user enters the URL for an agent resource, the request is routed to the L4 server, and the L4 routes the user to one of the agents in the cluster, as traffic necessitates.

A cluster is created by assigning one or more agents to a cluster configuration. The agents must all belong to one type. For example, you can have a cluster of WebLogic agents, but not a cluster with both JBoss agents and WebLogic agents.

3.1 Prerequisites

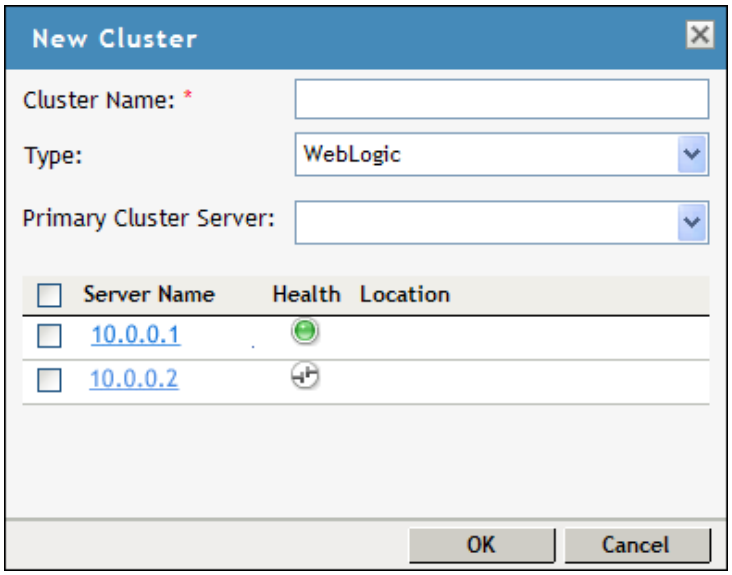
- An L4 switch installed. You can use the same switch for clustering all Access Manager devices.
- The LB algorithm of the L4 switch can be anything (hash/sticky bit), defined at the Real server level.
- Persistence (sticky) sessions enabled on the L4 server. You usually define this at the virtual server level.
- One or more Agents installed. They must all be of one type.
- The base URL DNS name of this configuration must be the virtual IP address of the L4 server. The L4 balances the load between the J2EE Agents in the cluster.
- The application server on which the J2EE Agents reside must support clustering.
- Your DNS server must to be configured to resolve the base URL of the agent cluster to the L4 switch.

3.2 Creating a Cluster Configuration

To create a new cluster of J2EE Agents:

1. In the Administration Console, click *Devices* > *J2EE Agents*.
2. Select the J2EE Agent that you want to add to the cluster, then click *New Cluster*.

The *New Cluster* dialog box appears.



3 Specify the following information:

Cluster Name: Specify a name for the cluster configuration.

Type: Specify if the J2EE agent is a WebLogic Agent, JBoss Agent, or WebSphere Agent. A list of servers is displayed, depending on the selection you make here.

Primary Cluster Server: Select a primary server from the list of servers displayed.

4 Click *OK*.

The status icons for the configuration and the J2EE Agent should turn green. It might take several seconds for the J2EE Agent to start and for the system to display a green status.

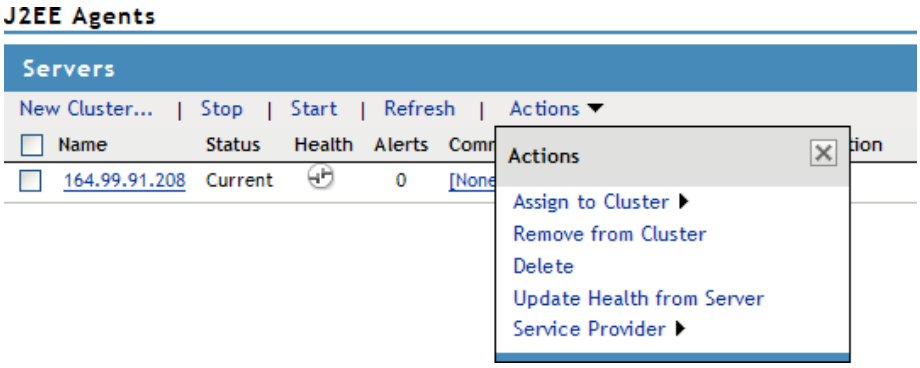
IMPORTANT: Clustering of WebSphere Application Server instances running on the same physical machine is not supported.

3.3 Assigning a J2EE Agent to a Cluster

After you create a cluster, you can assign other J2EE Agents to it. A cluster uses any shared settings you have specified for the primary cluster server.

1 In the Administration Console, click *Devices* > *J2EE Agents*.

2 On the Servers page, select the server's check box, then choose *Actions* > *Assign to Cluster*.



To select all the servers in the list, select the top-level Server check box.

3 Select the configuration's check box, then click *Assign*.

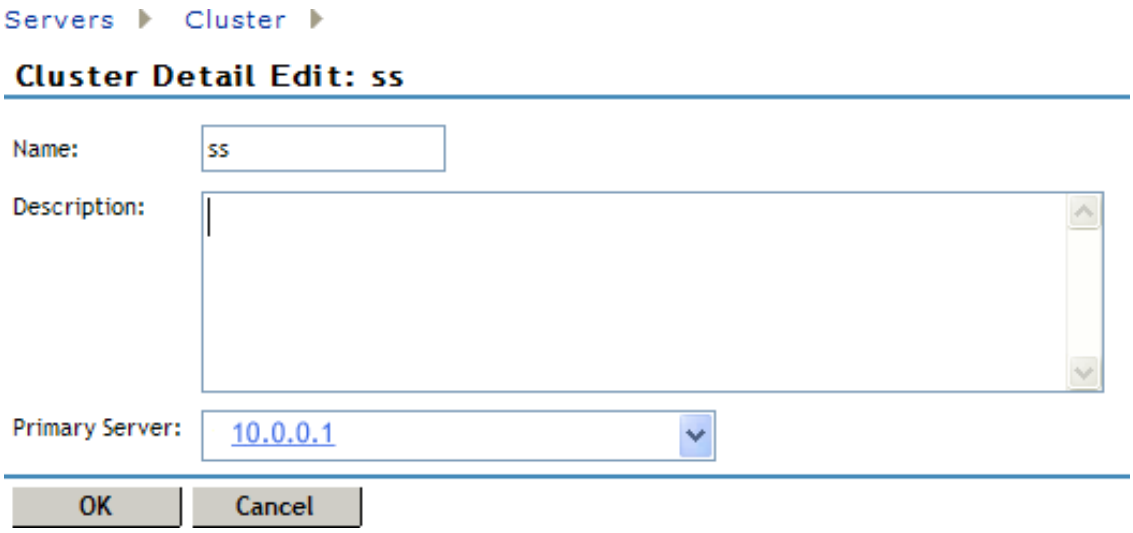
The status icon for the J2EE Agent should turn green. It might take several seconds for the J2EE Agent to start and for the system to display the green status.

3.4 Modifying Cluster Details

1 In the Administration Console, click *Devices* > *J2EE Agents*.

2 To modify the details, click the cluster name.

3 On the Cluster Details page, click *Edit*.



4 Fill in the following fields as required:

Name: Specifies the name of the J2EE Agent cluster configuration. You can modify the name of the cluster.

Description: Specify a brief description of the J2EE Agent cluster.

Primary Server: Specify the IP address of the primary server in that J2EE Agent cluster.

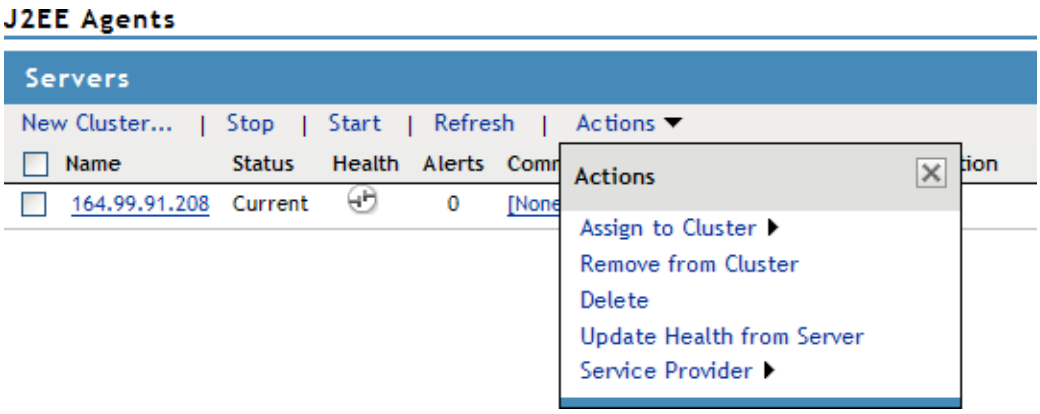
The *Cluster Members* section displays the IP address and other details of the J2EE Agents that are assigned to the cluster.

5 Click *OK*.

3.5 Removing a J2EE Agent from a Cluster

Removing a J2EE Agent from a configuration disassociates the J2EE Agent from the cluster configuration. The configuration remains unchanged and can be reassigned later or assigned to another cluster. You can either remove one member from a cluster or remove all of them at once.

1. In the Administration Console, click *Devices* > *J2EE Agents*.
2. Select the server, then click *Stop*. Wait for the *Health* tab to show a red icon, indicating that the server has stopped.
3. Select the server, then choose *Actions* > *Remove from Cluster*.



4. Click *OK*.

IMPORTANT: If you are not going to assign the agent to another cluster, you need to reconfigure it. You also need to reconfigure the L4 switch and remove this agent from the cluster list.

4 Preparing the Applications and the J2EE Servers

After installing a J2EE Agent and configuring it to use an Identity Server for authentication, you need to configure your applications to use the Identity Server authentication and to configure the security of the J2EE server to interact with the J2EE Agent for authentication and authorization.

4.1 Preparing the Application for the Agent

For each Web application that you want to use with the J2EE Agent, you need to configure the Web application to use the J2EE Agent for login and for logout. You do this by configuring the application’s `web.xml` file:

The `web.xml` file of the sample application (`PayrollApp.ear`) has these modifications. The location of this sample payroll application is platform-specific:

- On a Linux J2EE server, this application is copied to the `/opt/novell/nids_agents/examples` directory.
- On a Windows J2EE server, this application is copied to the `<Install_Directory>\sampleapp` directory.

For more information on the sample payroll application, see Section 1.2, “Overview of the Sample Payroll Application,” on page 10 and Chapter 7, “Deploying the Sample Payroll Application,” on page 93.

4.1.1 Configuring for Login

The Web application needs to be able to log in to the Identity Server that you have configured the J2EE Agent to trust. You accomplish this by specifying that the Web application uses FORM authentication. This is specified in the `<login-config>` section of the application's descriptor in the `WEB-INF/web.xml` file. For example:

```
<login-config>
    <auth-method>FORM</auth-method>
    <form-login-config>
        <form-login-page>/login</form-login-page>
        <form-error-page>/login</form-error-page>
    </form-login-config>
</login-config>
```

The <form-login-page> and <form-error-page> elements need to be set to a URL that is mapped to the following servlet class:

```
com.novell.nids.agent.auth.LoginServlet
```

The above <login-config> element specifies /login as the login page and the error page. The /login URL needs a servlet mapping within the application's web.xml file:

```
<servlet>
    <servlet-name>LoginServlet</servlet-name>
       <servlet-class>
          com.novell.nids.agent.auth.LoginServlet
       </servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/login</url-pattern>
</servlet-mapping>
```

4.1.2 Configuring for Logout

As part of single sign-on and single logout, the J2EE Agent supports the following:

- Notifying the Identity Server about application-level logout events.
- Informing the J2EE applications when the Identity Server logs a user out.

For global logout to function, you need to add a logout servlet and its servlet mapping to the web.xml file:

```
<servlet>
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>
        com.novell.nids.agent.auth.LogoutServlet
    </servlet-class>
      <init-param>
        <param-name>postLogoutURL</param-name>
        <param-value>/loggedOut</param-value>
     </init-param>
     <init-param>
         <param-name>websphereLTPAMechanism</param-name>
         <param-value>false</param-value>
    <description>
ThisshouldbesettotrueinordertoclearLTAPcookiesandtokensin
case of websphere with LTPA as authentication mechanism
    </description>
         </init-param>
</servlet>

<servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/logout</url-pattern>
</servlet-mapping>
```

Two parameters are defined in this servlet: the `postLogoutURL` parameter and the `WebsphereLTPAMechanism` parameter.

- The URL pattern of the LogoutServlet can be customized for the application's requirements. To cause the LogoutServlet to notify the Identity Server about a user logging out, the user is redirected to the URL in the Web module as specified by the postLogoutURL servlet initialization parameter. If it is not specified, the LogoutServlet defaults the postLogoutURL to `/`.
- The `<param-value>` for the `WebsphereLTPAMechanism` parameter is set to `false` by default. When the WebSphere server is configured to use the LTPA authentication mechanism, the `<param-value>` must be set to `true` so that when the global logout is performed, the J2EE Agent clears the LTPA cookie.

  If the `<param-value>` is not set to `true` and the LTPA cookie is not cleared during the logout, the users have problems connecting from a browser that was not closed after a previous logout.

  This`<param-value>` is also available in the `web.xml` file of the sample PayrollApps.

More than one `<url-pattern>` value can be specified for the LogoutServlet. The function of the LogoutServlet is to notify the Identity Server about the application logout. The Identity Server is responsible for notifying all other components about the logout.

4.2 Configuring Applications on the JBoss Server

4.2.1 Configuring a Security Domain

JBoss needs to know that your Web application is a part of the security domain that requires the Identity Server JAAS login module. You do this by specifying your application's security domain in the `<jboss-web>` element of the `jboss-web.xml` file located in your application’s `WEB-INF` directory. You might need to create this file, if your application hasn’t already required you to create it.

The J2EE Agent installation program modifies the `login-config.xml` file in the `${JBOSS_HOME}/server/default/conf` directory and sets the name attribute of the `<application-policy>` element to `novell-idp`.

You need to set the `<security-domain>` element in the `jboss-web.xml` file to this value. Add the following lines to this file:

```
<jboss-web>
    <security-domain>java:jaas/novell-idp</security-domain>
</jboss-web>
```

The `jboss-web.xml` file of the sample application (`PayrollApp.ear`) has these modifications. (For the location of this application, see Section 2.1, “Prerequisites,” on page 43.)

4.2.2 Configuring Security Constraints

If you specify a security constraint similar to the following in the `web.xml` file of an application, the users are redirected for authentication as soon as they access any URL of the application:

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>All web resources</web-resource-name>
      <url-pattern>/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>Manager</role-name>
   </auth-constraint>
</security-constraint>
```

After authenticating to the Identity Server, all users receive an error:

- If the user has the Manager role, the user sees a 404 error stating that j_security_check is not available.
- If the user does not have the Manager role, the user sees a 403 Access Denied error to the login servlet.

When us the J2EE Agent with a JBoss server, you cannot give the `<url-pattern>` element a value of / * or / for a login page that requires authentication. The JAAC provider in the JBoss server is not informed about the login servlet. For example, suppose that the login page for the application has a configuration similar to the following:

```
<login-config>
   <auth-method>FORM</auth-method>
   <form-login-config>
      <form-login-page>/login</form-login-page>
      <form-error-page>/error.jsp</form-error-page>
   </form-login-config>
</login-config>
```

You need to configure the `/login` directory to allow access. For example:

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>Allow Form Login page</web-resource-name>
      <url-pattern>/login</url-pattern>
   </web-resource-collection>
</security-constraint>
```

4.2.3 Configuring for Roles

For the J2EE Agent to enforce authentication for a `.war` file, the JBoss server must have a `web.xml` file that contains a URL with a role restriction. You can use the generic authenticated role for this URL. This policy triggers authentication, and the J2EE Agent policies can then be used to determine authorization. The following is a sample security constraint for a `web.xml` file that triggers authentication for any path below the `protected` directory:

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>Protected Content</web-resource-name>
      <url-pattern>/protected/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>authenticated</role-name>
   </auth-constraint>
</security-constraint>

<security-role>
    <description></description>
    <role-name>authenticated</role-name>
</security-role>
```

The role must be declared with the `<security-role>` tags when it is used inside a security constraint.

4.3 Configuring Applications on the WebSphere Server

4.3.1 Configuring for Authentication

You need to create policies that deny access to the anonymous user. You can do this either with the `web.xml` file within the `.war` file or with Access Manager policies. In Access Manager, you deny access to the anonymous user by creating an authorization policy that denies access to anyone who has not been assigned the `authenticated` role. Anonymous users who haven't authenticated do not have this role, and users who have authenticated to Access Manager are automatically assigned this role.

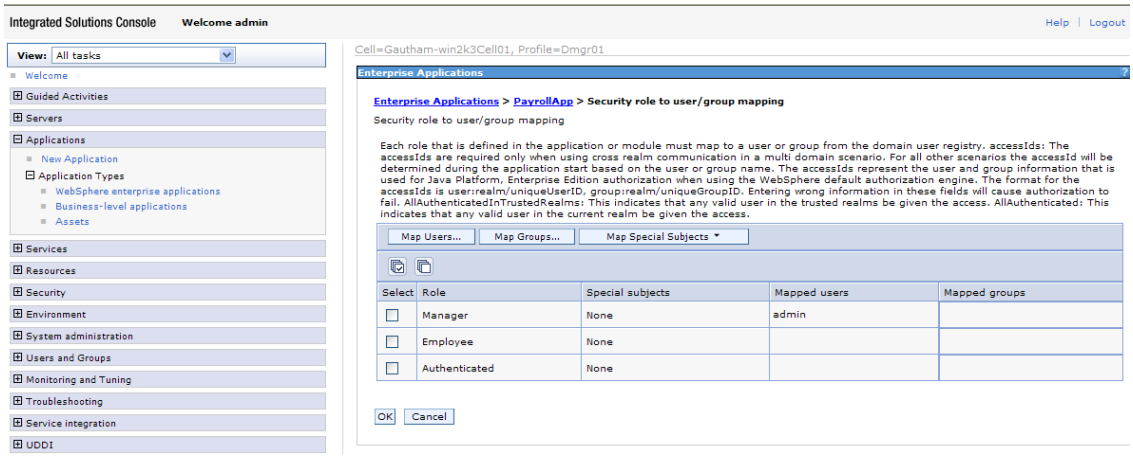
If you have pages that call Enterprise JavaBeans that are protected, you should assign a policy to these pages that denies access to users who have not authenticated.

NOTE: Both Section 4.3.2, “Configuring Security Role to User/Group Mapping,” on page 69 and Section 4.3.3, “Configuring for User RunAs Roles,” on page 70 are applicable only to sample payroll application.

4.3.2 Configuring Security Role to User/Group Mapping

You need to configure security role for user or group and map the roles.

1 In the Integrated Solution Console, Select *Applications* > *Websphere Enterprise Applications* > *Payrollapp* > *Detail Properties* >*Security Role to User/Group Mapping*.



2 Click *Map Users* tab.

3 Select Manager role check box to map the user.

4 Enter the user name in the search string. For Example - admin

5 Click *Search*.

6 Select the users from the *Available List* to the *Selected List*.

7 Click *OK*.

8 Click *Save* to configure the changes.

9 The mapped user is displayed for the selected role.

4.3.3 Configuring for User RunAs Roles

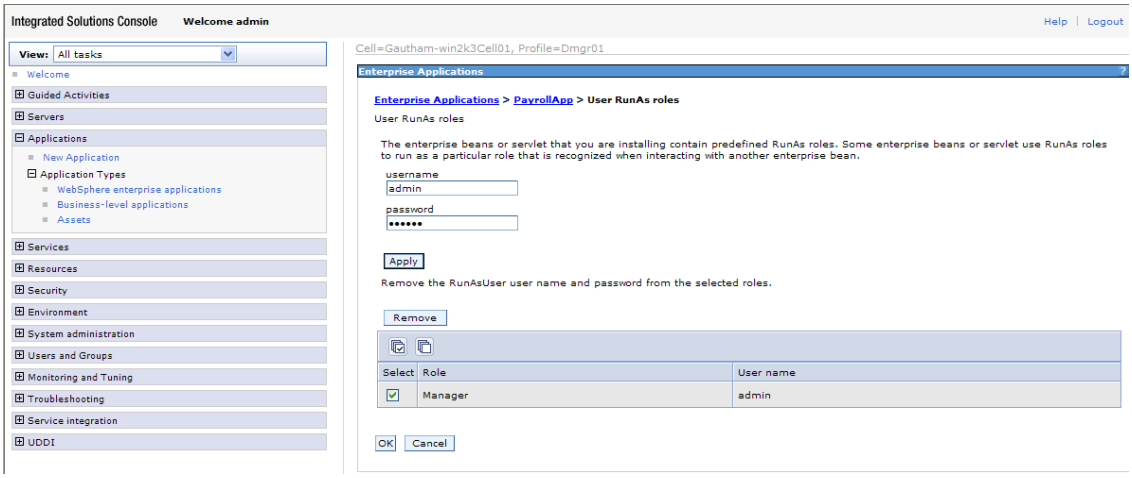
An Enterprise JavaBean deployment descriptor can state that an Enterprise JavaBean must run with a particular role. The the sample application (`PayrollApp.ear`) includes such a statement in its descriptor:

```
<security-identity>
    <run-as>
       <role-name>Manager</role-name>
    </run-as>
</security-identity>
```

Without configuring WebSphere to map a RunAs role to a user, WebSphere ignores this statement. If a user is mapped to a RunAs role, the agent cannot know which J2EE roles the user has unless the role is also mapped.

To configure mapping for RunAs roles, complete the following during WebSphere deployment:

1 In the Integrated Solution Console, Select *Applications* > *Websphere Enterprise Applications* > *Payrollapp* > *Detail Properties* > *User RunAs Roles*.



2 Enter the user name and the password. For Example - admin

3 Select the User role check box. For Example - Manager.

4 Click *Apply*.

5 Click *OK*.

6 Click *Save* to configure the changes.

7 The mapped user is displayed for the selected role.

The WebSphere server uses this mapping to assign a user to execute an Enterprise JavaBeans method.

When you configure the WebSphere mapping for RunAs roles, WebSphere updates the J2EE agent configuration on the WebSphere server. To ensure that this was updated properly or to update the configuration manually, see Section 9.8, "Authorization Fails in the WebSphere Application," on page 120.

NOTE: The J2EE Agent may not send the user roles to application when running on WebSphere. To fix this issue, follow the steps provided in Section 9.8, "Authorization Fails in the WebSphere Application," on page 120.

4.3.4 Configuring the Trust Association Interceptor Module for WebSphere Application

The Trust Association Interceptor (TAI) module is a plug-in for WebSphere Application Server. It is a Java class, packaged into a Java Archive file (JAR) that is placed within the application code sections of the WebSphere Server file system.

The TAI module, which has a class name of `com.novell.consulting.nl.-accessmanager.tai.Roller`, provides role provisioning services to WebSphere Application Server (WAS) and WebSphere Portal Server (WPS). HTTP requests from end-user Web browsers are intercepted by Access Manager enriched with supporting information, passed on by Access Manager to WebSphere Application Server and offered to the TAI for validation.

How Does the TAI Module Works?

The TAI module provides role provisioning services to WAS and WPS.The supporting information placed in each request consists of the following fields:

- **Secret Key String:** This is used by the TAI to validate that the process has actually gone through Access Manager.
- **User Name:** This is the short user name retrieved by Access Manager when the user authenticates.
- **User ID:** This is the long user name in LDAP format.The fully qualified distinguished name of the authenticated user within the LDAP store to which Access Manager, WebSphere Application Server, and WebSphere Portal Server are connected.
- **Cache Key:** This is the identifier for the user session, as generated by Access Manager.
- **User Roles:** This is a list of the iManager roles for the user.

All fields are fixed strings, stored within the `HttpServletRequest` as retrievable HTTP headers. When TAI receives a request, it takes the following actions:

- It verifies that the request contains a secret key value that matches the value specified in the TAI configuration.
- It prepares for the username, user ID, and cache key to be passed on to WebSphere Application Server, as attributes (`WSCREDENTIAL_SECURITYNAME`, `WSCREDENTIAL_UNIQUEID` and `WSCREDENTIAL_CACHE_KEY`) of a subject within a `TAIResult`. If a cache key was not provided by Access Manager, the TAI generates one that is based on the current server system time.
- It filters certain roles out of the list provided by Access Manager, and formulates a full LDAP group DN for each of those. The resulting set of group distinguished names is placed in the `TAIResult` as the `WSCREDENTIAL_GROUPS` subject attribute. WebSphere Application Server can perform a lookup in an LDAP store to which it is connected.
- It filters other roles out of the Access Manager list, constructs another LDAP group DN for each of them, and then establishes the actual membership for the groups by performing LDAP query and update transactions against the LDAP store with which it is configured. With the group objects prepared in this way, WebSphere Portal Server can subsequently query the LDAP store for the members, thus indirectly consuming the role information previously gathered by Access Manager.

Methods

The TAI classes implement five methods:

- **initialize(Properties):** Module initialization, based on a configuration that is provided to the TAI as a java.util.Properties set.
- **getType():** Returns the module's Java class name, thereby identifying it to WebSphere Application Server (WAS).
- **get Version():** Returns the module's version number, normally a fixed string.
- **isTargetInterceptor(HttpServletRequest):** Establishes whether this particular TAI instance (of wnegotiateValidateandEstablishTrust(HttpServletRequest, HttpServletResponse)negotiateValidateandEstablishTrust(HttpServletRequest, HttpServletResponse) performs the validation of a particular HTTP request, throwing an WebTrustAssociationFailedException on failure.
- **cleanup():** Releases any resources held by the TAI while in its active state.

Configuration Properties

The initialize() method of the TAI currently recognizes the following configuration properties:

- **Secret-Value:** Value of the authentication secret placed into requests by Access Manager.
- **Secret-Header:** Name of the HTTP request header in which the secret value is placed. As with the other...-header properties, Access Manager. is expected to set the header in question.
- **User-Name Header:** Name of the HTTP request header that contains the short user name. It is passed on to WebSphere Application Server as the WSCREDENTIAL_SECURITYNAME attribute.
- **User-ID-Header:** Name of the HTTP request header that contains the fully distinguished user name in LDAP format. It is passed on to WebSphere Application Server as the WSCREDENTIAL_UNIQUEID attribute, and used in the arrangement of group membership for role determination by WebSphere Portal Server.

- **Cache-Key-Header:** Name of the HTTP request header that contains the cache key for the session. It is initialized by the TAI to the current system time in milliseconds, and expressed as a decimal number when it is unset by Access Manager. It is passed on to WebSphere Application Server as the WSCREDENTIAL_CACHE_KEY attribute.
- **Role-Header:** Name of the HTTP request header that lists the user's roles.
- **Role-Separator:** Fixed character string that separates individual role names. Used with the role-header, update-roles, and presentation-roles values, all concatenations of role names.
- **Presentation-Roles:** Names of the roles that should be presented to WebSphere Application Server (through the WSCREDENTIAL_ GROUPS attribute) as the names of LDAP groups of which the user is a member. Individual presentation role names are separated by the (global) role separator string.
- **Update-Roles:** Names of the roles for which the TAI should prepare corresponding LDAP group objects for direct LDAP readout by Websphere Portal Service.
- **Presentation-Container:** Distinguished name of the LDAP container that is expected to contain the WebSphere Application Server groups; that is the objects that are merely presented as being groups of which the user is a member.
- **Update-Container:** Distinguished name of the LDAP container that is expected to contain the groups for which the TAI should actually manipulate the membership.
- **Update-Connection** :URL, in RFC 2255 format, of the LDAP server/store on which group membership should be established for readout by WebSphere Portal Server.Values for this property would typically look like ldap://localhost:1389. The port section of the URL is optional, and only plain-text LDAP is currently supported.
- **Update-User:** Distinguished name of the user that logs in to the LDAP group server.
- **Update-Password:** Password of the user that logs in to the LDAP group server.
- **Debug-Level:** Debug level for the textual output generated by the TAI. It can be Off, Severe, Warning, Info, Config, Fine, Finer, Finest, or All.

Selective Deployment

By selectively leaving out specific configuration properties from the TAI's configuration, the TAI can be configured to refrain from certain activities.

- When presentation-roles or presentation-container is left unspecified, the TAI does not formulate a WSCREDENTIAL_GROUPS attribute for WebSphere Application Server.
- When update-user or update-password is left unspecified, the TAI can connect to an LDAP store to manipulate group membership for read-out by WebSphere Portal Server, but does not authenticate (bind) across that connection. Instead, it performs what amounts to an anonymous login. This approach decreases security but improves performance.
- When update-connection or update-container is left unspecified, the TAI performs no updates at all. For example, there would be no need for the TAI to update groups within an LDAP store if it is deployed in an environment where only WebSphere Application Server (and not WebSphere Portal Server) is used in conjunction with Access Manager.

Update Behavior

One of the TAI's key pieces of functionality is the establishment of group memberships for the currently logged-in user (as identified by Access Manager.) within an LDAP store that is queried by WebSphere Portal Server. Here, the LDAP store (normally an instance of eDirectory) is used as a means of indirect communication between Access Manager and the WebSphere Portal Server. Before control is passed on to WebSphere Application Server (which in turn calls upon WebSphere Portal

Server), the TAI ensures that the group situation in the LDAP store is in line with the role definition as provided by Access Manager., through WebSphere Application Server.Through its configuration, the TAI has been notified of the names of roles that can be mapped directly to LDAP groups, which give TAI access to a virtual catalog of roles and group objects. Each separate service request (initiated by Access Manager and delivered to the TAI by WebSphere Application Server) lists the roles to which Access Manager. believes the identified user belongs. From that actual list of roles, the TAI derives whether or not the user should be a member of each of the groups it knows. It then needs to proceed with implementing the group membership, which can include the following:

- Adding the user's distinguished name to the member attribute of group objects.
- Removing the user's distinguished name from the member attribute of group objects.
- Adding information to or removing information from the groupMembership attribute of the user object.

The TAI could also simply replace the previous member and groupMembership values for all objects involved. However, that would require a great number of writes to the LDAP store, operations that (in terms of time) are much more costly than reads. The TAI therefore performs a series of queries, one for each group, to determine whether the user is currently a member. If membership is not what it should be, the TAI synthesizes a modification of the individual group object.In the reverse direction, a similar optimization is applied, in which updates to the groupMembership back reference attribute are combined into a single joint LDAP modification.

Implementing the Trust Association Interceptor Module

The TAI module is implemented in eDirectory, WebSphere Application Server, and Access Manager.

Configuring eDirectory

Use the following configuration for eDirectory:

- Place all application groups inside a container. For example, `ou=Groups,o=MyOrg`
- Create a wpstaibind user. For example, cn=wpstaibind,ou=Admins,ou=Services,o=MyOrg. This user updates the LDAP groups for the TAI module. Assign the following rights to this user:
  - Create and Modify rights to the ou=Groups,o=MP container.
  - Modify rights to the Membership attribute of all users under the user container.
- Create a cn=wasadmins,ou=Groups,o=MyOrg group for all WebSphere Application Server administrators.

NOTE: The exact location of WebSphere Portal Server groups can change to a specific application container below the ou=Groups,o=MyOrg container.

Configuring the WebSphere Application Server

Copy the following files to the `/usr/WebSphere/AppServer/lib` folder:

- `ldap.jar`
- `utilities.jar`
- `roller.jar`

NOTE: The `ldap.jar` and `utilities.jar` files are found in the Novell LDAP SDK, located at LDAP Classes for Java (http://developer.novell.com/wiki/index.php/LDAP_Classes_for_Java). The `roller.jar` file is available in `novell-access-manager.tar.gz` (Linux installer).

To configure and enable the TAI module:

1. Log in to the WebSphere Application Server Admin Console and go to Security / Global Security.
2. Select *Authentication Mechanism* > *Authentication*.
3. Select *LTPA*.
4. Select *Trust Association*.
5. Enable the *enable trust association* check box.
6. Click *Apply* to save the changes.
7. Select *Interceptors*.
8. Remove both default TAI modules:

   `com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus`

   `com.ibm.ws.security.web.WebSealTrustAssociationInterceptor`
9. Click *New.*
10. Specify the following *Interceptor class name*:

    `com.novell.consulting.nl.accessmanager.tai.Roller`
11. Select *Apply*.
12. Select *Custom Properties*.
13. Select *New* to add the following name/value pairs:

    `user-name-header = X-Novell-TAI-UserName`

    `user-id-header = X-Novell-TAI-UID`

    `secret-header = X-Novell-TAI-ID`

    `secret-value = 23870790790732232` (Use whatever value you want)

    `cache-key-header = X-Novell-TAI-Cookie`

    `role-header = X-Novell-TAI-Roles`

    `role-separator = ;`

    `presentation-container =` (for example, ou=Groups,o=MP)

`update-connection = ldap://<ldapserver DNS name>:389`

`update-user =` Provide the DN of user in the same format that was created in eDirectory (for example, cn=wpstaibind,ou=Admins,ou=Services,o=MyOrg)

`update-password = <password of wpstaibind user>`

`update-container = <Container where groups are to be stored>` (for example, ou=Groups,o=MyOrg)

`update-roles = role1;role2;role3;role4;` (Roles should be separated by semicolons; do not embed white space in role names)

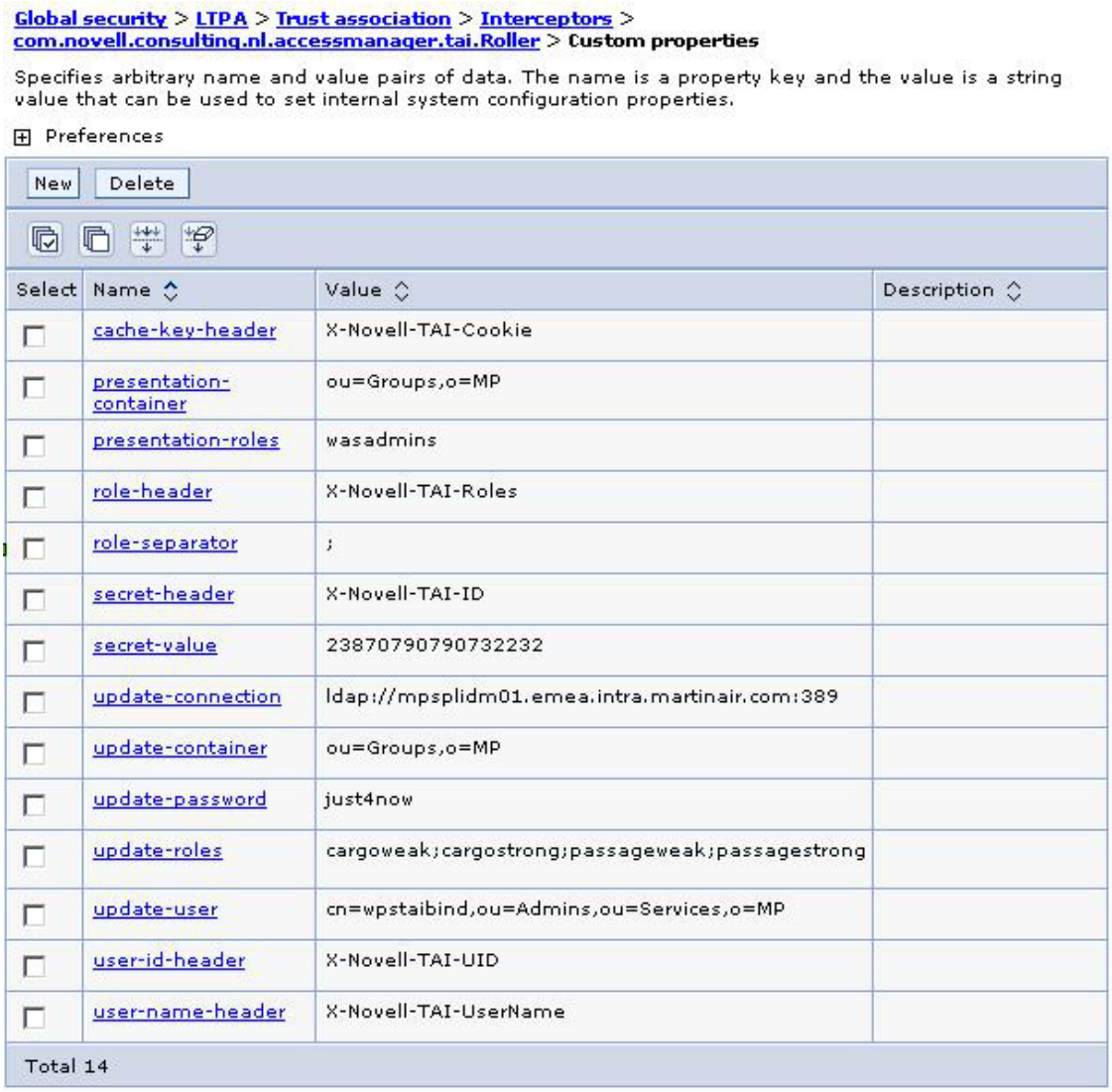
`presentation-roles = wasadmins`

`debug-level = info`

14. Save the changes.

WebSphere Portal Server and WebSphere Application Server need to be restarted before the TAI is enabled. Logging is placed in the `SystemOut.log` file.

Figure 4-1 *TAI Configuration*



Mapping WebSphere Application Server Roles to LDAP Groups

1. From the WebSphere Application Server select *System Administration* > *Console Settings* > *Console Groups*.
2. Click *Add* and add the wasadmins group.
3. Assign the role of Administrator to this group.

Editing Cache Settings

1. Edit the `/usr/WebSphere/PortalServer/wmm/wmm.xml` file and change the following:
   - `cacheAttributes="true"`
   - `attributesCacheSize="2000"`
   - `attributesCacheTimeOut="10"`

- `cacheNames="true"`
- `namesCacheTimeOut="10”`

2. Stop WebSphere Portal Server and WebSphere Application Server.
3. Issue a `/usr/WebSphere/PortalServer/config/WPSconfig.sh update-properties` command to let WebSphere reread the configuration.
4. Restart WebSphere Portal Server and WebSphere Application Server.

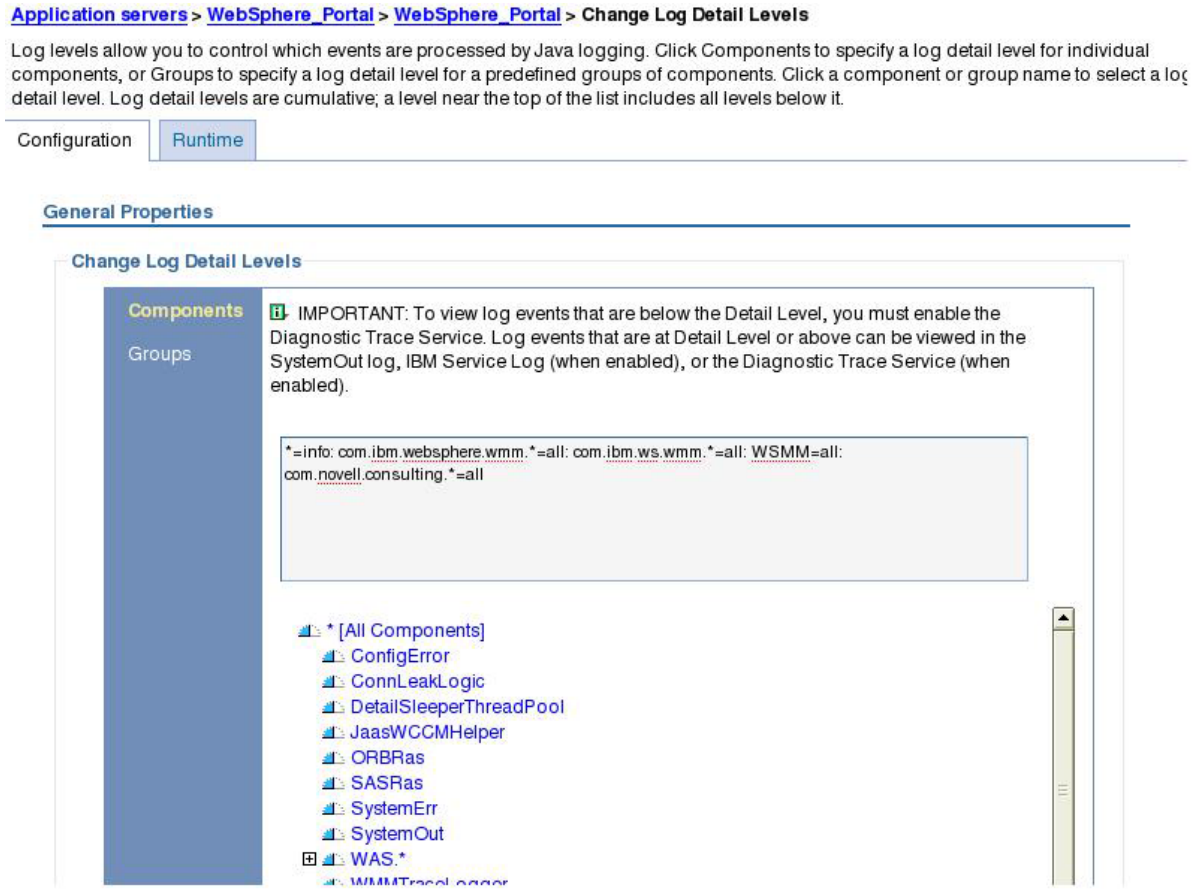
Enabling Logging Levels

1. From the WebSphere server, select *Application Servers* > *WebSphere_Portal* > *WebSphere_Portal* > *Change log level* details.
2. Select *com.novell.consulting*.
3. Set the appropriate log level and save changes.

NOTE: If *com.novell.consulting* is not available, you probably have not yet restarted WebSphere Portal Server. The TAI module is not yet initialized.

All logging is found in the `/usr/WebSphere/PortalServer/log/SystemOut.log` file.

Figure 4-2 *Example Log Configuration*



Configuring Access Manager

Access Manager passes all required details to the TAI module via the HTTP header.There are three places where configuration is required:

- Additions to the Roles policy.
- Creating an Identity Injection policy for protected WebSphere Portal Server application resources.
- Assigning the Identity Injection policy to the WebSphere Portal Server application resources.

- "Configuring the Roles Policy" on page 78
- "Configuring the Identity Injection Policy for WebSphere Portal Server Application Resources" on page 79

Configuring the Roles Policy

Because WebSphere Portal Server requires multiple roles, a separate Roles policy named WPS_roles can be created.This policy needs to be assigned to the IDP clusters. Add the following information to the WPS_roles policy.

Pair / Value:		
Role Description:	Role1	
Priority:	1	
If		
Condition Group 1		
Authentication Contract	[Current]	
Comparison:	String	Equals
Mode:	Case Sensitive	
Value:	Authentication Contract	**Int_IDP** – Secure Name/Password-Form
Result on Condition Error:	False	
AND		
LDAP Group	Current	
Comparison:	LDAP Group	Is member of
Mode:	Case Sensitive	
Value:	Data Entry Field	cn=Role1,ou=groups,o=MyOrg
Result on Condition Error:	False	
Or		
Condition Group 2		
Authentication Contract	[Current]	
Comparison:	String	Equals
Mode:	Case Sensitive	
Value:	Authentication Contract	**Ext_IDP** – Secure Name/Password-Form
Result on Condition Error:	False	
AND		
LDAP Group	Current	
Comparison:	LDAP Group	Is member of
Mode:	Case Sensitive	
Value:	Data Entry Field	cn=Role1,ou=groups,o=MyOrg
Result on Condition Error:	False	
Activate role	Role1	

Create the policies you would require and associate them with the appropriate Authentication contracts.

Configuring the Identity Injection Policy for WebSphere Portal Server Application Resources

Add the following information to the WPS_roles policy, then use the Administration Console to assign the injection policy for the appropriate protected application in Access Manager.

Pair / Value:			
Policy Name:	II_TAI_Header		
Description:	X-Novell-TAI Headers		
Rule1:			
Description:			
Priority:	1		
DO			
Inject into Custom Header			
	Custom Header Name:	X-Novell-TAI-UserName	
	Value:	Credential Profile: LDAP Credentials: LDAP User Name	
	Multi-Value Separator:	;	
	DN Format:	LDAP	
AND			
Inject into Custom Header			
	Custom Header Name:	X-Novell-TAI-UID	
	Value:	Credential Profile: LDAP Credentials: LDAP User DN	
	Multi-Value Separator:	;	
	DN Format:	LDAP	
AND			
Inject into Custom Header			
	Custom Header Name:	X-Novell-TAI-ID	
	Value:	String Constant:	23870790790732232
	Multi-Value Separator:	;	
	DN Format:	LDAP	
AND			
Inject into Custom Header			
	Custom Header Name:	X-Novell-TAI-Cookie	
	Value:	Proxy Session Cookie	
	Multi-Value Separator:	;	
	DN Format:	LDAP	
AND			
Inject into Custom Header			
	Custom Header Name:	X-Novell-TAI-Roles	
	Value:	Roles for Current User	
	Multi-Value Separator:	;	
	DN Format:	LDAP	

4.4 Configuring Applications on the WebLogic Server

If the application is using RunAs roles in the `weblogic-ejb-jar.xml` file, the role needs to be mapped to a user in the WebLogic domain. To enable this configuration on the server, two elements need to be added to this file:

- `<run-as-principal-name>` element for the EJB that is configured to use RunAs roles
- `<security-role-assignment>` element for the role

Run-As-Principle-Name Element

The `<run-as-principal-name>` element resides inside the `<weblogic-enterprise-bean>` element for the EJB. The element tells the server to run the EJB as the specified user. The sample below uses weblogic as the username because this is the default name of the WebLogic admin user. The entry should look similar to the following:

```
<run-as-principal-name>weblogic</run-as-principal-name>
```

The value (`weblogic`) must be the name of a user that exists in the domain. When this user is mapped to the Manager role, all users with the Manager role can run the EJB. The `<weblogic-enterprise-bean>` section of the file should look similar to the following for the sample payroll application. These sample lines configure the EmployeeSessionEJB:

```
<weblogic-enterprise-bean>
        <ejb-name>EmployeeSessionEJB</ejb-name>
        <reference-descriptor>
            <ejb-local-reference-description>
                <ejb-ref-name>ejb/EmployeeEJB</ejb-ref-name>
                <jndi-name>ejb.EmployeeEJB</jndi-name>
            </ejb-local-reference-description>
        </reference-descriptor>
        <enable-call-by-reference>True</enable-call-by-reference>
        <run-as-principal-name>weblogic</run-as-principal-name>
        <jndi-name>ejb.EmployeeSessionEJB</jndi-name>
    </weblogic-enterprise-bean>
```

Security-Role-Assignment Element

The `<security-role-assignment>` element needs to be placed outside of the `<weblogic-enterprise-bean>` element, and it needs to map the Manager role to the weblogic user specified in the `<run-as-principal-name>` element. It should look similar to the following for the sample payroll application:

```
<security-role-assignment>
    <role-name>Manager</role-name>
    <principal-name>weblogic</principal-name>
</security-role-assignment>
```

5 Configuring the Basic Features of a J2EE Agent

This section describes how to configure a J2EE Agent for the following features:

For information about configuring a J2EE Agent for authentication and access control, see the following:

5.1 Enabling Tracing and Auditing of Events

You can use either a Novell Audit server or the J2EE server log files to record information about what is being processed by the J2EE Agent.

5.1.1 Tracing Events to Log Files

Tracing adds more information about events (such as logins, logouts, and policy enforcement) to the J2EE server log files.

To enable tracing:

1. In the Administration Console, click *Devices* > *J2EE Agents* > *Edit*.
2. Select the *Enable Tracing* option. The messages are sent to the following log files, depending upon the type of application server you are using:
   - **JBoss Server:** For a JBoss server, the log messages are logged to the `$JBOSS_HOME/log/jboss.log` file if you launched the JBoss server using the `run.sh` script found in the `bin` folder. Messages are also sent to the console, so you should check the console or the `$JBOSS_HOME/server/default/log/server.log` file.

- **WebSphere Server:** For a WebSphere server, the log messages are logged to files in the `$WAS_BaseDir/profiles/$ProfileName/logs` directory. Check the `SystemOut.log` and `SystemErr.log` files.
- **WebLogic Server:** For a WebLogic server, the log messages are sent to standard out.

3 Click *Apply Changes*.

4 To trace policy enforcement, you also need to enable and set the level of logging for the Embedded Service Provider.

5.1.2 Enabling the Auditing of Events

The Access Manager ships with a Novell Audit server that is installed when you install the first instance of the Administration Console. You can configure the J2EE Agent to send events to this audit server or to another Novell Audit server on your network. (To configure access to the Novell Audit server.

1 In the Administration Console, click *Devices* > *J2EE Agents* > *Edit*.

2 In the *Audit Configuration* section, select from the following events:

Event	Description
Startup, shutdown, and reconfigure	Generated when the agent is started or stopped and when the configuration of the agent is modified.
Successful authentications	Generated when someone successfully authenticates to the agent.
Allowed EJB access	Generated when someone is granted access to Enterprise JavaBeans.
Allowed web resource access	Generated when someone is granted access to a Web resource.
Allowed clear text access	Generated when a user is granted clear text access to a Web resource.
Denied clear text access	Generated when someone is denied clear text access to a Web resource.
Unsuccessful authentications	Generated when someone is unsuccessful in attempting to authenticate.
Denied EJB access	Generated when someone is denied access to Enterprise JavaBeans.
Denied web resource access	Generated when someone is denied access to a Web resource.

3 Click *OK*, then click *Update* > *OK*.

5.2 Managing Embedded Service Provider Certificates

You can view and modify the private keys, certificate authority (CA) certificates, and certificate containers associated with the Embedded Service Provider. The Embedded Service Provider module is the J2EE Agent module that communicates with the Identity Server. This module handles all the authentication requests that need to be forwarded to the Identity Server for verification.

1. In the Administration Console, click *Devices* > *J2EE Agents* > *Edit*.
2. To view the assigned certificates, click one of the following keystores in the *Service Provider Certificates* section:

   Signing: The signing certificate keystore. Click this link to access the keystore and replace the signing certificate as necessary. The signing certificate is used to sign the assertion or specific parts of the assertion.

   Mutual SSL: The mutual SSL connector keystore. Click this link to access the keystore and replace the certificate. This certificate is used for mutual SSL connections with the Identity Server. If you set up services on the Identity Server that require mutual SSL, the Identity Server uses this certificate to established the mutual SSL connection.

   The Web Services Framework allows each service (such as a personal profile or employee profile) defined on the Identity Server to specify various security mechanisms that are a combination of transport-level and messages-level security as depicted in Liberty ID-WSF specification. This can be selected by the administrator, depending upon the nature of data and optimizations. If a service on the Identity Server specifies that any Web service consumer (which includes the Embedded Service Provider) must authenticate itself using a client certificate, the Web service consumer needs to support mutual SSL. For information on how to set up a profile to require mutual SSL,

   The Access Manager automatically populates this keystore with the certificate that you select when enabling SSL between the agent and the Identity Server. If you replace this certificate, you need to replace it with a certificate whose subject name (cn) matches the DNS name of the agent.

   Trusted Roots: The trusted root certificate container for CA certificates associated with the agent. Click this link to access the trust store, where you can change the password or add trusted roots to the container.

   The Embedded Service Provider must trust the certificate of the Identity Server that the agent has been configured to trust. The public certificate of the CA that generated the Identity Server certificate must be in this trust store. If you configured the Identity Server to use a certificate generated by a CA other than the Access Manager CA, you must add the public certificate of this CA to the Trusted Roots store.
3. Click *OK*, then click *Update* > *OK*.

5.3 Configuring SSL Certificate Trust

The Identity Server must be configured to trust the CA that created the SSL key pair certificate of your application server. The public key of this CA needs to be added to the NIDP Trust Store of the Identity Server. For instructions, select the NIDP Trust Store, and specify the IP address and port of your application server.

The Embedded Service Provider of the agent, which the agent uses for communication with the Identity Server, must be configured to trust the CA that generated the certificate for the Identity Server. If you configured the Identity Server to use a certificate generated by a CA other than the Access Manager CA, you must add the public certificate of this CA to the trusted roots store of the Embedded Service Provider. See Section 5.2, “Managing Embedded Service Provider Certificates,” on page 85.

5.4 Modifying the Display Name and Other Details

1. In the Administration Console, click *Devices* > *J2EE Agents* > *[Name of Agent]* > *Edit*.
2. (Optional) Modify the following fields:

   Name: Specifies the console display name for the agent. The default name is a randomly generated unique number. You should probably modify this name to one that you can pronounce. You cannot leave this field blank.

   The name must use alphanumeric characters and can include spaces, hyphens, and underscores.

   Location: (Optional) Specifies the physical location of this J2EE Agent.

   Description: (Optional) Describes the purpose of this agent. This is a useful field if your network has multiple J2EE Agents.
3. To save your changes, click *OK*.

To change the Management IP Address, see Section 5.5, "Changing the IP Address of a J2EE Agent," on page 86.

5.5 Changing the IP Address of a J2EE Agent

If you configure your J2EE server to use a different IP address after you have installed a J2EE Agent, the communication channel between the Administration Console and the J2EE Agent breaks. The Administration Console needs to be updated to use the new IP address for communication.

IMPORTANT: The agent must be informed of the pending change in the IP address before you actually change the address on the J2EE server. If you change the address on the J2EE server before configuring the change in the Administration Console, you must uninstall the agent and reinstall it to establish communication with the Administration Console.

---

1. In the Administration Console, click *Devices* > *J2EE Agents* > *[Name of Agent]* > *Edit*.
2. In the *Management IP Address* option, specify the IP address of the J2EE server. If you have changed the IP address of the J2EE server, specify the new address here.
3. To save your changes, click *OK*.
4. To verify your settings for the *J2EE Application Server URL* option, click *J2EE Agents* > *Edit*.

   If you used a DNS name for the *J2EE Application Server URL*, make sure your DNS server has been updated to resolve the DNS name to the new IP address.

6 Protecting Web and Enterprise JavaBeans Modules

The J2EE Agent mechanisms for protecting Web and EJB (Enterprise JavaBeans) modules have far more granularity than what you can configure on the J2EE application server. With the agent, you can be selective of what you are protecting. For a Web application, you can select to protect a specific page or group of pages. For an Enterprise JavaBean, you can select to protect a bean, an interface, a method, or a parameter. After selecting the granularity of the resource you want to protect, you can then configure a policy that grants access to this resource. You can use roles as part of this policy, but you can refine it by using other criteria such as LDAP attributes, credential profile attributes, or the day of the week.

The J2EE Agent also allows you to decide how you want the authorization to be handled. You can use the security settings configured on the application server, use the Authorization policies configured on the J2EE Agent, or use both methods.

The following sections explain how to set up security for your J2EE resources:

6.1 Configuring Access Control

The access control configuration determines which Authorization policies are used to allow access to resources. The application server must be configured to allow the J2EE Agent to enforce authorization:

- Section 4.2, "Configuring Applications on the JBoss Server," on page 67
- Section 4.3, "Configuring Applications on the WebSphere Server," on page 69
- Section 4.4, "Configuring Applications on the WebLogic Server," on page 81

After you have configured the J2EE server for authorization, you need to configure the J2EE Agent for access control:

1. In the Administration Console, click *Devices* > *J2EE Agents* > *Edit*.
2. In the *Access Control Configuration* section, select one or more of the following:

   Enforce application server policy: Allows access based on the policy of the application server. These policies are defined on the application server in a `web.xml` file for a `.war` file and in a `ejb-jar.xml` file for a `.jar` file.

   IMPORTANT: If you select this option and you are using a JBoss server, see Section 4.2.2, "Configuring Security Constraints," on page 67 for additional information.

Enforce additional authorization policies: Allows access based on the policies assigned to the protected resources. If you do not configure any protected resources, users are denied access to all resources. If a resource does not match any of the protected resource configurations, all users are denied access to that resource.

You can enable both of these options, only one, or none. If you select neither, any user can access the resources on the application server.

If you select only the J2EE Agent policies for authorization and you disable the *Enforce application server policy* option, remember that authentication is triggered by the Web page for a `.jar` file and by the `web.xml` file for a `.war` file.

IMPORTANT: Do not disable *Enforce application server policy* until you have configured and tested the J2EE Agent policies and know that they are enforcing the security you require and that users have access to the resources they require.

---

3 If you decided to use just the application server policies, click *OK*, then click *Update* > *OK*.

or

If you enabled *Enforce additional authorization policies*, click *Define authorization policies* and continue with one of the following:

- Section 6.2, "Protecting Web Resources," on page 88
- Section 6.3, "Protecting Enterprise JavaBeans Resources," on page 90

6.2 Protecting Web Resources

Because you can define multiple protected resources for each Web application, you can protect some URLs with one policy and other URLs with a different policy.For example, you might have some pages in the application that you want all employees to access, and some pages that you want only managers to access. For this application, you would create two protected resources, one for all employees and one for managers. You would then assign a policy to each protected resource. The following sections explain this process:

- Section 6.2.1, "Creating a Protected Resource for a Web Application," on page 88
- Section 6.2.2, "Assigning a Web Authorization Policy to the Resource," on page 90

6.2.1 Creating a Protected Resource for a Web Application

1 In the Administration Console, click *Devices* > *J2EE Agents* > *Edit* > *Manage authorization policies*.

2 Click *New* and supply the following information:

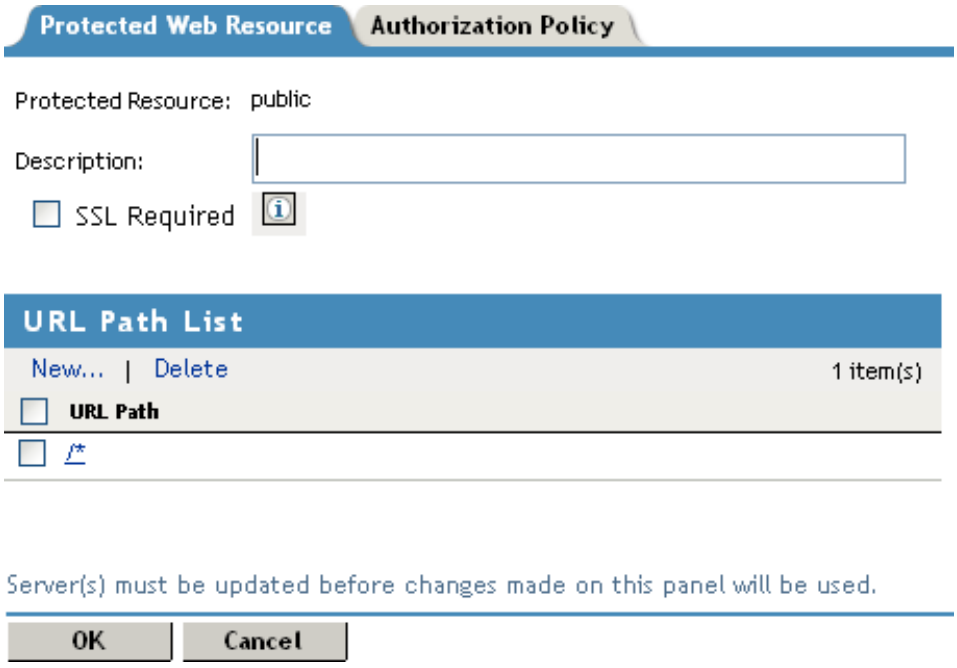
Module File Name: The filename of the application. Specify the name of the file you are protecting, including the file extension (`.war` for a Web application).

Type: The type of the application. Select *Web Module* for a Web application.

3 Click *OK*.

4 To add a protected resource to the list, click *New*, specify a display name for the resource, then click *OK*.

If possible, this name should indicate the URLs that you are going to configure for this resource.



5 Fill in the following fields:

Description: (Optional). A text box where you can specify a description of the protected resource. You can also use the field to briefly describe the purpose of protecting this resource.

SSL Required: If this option is selected, the J2EE Agent sets up an SSL connection between the client and the application.

IMPORTANT: If the Web pages that you are now protecting with SSL have been publicly available over HTTP, they remain publicly available over HTTP until you either restart the Web server or reinstall the application. If this is a new application, reinstalling the application might be less disruptive to your network environment than restarting the Web server.

For the JBoss Agent, selecting the *SSL Required* option is only part of the process. On JBoss, you must also either disable the HTTP port and enable the SSL port or configure SSL in the `web.xml` file.

---

6 In the *URL Path List*, configure the paths that this resource protects. To add a path, click *New*, specify the path, then click *OK*.

For example, to allow access to all the pages in the `public` directory on the Web server, specify the following path:

```
/public/*
```

To allow access to everything on the Web server, specify the following path:

```
/*
```

To use this protected resource to protect a single page, specify the path and the filename. For example, to protect the login.html page in the `/login` directory, specify the following:

```
/login/login.html
```

7 Click *Configuration Panel* > *OK*.

8 On the Configuration page, click *OK*, then click *Update* > *OK*.

9 Continue with Section 6.2.2, "Assigning a Web Authorization Policy to the Resource," on page 90.

Until you have assigned an Authorization policy to the resource, which restricts access to this resource, all authenticated users have access to the resource.

6.2.2 Assigning a Web Authorization Policy to the Resource

The following instructions assume that you have already created your Authorization policy for the Web resource. For general information about Authorization policies, and for information about creating a Web Authorization policy.

To assign an Authorization policy:

1 In the Administration Console, click *Devices* > *J2EE Agents* > *Edit* > *Manage authorization policies* > *[Name of Web Module]* > *[Name of Protected Resource]* > *Authorization Policy*.

2 To enable a policy, select a policy in the list, then click *Enable*.

If no policies appear in the list, you haven't created any. Click *Manage Policies*. For configuration information.

3 Click *Configuration Panel* > *OK*.

4 On the Configuration page, click *OK*, then click *Update* > *OK*.

6.3 Protecting Enterprise JavaBeans Resources

Because you can define multiple protected resources for each JavaBean, you can create one policy that protects the module and another policy that protects specific interfaces or methods. For example, you can create two protected resources and two policies for an EJB. The first resource and policy combination grants general access to the EJB to all the users that meet the criteria in the Authorization policy. If the EJB contains areas that only a few users should access, then you create a second protected resource and policy combination that restricts access to these resources to these users. The following sections explain this process:

- Section 6.3.1, "Creating a Protected Enterprise JavaBean Resource," on page 90
- Section 6.3.2, "Assigning an Enterprise JavaBeans Authorization Policy to a Resource," on page 92

6.3.1 Creating a Protected Enterprise JavaBean Resource

1 In the Administration Console, click *Devices* > *J2EE Agents* > *Edit* > *Manage authorization policies*.

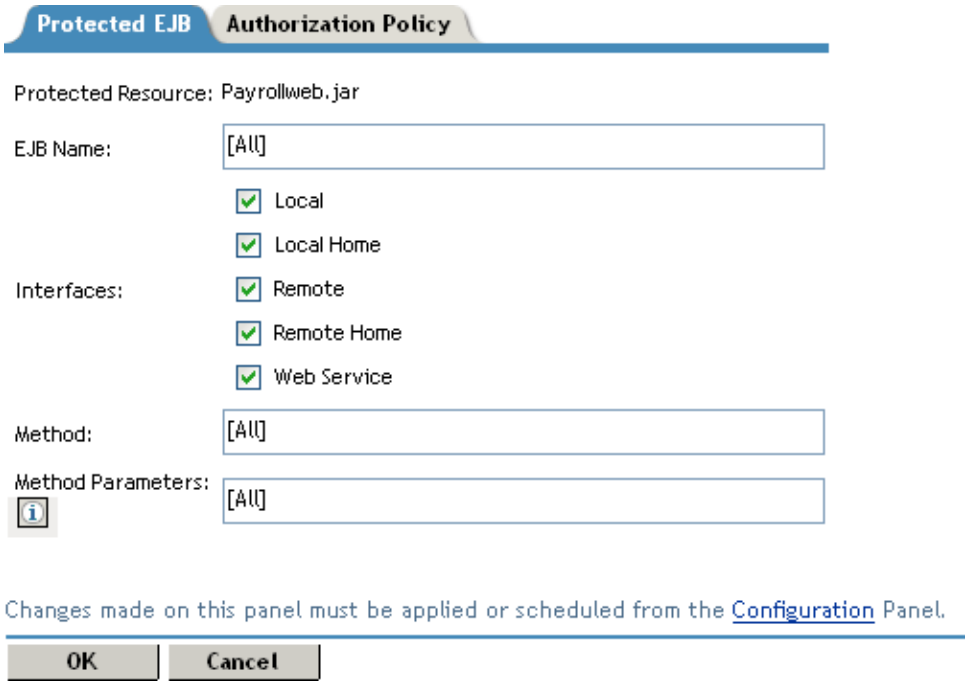
2 Click *New* and supply the following information:

Module File Name: The filename of the EJB. Specify the name of the EJB module you are protecting, including the file extension (`.jar` for an EJB Module).

Type: The type of the application. Select *EJB Module* for an EJB module.

3 Click *OK*.

4 To add a protected resource to the list, click *New*, specify a display name for the EJB resource, then click *OK*.



5 Fill in the following fields:

EJB Name: The module name to protect. Select *[All]* to protect all modules.

Interfaces: The interfaces to protect. Select one or more of the following:

- Local
- Local Home
- Remote
- Remote Home
- Web Service

Method: The method to protect. Select *[All]* to protect all methods.

Method Parameters: The parameters of the method to protect.

- If *[All]* is specified, the policy is applied to all methods listed in the *Method* field.
- If the list is empty, the policy is applied only to the methods that have an empty set of parameters.
- If the field contains parameter names, the policy is applied only to the methods that have the specified parameters.

6 Click *Configuration Panel* > *OK*.

7 On the Configuration page, click *OK*, then click *Update* > *OK*.

8 Continue with Section 6.3.2, "Assigning an Enterprise JavaBeans Authorization Policy to a Resource," on page 92.

Until you have assigned an Authorization policy to the resource to restrict access to this resource, all authenticated users have access to the resource.

6.3.2 Assigning an Enterprise JavaBeans Authorization Policy to a Resource

The following instructions assume that you have already created your Authorization policy for the Web resource. For general information about Authorization policies, and for information about creating an EJB Authorization policy.

1. In the Administration Console, click *Devices* > *J2EE Agents* > *Edit* > *Manage authorization policies* > *[Name of EJB Module]* > *[Name of EJB]* > *Authorization Policy*.
2. To enable a policy, select a policy in the list, then click *Enable*.

   If no policies appear in the list, you haven't created any. Click *Manage Policies*.

   WARNING: EJBs that are configured to run as a role can only use limited conditions in an EJB Authorization policy. The Current Roles of User and the time conditions can be used in the policy, but the conditions requiring user information cannot be used. This is because the RunAs role subjects do not contain the Liberty profile, LDAP attribute, or LDAP credential information that these conditions require. When unsupported conditions are defined in a policy and that policy is assigned to a RunAs role EJB, the user is denied access to the EJB resource.

   ---
3. Click *Configuration Panel* > *OK*.
4. On the Configuration page, click *OK*, then click *Update* > *OK*.

7 Deploying the Sample Payroll Application

The sample payroll application has been configured to grant access based on whether the user has an Employee role or a Manager role. You can configure your J2EE Agent to use the authorization policies of the J2EE server or to use the policies of Access Manager.

During installation, the sample payroll application is copied to the following location:

- On a Linux J2EE server, this application is copied to the `/opt/novell/nids_agents/examples` directory.
- On a Windows J2EE server, this application is copied to the `<Install_Directory>\sampleapp` directory.
- On a Solaris J2EE server, this application is copied to the `/opt/novell/nids_agents/examples` directory.
- On an AIX J2EE server, this application is copied to the `/opt/novell/nids_agents/examples` directory.

This section has the following information:

7.1 Deploying the Sample Payroll Application

This example deploys the sample application to WebSphere.

1. In the WebSphere Administration console, click *Applications* > *Enterprise Applications*.
2. Click *Install*.
3. Select *Remote File System*.
4. Browse and select the payroll application `PayrollApp.ear` from one of the following locations:
   - `/opt/novell/nids_agents/examples` directory on Linux.
   - `<Install_Directory>\sampleapp` directory on Windows.
5. Click *Next*.
6. Accept the default settings, then click *Finish*.
7. To start the Payroll application, click *Start*.
8. (Optional) Restart the WebSphere server if there any problems.

For more information on testing the configuration, see Section 7.4.4, "Testing the Configuration," on page 104.

7.2 Preparing the Sample Application for the Agent

Each Web application that you want to use with the J2EE Agent must be able to log in and log out of the Identity Server that you have configured the J2EE Agent to trust. You do this by configuring the `web.xml` file of the application.

The following sections describe the procedure to configure the `web.xml` file of the sample application (`PayrollApp.ear`):

- Section 7.2.1, "Configuring for Login," on page 94
- Section 7.2.2, "Configuring for Logout," on page 94

7.2.1 Configuring for Login

In order to configure the login, you must specify in the `web.xml` file that the Web application uses FORM authentication. This is specified in the `<login-config>` section of the application descriptor in the `WEB-INF/web.xml` file as follows:

```
<login-config>
    <auth-method>FORM</auth-method>
    <form-login-config>
        <form-login-page>/login</form-login-page>
        <form-error-page>/login</form-error-page>
    </form-login-config>
</login-config>
```

The `<form-login-page>` and `<form-error-page>` elements need to be set to a URL that is mapped to the following servlet class:

```
com.novell.nids.agent.auth.LoginServlet
```

The `<login-config>` element in the example above specifies `/login` as the login page and the error page. The `/login` URL needs a servlet mapping within the application's `web.xml` file:

```
<servlet>
    <servlet-name>LoginServlet</servlet-name>
       <servlet-class>
          com.novell.nids.agent.auth.LoginServlet
       </servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/login</url-pattern>
</servlet-mapping>
```

7.2.2 Configuring for Logout

To add a logout servlet and its servlet mapping to the `web.xml` file, modify the contents of `web.xml` as follows:

```
    <servlet>
        <servlet-name>LogoutServlet</servlet-name>
        <servlet-class>
            com.novell.nids.agent.auth.LogoutServlet
        </servlet-class>
          <init-param>
            <param-name>postLogoutURL</param-name>
            <param-value>/loggedOut</param-value>
         </init-param>
         <init-param>
             <param-name>websphereLTPAMechanism</param-name>
             <param-value>false</param-value>
        <description>
ThisshouldbesettotrueinordertoclearLTAPcookiesandtokensin
case of websphere with LTPA as authentication mechanism
        </description>
             </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>LogoutServlet</servlet-name>
        <url-pattern>/logout</url-pattern>
    </servlet-mapping>
```

7.3 Using the J2EE Server to Enforce Authorization

The following procedure explains how you can configure Access Manager to use the authorization policies of the J2EE server:

1. Deploy the sample payroll application on your J2EE server.
2. On your J2EE server, prepare the application to use the agent for login and logout. See Section 4.1, "Preparing the Application for the Agent," on page 65.

   These steps have already been performed for the sample application. See the `web.xml` file in the application's `WEB-INF` directory.
3. Complete any platform-specific configuration:
   - **JBoss:** These tasks have already been performed for JBoss. To understand what was modified, see Section 4.2, "Configuring Applications on the JBoss Server," on page 67.
   - **WebSphere:** You need to configure the RunAs Roles feature. See Section 4.3.2, "Configuring Security Role to User/Group Mapping," on page 69 and Section 4.3.3, "Configuring for User RunAs Roles," on page 70.
   - **WebLogic:** You need to configure the RunAs Roles feature. See Section 4.4, "Configuring Applications on the WebLogic Server," on page 81.
4. In Access Manager, create role policies for an Employee role and a Manager role.

   For more information, see "Creating Role Policies" in the *NetIQ Access Manager 3.2 Policy Guide*.
5. Configure the agent for authentication. For more information, see Chapter 2, "Configuring the Agent for Authentication," on page 43.
6. Make sure that the *Enforce application server policy* option is selected. In the Administration Console, click *Devices* > *J2EE Agents* > *Edit*.
7. To test this configuration, send the following request from a browser:

   `http://<Application_Server_DNS_Name>:<port>/payroll`

   Replace `<Application_Server_DNS_Name>` with the DNS name or the IP address of your application server.

   Replace `<port>` with the port number you have configured the J2EE Agent to use.

8 Log in as a user who matches the condition to receive the Employee role and access the *My Page* and the *Manager Page*.

9 Log out and log in as a user who matches the condition to receive the Manager role. Access the *My Page* and the *Manager Page*.

As a manager, you can add Employee Records so that when employees log in, their records are displayed on *My Page*.

7.4 Using Access Manager Policies to Enforce Authorization

The following procedure explains how to set up Access Manager policies that permit Managers to access the manager pages in the sample payroll application, deny Employees access to the manager pages, but permit Employees and Managers access to their own information pages. These policies do not require any J2EE server configuration to correctly enforce the policies.

7.4.1 Creating an Employee Role and a Manager Role

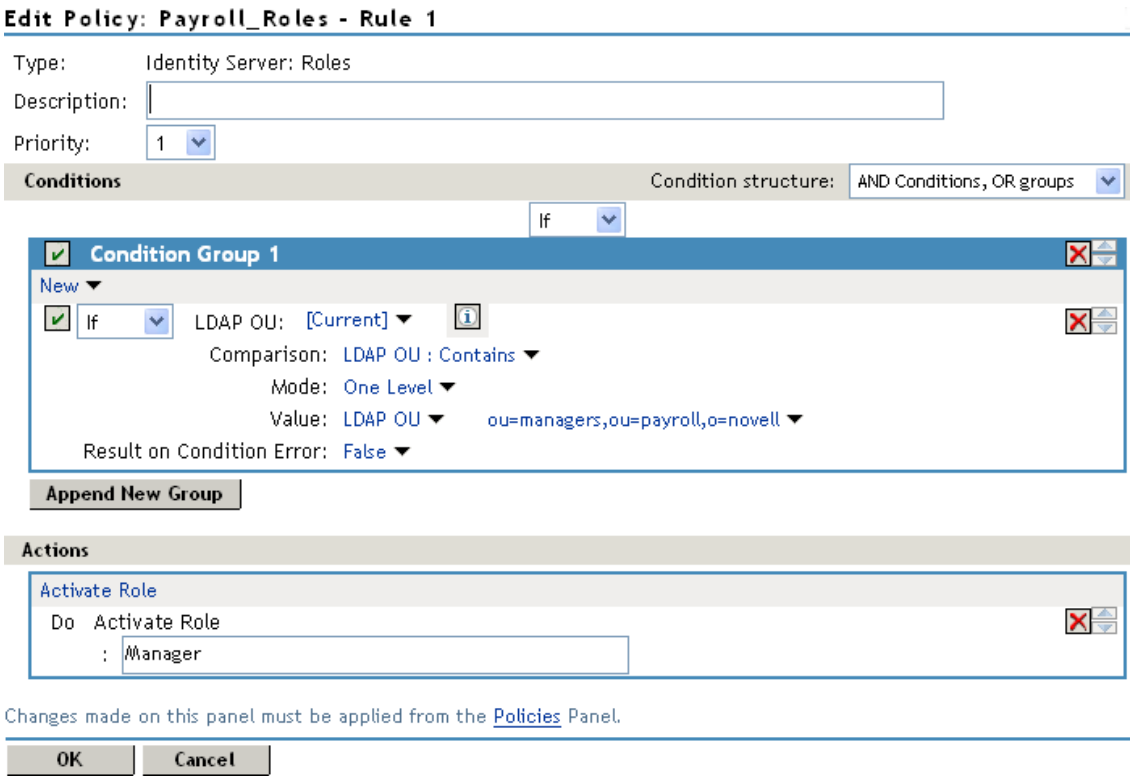
If you have a particular application that requires more than one role, and it is the only application using these roles, you can create one role policy that assigns users to the required roles. The following steps explain how to create one role policy that assigns users to the Manager role and the Employee role.

1 In the Administration Console, click *Devices* > *Policies*.

2 Click *New*, specify a name for the role policy, select *Identity Server: Roles* as the type, then click *OK*.

3 For the first rule, click *New*, create a condition that matches your managers but not your employees, activate the Manager role, then click *OK*.

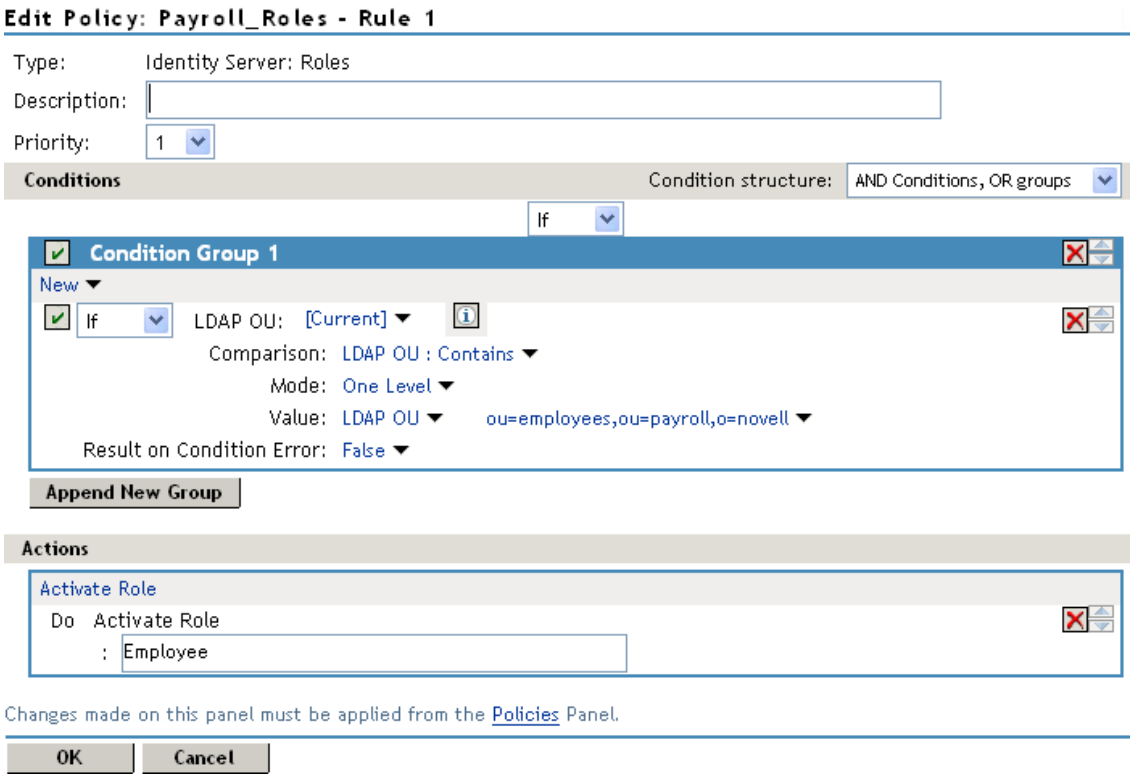
The following rule uses the LDAP OU condition to determine whether the user is a manager. It assumes that all managers are in the ou=managers,ou=payroll,o=novell container.



4 To create the second rule of the policy, click *New*.

5 In Condition Group 1, click *New*, create a condition that matches your employees but not your managers, activate the Employee role, then click *OK*.

The following rule uses the LDAP OU condition to determine whether the user is an employee. It assumes that all employees are in the ou=employees,ou=payroll,o=novell container.



6 To save your Role policy, click *OK* > *Apply Changes*.

7 Activate the Role policy for your Identity Server cluster configuration. Click *Identity Servers* > *Edit* > *Roles*.

8 Select the name of your Role policy, click *Enable*, then click *OK*.

9 Click *Identity Servers* > *Update* to update the Identity Server.

10 Continue with Section 7.4.2, "Creating Authorization Policies," on page 98.

7.4.2 Creating Authorization Policies

The payroll application is a `.ear` file that contains both an EJB module and a Web (`.war`) module. Each module type requires its own type of Authorization policies, and to fully protect the application, you must create the following policies:

- "Creating EJB Authorization Policies" on page 98
- "Creating Web Authorization Policies" on page 101

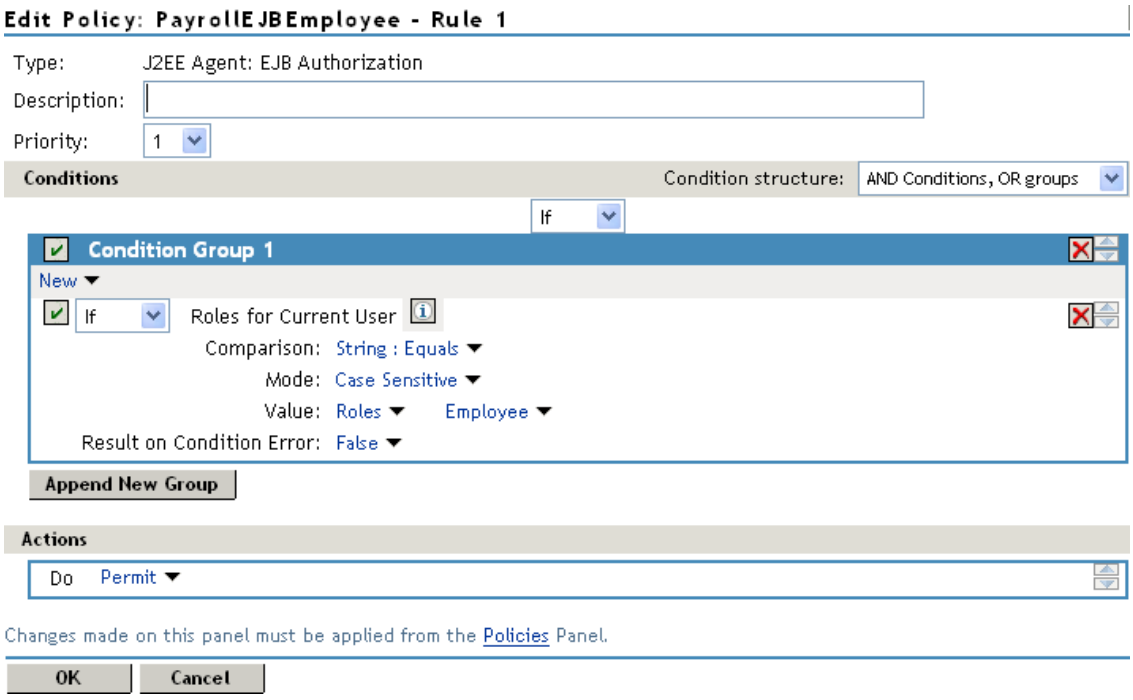
Creating EJB Authorization Policies

You need to create two policies: one that permits Managers to access EJB resources and one that permits Employees to access EJB resources.

1 In the Administration Console, click *Devices* > *Policies*.

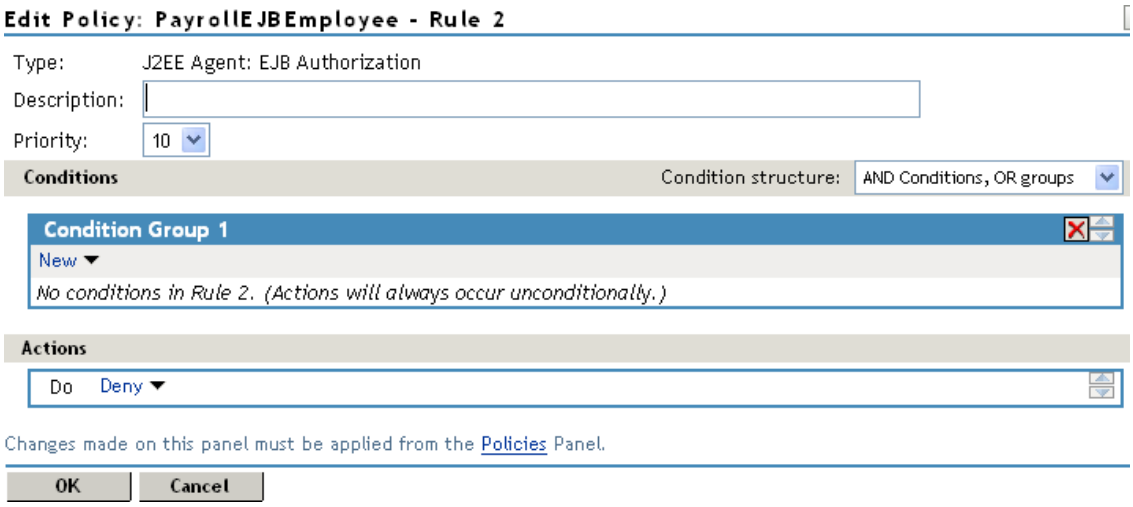
2 To create an Authorization policy for the employees, click *New*, specify a name for the policy, select *J2EE Agent: EJB Authorization* as the type, then click *OK*.

3 For the first rule, click *New*, set up a condition that permits access if the user has been assigned the Employee role, then click *OK*. Your rule should look similar to the following:



4 To create the second rule in the policy, click *New*.

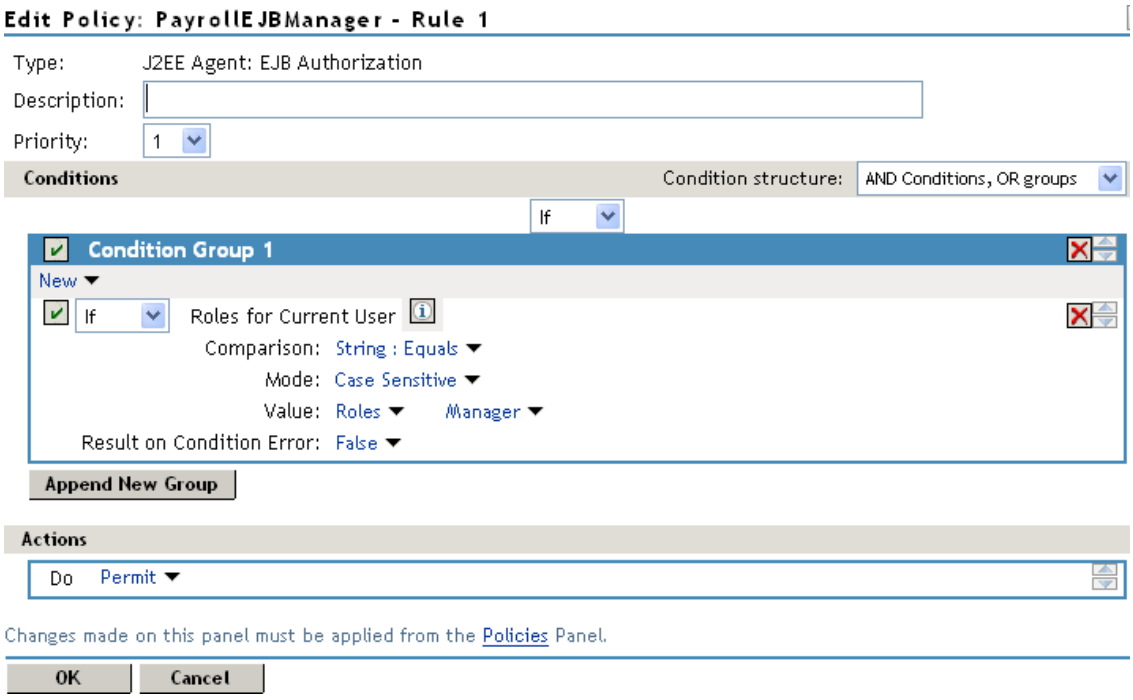
5 To create a generic deny rule, assign a deny action, then click *OK*. Your rule should look similar to the following:



6 To save your employee policy, click *OK* > *Apply Changes*.

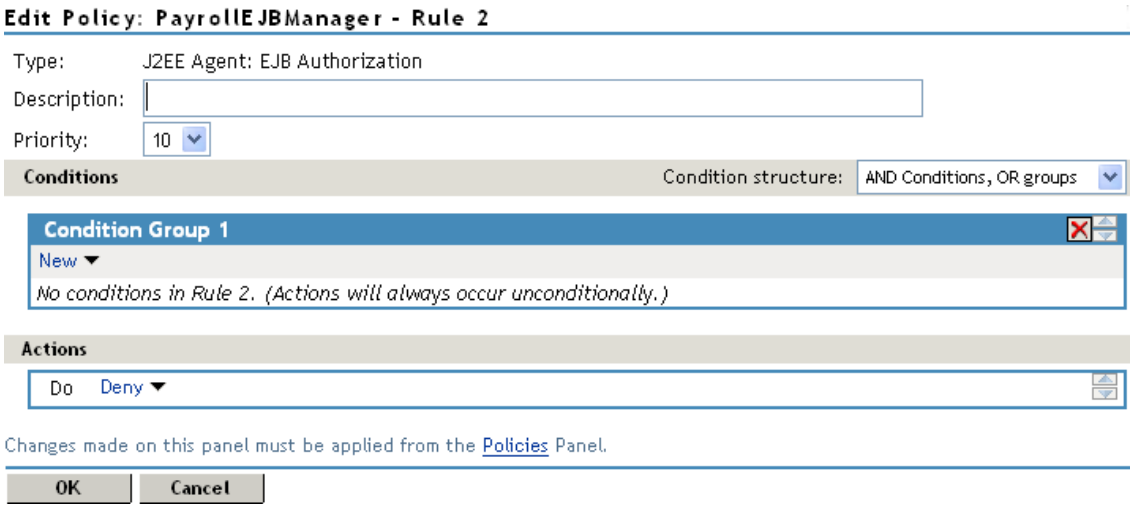
7 To create a policy for the managers, click *New*, specify a name for the policy, select *J2EE Agent: EJB Authorization* as the type, then click *OK*.

8 For the first rule, click *New*, set up a condition that permits access if the user has been assigned the Manager role, then click *OK*. Your rule should look similar to the following:



9 To create the second rule in the policy, click *New*.

10 To create a generic deny rule, assign a deny action, then click *OK*. Your rule should look similar to the following:



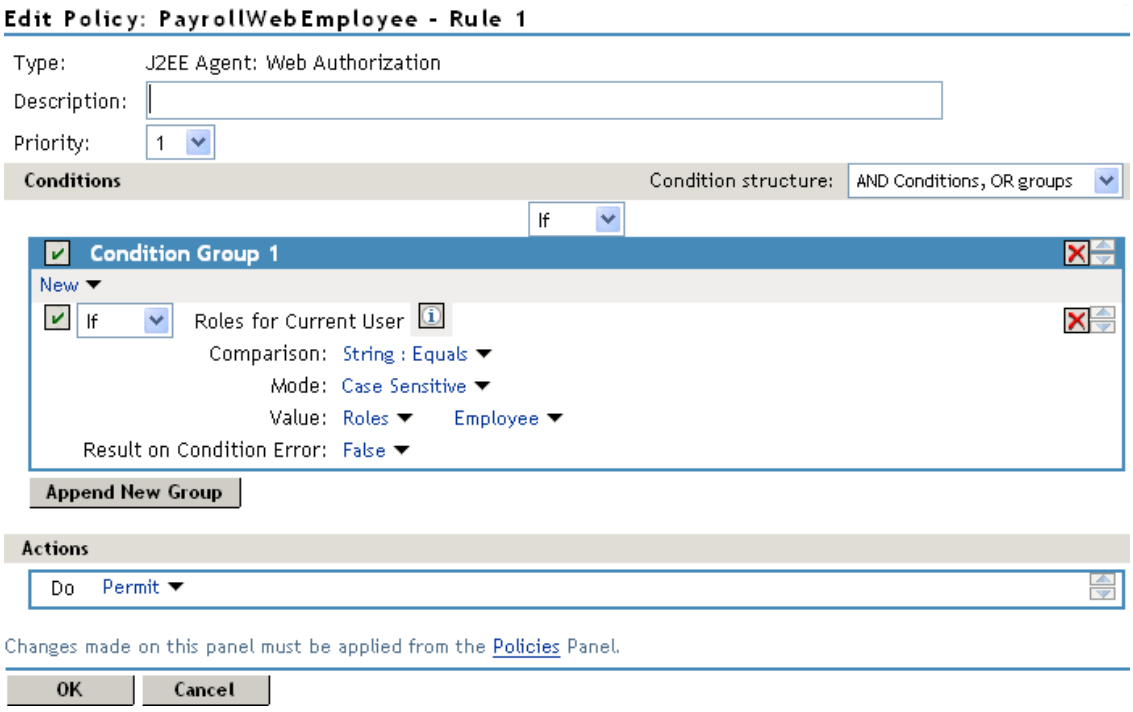
11 To save your manager policy, click *OK* > *Apply Changes*.

12 Continue with "Creating Web Authorization Policies" on page 101.

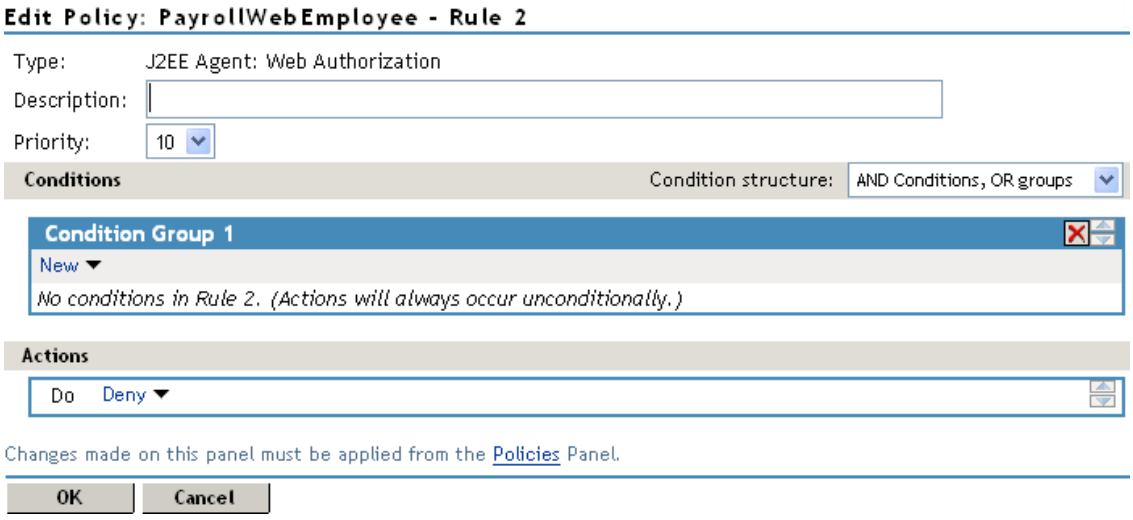
Creating Web Authorization Policies

You need to create two policies: one that permits Managers to access resources and one that permits Employees to access resources.

1. In the Administration Console, click *Devices* > *Policies*.
2. To create an Authorization policy for the employees, click *New*, specify a name for the policy, select *J2EE Agent: Web Authorization* as the type, then click *OK*.
3. For the first rule, click *New*, set up a condition that permits access if the user has been assigned the Employee role, then click *OK*. Your rule should look similar to the following:



4. To create the second rule in the policy, click *New*.
5. To create a generic deny rule, assign a deny action, then click OK. Your rule should look similar to the following:

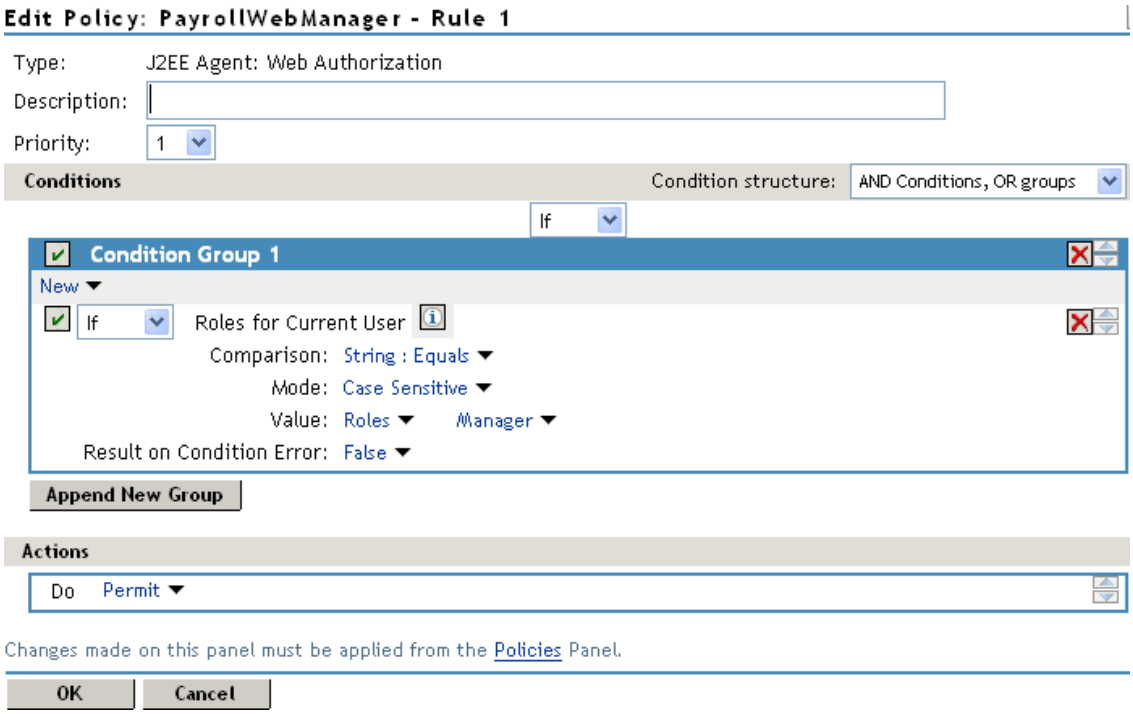


When you create a policy with one or more permit rules and you end it with a deny rule with a priority of 10, the logic of the policy is clear. Users who match a permit rule are allowed access; everyone else is denied access.

6 To save your employee policy, click *OK* > *Apply Changes*.

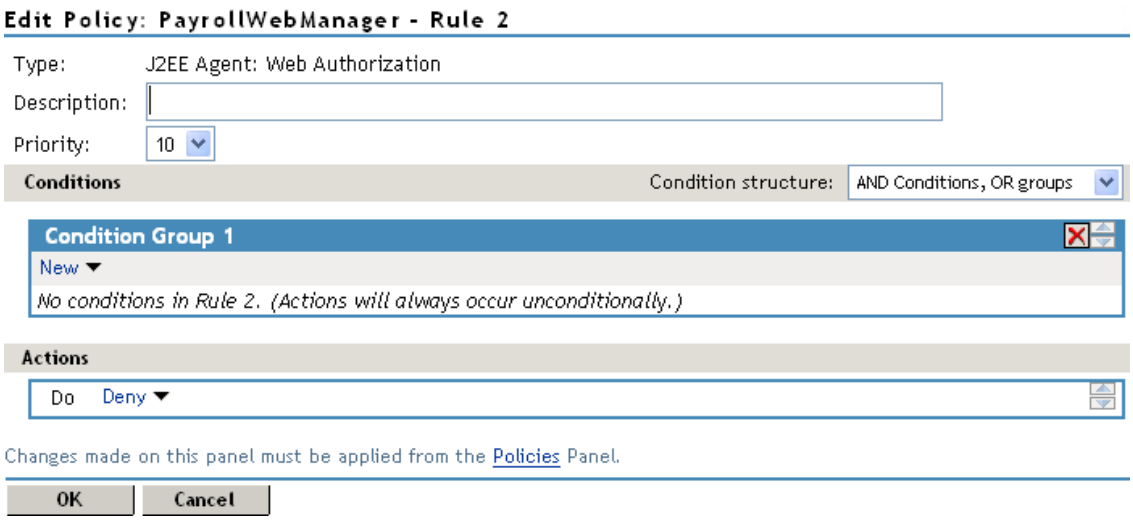
7 To create a policy for the managers, click *New*, specify a name for the policy, select *J2EE Agent: Web Authorization* as the type, then click *OK*.

8 For the first rule, click *New*, set up a condition that permits access if the user has been assigned the Manager role, then click *OK*. Your rule should look similar to the following:



9 To create the second rule in the policy, click *New*.

10 To create a generic deny rule, assign a deny action, then click *OK*. Your rule should look similar to the following:



11 To save your manager policy, click *OK* > *Apply Changes*.

12 Continue with Section 7.4.3, "Assigning Policies to Protected Resources," on page 103

7.4.3 Assigning Policies to Protected Resources

After creating the Authorization policies, you need to create protected resources for the payroll application, then assign the policies to the protected resources.

Assigning the Authorization Policies to Protected Web Resources

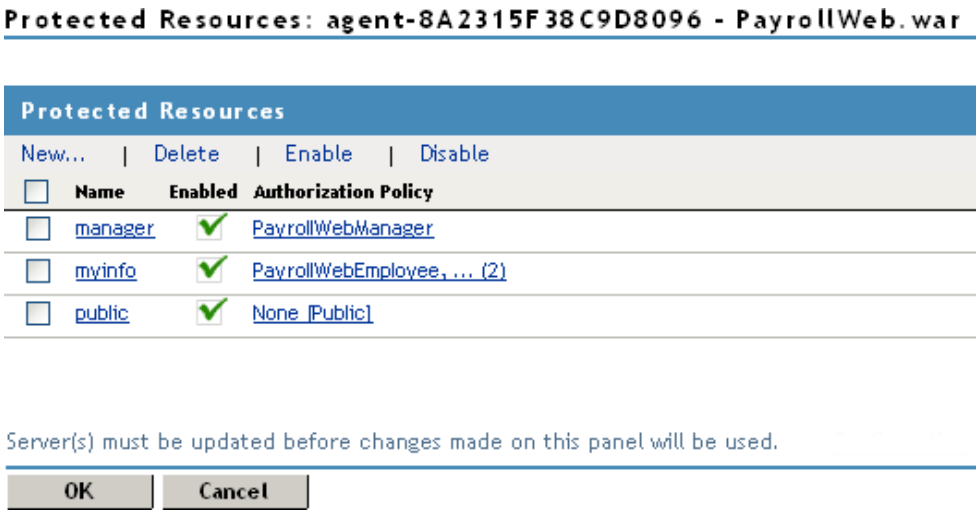
To allow the J2EE Agent to enforce authorization for the payroll Web module, you need to create three protected resources for the payroll application.

1 Click *Devices* > *J2EE Agents* > *Edit*.

2 In the Access Control Configuration section, deselect *Enforce application server policy*, select *Enforce additional authorization policy*, then click *Manage authorization policies*.

3 Click *New*, specify the name of the payroll `.war` file (`PayrollWeb.war`), select *Web Module* as the *Type*, then click *OK*.

4 Click *New* to create the required protected resources.



5 Select all three protected resources.

The *manager* protected resource has `/manager/*` as its URL path and enables the PayrollWebManager Authorization policy. This policy allows only managers to access the manager pages. Everyone else is denied access.

The *myinfo* protected resource has `/myInformation.jsp` and `/payserv` as its URL paths. Both the PayrollWebEmployee and PayrollWebManager Authorization policies are enabled for this resource. This allows both employees and managers to view their own information pages.

The *public* protected resource uses `/*` for its URL path and is not assigned an Authorization policy. This allows everyone who can log in to the Identity Server to have access to the public pages of the application.

6 To save your changes, click *Configuration Panel*, then click *OK*.

7 On the J2EE Agents page, click *Update*.

Assigning the Authorization Policies to Protected EJB Resources

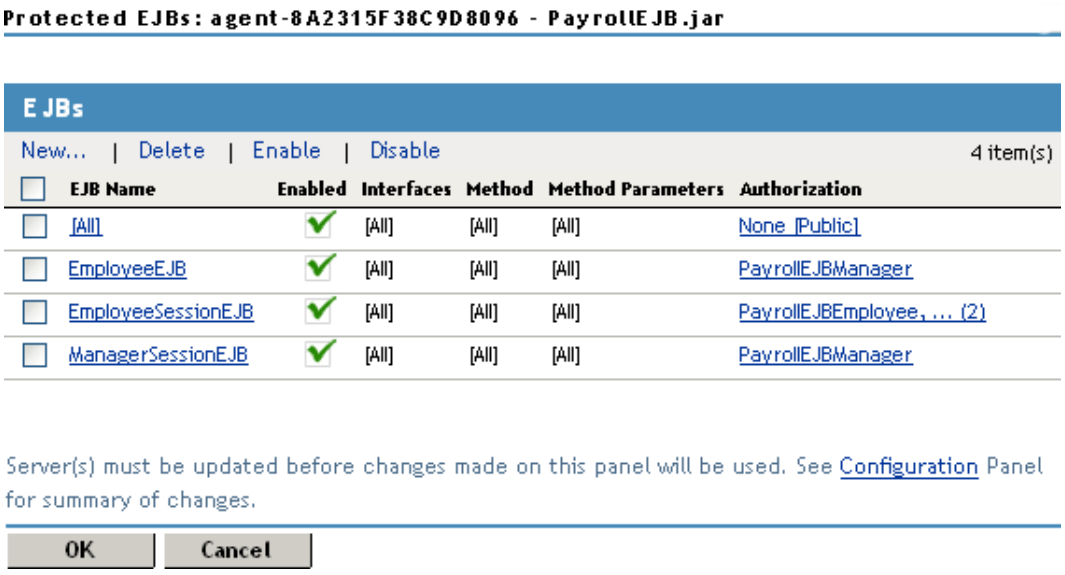
To allow the J2EE Agent to enforce authorization for the payroll EJB module, you need to create policies for four EJBs.

1 Click *Devices* > *J2EE Agents* > *Edit*.

2 In the Access Control Configuration section, deselect *Enforce application server policy*, select *Enforce additional authorization policy*, then click *Manage authorization policies*.

3 Click *New*, specify the name of the payroll `.jar` file (`PayrollEJB.jar`), select *EJB Module* as the *Type*, then click *OK*.

4 Click *New* to create the required EJB modules for this application.



5 Select all four EJB modules.

The *[All]* EJB is not assigned an Authorization policy. This allows everyone who can log in to the Identity Server to have access to the public EJBs of the application.

The *EmployeeEJB* enables the PayrollEJBManager Authorization policy. This policy allows only managers to change sensitive employee information, such as an employee's salary.

The *EmployeeSessionEJB* enables both the PayrollEJBEmployee and PayrollEJBManager Authorization policies for this resource. This allows both employees and managers to view their own employee information.

The *ManagerSessionEJB* enables the PayrollEJBManager Authorization policy. This policy allows only managers to manage employee information. Everyone else is denied access.

6 To save your changes, click *Configuration Panel*, then click *OK*.

7 On the J2EE Agents page, click *Update*.

7.4.4 Testing the Configuration

1 Deploy the sample payroll application on your J2EE server.

The location of the sample application is platform-specific:

- On a Linux, Solaris, or AIX J2EE server, the application is copied to the `/opt/novell/nids_agents/example` directory.
- On a Windows J2EE server, the application is copied to the `<Install_Directory>\sampleapp` directory.

2 On your J2EE server, prepare the application to use the agent for login and logout. (See Section 4.1, "Preparing the Application for the Agent," on page 65).

These steps have already been performed for the sample application. See the `web.xml` file in the application's `WEB-INF` directory.

3 Enable the RunAs role feature on your J2EE server.

- **JBoss:** This tasks have already been performed for JBoss. To understand what was modified, see Section 4.2, "Configuring Applications on the JBoss Server," on page 67.
- **WebSphere:** See Section 4.3.3, "Configuring for User RunAs Roles," on page 70.
- **WebLogic:** See Section 4.4, "Configuring Applications on the WebLogic Server," on page 81.

4 To test this configuration, send the following request from a browser:

```
http://<Application_Server_DNS_Name>:<port>/payroll
```

Replace `<Application_Server_DNS_Name>` with the DNS name or the IP address of your application server. Replace `<port>` with the port number you have configured the J2EE Agent to use.

5 Log in as a user who matches the condition to receive the Employee role. Access the *My Page* and the *Manager Page*.

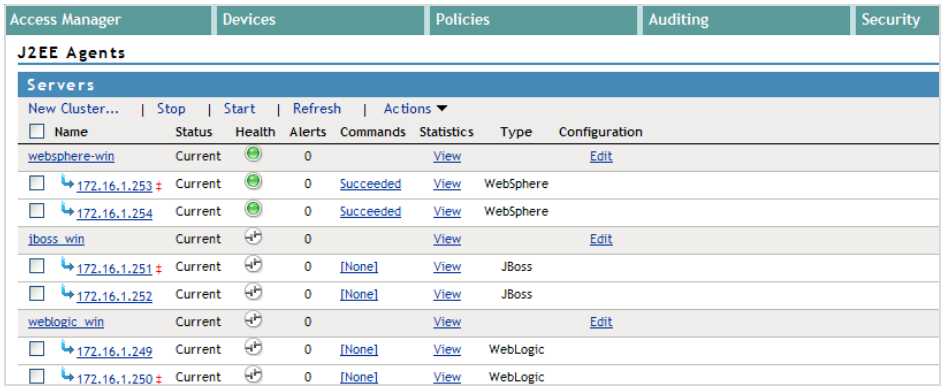
6 Log out and log in as a user who matches the condition to receive the Manager role. Access the *My Page* and the *Manager Page*.

8 Managing a J2EE Agent

The following sections describe the options available for managing a J2EE Agent.

8.1 Viewing General Status Information

1 In the Administration Console, click *Devices* > *J2EE Agents*.



The table contains general information about each installed agent.

Column	Description
Name	Displays a list of all the J2EE Agents that can be managed from this console. Click the link of a particular agent to view or modify its general details. For more information, see Section 8.8, “Viewing Platform Information,” on page 114.
Status	Indicates the configuration status of the agent. The possible states are pending, update, and current. ◆ Current indicates that all configuration changes have been applied. ◆ Update indicates that a configuration change has been madebut not applied. Click this link to apply the changes. You can select to have the agent read its complete configuration file (all configuration). When the embedded service provider (ESP) logging settings have been modified on the Identity Server, the update logging settings option is available. ◆ Pending indicates that the agent is processing a configuration change but has not completed the process.
Health	Indicates whether the J2EE Agent is functional. Click the icon to view information about the operational status of an agent. For more information, see Section 8.2, “Managing the Health of an Agent,” on page 108.
Alerts	Indicates whether any alerts have been sent. If the alert count is non-zero, click the link for information. For more information, see Section 8.4, “Managing Alerts,” on page 111.
Commands	Indicates whether any commands are pending. Click the link for information. For more information, see Section 8.9, “Viewing the Status of Recent Commands,” on page 114.
Statistics	Provides a link to statistics pages. For more information, see Section 8.6, “Viewing Statistics,” on page 113.
Type	Indicates the type of agent: JBoss, WebLogic, or WebSphere.
Configuration	Provides a link to the configuration page. For more information, see Chapter 5, “Configuring the Basic Features of a J2EE Agent,” on page 83.

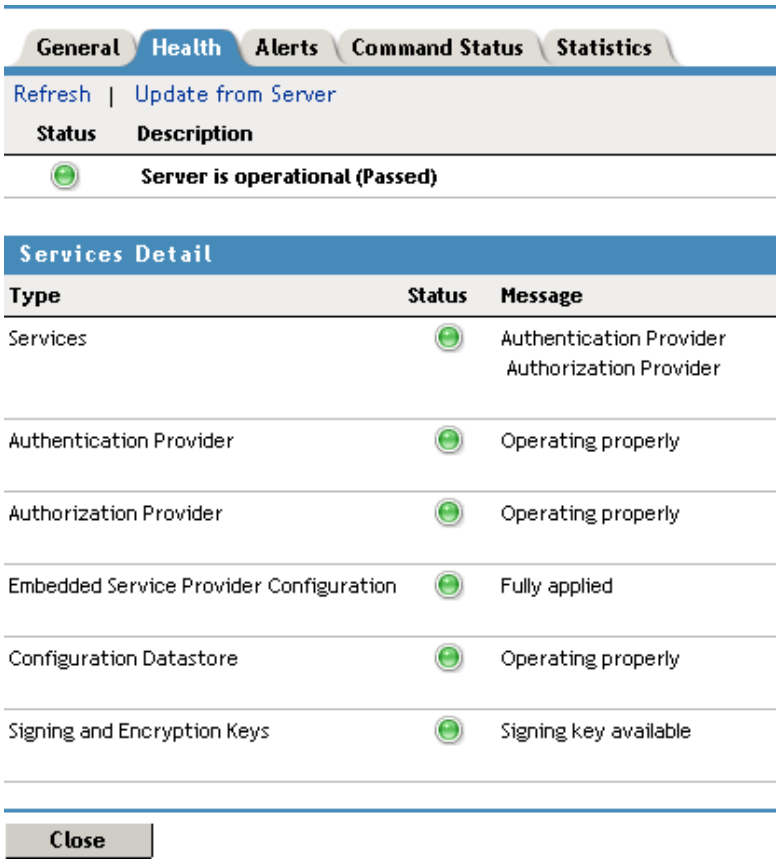
2 To view information about one of the displayed options, click the corresponding link or icon.

3 To update the list of agents and their health status, click *Refresh*.

8.2 Managing the Health of an Agent

If a J2EE Agent is functioning normally, its health icon is green. If the icon is any other color, you need to discover the cause.

1 In the Administration Console, click *Devices* > *J2EE Agents* > *[Name of Agent]* > *Health*.



2 If you think the information on the page might be stale, click *Refresh*.

3 If you want to have the page refreshed with the information sent from the agent, click *Update from Server*.

4 If the status icon does not turn green, view the information in the Services Detail section.

For an agent, this includes information such as the following:

Status Category	If Not Healthy
Services: Lists the Access Manager services that this agent has been configured to use.	Check the status of the listed services.
Authentication Provider: Indicates whether the agent has been configured to use an authentication contract and assigned a base URL.	See Section 2.3, “Configuring the Agent for Direct Access,” on page 45.
Authorization Provider: Indicates whether the agent has been configured to use authorization policies before granting access.	To view your configuration, click *Devices* > *J2EE Agents* > *Edit* > *Manage authorization policies*. For configuration information, see Section 6.1, “Configuring Access Control,” on page 87.
Enterprise Service Provider Configuration: Indicates whether the agent has a trusted relationship with an Identity Server. At least one Identity Server must be configured and set up as a trusted authentication source for the agent.	See Section 2.3, “Configuring the Agent for Direct Access,” on page 45 and configure the *Trusted Identity Configuration* field.
Configuration Datastore: Indicates whether the configuration datastore is functioning correctly.	If it isn’t functioning correctly, you might need to restore the data from a backup.
Signing and Encryption Keys: Indicates whether the Signing keystore contains a key.	Click *Devices* > *J2EE Agents* > *Edit* > *Service Provider Certificates* > *Signing*, and replace the signing key in this keystore.

5 Click *Close*.

If the status is not green, you should also check the following:

- Section 8.4, “Managing Alerts,” on page 111
- Section 8.9, “Viewing the Status of Recent Commands,” on page 114

8.3 Managing the Health of a Cluster

1 In the Administration Console, click *Devices* > *J2EE Agents*.

2 Click the Agents cluster.

3 Select the *Health* tab. The Health page displays the current health status of a cluster configuration.

Server Name: Displays the IP address that identifies the J2EE Agent.

Health: Displays the health status of the server.

Description: Displays the description of the health status of the server.

8.4 Managing Alerts

The J2EE Agent sends alerts when it is not functioning correctly. After discovering the cause of an alert and then correcting the problem, you should clear the alert from the list.

1. In the Administration Console, click *Devices* > *J2EE Agents* > *[Name of Agent]* > *Alerts*.
2. To send an acknowledgement, select the check box by the alert, then click *Acknowledge Alert(s)*. When you acknowledge an alert, you clear the alert from the list.
3. The J2EE Agent sends the following alerts when it is not functioning correctly:

Alert Message	Solution
`The Embedded Service Provider base URL is not set. Configure the Embedded Service Provider base URL.`	Click *Devices* > *J2EE Agents* > *Edit,* and configure the *J2EE Application Server URL* field. For configuration information, see Section 2.3, “Configuring the Agent for Direct Access,” on page 45.
`The Embedded Service Provider returned not OK. Check that the Embedded Service Provider is running properly.`	Restart the agent. Click *Devices* > *J2EE Agents* > *[Server Name]* > *Stop* | *Start*.
`The Embedded Service Provider base URL is invalid. Configure the Embedded Service Provider base URL.`	Click *J2EE Agents* > *Edit* , and configure the *Devices* > *J2EE Application Server URL* field. For configuration information, see Section 2.3, “Configuring the Agent for Direct Access,” on page 45.
`The Embedded Service Provider could not be contacted due to an SSL exception. Check that certificates are set up properly.`	Check the SSL certificate trust configuration. See Section 5.2, “Managing Embedded Service Provider Certificates,” on page 85 and Section 5.3, “Configuring SSL Certificate Trust,” on page 85.
`The Embedded Service Provider could not be contacted due to a socket exception. Check that the Embedded Service Provider is running properly.`	Indicates a network problem. Verify that the J2EE server is running. Restart the J2EE Agent by clicking *J2EE Agents*, select the agent, then click *Stop* | *Start*.
`The Embedded Service Provider could not be contacted due to a general IO exception. Check that the Embedded Service Provider is running properly.`	Restart the agent. Click *J2EE Agents*, select the agent, then click *Stop* | *Start.*
`Not running. Start the J2EE Agent.`	Click *J2EE Agents*, select the agent, then click *Stop* | *Start*.
`Failed to construct the policy enforcement points. Check the J2EE Agent configuration and restart.`	Click *Policies* and check your J2EE Agent policies.
`WebSphere global security is not enabled. Enable WebSphere's global security.`	This is enabled during installation. See your WebSphere documentation.
`WebSphere server security is not enabled. Enable WebSphere's server security.`	This is enabled during installation. See your WebSphere documentation.
`The JACC PolicyConfigurationFactory was not initialized. Configure the J2EE Application Server to use the proper PolicyConfigurationFactory.`	Contact NetIQ Support.

4 Click *Close*.

8.5 Managing Cluster Alerts

The Alerts page allows you to view information about current Java alerts and clear them. An alert is generated whenever the configuration detects a condition that prevents it from performing normal system services.

1. In the Administration Console, click *Devices* > *J2EE Agents*.
2. Click the Agents cluster.
3. Select the *Alerts* tab. The cluster alerts page is displayed with the following information:

   Server Name: Displays the IP address of the J2EE Agent server.

   Severe: Displays the severity of the alert sent.

   Warning: Displays the warning.

   Information: Displays information on the alert.

8.6 Viewing Statistics

The following statistics allow you to monitor the sessions and run time of the J2EE Agent.

1. In the Administration Console, click *Devices* > *J2EE Agents* > *[Name of Agent]* > *Statistics.*
2. Select whether to monitor live or static statistics:
   - **Statistics:** Select this option to view the statistics as currently gathered. The page is static, and the statistics are not updated until you click *Live Statistics Monitoring*.
   - **Live Statistics Monitoring:** Select this option to view the statistics as currently gathered and to have them refreshed at the rate specified in the *Refresh Rate* field.
3. Check the following statistics:

Column	Description
Active Sessions	Displays the number of sessions currently established on the J2EE server through the Access Manager. To view the most popular time for establishing sessions, click *Graphs*.
Start Up Time	Displays the time when the J2EE Agent was last started.
Up Time	Displays how long the J2EE Agent has been running since it was last started.

4. Click *Close*.

8.7 Viewing Cluster Statistics

1. In the Administration Console, click *Devices* > *J2EE Agents*.
2. Click the *Agents* cluster.
3. Select the *Statistics* tab. The cluster statistics page is displayed with the following information:

   Server Name: The IP address identifying the J2EE Agents in the cluster. Click the *Edit* link to edit server information.

   Up Time: Displays the time for which the J2EE Agents server is up.

   Active Sessions: Displays the number of current active sessions.

Statistics: Click the *View* link to get a summary of the statistics of individual servers in a cluster.

8.8 Viewing Platform Information

The General page displays version and platform information:

1. In the Administration Console, click *Devices* > *J2EE Agents* > *[Name of Agent]*.

   The fields that contain links transfer you to the page where you can edit the information. If the field is empty, click *Edit* to add a value.

2. To view platform and version information, look at the following fields:

   Server Version: Specifies the version of the software currently installed on this J2EE Agent.

   Server Type: Specifies the type of server on which the J2EE Agent is installed (JBoss, WebLogic, or WebSphere for this release). Other types are in development.

   Server Platform: Specifies the operating system of the J2EE server.

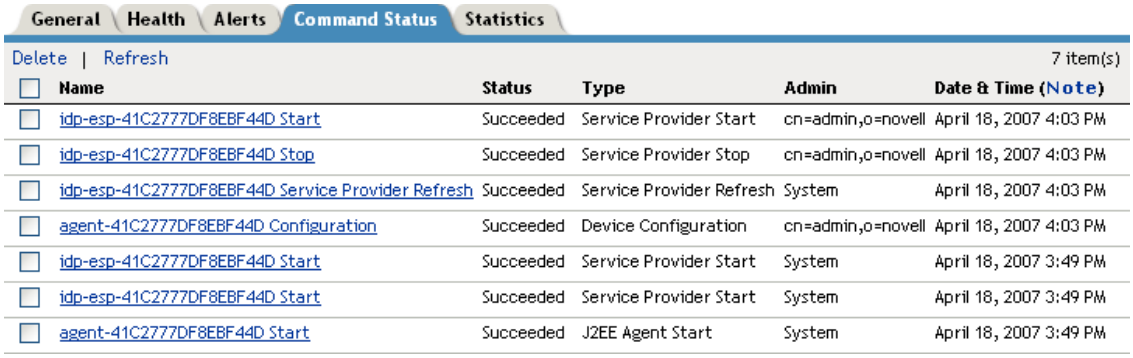
3. Click *Close*.

For information on how to modify the fields with links, see Section 5.4, "Modifying the Display Name and Other Details," on page 86 and Section 5.5, "Changing the IP Address of a J2EE Agent," on page 86.

8.9 Viewing the Status of Recent Commands

Agent commands are issued when the configuration of the agent is modified and when the agent is stopped, started, or refreshed.

1. In the Administration Console, click *Devices* > *J2EE Agents* > *[Name of Agent]* > *Command Status*.



2. Select one of the following actions:

   - **Delete:** To delete a command, select the check box for the command, then click *Delete*. The selected command is cleared.
   - **Refresh:** To update the current cache of recently executed commands, click *Refresh*.
   - **Name:** To select all the commands in the list, click *Name*, then click *Refresh* or *Delete*.

3. View the information. The following columns display information about each command:

Column	Description
Name	Contains the display name of the command. Select this link to view additional details about the command.
Status	Specifies the status of the command and includes states such as Pending, Incomplete, Executing, Succeeded, Failed, and Unsuccessful.
Type	Specifies the type of the command.
Admin	Specifies whether the system or a user issued the command. If a user issued the command, the field contains the DN of the user.
Date & Time	Specifies when the command was issued. The date and time are displayed in local time.

4 To view additional information about a command, click the name of the command.

5 Click *Close*.

8.10 Stopping and Starting the Agent

When you stop a J2EE Agent, all the resources it is protecting are not available until the agent is started again.

To stop or start a selected J2EE Agent:

1 In the Administration Console, click *Devices* > *J2EE Agents.*

2 To stop the agent, select the agent, then click *Stop* > *OK*.

3 To start the agent, select the agent, then click *Start* > *OK*.

8.11 Stopping and Starting the Embedded Service Provider

When you stop the Embedded Service Provider of a J2EE Agent, the provider closes the application session for logged-in users. The actual user session is on the Identity Server so the user can access the resources without logging in again after the Embedded Service Provider has started. For example, if a user is adding items to a shopping cart when the action to stop and start the Embedded Service Provider occurs, the user loses the items in the shopping cart but can continue shopping and adding new items without logging in again.

To stop or start the Embedded Service Provider of a J2EE Agent:

1 In the Administration Console, click *Devices* > *J2EE Agents.*

2 To stop the Embedded Service Provider, select the agent, then click *Actions* > *Service Provider* > *Stop Service Provider* > *OK*.

3 To start the Embedded Service Provider, select the agent, then click *Actions* > *Service Provider* > *Start Service Provider* > *OK*.

4 To restart the Embedded Service Provider, select the agent, then click *Actions* > *Service Provider* > *Restart Service Provider* > *OK*.

8.12 Deleting an Agent from the Administration Console

When you delete an agent from the Administration Console, the configuration file for the selected agent is deleted and you can no longer manage it. Usually you delete an agent only if you are removing the agent from the J2EE server or if you want another console to manage the agent. After you have deleted an agent, the only way to trigger an import into a different Administration Console is to reinstall the agent.

To delete a J2EE Agent from the Administration Console:

1. In the Administration Console, click *Devices* > *J2EE Agents*.
2. Select the agent, then click *Actions* > *Delete*.
3. Click *OK*.

9 Troubleshooting the J2EE Agent

This section has the following information:

9.1 Troubleshooting the J2EE Agent Import

If the J2EE Agent does not appear in the Administration Console after the installation has finished, try one or more of the following:

- If the import started and failed to complete, a *repair import* link appears at the bottom of the table on the J2EE Agents page. Click this link to repair the import.
- If your J2EE server is not running, the Administration Console cannot import the J2EE Agent. Start J2EE server and wait 30 seconds before trying to configure the agent in the Administration Console.
- If you installed the J2EE Agent on a WebSphere server, make sure you have restarted the WebSphere server. The J2EE Agent does not import into the Administration Console until WebSphere is restarted.
- If you are running WebSphere with additional Java 2 security checks, the agent cannot import into the Administration Console. In the WebSphere console, turn off the additional Java 2 security checks or create a policy that grants full access to the nesp application.

9.2 Authorization Policies Fail for Some Attributes

The authorization policy fails if they are configured based on any of the following attributes:

- LDAP Attribute
- Liberty User profile
- Authentication Contract
- Credential Profile

To work around this problem, configure the roles in the Identity Server based on these attributes, and configure the authorization policies for the J2EE agents based on these identity roles.

9.3 The Health Status Displays as “Server Is Not Responding

When J2EE gents installed on the WebSphere Application Server v6.1, the health status might be reported as `Server is Not Responding` under high stress loads with frequent connections opening and closing the clients. This is because of a known issue with 6.1. To workaround this problem, upgrade to Fix Pack 17 of 6.1.

9.4 Auto-import Agents Fails on WebLogic Running on RedHat

When you install the J2EE Agents on a WebLogic server running on RedHat Enterprise Linux, auto-import agents might fail with the following error message:

This is because JCC fails to start automatically. To manually restart JCC, the following command:

```
/etc/init.d/novell-jcc start
```

9.5 Error: Invalid Administration Server IP Address

While installing the J2EE gents you might see the `Invalid Administration Server IP address` error, even if the IP address of the Administration Console is valid. This error could be appearing because of the following reasons:

- Section 9.5.1, “JRE Version is Wrong,” on page 119
- Section 9.5.2, “Issues With the Administration Console,” on page 119

9.5.1 JRE Version is Wrong

Enter the following command to verify the version of JRE being used by the J2EE Installer:

```
java -version
```

The J2EE installer Java 1.5 or later. To fix the problem, download and install JRE 1.5. Set the `PATH` environment variable to point to the `bin` directory of the newly installed JRE.

For example:

- In Linux or UNIX: `export PATH = <JREDirectory>/bin:$PATH`
- In Windows: `set PATH = <JREDirectory>/bin;%PATH%`

run the J2EE installer.

9.5.2 Issues With the Administration Console

Administration Console has the following issues:

- The installation directory of the administration console reachable LDAP.
- Check if firewall block the ports. If the port.
- eDirectory is running.
- The Administration Console is installed properly.

9.6 Installer Stops Responding While Installing on WebSphere

If the J2EE installer stops responding while installing on the WebSphere server, check if the installation was performed on a new instance of the WebSphere Application Server that is part of the WebSphere Cell. If it is, the possible cause could be that the installer uses the `wsadmin` script provided by WebSphere to perform configuration changes to the application server. When the installer is run on a new server instance of the WebSphere Application Server instance that is part of the WebSphere Cell, WebSphere requests the user to authorize signer certificates given by the deployment manager. The installer eventually connects to this deployment manager. To verify this, run the `wsadmin` script from a command line.

To work around this problem:

1. Kill the installer process.
2. Run the `wsadmin` script from a command line interpreter. If you are prompted to confirm the signer certificate, confirm it.
3. Run the J2EE Agent installer.
4. If you are prompted to confirm overwriting some of the files that were installed during the previous failed attempt, click *OK*.

Contact NetIQ Support if the problem persists.

9.7 Unable to Federate WebSphere Custom Profile If Agent Already Installed

When a WebSphere server instance that is created one machine is federated into the deployment manager of another machine that already has server instances with J2EE Agent installed, the newly federated server instance not start. This is because, the security configuration changes that are performed apply to all the instances of the deployment manager and all the application server instances that are part of it.

To work around this problem, copy the following files from the `$WAS_HOME$/lib` folder of the machine that has agents installed, to the `$WAS_HOME$/lib` folder of the new machine, then restart the server:

- `NidsCommonAgent-unsign.jar`
- `NidsWebSphere-unsign.jar`
- `nxpe.jar`
- `jcc-unsign.jar`
- `nxpe-toolkit-unsign.jar`

9.8 Authorization Fails in the WebSphere Application

Entries in the `NidsJaccRoles.xml` file indicate whether the `RunAs` roles and user/grouptorole mappings are automatically propagated to the JAAC module. If you use SLES as your WebSphere host, the file is located in a path similar to the following example:

```
/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/novell/cells/sles10Node01Cell/
nodes/sles10Nodeo1/servers/server1/NidsJaccRoles.xml
```

The entries look similar to the following:

```
<J2EERole roleId="Manager">
<User Name="
```

If you have configured WebSphere to map roles, the authorization of the user might occasionally fail. This could be because, when `Run As` roles and user/grouptorole mappings are configured after the J2EE Agent is installed, they fail to be propagated to the JAAC module even after a restart.

To workaround this issue:

1. Browse to the folder where the J2EE Agent is installed.
2. Open `uDontKnowJacc.jy`, which is located in the `/novell/nids_agents/bin` folder.
3. Delete the first line.
4. Modify `member1` to *<application server name>*.

   Replace *<application server name>* with the name of the application server instance where NIDPJ2EEApp is installed.
5.
6. Execute the following command at the shell prompt:

   <path-to-websphere>`/bin/wsadmin.sh -`*username* *<adminusername>* `-password` *<adminpassword>* `-lang jacl -f <path-to-nids_agents-folder>/uDontKnowJacc.jy`

   Replace *<path-to-websphere>* with the path where the WebSphere server is installed.

Replace <*adminusername*> with the name of the WebSphere administrator.

Replace <*adminpassword*> with the password of the WebSphere administrator.

NOTE: For more information about updating a security policy, see "Propagating a Security Policy" (http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tsec_jaccmigrate.html).

9.9 Audit Log Event Problems on 64-Bit Platforms

No audit log events occur on 64-bit platforms. There is currently no workaround for the WebSphere Agent. For the JBoss and WebLogic Agents, you can enable log events on 64-bit platforms by deleting the `LogEvent.jar` file and replacing it with the `NAuditPA.jar` file.

On Windows, the `NAuditPA.jar` file is located in `Program Files\novell\Nsure Audit` directory. On Linux, the file is located in `/opt/novell/naudit/java/pa` directory.

- Section 9.9.1, "JBoss Agent," on page 121
- Section 9.9.2, "WebLogic Agent," on page 121

9.9.1 JBoss Agent

Delete the `LogEvent.jar` file in the server configuration `lib` directory (the location for the default configuration is the `JBoss/server/default/lib` directory). Copy the `NAuditPA.jar` file to this directory.

The `LogEvent.jar` file also needs to be deleted from the ESP directory (`JBoss/server/default/deploy/nesp.ear/nesp.war/WEB-INF/lib`). The `NAuditPA.jar` does not need to be added to this directory.

9.9.2 WebLogic Agent

Linux: Edit the `WL_HOME/common/bin/commEnv.sh` file. Change the `${AGENT_LIB}/LogEvent.jar` path variable to `/opt/novell/naudit/java/pa/NAuditPA.jar` variable.

Delete the `LogEvent.jar` file from the ESP directory (`nesp.ear/nesp.war/WEB-INF/lib`).

Windows: Edit the `WL_HOME/common/bin/commEnv.cmd` file. Change the`%AGENT_LIB%\LogEvent.jar` path variable to `Program Files\novell\audit\NAuditPA.jar` variable.

Delete the `LogEvent.jar` file from the ESP directory (`nesp.ear/nesp.war/WEB-INF/lib`).

9.10 JBoss and SSL

If you want to restrict access to SSL on JBoss, you need to either disable the HTTP port in JBoss and enable only the SSL port or configure SSL in the `web.xml` file. It is not enough to select *Require SSL* on the Protected Web Resource page. In the Administration Console, click *Devices* > > *J2EE Agents* > *Edit* > *Manage authorization policies* > *[Name of Web Module]* > *[Name of Protected Resource]*.

9.11 Viewing Log Files

The J2EE agent logs messages to the J2EE server log files. For verbose messages, including policy evaluation messages, you need to enable tracing. In the Administration Console, click *Devices* > *J2EE Agents* > *Edit* > *Enable tracing*.

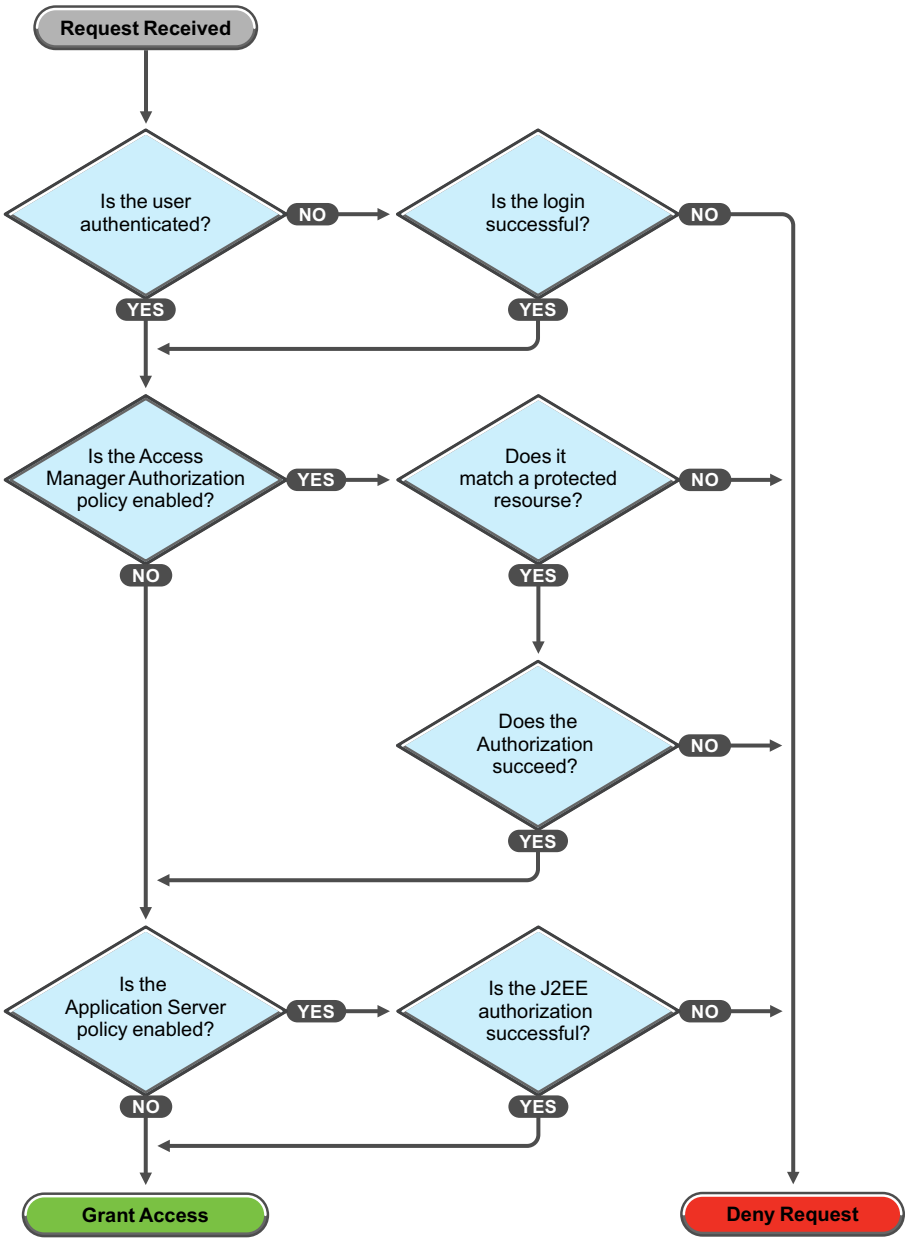
The location of the log files for each J2EE server is implementation-specific:

- **JBoss Server:** For a JBoss server, the log messages are logged to the `$JBOSS_HOME/log/jboss.log` file if you launched the JBoss server by using the `run.sh` script found in the `bin` folder. Messages are also sent to the console, so you should check the console or the `$JBOSS_HOME/server/default/log/server.log` file.
- **WebSphere Server:** For a WebSphere server, the log messages are logged to files in the `$WAS_BaseDir/profiles/$ProfileName/logs` directory. Check the `SystemOut.log` and `SystemErr.log` files.
- **WebLogic Server:** For a WebLogic server, the log messages are sent to standard out. If you launch the server in a console window, the messages appear in this window. If you want the messages logged to the server log file, you need to configure the server to send standard out to this file. This can be done from the WebLogic Administration Server console application in the *Logging* tab under *Servers*.

9.12 Troubleshooting Access Control

When a user requests access to a resource protected by the J2EE Agent, the request flows through the policy enforcement points illustrated in Figure 9-1.

Figure 9-1 *Access Control Flow*



If users are not getting access to a resource when they should, you need to enable tracing (see Section 9.11, "Viewing Log Files," on page 122) and view the log files to determine where the error is occurring.

- **Login:** The Identity Server supports a variety of contracts that can be used for logging in. You need to create a contract that is compatible with the J2EE server, if it has been configured to verify login credentials. You can select an *Any Contract* option, but if you configure the J2EE Agent to use this option, be sure that all defined contracts are compatible with the J2EE server. If a user logs into another Access Manager resource with a contract that is not compatible, the *Any Contract* option allows the J2EE Agent to accept those login credentials, but the J2EE server denies access.

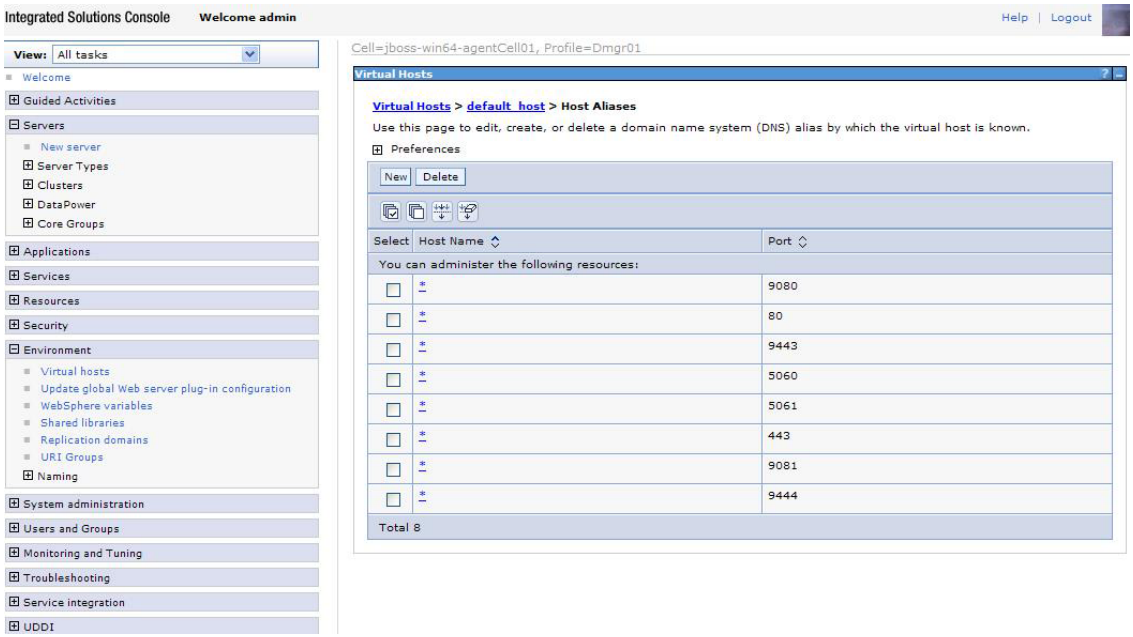
- **Access Manager Authorization Policy:** To enable an Access Manager authorization policy, you must select the *Enforce additional authorization policy* option, create a protected resource, create a policy for the resource, then enable the policy.
- **Protected Resource:** If you have enabled the *Enforce additional authorization policy* option but have not created a protected resource that matches the requested application URL or JavaBean, the user is denied access to the resource.
- **Web Authorization Policy or Enterprise JavaBean Authorization Policy:** If the only requirement you have for granting access is authentication, you should create a policy that grants access based on the authenticated role. All users are assigned this role when they successfully authenticate to the Identity Server.
- **Application Server Authorization Policy:** To enable the policies you have configured on the J2EE server, you must enable the *Enforce application server* policy option. You must also create Access Manager Role policies for the roles that you have configured the J2EE server to use for authorization. Depending upon the application, role names can be case sensitive, so when you create the role, make sure to use the case the application expects.

9.13 Adding the Listening Port in Host Aliases

If more than one Websphere server instance is created on the non-default port and J2EE Agent is installed on that instance, then you must add the listening port in Host aliases.

For example, if you already have a WebSphere server1 running and if a second Websphere server instance, server2 is created at port 9081(non-ssl) and 9444(ssl), then port 9081 and 9444 must be added in host aliases.

1 Go to *Websphere Administration Console* > *Environment* > *Virtual hosts* > *Default host* > *Host Aliases*.



2 Click *New.*

3 Change the <Port> to the listening port of the WebSphere server2 instance.

4 Click *Apply* and click *Save* to save the configuration changes.

9.14 J2EE Agents Deny New Authentication Because of Low System Memory

To work around this issue, increase the java heap size and set the threshold memory value to zero.

Complete the following steps:

1. Log in to Websphere Administration Console.
2. Click Websphere Application Server.
3. Under *Server Infrastructure*, select *Java and Process Management* > *Process Definition*.
4. Set *Maximum heap size* to 2GB.
5. Set the threshold value to zero using the following arguments:

   append `-Dnids.freemem.threshold=0`to the *Debug arguments* field.add `-Dnids.freemem.threshold=0` to the *Generic JVM arguments* field.

   For example, debug arguments: `-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7777 -Dnids.freemem.threshold=0`

   Generic JVM arguments: `-Dnids.freemem.threshold=0`
6. Restart the application server.