J2EE Policy Agent Guide

Sun™ ONE Identity Server 5.1

Version 1.0

816-6690-10
January 2003
Fourth Edition

Contents

About This Guide

This *J2EE Policy Agent Guide* offers an introduction to Sun™ ONE Identity Server Policy Agent and describes how to install and configure Sun ONE Identity Server Policy Agent on Application Servers.

This preface contains the following sections:

- What You Are Expected to Know
- Documentation Conventions Used in This Manual
- Related Information

What You Are Expected to Know

Before you begin, you should already be familiar with the following topics:

- Application Server
- Sun ONE Identity Server
- Java programming and J2EE
- Windows 2000, Solaris 8, Solaris 9, or HP-UX 11 operating systems

Documentation Conventions Used in This Manual

There are certain typographic and terminology conventions used to simplify discussion and to help you better understand the material. These conventions are described below.

Typographic Conventions

This book uses the following typographic conventions:

- *Italic type* is used within text for book titles, new terminology, emphasis, and words used in the literal sense.
- `Monospace font` is used for sample code and code listings, API and language elements (such as function names and class names), filenames, pathnames, directory names, HTML tags, and any text that must be typed on the screen.
- *Italic serif font* is used within code and code fragments to indicate variable placeholders. For example, the following command uses *filename* as a variable placeholder for an argument to the gunzip command:

  gunzip -d *filename*.tar.gz

Terminology

Below is a list of the general terms that are used in the Sun ONE Identity Server Policy Agent for WebLogic 6.1 SP2 documentation:

- *Agent_Install_Dir* is a variable placeholder for the directory where you have installed the Sun ONE Identity Server Policy Agent.
- *WebLogic_Install_Dir* is a variable placeholder for the home directory where you have installed WebLogic Server.
- *WAS_root_dir* is a variable placeholder for the home directory where you have installed WebSphere Application Server. For example: `/opt/WebSphere/AppServer`

Related Information

This section lists additional sources of information.

Sun ONE Identity Server Documentation

You can find Sun ONE products documentation at the following site:

`http://docs.sun.com/db/prod/s1idsrv#hic`

J2EE Documentation

You can find the J2EE documentation at the following site:

```
http://java.sun.com/j2ee
```

WebLogic Server Documentation

You can find the WebLogic Server documentation at the following site:

```
http://www.bea.com
```

Sun ONE Products Documentation

You can find Sun ONE products documentation at the following site:

```
http://docs.sun.com/
```

Sun ONE Technical Support

You can contact Sun ONE Technical Support through the following location:

```
http://www.sun.com/service/support/software/iplanet/index.html
```

Chapter 1

Read This First

The Sun ONE Identity Server Policy Agent enables Application Servers to enforce authentication and authorization using Sun ONE Identity Server services, thereby securing client access to the hosted J2EE applications and enforcing J2EE security policies defined in the deployed application's Deployment Descriptors.

This chapter provides a brief overview of Sun ONE Identity Server Policy Agent, as well as some concepts you will need to understand before proceeding with the Installation program. The information in this chapter is common to all Application Servers.

Topics include:

- Uses of Policy Agent for Application Server
- Supported Servers

Uses of Policy Agent for Application Server

The Sun ONE Identity Server Policy Agent for Application Server may be installed for protecting a variety of hosted J2EE applications which may require a varying set of security policy implementation. The security infrastructure of J2EE provides declarative as well as programmatic security which are platform independent and are supported by all the compliant J2EE application servers. For details on how to use J2EE platform's declarative as well as programmatic security, refer to J2EE documentation which can be found at `http://java.sun.com/j2ee`.

The Agent provides the ability to enable role-to-principal mapping for protected J2EE applications with Sun ONE Identity Server principals. Thus at runtime, when a J2EE policy is evaluated, it is done against the information available in Sun ONE Identity Server. Using this functionality, administrators may configure their hosted J2EE applications to be protected by the Agent which provides real security services and also other key features such as single sign-on.

Examples

A Commerce Application

A commerce application may have a variety of specialized Enterprise JavaBeans components that offer a spectrum of services to the clients. For instance, there could be a specialized component that provides the ability to create purchase orders. Similarly, there could be a specialized component that provides the ability to approve a purchase order. While such components provide the basic business services for the application to function, the very nature of tasks that they accomplish require a security policy to enforce appropriate use of such services.

Using the deployment descriptors, the application vendor or developer can express this intent by protecting such components using abstract security role names. For example, there could be a role called “Buyer” which protects the component that provides the ability to create a purchase order. Similarly, there could be a role called “Approver” which protects the component that provides the ability to approve a purchase order. While these roles convey the intent of the application vendor or developer to enforce such security policies, they will not be useful unless these abstract role names are mapped to real life principals such as actual users or actual roles that reside in Identity Server.

The Agent provides the ability to the container to enforce such a runtime linkage of abstract security roles to real life principals. Once the Agent is installed and configured, the Application security roles can be mapped to real principals. For example, the role “Buyer” may be mapped to a Identity Server role called “Staff”. Thus when a user “Arvind” tries to access the application's protected resources, the Agent will allow this access if and only if the actual user “Arvind” is a member of the mapped role “Staff”.

An Intranet Employee Portal

An intranet employee portal may offer services such as payroll information, online benefits administration etc. While such services may be offered in a read-only manner to regular employees, administrators may have special privileges that can allow them to update the associated data. For instance, there could be a specialized

Enterprise JavaBeans component that provides two services – one for reading payroll information and the other for updating payroll information. Using the Agent to protect this application, it will be possible to grant the administrators the privileges necessary to update payroll information, while the employees may only have read-only access.

A Content-Based Web Application

A content based web application can offer pay per-view services. The application may be partitioned into two domains—the public domain which is accessible to anonymous users, and the private domain which is accessible only to the subscribers of this particular service. Using the Agent, it will be possible to enforce that only authenticated and authorized users may be allowed to access the private domain of the application, while any user has the ability to access the public domain. Specific Servlets and JSPs that provide application functionality will be protected by the Agent by enabling the mapping of the associated security roles with actual Identity Server principals.

Supported Servers

Sun ONE Identity Server Policy Agent is supported on the following servers:

- WebLogic 6.1 SP2
- WebSphere 4.0.3 AE
- Sun ONE Application Server 7.0
- PeopleSoft 8.3

Policy Agent for WebLogic 6.1 SP2

This chapter provides a brief overview of WebLogic Agent and describes how to install and configure Sun ONE Identity Server Policy Agent for WebLogic 6.1 SP2 Application Server running on Solaris 8, Windows 2000, and HP-UX 11 operating systems. Topics include:

- Overview
- Guidelines
- Installing the Agent
- WebLogic Server Configuration
- Application Configuration
- Agent Configuration
- Uninstalling the Agent

Overview

The Sun ONE Identity Server Policy Agent enables WebLogic Server to enforce authentication and authorization using Sun ONE Identity Server services, thereby securing client access to the hosted J2EE applications and enforcing J2EE security policies defined in the deployed application's deployment descriptors

The Sun ONE Identity Server Policy Agent for WebLogic Server consists of two main components that affect the operation of Application Server as well as the behavior of the protected application. These components are:

- **Agent Realm.** The Agent Realm component provides the ability to Application Server to interact with Identity Server's user and role information. This acts as the core of the Agent and has to be configured correctly in order for the Agent to function.
- **Agent Filter.** The Agent Filter component provides the ability to the hosted application to enforce Sun ONE Identity Server based authentication and is responsible for creating the Security Principal associated with the logged on user. Every application that has to be protected by the Agent must have its Deployment Descriptors changed to reflect that it is now configured to use the Agent Filter component. Applications that do not have this setting will not be protected by the Agent and may malfunction or become unusable if deployed on a Application Server where the Agent Realm component is installed.

Together, the Agent Realm and Agent Filter components work in tandem with Identity Server and enforce authentication and authorization for clients trying to access protected J2EE Applications.

Guidelines

The following guidelines will help you use Identity Server Policy Agent for WebLogic most optimally:

- **Use Agent-Based Authentication**

  After the Agent is installed and the application has been configured to use Agent Filter component, the Agent Filter component enforces authentication for all web based access to the protected application's enforced portions. Working in tandem with the Agent Realm component, the Agent Filter ensures that the J2EE policies defined for the protected application get evaluated correctly based on the set role-to-principal mappings, at the same time offering other key services like Single Sign-On etc. Therefore, it is recommended that protected applications do not use their own authentication mechanism or any container based authentication mechanisms which would result in the Agent Filter component being bypassed during the application operation.

- **Create Enhanced Security Aware Applications**

  The Agent provides the rich APIs offered by Sun ONE Identity Server SDK libraries, which are available for use within the protected application. Using these APIs, the application architect can create enhanced security aware applications that are customized to work in the security framework offered by Identity Server. For more information on how to use the Sun ONE Identity Server SDK, refer *Sun ONE Identity Server Programmer's Guide.*

Installing the Agent

The Sun ONE Identity Server Policy Agent for WebLogic Server 6.1 SP2 may be installed on these platforms—Solaris 8, Windows 2000, or HP-UX 11. The installation program for Sun ONE Identity Server Policy Agent for WebLogic Server 6.1 SP2 should be launched according to the following steps as applicable to the respective platform in use. Once the installation program is launched successfully, you may skip to the next section, which details the steps necessary to install the Agent.

Pre-installation Tasks

The following tasks must be performed before installing the Sun ONE Identity Server Policy Agent for WebLogic Server 6.1 SP2:

1. Install the WebLogic Server 6.1 SP2.

   See the WebLogic Server documentation for the necessary details. When the server is installed, test the installation by using the provided sample application to ensure that the server is installed correctly.

2. Test the deployment of application that you intend to protect.

   Before installing the Agent, it is important that you deploy and test the application that must be protected for simple functionality. Once it is established then the application can be deployed successfully, you are ready to install the Agent.

Launching the Installation Program on Solaris 8

The binaries for Sun ONE Identity Server Policy Agent for WebLogic Server 6.1 SP2 for Solaris platform are provided as a tar-gzip archive. Copy this archive to the machine where WebLogic Server is installed. Follow the steps to launch the installation program:

1. Login as root.

2. Unzip the binary archive using the following command:

```
# gzip -dc
j2eeagents-1.0-domestic-us.sparc-sun-solaris2.8.tar.gz | tar xvf
-
```

3. Set your JAVA_HOME environment variable to JDK version 1.3.1 or higher. If your system does not have the required version of JDK, use the JDK supplied with WebLogic Server 6.1 SP2 server. This JDK is located at:

   WebLogic_Install_Dir/bea/jdk131

   The installation program provides two types of interfaces—a graphical user interface (GUI) and a command line interface. In most cases, the GUI installation program can be used for installing the Agent. However, in cases when you are installing the Agent over a telnet session on a remote server and do not have windowing capabilities, then it is recommended that you use the command-line installation program for installing the Agent. You can launch this by executing the following command:

   ```
   # ./setup -nodisplay
   ```

   However, if you choose to use the GUI installation program, then it is required that you set your DISPLAY environment variable to ensure that the GUI installation program window appears on the correct console.

NOTE If you choose to use the command line installation program using the -nodisplay option, you may skip the following step and proceed directly to the section "Installing the Agent Using GUI," on page 22, which details out the installation procedure.

4. Launch the GUI installation program by invoking the setup script as follows:

   ```
   # ./setup
   ```

   - The installation program requires that you set up your JAVA_HOME variable correctly as pointed out in the Step 3. However, if you have incorrectly set the JAVA_HOME variable, the setup script will prompt you for the correct JAVA_HOME value: oblige

     ```
     Enter JAVA_HOME location (Enter "." to abort):
     ```

     Type the full path to the JDK installation directory for launching the installation program. Otherwise, enter a period (.) to abort the installation.

   - In order that the GUI installation program be displayed on your console, the DISPLAY environment variable of your shell must be set correctly. If your DISPLAY environment variable is not set at the time of invoking the setup script, the installation program will prompt you for the DISPLAY environment variable value as follows:

```
Please enter the value of DISPLAY variable  (Enter "." to
abort):
```

Provide the `DISPLAY` value to the installation program by typing the exact value at the above prompt. Otherwise, enter a period (`.`) to abort the installation.

Launching the Installation Program on Windows 2000

The binaries for Sun ONE Identity Server Policy Agent for WebLogic Server 6.1 SP2 for Windows platform are provided as a zip archive. Copy this archive to the machine where WebLogic Server is installed and follow the steps to launch the installation program:

1. You must have administrative privileges when you run the installation program. If you do not have administrative privileges, either log on as "Administrator" user or request such privileges to be granted to your account by the system administrator of the machine or domain as applicable.

2. Unzip the Agent binaries in a convenient location using the Zip utility. This operation results in two executable files `setup.bat` and `setup.exe`, which may be used to launch the installation program. Each of these files provide different features for launching the installation program. You may choose either of these two files depending upon your installation requirements.

   Using setup.bat

   In order to use the `setup.bat` file to launch the installation program, you must have JDK version 1.3.1 or higher. This can be verified by typing the following command in a command prompt window:

   ```
   C:\> java -version

   java version "1.3.1_02"

   Java(TM) 2 Runtime Environment, Standard Edition (build
   1.3.1_02-b02)

   Java HotSpot(TM) Client VM (build 1.3.1_02-b02, mixed mode)
   ```

   If you do not have JDK of required version in your system path, you can use the JDK supplied with WebLogic Server 6.1 SP2 server located at:

   WebLogic_Install_Dir`\bea\jdk131`

   The `setup.bat` may be executed by typing the file name in at the command

prompt window in a directory where it is present, or by double clicking the file in Windows Explorer. For example:

```
C:\>setup.bat
```

The installation program provides two types of interfaces—a graphical user interface (GUI) and a command line interface. By invoking `setup.bat` file from a command prompt window as shown above or by double clicking it in Windows Explorer, you can launch the installation program in the GUI mode. To launch the installation program with the command line argument `-nodisplay`, type the following:

```
C:\>setup.bat -nodisplay
```

Using setup.exe

Using `setup.exe` relieves you from the task of setting up your environment path to include a valid version of JDK. This program first checks your system for the presence of a compatible JDK version and uses the one that was found. However, if no compatible version is found, this program installs the necessary runtime and uses it to launch the installation program.

You can invoke `setup.exe` either from the command prompt or by double clicking the file from Windows Explorer. Unlike `setup.bat`, the `setup.exe` does not take the `-nodisplay` command line argument and can launch only the GUI installation program.

NOTE Since the command line installation program cannot be launched using `setup.exe`, in cases where the command line installation program is required, it is recommended that you use `setup.bat` to launch the command line installation program.

Launching the Installation Program on HP-UX 11

The binaries for Sun ONE Identity Server Policy Agent for WebLogic Server 6.1 SP2 for HP-UX 11 platform are provided as a tar-gzip archive. Copy this archive on the machine where WebLogic Server is installed. Follow the steps to launch the installation program:

1. Login as root.

2. Unzip the binary archive using the following command:

   ```
   # gzip -dc
   j2eeagents-1.0-domestic-us.hppa1.0-hp-hpux11.00.tar.gz | tar xvf
   -
   ```

3. Set your JAVA_HOME environment variable to JDK version 1.3.1 or higher. If your system does not have the required version of JDK, use the JDK supplied with WebLogic Server 6.1 SP2 server. This JDK is located under:

   WebLogic_Install_Dir/bea/jdk131

4. The installation program provides two types of interfaces—a graphical user interface (GUI) and a command line interface. In most cases, the GUI installation program can be used for installing the Agent. However, in cases when you are installing the Agent over a telnet session on a remote server and do not have windowing capabilities, then it is recommended that you use the command line installation program for installing the Agent. You can launch this by executing the following command:

   ```
   # ./setup -nodisplay
   ```

   However, if you choose to use the GUI installation program, then it is required that you set your DISPLAY environment variable to ensure that the GUI installation program window appears on the correct console.

NOTE If you choose to use the command line installation program using the -nodisplay option, you may skip the following step and proceed directly to the next section, which details out the installation procedure.

5. Launch the GUI installation program by invoking the setup script as follows:

   ```
   # ./setup
   ```

   - The installation program requires that you set up your JAVA_HOME variable correctly as pointed out in the Step 3. However, in case you have incorrectly set the JAVA_HOME variable, the setup script will prompt you for the correct JAVA_HOME value:

     ```
     Enter JAVA_HOME location (Enter "." to abort):
     ```

     Type the full path to the JDK installation directory for launching the installation program. Otherwise, enter a period (.) to abort the installation.

- In order that the GUI installation program be displayed on your console, the `DISPLAY` environment variable of your shell must be set correctly. In case your `DISPLAY` environment variable is not set at the time of invoking the setup script, the installation program will prompt you for the `DISPLAY` environment variable value as follows:

  ```
  Please enter the value of DISPLAY variable  (Enter "." to
  abort):
  ```

  Provide the `DISPLAY` value to the installation program by typing in the exact value at the above prompt. Otherwise, enter a period (`.`) to abort the installation.

Installing the Agent Using GUI

The installation program begins with a Welcome screen. Click Next to step through the installation screens and answer the required questions.

1. Read the License Agreement. Click Yes (Accept License) to continue with the Installation.

   If you do not accept all the terms of the Software License Agreement, click No to end the Installation program without installing the Agent.

2. In the Select Installation Directory screen, enter the path where you want to install.

   If you wish to install the Agent in a directory different from the default directory, click the Browse button and choose the directory. Once you have selected the appropriate directory, click the Next button to proceed to the next screen.

NOTE If you select a directory that does not exist on your system, the installation program will prompt you to specify if the new directory should be created. You can either choose to create this new directory by clicking Create Directory button or select a new directory by clicking Choose another Directory button.

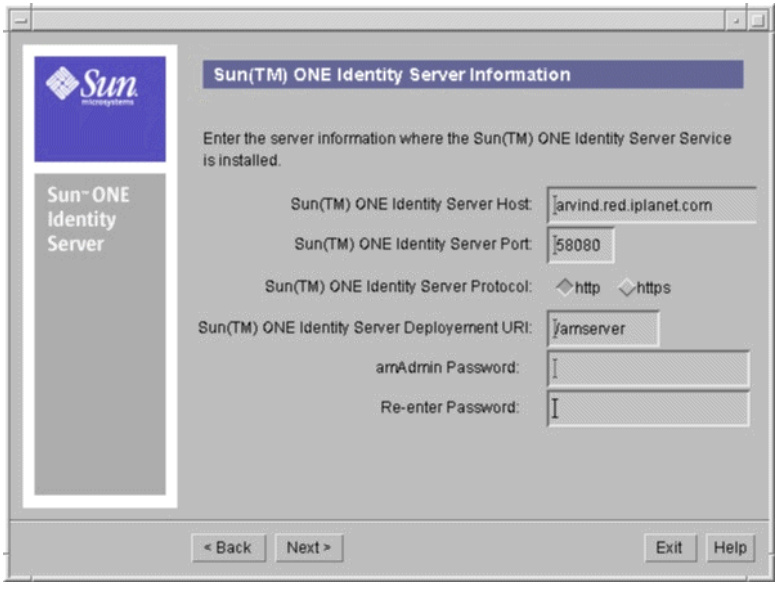
3. In the Select an Agent screen, select the component you wish to install in this by selecting the check box against the component name. The only component available for installation is the Sun ONE Identity Server Policy Agent for WebLogic Server 6.1 SP2, which is selected by default.

Figure 2-1 Component Selection Screen

NOTE For a given system, only one installation of Sun ONE Identity Server Policy Agent for WebLogic Server 6.1 SP2 is allowed at a time. If a previous installation of the Agent has not been removed completely from this system, the selection box will be disabled and no selections can be made. In that case it is recommended that you exit the installation program by clicking the Exit button and remove the old installation completely before starting the install again.

4. In the Sun ONE Identity Server Information screen, provide the following information about the Sun ONE Identity Server and click Next.

Figure 2-2 Sun ONE Identity Server Information Screen



Sun ONE Identity Server Host: Enter the fully qualified host name of the computer system where Sun ONE Identity Server is installed.

Sun ONE Identity Server Port: Enter the port number for the Web Server that runs Sun ONE Identity Server Services.

Sun ONE Identity Server Protocol: Select the protocol that will be used by the Agent to communicate with Sun ONE Identity Server services. This protocol may either be HTTP or HTTPS.

Sun ONE Identity Server Deployment URI: Enter the URI that should be used for accessing Sun ONE Identity Server Services.

amAdmin Password: Enter the password for `amAdmin` user.

Re-enter Password: Re-enter the password for `amAdmin` user for confirmation.

NOTE The password supplied during installation is recorded by the Agent in a secure manner. However, if in the future you change this password in Identity Server, you will have to update the password in the Agent as well. This can be done by using the `agentadmin` tool provided with the Agent. Once the Agent has been installed on the system, the `agentadmin` tool can be invoked from the location: *Agent_Install_Dir*`/SUNWam/wlAgent/bin/agentadmin`

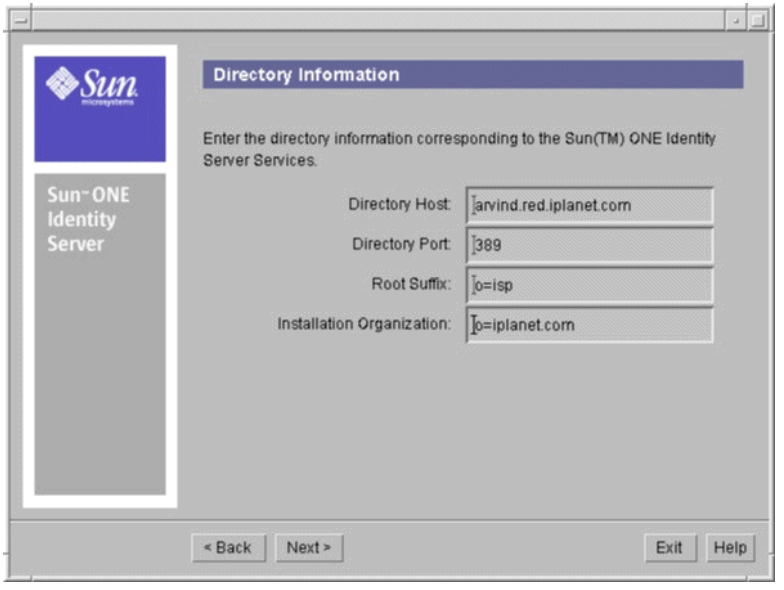
The `agentadmin` tool is available as `agentadmin.bat` on the Windows platform.

To change the password, invoke this tool as follows:

`#./agentadmin -password` *oldpassword newpassword*

5. In the Directory Server Information screen, provide the following information about the Directory Server that is associated with Sun ONE Identity Server services.

Figure 2-3 Directory Information Screen



Directory Host: Enter the fully qualified host name of the computer system where the Directory Server is installed.

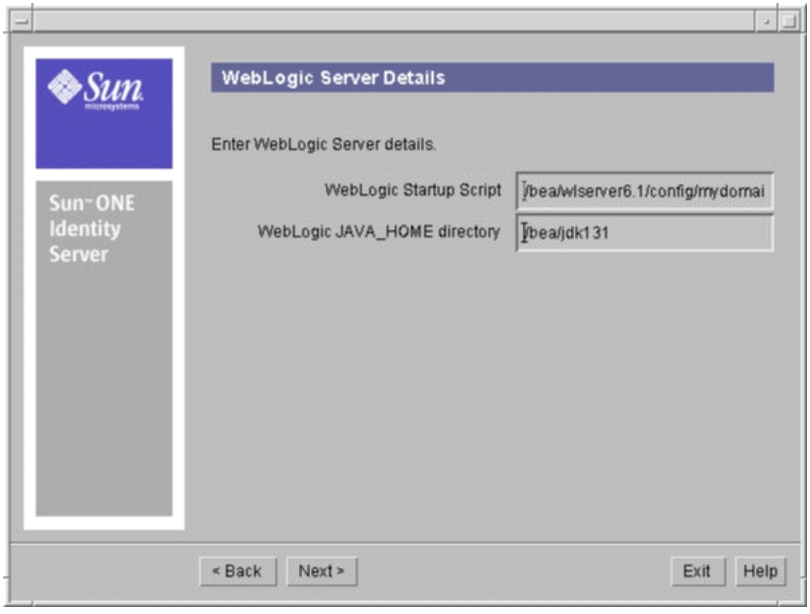
Directory Port: Enter the port number used by the Directory Server.

Root Suffix: Enter the root suffix to be used with this Directory Server.

Installation Organization: Enter the name of the installation organization as used when installing the Sun ONE Identity Server

6. In the WebLogic Server Details screen, provide the following information about the WebLogic Server on which the Agent is installed.

Figure 2-4 WebLogic Server Details Screen



WebLogic Startup Script: Enter the full path to the location of the script used to start the WebLogic Server.The WebLogic Server Startup Script on Solaris platform is a shell script that is used to start the WebLogic Server. On Windows platform, this is a CMD script which serves the same purpose. This script is typically located under the following directory:

WebLogic_Install_Dir`/bea/wlserver6.1/config/`*server-domain-name/*

WebLogic JAVA_HOME directory: Enter the full path to the location of the JDK installation home used by the WebLogic Server. The WebLogic `JAVA_HOME` directory refers to the JDK installation that is used by the WebLogic Server. Typically the value of this is the full path to the following directory:

WebLogic_Install_Dir`/bea/jdk131`

In case you are not sure of the exact location of this directory, open the WebLogic Startup Script that is used for starting the WebLogic Server and locate the value of `JAVA_HOME` variable as specified in this file.

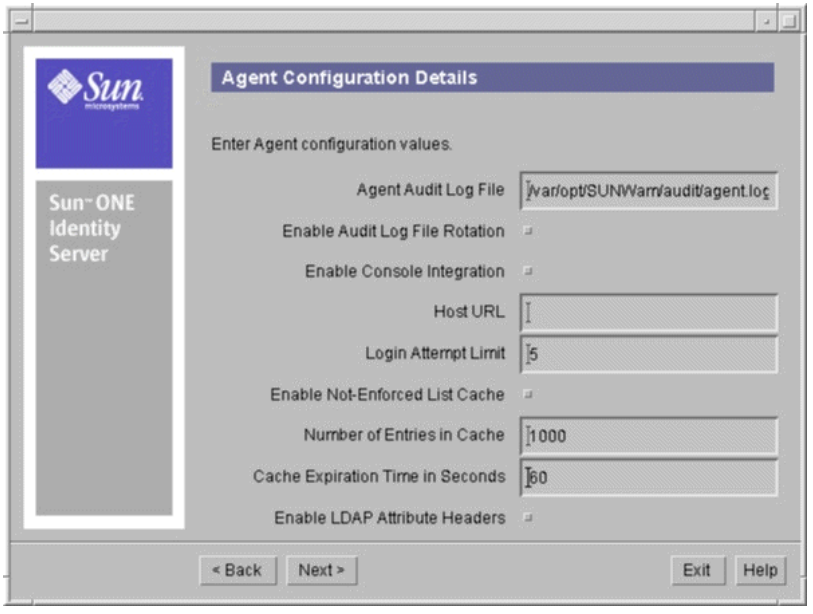
CAUTION

- The installation program modifies the WebLogic Server Startup Script to include certain libraries in the WebLogic `CLASSPATH`, as well as adds certain required parameters to the command that loads the startup classes of WebLogic Server in the Java Virtual Machine. If you specify an incorrect value for the WebLogic Server Startup Script, the necessary classes and parameters will not get added, resulting in malfunction of the Agent, which can render the WebLogic Server unusable. To avoid this problem, ensure that the value you specify for WebLogic Server Startup Script is accurate.
- The installation program adds certain extensions to the JDK used by the WebLogic Server, which are needed by the Agent for successful execution. These extensions are installed in the JDK directory indicated by the above mentioned WebLogic `JAVA_HOME` value. If the value supplied is incorrect or is not the JDK used by the WebLogic Server, it will result in malfunction of the Agent and can render the WebLogic Server unusable. To avoid this problem, ensure that the value you specify for WebLogic `JAVA_HOME` is accurate.

7. In the Agent Configuration Details screen, provide the key configuration information necessary for the Agent to function correctly.

NOTE Read the section “Important Tips” before performing this step.

Figure 2-5 Agent Configuration Screen



Audit Log File: Enter the complete path to the log file to be used by the agent to record Audit messages.

Enable Audit log file rotation: Select this to enable rotation of Audit Log files.

Enable Console Integration: Select this to enable console level integration of Sun ONE Identity Server with WebLogic Server Administration console.

Host URL: Enter a valid URL to be used as the base URL by the Agent to redirect users as necessary. This value may be left blank.

Login Attempt Limit: Enter the number of unsuccessful access attempts in succession after which the user will not be allowed to access the requested URL temporarily for security purposes. Specify the value `0` to disable this feature.

Enable Not-Enforced List Cache: Select this to enable caching of Not-Enforced List evaluation results.

Number of Entries in Cache: Specify the number of entries that the cache can hold at a given instance.

Cache Expiration Time: Specify the time (in seconds) to be used as the maximum time limit for entries that are added in the Not-Enforced List cache.

Enable LDAP Attribute Headers: Select this item to enable the passing LDAP attributes associated with the current user as HTTP Headers.

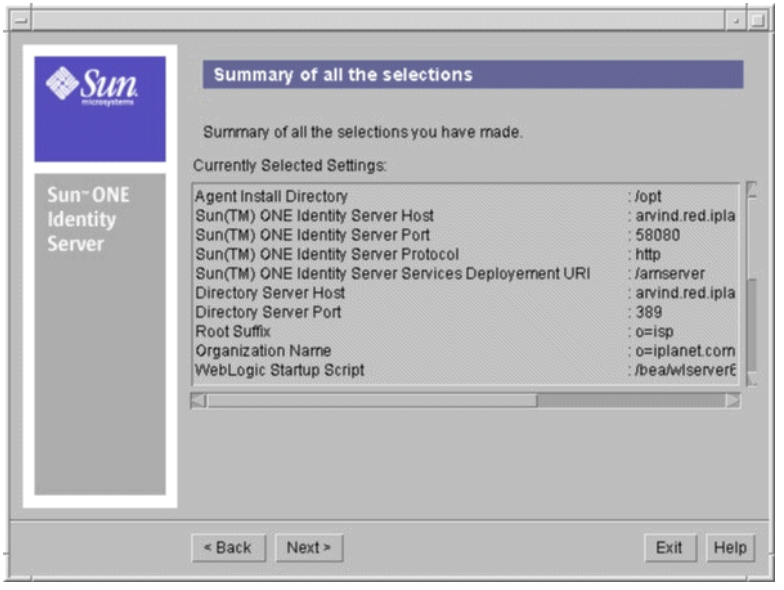
Important Tips

- The Audit Log file is a necessary requirement for the Agent. You may provide the name of a non-existing file on the system to be used as the Audit file. In which case, the Agent creates this file and the necessary directories in its path on first use. Alternatively, you can provide the file name of an existing file which can be used by the Agent as well. However, in either case, it is required that the specified file has write permissions for the WebLogic Server process because the Agent executes in the same process as the WebLogic Server. Providing an incorrect value for this will result in the malfunction of the Agent which can render the WebLogic Server unusable.
- Console Integration implies that the information regarding the names of all Users, Roles, and the Users who are assigned to these Roles in Sun ONE Identity Server will be visible on the WebLogic Server Administration Console. Therefore, this feature must be enabled with caution since the User / Role information will then be accessible to administrators who access the WebLogic Server Console.
- When a request is intercepted by the Agent without sufficient credentials, the Agent redirects the user to the Sun ONE Identity Server's authentication service. Along with this redirect, the Agent also passes information regarding the original request to the authentication services, which is used to redirect the user back to the original requested destination. A part of this request is the Host URL that is used to identify the web container / server which the user was originally trying to access. This Host URL can be specified by setting the Host URL value on this screen. If the Host URL value is left blank, the Agent reconstructs the original Host URL from the request and uses that when redirecting the user to authentication service. The implication of having a Host URL is that no matter which web container / server that the user was originally trying to access, the user will be sent to the specified Host URL after successful authentication. While this configuration value can be used to override the default behavior, an incorrect value may lead to the application becoming inaccessible. It is therefore recommended that this value be left empty unless there is a specific need based on the deployment scenario, which calls for setting this value appropriately.

- The Login Attempt Limit feature can be used to guard the hosted application from Denial-Of-Service attacks where the end user can overload the application server by repeated authentication requests. By disabling this feature, the system remains vulnerable to such attacks. Therefore this feature should not be disabled unless there is a specific requirement that necessitates the disabling of this feature.
- When the Agent must process a large list of Not-Enforced pattern rules specified in the configuration, every incoming request must be evaluated against every such rule to determine if the request can be allowed without authentication or not. In scenarios where the user load is high, the time spent in evaluating these rules can add up and degrade the overall performance of the system. To avoid this problem, it is recommended that Not-Enforced List Cache be enabled.
- Enabling the Not-Enforced List Cache may result in degraded performance if the values for Number of Entries in Cache and Cache Expiration time are not set up appropriately. In case when the cache expiration time duration is more than necessary, the cache will get filled up very fast and new requests will still be evaluated against all the specified pattern rules, resulting in no improvement in performance of the system. If the Number of Entries in Cache is set very high, it may result in excessive consumption of system memory, thereby leading to degraded performance. Therefore, these values must be set up after careful deliberation of the deployment scenario and should be changed as necessary to reflect changing usage scenario of the system. It is recommended that the system be tested with various values of these two parameters in a controlled environment to identify the optimal values, and only then be deployed in production.
- The Agent maintains two caches in its memory—one for recording the URIs that were evaluated as enforced and the other for recording the URIs that were evaluated as not-enforced. The specified values of Number of Entries in Cache and Cache Expiration time is equally applicable to both of these caches. This factor must be considered when setting the values for the size and expiration time of the cache.
- By enabling the LDAP Attribute Headers, for every incoming request, the Agent must retrieve the LDAP Attributes associated with the authenticated user and add them as Header values in the request. This feature should be used only in the case when the deployed application requires these header values for business logic implementation. Turning this feature on without an appropriate need will result in performance degradation of the system that can be otherwise avoided and serves no purpose.

8. In the Summary of all the selections screen, review and verify the installation options that you have specified for the Agent. If you need to make changes, click Back. Otherwise, Next to proceed.

Figure 2-6 Summary of all the selections Screen



9. In the Ready to Install screen, click Install Now button to begin the installation.

10. The Install Progress screen displays the progress of the installation as the installation program makes changes to your system. If necessary, you may interrupt this process by clicking the Stop button.

NOTE It is strongly recommended that you do not interrupt this process as that may lead to a partially installed product, which may also cause problems with uninstall, and may as well render the WebLogic Server unusable. This process should be interrupted only in cases where it is absolutely necessary to do so.

11. In the Installation Summary screen, click Details for a detailed summary of the configuration information that was processed during installation. Click Exit to end the program.

NOTE If the status of the installation is “Failed”, you must view the install log file by clicking the Details button to identify the unsuccessful installation task. In this situation, you may uninstall the Agent and try again after fixing the root cause of failure during this installation.

After the agent is installed, the next step is to configure the WebLogic Server and the deployed application appropriately as explained in the following sections.

WebLogic Server Configuration

Once the Sun ONE Identity Server Policy Agent for WebLogic Server 6.1 SP2 has been installed on your system, the WebLogic Server must be configured to use the Agent Realm provided as a part of the Agent.

Installing the Agent Realm

The Agent Realm is a Custom Security Realm that can be added to the WebLogic Server by using the WebLogic Server Administration Console. This section outlines the steps necessary to successfully add the Agent Realm to the WebLogic Server. It must be noted that the information provided in this section is only to facilitate the installation of Agent Realm and should not be taken as a substitute for the information provided in WebLogic Server documents. For a complete in-depth discussion on WebLogic Custom Realms, refer WebLogic Server documentation center located at `http://www.bea.com`.

In order to install the Agent Realm, the following steps must be performed:

1. Create a Custom Realm for Agent
2. Create a Caching Realm for Agent Realm
3. Configure the File Realm

Create a Custom Realm for Agent

The following steps outline how a new Custom Realm may be created for installing the Agent Realm in WebLogic Server:

1. Logon to the WebLogic Server Administration Console. Use the configured system username and password for logging on to the console.

2. In the left pane of the WebLogic Server Administration Console, expand the Security Node by clicking on the "+" sign.
3. In the left pane under Security node, click the "Realms" item to display a list of available Realms in the system on the right pane.
4. On the right pane, click "Configure a new Custom Realm" link. This displays a form that can be used to enter the information regarding the new Custom Realm that you are trying to create.
5. In this form, enter the following information:

   Name: Agent Realm

   Realm Class Name: `com.iplanet.amagent.weblogic.realm.AgentRealm`
6. Once this information has been entered, click the Create button.
7. After creating the new Realm, restart the WebLogic Server.

Once the WebLogic Server is restarted, using the Administration Console, go to the Security > Realms node. You should be able to see the newly created Agent Realm in the list of Realms displayed on the right hand pane.

Create a Caching Realm for Agent Realm

Follow these steps to create a new Caching Realm for installing the Agent Realm in WebLogic Server:

1. Logon to the WebLogic Server Administration Console. Use the configured system username and password for logging on to the console.
2. In the left pane of the WebLogic Server Administration Console, expand the Security Node by clicking on the "+" sign next to it.
3. In the left pane under Security node, click the Caching Realms to display a list of available Caching Realms in the system on the right pane.
4. On the right pane, click the "Configure a new Caching Realm" link. This displays a form that can be used to enter the information regarding the new Caching Realm that you are trying to create.
5. In this form, enter the following information:

   Name: Agent Caching Realm

   Basic Realm: Select Agent Realm from the pull down menu.

6. Click on the Create button. This refreshes the right pane and a new Caching Realm is created. The right pane now displays the configuration of this newly created Caching Realm.
7. In the right pane, disable all the caching attributes. This can be done as follows:
    a. Click the ACL Tab. This displays the ACL caching attributes.
    b. Uncheck the check box next to Enable ACL Cache
    c. Click on the Apply button.
    d. Repeat this process for the rest of the available tabs—Authentication, Groups, Users, and Permissions. Uncheck the appropriate Enable Cache check box and click the Apply button.
8. Restart the WebLogic Server.

Once the WebLogic Server is restarted, using the Administration Console, go to the Security > Caching Realms node. You should be able to see the newly created Agent Caching Realm in the list of Caching Realms displayed on the right pane.

Configure the File Realm

Once the Agent Caching Realm has been created, the WebLogic Server must be configured to use this new Caching Realm. This is done by configuring the File Realm. The following steps outline how the File Realm may be configured for this purpose:

1. Logon to the WebLogic Server Administration Console.
2. On the left pane click the Security node. This forces the console to display the Security configuration of the WebLogic Server in the right hand pane.
3. In the right pane, click the Filerealm tab. This causes the console to display the details of the current File Realm.
4. In the form that is displayed on the right pane, under Caching Realm, select the Agent Caching Realm from the pull down menu.
5. Click the Apply button.
6. Restart the WebLogic Server.

Once the File Realm is configured and the WebLogic Server is restarted, the Agent Realm has been successfully installed.

NOTE Once the Agent Realm has been successfully configured, it is recommended that you create a backup of the WebLogic Server's `config.xml` file. You may name this backup as `config.xml.withAgent`, so that the next time you install the Agent, you can simply copy this file on top of your existing `config.xml` and bypass the manual steps necessary to install the Agent Realm.

Troubleshooting the Installation

If after configuring the File Realm, the WebLogic Server does not start up correctly, the following could be the reasons:

- **The WebLogic Server Startup File was not modified successfully by the Agent Installation.**

  This can happen if you have modified the WebLogic Server startup file before installing the Agent, or if the installation program did not have sufficient permissions to modify this file. In either case, refer Appendix A to manually modify the startup file in order to recover from this problem.

- **The Agent installation program was unable to install the required extensions for the JDK being used by WebLogic Server.**

  To verify this, first look at the WebLogic Server startup file to determine the exact location of JDK used by the WebLogic Server. This can be inferred from the value of `JAVA_HOME` as set in the WebLogic Server startup file. Using this value as the JDK directory, ensure that it has the necessary extensions installed in it. Refer Appendix A to manually install the extensions if necessary in order to recover from this problem.

If the above reasons do not explain why the WebLogic Server did not startup correctly, it implies that there has been a serious installation error, which has resulted in the WebLogic Server being unusable. To recover from this problem, uninstall the Agent from your system. This should bring the WebLogic Server back to the state that it was in before the installation of the Agent.

Application Configuration

The Agent Realm component of Sun ONE Identity Server Policy Agent for WebLogic 6.1 SP2 provides runtime mapping of various principals in Identity Server. Abstract security role names are used by the hosted application in order to determine if the currently authenticated user is authorized to access a particular resource or is otherwise a member of a given role. This runtime evaluation can occur only if the user is authenticated as an Sun ONE Identity Server principal by the means of Identity Server's authentication service. Without the user being authenticated appropriately, the results of such evaluations done by the Agent Realm will always be negative, resulting in access being denied to the user for the requested resource.

It is the Agent Filter component that enforces authentication for users who try to access particular application resources, thereby enabling the Agent Realm component to correctly evaluate the principal mappings as desired.

Unlike the Agent Realm component which is installed in the core of WebLogic Server, the Agent Filter is installed in the deployed application which must be protected by Identity Server. This is true for every application that must be protected on the WebLogic Server using the Agent. It is recommended that applications that are not protected using the Agent should not be deployed on the WebLogic Server on which the Agent Realm has been installed. This is to ensure that such applications can independently enforce their own security requirements as necessary. The presence of Agent Realm will interfere with the security evaluations done by such applications resulting in their malfunction.

Installing the Agent Filter Component in an Application

The Agent Filter can be installed by simply modifying the deployment descriptor of the application that needs to be protected. The following steps outline the process to install the Agent Filter component for a given application:

1. If the application is currently deployed on the WebLogic Server, it must be removed using the WebLogic Server's Administration Console or by the use of WebLogic Server's deployment tools.
2. It is recommended that you create a backup of the deployment descriptor that will be edited in order to install the Agent Filter in this application.

3. Edit the application's `web.xml` deployment descriptor. Since the Filters were introduced in Servlet Specification 2.3, the `web.xml`'s `DOCTYPE` element must be changed to reflect that the deployment descriptor is a Servlet 2.3 compliant deployment descriptor. This can be done by setting the `DOCTYPE` element as:

   ```
   <!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
   Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
   ```

4. Once the `DOCTYPE` element has been changed, add the Filter elements in the deployment descriptor. This can be done by specifying the Filter element and the Filter-mapping element in the `web.xml` deployment descriptor immediately following the description element of the web-app element. The following is a sample `web.xml` with the Filter and Filter-mapping elements.

```
<web-app>
  <display-name>...</display-name>
  <description>...</description>

  <filter>
  <filter-name>Agent</filter-name>
  <display-name>Agent</display-name>
  <description>SunTM ONE Idenitity Server Policy Agent for WebLogic 6.1
SP2</description>
  <filter-class>com.iplanet.amagent.weblogic.filter.AgentFilter</filter-class>
  </filter>
  <filter-mapping>
  <filter-name>Agent</filter-name>
  <url-pattern>/*</url-pattern>
  </filter-mapping>
  ...
  ...
</web-app>
```

5. Once the `web.xml` deployment descriptor has been modified to reflect the new `DOCTYPE` and filter elements, the Agent Filter has been added to the application.

Creating Role-to-Principal Mappings

Once the application has been configured to have the Agent Filter component in it, the Agent Filter will enforce authentication, thereby enabling the Agent Realm to successfully resolve the role-to-principal mappings. However, these mappings must first be created in order that the hosted application may use them during runtime.

The following are two ways to create such mappings:

- Editing the WebLogic Server specific deployment descriptors.

  The WebLogic Server specific deployment descriptors may be edited to create the role to principal mappings. These descriptors are in `weblogic.xml` and `weblogic-ejb-jar.xml` files. Refer WebLogic Server reference documentation to learn the details of how these descriptors may be edited to create the role to principal mappings. Alternatively, you can refer Appendix C for sample descriptors, which create such mappings.

- Using the WebLogic Server Administration Console.

  The WebLogic Server administration console allows you to create the role to principal mappings by editing the deployed application's deployment descriptors on the fly. In order to use this facility, the application must be deployed and the WebLogic Server must be up and running at that time. Refer WebLogic Server documentation on how to use the Administration console to create such mappings for deployed applications.

Application Specific Agent Configuration

Oftentimes, the deployed applications are partitioned into public and protected parts, which have varying access restrictions. In most cases, the public portions of the application are accessible to anonymous users, whereas the protected portions of the application are accessible only to the registered users. The Agent can be configured to allow this type of access by letting anonymous users access the public portions of the application without requiring that they authenticate using Identity Server's authentication service. This information is provided as a part of the Agent's configuration properties file that is present in the following location:

Agent_Install_Dir`/wlAgent/amAgent/config/AMAgent.properties`

This file may be edited to provide general as well as application-specific configuration for the Agent.

NOTE

- The properties specified in the `AMAgent.properties` file are required for the Agent to function properly. Invalid values specified in this file can lead to malfunction of the Agent, application becoming inaccessible, or the entire system to become unusable. It is recommended that you use extreme caution when modifying the values in this file and always create backups before such modifications to ensure that you can back-out your changes to restore the system to its original state.
- The properties specified in the `AMAgent.properties` file are loaded during the WebLogic Server startup time. Any changes made to this file while the WebLogic Server is running will not take effect until the server has been restarted.

Providing Application-Specific Not-Enforced List

In order to allow anonymous users to access portions of the application, the `AMAgent.properties` file must include an entry for the specific application. The entry that specifies this property is called the application not-enforced list and is identified by the string:

`com.iplanet.amagent.config.filter.`*AppName*`.notEnforcedList[`*index*`]=`*pattern*

This property requires that you format it correctly in order that it can be used by the Agent during runtime. The entries appearing in italics within this string should be replaced by their appropriate values as follows:

AppName: This string should be replaced by the deployed application's context path without its leading forward-slash character. The context path is the first URI segment that is used to identify which application the user is trying to access. For example, if the user accesses your application by typing the URL

`http://myserver.mydomain.com/SomeApp/index.html`, or

`http://myserver.mydomain.com/SomeApp/SomeModule/doSometing.jsp`

then, in both these cases, the *AppName* is `SomeApp`.

index: This is an integer value starting from 0 for every deployed application and is unique for every entry in the application not-enforced list. For example, the following are two entries of application not-enforced list for an application with context path `/SomeApp`:

```
com.iplanet.amagent.config.filter.SomeApp.notEnforcedList[0]=/SomeApp/public/*
```

```
com.iplanet.amagent.config.filter.SomeApp.notEnforcedList[1]=/SomeApp/images/*
```

pattern: This is a pattern string that will be matched with the incoming request to evaluate if the request should be allowed to pass without enforcing authentication or not. The pattern string could be a specific URI, for example `/SomeApp/public/RegistrationServlet`, or could be a generic pattern using the wild card character '`*`' that can be used for denoting 0 or more characters in the request URI; for example `/SomeApp/public/*` will match with any URI that begins with `/SomeApp/public/`.

Using this property, you could specify none or many pattern strings and URIs that the Agent will treat as not-enforced. In other words, user requests that match these particular patterns will be allowed to pass through without enforcing authentication.

Providing Application Specific Access Denied URI

In cases when the Login Attempt Limit is enabled (refer "Installing the Agent," on page 17 for details on how this feature is configured, or refer "Agent Configuration," on page 44 for details on this feature), the Agent is required to block the user's access under certain circumstances. The default behavior of the Agent in this situation is to send an HTTP Status Code 403 Forbidden. In such a situation, the web container can display its preconfigured Forbidden page or simply send the status code in which case the user's browser displays the details of the error message in its own manner. While this is the default behavior of the Agent, it can be changed to suit the needs of the application by allowing the Agent to use an application-specific URI that will be used as the access denied error page.

This can be done by setting the following property in `AMAgent.properties` file:

`com.iplanet.amagent.config.filter.`*AppName*`.accessDeniedURI=/`*URI to use*

This property requires that you format it correctly in order that it can be used by the Agent during runtime. The entries appearing in italics within this string should be replaced by their appropriate values as follows:

AppName: This string should be replaced by the deployed application's context path without the leading forward-slash character (/). The context path is the first URI segment that is used to identify which application the user is trying to access. For example, if the user accesses your application by typing the URL

`http://myserver.mydomain.com/SomeApp/index.html`, or

`http://myserver.mydomain.com/SomeApp/SomeModule/doSometing.jsp`

then, in both these cases, the *AppName* is `SomeApp`.

URI to use: is an application specific URI that the Agent will use to locate the display page for blocking the user request. This URI can be a static HTML page, or a JSP or even a Servlet. However, this URI must be a part of the application itself. In other words, this URI must begin with */AppName/rest of the URI*.

Special Case: Default Web Application

A default web application in WebLogic Server is accessible without providing any context path in the request URI. For example, the following URL is that of a Default web application:

```
http://myserver.mydomain.com/index.html
```

The above URL does not have an associated context path.

For such applications, the Agent provides a convenient means of identifying that an entry is specific to the default web application. This is done in two steps as follows:

1. The following property is set to a name that represents the default web application:

   ```
   com.iplanet.amagent.config.filter.defaultWebAppName=
   DefaultWebApp
   ```

2. This name is then used to specify the application not-enforced list as well as the application's access denied URI as follows:

   ```
   com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList[
   0]=/index.html
   ```

   ```
   com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList[
   1]=/about.html
   ```

   ```
   com.iplanet.amagent.config.filter.DefaultWebApp.accessDeniedURI=
   /URLAccessDenied.html
   ```

Using this scheme, the default web application that does not have a context path associated with it, may be configured just like any other application that has a context path. The same rules apply to the default web application for specifying the not-enforced list entries and access-denied URI as are applicable for the rest of the applications. However, the only difference is that the access-denied URI of the default web application as well as the not-enforced list entries cannot begin with the `/DefaultWebApp/` path segment since such a path segment does not exist on

the application server in reality. The *AppName* in this case where the actual context path is an empty string, is only provided as a convenience to specify the properties associated with the default web application and should not be used in specifying their values.

Global Agent Configuration

The `AMAgent.properties` file provides a way to specify a global not-enforced list, which will be applicable to all the protected applications that are deployed on the server. This list is specified by using the following property:

`com.iplanet.amagent.config.filter.global.notEnforcedList[`*index*`]=` *pattern*

pattern is either an exact URI or a pattern specified by using the wild card character '*' that can be substituted for zero or more characters in the request URI.

index is an integer value starting from 0 and unique for every entry.

Not-Enforced List Usage Considerations

Although the use of Not-Enforced List can be extremely helpful in partitioning your application for public and protected domains, it can also lead to undesirable effects if not used appropriately.

For example, if a request URI that represents a Servlet is matched by some not-enforced list pattern, then the Agent Filter will not enforce authentication for users who try to access that particular Servlet. However, consider the case where this Servlet accesses an Enterprise JavaBeans component that is protected by the Agent using role to principal mapping. In such a case, since the user is not authenticated, the access to the protected component will result in a security violation exception being generated by the application server. Therefore, before an entry is added to the not-enforced list, it must be ensured that it does not in any way cover a resource that may be protected or may try to access a protected resource.

Another interesting aspect of the use of non-enforced list are the images. Typically in a web page there are many images for various purposes like buttons, place holders, banners, and logos. Every time the user accesses this page, the browser issues a request to the application server to get the images contained in this page. Each of such requests are treated as individual requests coming from the client and goes through the same evaluation mechanism for authentication and not-enforced list check as does any other request. This results in one client generating multiple

calls to the server for displaying a single page. Considering the overhead involved in enforcing authentication for every such request, it can impact the overall performance of the system. A solution to this problem is to have a global not-enforced list entry or entries that match all images.

For example:

```
com.iplanet.amagent.config.filter.global.notEnforcedList[0]=*.gif
```

```
com.iplanet.amagent.config.filter.global.notEnforcedList[1]=
/images/*
```

This indicates that any request URI that ends with `.gif` will be not enforced and nor will be any URI that begins with `/images/`. In heavy user load situations, this can significantly increase the performance of the system.

Agent Configuration

The core configuration needed by the Sun ONE Sun ONE Identity Server Policy Agent for WebLogic Server 6.1 SP2 is provided in the `AMAgent.properties` file located in the following directory:

Agent_Install_Dir`/wlAgent/amAgent/config`

This property file provides many configuration settings that can be modified in order to customize the Agent's operation for your deployment scenario.

NOTE Before proceeding, it is important to note that this file and the information within it are critical for the operation of the Agent. It is strongly recommended that you always create backup of this file before modifying it. Also it is strongly recommended that you do not modify this file unless it is absolutely necessary. Note that invalid data entries present in this file can lead to the malfunction of the Agent, malfunction of the deployed applications, and could render the entire system unusable.

The settings provided in this file can be classified into the following categories:

- Common Configuration
- Audit Configuration
- Realm Configuration
- Global Filter Configuration
- Application Filter Configuration

- Debug Engine Configuration

Following sections detail the settings for each of these classifications.

Common Configuration

Settings in this section are general settings that affect the behavior of the Agent as a whole.

Organization Name

Key: `com.iplanet.amagent.config.org`

Description: This property specifies the organization name to be used when searching for principals in Sun ONE Identity Server.

Valid Values: A string that represents the organization name in Sun ONE Identity Server. This property is set during Agent installation and need not be changed unless absolutely necessary.

Example: `com.iplanet.amagent.config.org=iplanet.com`

Root Suffix

Key: `com.iplanet.amagent.config.rootsuffix`

Description: This property specifies the root suffix to be used when searching for principals in Sun ONE Identity Server

Valid Values: A string that represents the root suffix in Sun ONE Identity Server. This property is set during Agent installation and need not be changed unless absolutely necessary.

Example: `com.iplanet.amagent.config.rootsuffix=o=isp`

People Container Level

Key: `com.iplanet.amagent.config.realm.peopleContainerLevel`

Description: This property specifies the people container level to be used when searching for principals in Sun ONE Identity Server.

Valid Values: Non-zero unsigned integer representing the People Container Level in Sun ONE Identity Server, which may be used when searching for principals. This property is set during Agent installation and need not be changed unless absolutely necessary.

Example: `com.iplanet.amagent.config.realm.peopleContainerLevel=1`

Audit Configuration

These settings are exclusively used to configure the Audit Engine used by the Agent.

Language Code

Key: `com.iplanet.amagent.config.audit.localeLanguageCode`

Description: This property specifies the Locale for Audit log messages.

Valid Values: The `localeLanguageCode` must be a valid ISO Language Code. Default value of this property is `en`

NOTE For more information, refer ISO 639 specification at

`http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt`

Example: `com.iplanet.amagent.config.audit.localeLanguageCode=en`

Country Code

Key: `com.iplanet.amagent.config.audit.localeCountryCode`

Description: This property specifies the Locale for Audit log messages.

Valid Values: The `localeCountryCode` must be a valid ISO Country Code. The default value of this property is `US`

NOTE For more information, refer ISO 3166 specification:

`http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html`

Example: `com.iplanet.amagent.config.audit.localeCountryCode=US`

Audit Log File

Key: `com.iplanet.amagent.config.audit.logfile.name`

Description: This property specifies the Audit log file to be used for recording Audit messages.

Valid Values: A string that represents the complete path name of the file to be used by Agent to record Audit messages.

NOTE

- Be sure that the WebLogic Server process has sufficient permissions to write to this file.
- Invalid value specified for this property may result in the failure of the system to start up correctly.

Example:
`com.iplanet.amagent.config.audit.logfile.name=/audit/agent.log`

Audit Log File Rotation Flag

Key: `com.iplanet.amagent.config.audit.logfile.rotate`

Description: This property specifies if the Audit log file should be rotated by the Agent.

Valid Values: true / false. The default value of this property is `false` and should be changed as necessary.

Example: `com.iplanet.amagent.config.audit.logfile.rotate=false`

Audit Log File Rotation Size

Key: `com.iplanet.amagent.config.audit.logfile.rotate.size`

Description: This property specifies the approximate size of the Audit log file in bytes, which should be used to evaluate when the log file needs to be rotated.

Valid Values: Non-zero unsigned integer indicating the size in bytes to be used to evaluate when the log file needs to be rotated. The default value of this property is `52428800` bytes (~ 50 MB) and should be changed as required.

Example:
`com.iplanet.amagent.config.audit.logfile.rotate.size=52428800`

Realm Configuration

These settings are used to configure the Agent Realm component.

Allow Console Integration Flag

Key: `com.iplanet.amagent.config.weblogic.allowConsoleIntegration`

Description: This property specifies if the Agent Realm should allow console level integration of Sun ONE Identity Server with WebLogic Server.

Valid Values: `true/false`

NOTE

- When set, the result of console level integration is that the WebLogic Server Administrator will be able to see the list of Sun ONE Identity Server principals in WebLogic Server Console.
- Since this setting enables the WebLogic Server Administrator to see Sun ONE Identity Server principals in WebLogic Server console, it should be used with caution.
- The default value of this setting is false and should be changed as necessary.

Example:
`com.iplanet.amagent.config.weblogic.allowConsoleIntegration=true`

Global Filter Configuration

These settings are used to configure the Agent Filter component.

SSO Token Name

Key: `com.iplanet.amagent.config.filter.ssoTokenName`

Description: This property specifies the name of the Cookie that represents SSO Token.

Valid Values: A string that represents the name of SSO Token Cookie issued by Sun ONE Identity Server authentication service. This property is set during Agent installation and need not be changed unless absolutely necessary.

Example:
`com.iplanet.amagent.config.filter.ssoTokenName=iPlanetDirectoryPro`

Login URL

Key: `com.iplanet.amagent.config.filter.loginURL`

Description: This property specifies the login URL to be used by the Agent to redirect incoming users without sufficient credentials to the Sun ONE Identity Server authentication service.

Valid Values: A string that represents the complete URL to be used as the redirect URL in order to send users without sufficient credentials to Sun ONE Identity Server authentication service. This property is set during Agent installation and need not be changed unless absolutely necessary.

Example:
`com.iplanet.amagent.config.filter.loginURL=http://myserver.mydomain.com:58080/amserver/login`

Host URL

Key: `com.iplanet.amagent.config.filter.hostURL`

Description: This property specifies the host URL to be used by the Agent to reconstruct the request issued by the browser. This value is used by Sun ONE Identity Server Authentication service to redirect the user back to the original request destination after successful authentication.

Valid Values: A string value that represents the host URL that the user is expected to access in order that the Agent can intercept. This value must match the format of this property. The format of this property value is as follows:

protocol`://`*hostname*`.`*optional-sub-domain*`.`*domain*`:`*port*

where:

protocol can be http or https.

hostname`.`*optional-sub-domain*`.`*domain* is the fully qualified host name that the user is expected to access in order that the Agent may intercept.

port is the port number on which the receiving web server is listening.

If left unspecified, the Agent will try to reconstruct the host URL from the request.

Example:
`com.iplanet.amagent.config.filter.hostURL=http://www.iplanet.com:80`

Login Attempt Limit

Key: `com.iplanet.amagent.config.filter.loginAttemptLimit`

Description: This property specifies the number of login attempts that a user can make using a single browser session.

Valid Values: Unsigned integer value, including 0, which indicates the number of login attempts allowed for any user trying to gain access to protected resources.

NOTE
- This option can be disabled by setting the value to 0
- The default value of this property is 5

Example: `com.iplanet.amagent.config.filter.loginAttemptLimit=5`

Login Counter Cookie Name

Key: `com.iplanet.amagent.config.filter.loginCounterCookieName`

Description: This property specifies the name of the cookie that will be used to track the number of unsuccessful login attempts made by the user.

Valid Values: A string that represents the name of the cookie to be issued by Agent in order to track the number of unsuccessful login attempts made by the user. This property is set during Agent installation and need not be changed unless absolutely necessary.

Example:
`com.iplanet.amagent.config.filter.loginCounterCookieName=iPlanetLoginAttemptID`

Not-Enforced-List Cache Enable Flag

Key: `com.iplanet.amagent.config.filter.notEnforcedList.cache`

Description: This property specifies if the requests URIs that are evaluated as enforced or not-enforced may be cached to increase performance of the system.

Valid Values: `true/false`. The default value of this property is `true`.

Example:
`com.iplanet.amagent.config.filter.notEnforcedList.cache=true`

Not-Enforced-List Cache Size

Key: `com.iplanet.amagent.config.filter.notEnforcedList.cacheSize`

Description: This property specifies the number of entries that will be kept in the cache of not-enforced URIs and enforced URIs by the Agent.

Valid Values: Non-zero unsigned integer indicating the number of enforced as well as not enforced request URIs to be cached during runtime.

NOTE

- Values that are valid but not suited for the deployment scenario may result in degradation of system performance.
- The value for optimal system performance will depend on the type of application deployed, the number of possible request URIs in the deployed application, the user load on the system, the expiration time set for cache entries and a host of other deployment specific factors.
- To determine the most optimal value of this property, the application should be load tested in a controlled test environment before being deployed for production.

Example:
`com.iplanet.amagent.config.filter.notEnforcedList.cacheSize=1000`

Not-Enforced-List Cache Expiration Time

Key: `com.iplanet.amagent.config.filter.notEnforcedList.cacheTime`

Description: This property specifies the amount of time in seconds that will be used to evaluate if a cached entry can be removed from the cache to free up resources for new cache entries.

Valid Values: Non-zero unsigned integer indicating the time in seconds that will be used as the cache expiration time for entries in the cache during cleanup operation.

NOTE

- Values that are valid but not suited for the deployment scenario may result in degradation of system performance.
- The value for optimal system performance will depend on the type of application deployed, the number of possible request URIs in the deployed application, the user load on the system, the expiration time set for cache entries and a host of other deployment specific factors.
- To determine the most optimal value of this property, the application should be load tested in a controlled test environment before being deployed for production.

Example:
`com.iplanet.amagent.config.filter.notEnforcedList.cacheTime=60`

LDAP Attribute Header Enable Flag

Key: `com.iplanet.amagent.config.filter.enableLDAPAttributeHeaders`

Description: This property specifies if the Agent should populate the `HttpServletRequest` with LDAP Attributes associated with the currently authenticated user.

Valid Values: `true/false`. The default value of this property is `false` and should be changed as necessary.

Example:
`com.iplanet.amagent.config.filter.enableLDAPAttributeHeaders=true`

LDAP Attribute Header Map

Key: `com.iplanet.amagent.config.filter.ldapAttribute[`*attr-name*`]`

Description: This property specifies the LDAP Attributes to be populated under specific header names for the currently authenticated user.

Valid Values: Valid values must comply with the syntax of this property. The specified LDAP Attribute should be a valid attribute. The specified HTTP Header name should conform to HTTP Header name conventions. The format for specifying this property is as follows:

`com.iplanet.amagent.config.filter.ldapAttribute[`*attr-name*`]=`
header-name

where *attr-name* is the name of the LDAP Attribute to be looked up for the authenticated user, and *header-name* is the name of the Header that will be used to store this value.

NOTE

- Be sure that the specified Header Names do not conflict with existing Header names.
- Any number of such properties may be specified as long as they are valid properties conforming to the above stated requirements.

Example:

```
com.iplanet.amagent.config.filter.ldapAttribute[cn]=CUSTOM-Common-Name

com.iplanet.amagent.config.filter.ldapAttribute[ou]=CUSTOM-Organization-Unit

com.iplanet.amagent.config.filter.ldapAttribute[o]=CUSTOM-Organization

com.iplanet.amagent.config.filter.ldapAttribute[c]=CUSTOM-Country
```

```
com.iplanet.amagent.config.filter.ldapAttribute[mail]=CUSTOM-Email

com.iplanet.amagent.config.filter.ldapAttribute[employeenumber]=CUS
TOM-Employee-Number
```

LDAP Date Header Attribute Format String

Key: `com.iplanet.amagent.config.filter.ldapAttributeDateHeaderFormat`

Description: This property specifies the format of Date/Time value to be expected as a result of an attribute lookup. This is required when using the specialized get methods of the `javax.servlet.http.HttpServletReqeust` interface that return Date values for headers.

Valid Values: Valid `java.text.SimpleDateFormat` Time Format Syntax string. For more information, refer URL:

```
http://java.sun.com/j2se/1.3/docs/api/java/text/SimpleDateFormat.ht
ml
```

The default value of this property is set to `EEE, d MMM yyyy hh:mm:ss z` and should be changed as necessary.

NOTE Invalid value of this property may result in runtime exceptions in the application.

Example:

```
com.iplanet.amagent.config.filter.ldapAttributeDateHeaderFormat=
EEE, d MMM yyyy hh:mm:ss z
```

Authentication Session Binding Flag

Key: `com.iplanet.amagent.config.filter.authSessionBinding`

Description: This property specifies if the Agent will enforce session binding with authentication.

Valid Values: `true/false`. The default value of this property is set to `false`.

Example: `com.iplanet.amagent.config.filter.authSessionBinding=false`

SSO Token URL Decode Flag

Key: `com.iplanet.amagent.config.filter.urlDecodeSSOToken`

Description: This property indicates if the SSO Token needs to be URL Decoded by the Agent before it may be used.

Valid Values: `true/false`. The default value of this property is set to `true`.

NOTE Valid but inappropriate value of this property may lead to the application being unreachable by the users.

Example: `com.iplanet.amagent.config.filter.urlDecodeSSOToken=true`

Default Web Application Name

Key: `com.iplanet.amagent.config.filter.defaultWebAppName`

Description: This property specifies a name for the Default Web Application deployed on the application server.

Valid Values: A string consisting of lower case and or upper case letters that can be used as the name for default web application. The default value of this property is `DefaultWebApp`

NOTE This property is necessary if the protected application is deployed as the default web application.

Example:
`com.iplanet.amagent.config.filter.defaultWebAppName=DefaultWebApp`

Global Not-Enforced List

Key: `com.iplanet.amagent.config.filter.global.notEnforcedList[`*index*`]`

Description: This property specifies a list of patterns that can be used to evaluate if the requested URI does not require the protection enforced by the Agent.

Valid Values: Valid values must comply with the syntax of this property. The valid values can be exact URIs or patterns consisting of wild-card character '*' to indicate zero or more characters.The syntax of this property is as follows:

`com.iplanet.amagent.config.filter.global.notEnforcedList[`*index*`]=`*pattern*

index is an integer that starts from 0 and increments for every entry in this property list

pattern is a string that represents request URIs that are not enforced by Agent.

The *pattern* string may consist of wild card character '*', which may match zero or more characters.

The *index* must start from zero for the first entry and continue till the last in a sequence. Missing index values in this list will result in partial or complete loss of list entries.

NOTE No value for this property indicates empty global not-enforced list.

Example:

```
com.iplanet.amagent.config.filter.global.notEnforcedList[0]=*.gif
com.iplanet.amagent.config.filter.global.notEnforcedList[1]=
public/*
com.iplanet.amagent.config.filter.global.notEnforcedList[2]=
/images/*
```

Application Filter Configuration

These settings are used to configure the Agent Filter for a particular application.

Access Denied URI

Key: `com.iplanet.amagent.config.filter.`*AppName*`.accessDeniedURI`

Description: This property specifies the application-specific access denied URI for the protected application.

Valid Values: The URI within the deployed application that must be used as the access denied URI to block in coming requests when necessary.

NOTE

- This property is specific to the protected application. Therefore if there are more than one protected applications deployed on the system, there should be one property for each such application.
- This property must specify a URI that is within the application. Failure to do so can result in runtime internal server errors.
- The format for specifying this property is as follows:

  `com.iplanet.amagent.config.filter.`*`AppName`*`.accessDeniedURI=`*`URI`*

  where *AppName* is the context path name for the deployed application and *URI* is the URI to be used. Note that the difference between *AppName* and context path of the application is that the context path has a leading / character.
- In case, the protected application is the default web application, the *AppName* should be set to the same string as specified in for the value of the property Default Web Application Name.
- In case when this property is not specified for a given application, the Agent uses HTTP Status Code 403 (Forbidden) to indicate a blocked access.

Example:

```
com.iplanet.amagent.config.filter.Portal.accessDeniedURI=/Portal/AccessDenied.html

com.iplanet.amagent.config.filter.BankApp.accessDeniedURI=/BankApp/Block.jsp

com.iplanet.amagent.config.filter.DefaultWebApp.accessDeniedURI=/URLAccessDenied.htm
```

Application Not-Enforced-List

Key:
`com.iplanet.amagent.config.filter.`*`AppName`*`.notEnforcedList[`*`index`*`]`

Description: This property specifies a list of patterns that can be used to evaluate if the requested URI does not require the protection enforced by the Agent for a particular application.

Valid Values: Valid values must comply with the syntax of this property. The valid values can be exact URIs or patterns consisting of wild-card character '*' to indicate zero or more characters.The syntax of this property is as follows:

`com.iplanet.amagent.config.filter.`*AppName*`.notEnforcedList[`*index*`]=` *pattern*

AppName is the context path name without the leading forward-slash character (/) for the deployed application.

index is an integer that starts from 0 and increments for every specified property for the particular application.

pattern is a string that represents the URIs that are not enforced by the Agent.

NOTE

- The *pattern* string may consist of wild card character '*', which may match zero or more characters.
- The *index* must start from zero for the first entry and continue till the last in a sequence. Missing index values in this list will result in partial or complete loss of list entries. The index value will be independent for different *AppName*s specified in this property list.
- In case the protected application is the default web application, the *AppName* should be set to the same string as specified in for the value of the property Default Web Application Name.
- No value for this property indicates empty not enforced list.

Example:

```
com.iplanet.amagent.config.filter.Portal.notEnforcedList[0]=
/Portal/GuestPages/*

    com.iplanet.amagent.config.filter.Portal.notEnforcedList[1]=
    /Portal/Registration/*

    com.iplanet.amagent.config.filter.Portal.notEnforcedList[2]=
    /Portal/WebServices/PollServlet

    com.iplanet.amagent.config.filter.BankApp.notEnforcedList[0]=
    /BankApp/ModuleGuestTour/*

    com.iplanet.amagent.config.filter.BankApp.notEnforcedList[1]=
    /BankApp/index.html

    com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList0
    ]=/index.html
```

```
com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList[
1]=/about.html
```

Debug Engine Configuration

These settings are used to configure the Debug Engine to generate diagnostic information.

Debug Level

Key: `com.iplanet.amagent.config.debug.level`

Description: This property specifies the amount of debug messages that will be emitted by the Agent's Debug Engine.

Valid Values: Any of `0`, `1`, `3`, `7`, `15`, and `31`. These values indicate the following:

`0` = No debugging

`1` = Only Error messages

`3` = Error and Warning messages

`7` = Error, Warning and Brief Informational messages

`15`= Error, Warning and Verbose Informational messages

`31`= Error, Warning and Very Verbose Informational messages

NOTE

- For better performance of the system, this property should be set to a value 0. Values other than that will affect the system performance depending upon the amount of information the Debug Engine has to emit.
- Values other than the ones specified in the Valid Values list above will lead to invalid configuration of the Debug Engine, thereby affecting the volume of messages that will be emitted.

Example: `com.iplanet.amagent.config.debug.level=7`

Debug Log File

Key: `com.iplanet.amagent.config.debug.logfile.name`

Description: This property specifies the Debug log file to be used for recording Debug messages.

Valid Values: A string that represents the complete path name of the file to be used by Agent to record Debug messages.

NOTE

- Be sure that the WebLogic Server process has sufficient permissions to be able to write to this file.
- Invalid or empty value of this property will lead to Debug messages not being logged in the log file.

Example:

```
com.iplanet.amagent.config.debug.logfile.name=/debug/agent_debug.log
```

Debug Log File Rotation Flag

Key: `com.iplanet.amagent.config.debug.logfile.rotate`

Description: This property specifies if the Debug log file should be rotated by the Agent.

Valid Values: `true` / `false`. The default value of this property is `false` and should be changed as necessary.

Example:

```
com.iplanet.amagent.config.debug.logfile.rotate=false
```

Debug Log File Rotation Size

Key: `com.iplanet.amagent.config.debug.logfile.rotate.size`

Description: This property specifies the approximate size of the Debug log file in bytes, which should be used to evaluate when the log file needs to be rotated.

Valid Values: Non-zero unsigned integer indicating the size in bytes to be used to evaluate when the log file needs to be rotated. Default value of this property is `52428800` bytes (~ 50 MB) and should be changed as required.

NOTE This property is not used if the Debug Log File Rotation Flag is set to false

Example:

```
com.iplanet.amagent.config.debug.logfile.rotate.size=52428800
```

Debug Time/Date Format String

Key: `com.iplanet.amagent.config.debug.date.format`

Description: This property specifies the format of time stamp that is used to mark the exact time when the Debug message was recorded.

Valid Values: Valid `java.text.SimpleDateFormat Time Format Syntax` string. For more information, refer URL:

`http://java.sun.com/j2se/1.3/docs/api/java/text/SimpleDateFormat.html`

NOTE

- The default value of this property is set to `<MMM d, yyyy h:mm:ss a z>'`*Agent*`'` and should be changed as necessary.
- Invalid value of this property will result in loss of time stamp data with debug messages.

Example:

```
com.iplanet.amagent.config.debug.date.format=[yyyy/MM/dd HH:mm:ss
zzz]
```

Debug Print STDOUT Flag

Key: `com.iplanet.amagent.config.debug.print.stdout`

Description: This property specifies if the Debug Engine should print the debug messages on Standard Output stream.

Valid Values: `true`/`false`. The default value of this property is `true` and should be changed as necessary.

NOTE

- When set to true, the Debug Engine prints all debug messages on the Standard output stream. This results in the debug messages being displayed on the console window where the WebLogic Server startup script was executed from.
- This property has no affect on the ability of Debug Engine to write to debug log files.

Example: `com.iplanet.amagent.config.debug.print.stdout=true`

Uninstalling the Agent

When you install the Sun ONE Identity Server Policy Agent for WebLogic Server software, an uninstallation program is created in the installation directory. Using this uninstallation program the Agent can be removed completely from your system. While the uninstallation program deletes all the installed files from your system, certain files such as audit log messages are not deleted. You can delete them manually.

The uninstallation program for Sun ONE Identity Server Policy Agent for WebLogic Server should be launched according to the following steps for Solaris, Windows, or HP-UX platform as applicable.

Launching the Uninstallation Program on Solaris 8

The uninstallation program for Solaris platform may be launched by executing the generated uninstall script located in the installation directory. The following steps provide details on how to achieve this:

1. Login as root.
2. Go to the directory where the Agent is installed.
3. Set your `JAVA_HOME` environment variable to JDK version 1.3.1 or higher. If your system does not have required version of JDK, use the JDK supplied with WebLogic 6.1 SP2 server. This JDK is located under:

   WebLogic_Install_Dir`/bea/jdk131`
4. The uninstallation program provides two types of interfaces—a graphical user interface (GUI) and a command-line interface. In most cases, the GUI installer can be used for uninstalling the Agent. However, in cases when you are uninstalling the Agent over a telnet session on a remote server and do not have windowing capabilities, then the GUI uninstallation program cannot be used. In such a case it is recommended that you use the command line uninstallation program for uninstalling the Agent. This can be launched by executing the uninstall script and passing in a command line argument `-nodisplay` as follows:

   ```
   #./uninstall_wlagent -nodisplay
   ```

   However, if you choose to use the GUI uninstallation program, then it is required that you set your `DISPLAY` environment variable to ensure that the GUI uninstallation program window appears on the correct console.

NOTE If you choose to use the command line uninstallation program using the `-nodisplay` option, you may skip the next step and proceed directly to the next section, which details out the uninstallation procedure.

5. Launch the GUI uninstallation program by invoking the uninstall script as follows:

   ```
   # ./uninstall_wlagent
   ```

   The uninstallation program requires that you setup your `JAVA_HOME` variable correctly as pointed out in the Step 3. However, in case you have set the `JAVA_HOME` variable incorrectly, the uninstall script will prompt you for supplying the correct `JAVA_HOME` value:

   ```
   Enter JAVA_HOME location (Enter "." to abort):
   ```

   Type the full path to the JDK installation directory for launching the uninstallation program. Otherwise, enter a period (`.`) to abort the uninstallation.

   In order that the GUI uninstallation program be displayed on your console, the `DISPLAY` environment variable of your shell must be set correctly. In case your `DISPLAY` environment variable is not set at the time of invoking the uninstall script, the uninstallation program will prompt you for the `DISPLAY` environment variable value as follows:

   ```
   Please enter the value of DISPLAY variable  (Enter "." to abort):
   ```

   Provide the `DISPLAY` value to the installer by typing in the exact value at the above prompt. Otherwise, enter a period (`.`) to abort the uninstallation.

Launching the Uninstallation Program on Windows 2000

The uninstallation program for the Windows platform may be launched by executing the generated uninstall script located in the installation directory.

1. You must have administrative privileges when you run the uninstallation program. If you do not have administrative privileges, either login as “Administrator” or request such privileges to be granted to your account by the system administrator of the machine or domain as applicable.
2. Go to the directory where Agent is installed.

3. The uninstall script `uninstall_wlagent.bat` is located in this directory. In order to use the `uninstall_wlagent.bat` script to launch the uninstallation program, you must have JDK version 1.3.1 or higher. This can be verified by typing the following command in a command prompt window:

   ```
   C:\> java -version

   java version “1.3.1_02”

   Java(TM) 2 Runtime Environment, Standard Edition (build
   1.3.1_02-b02)

   Java HotSpot(TM) Client VM (build 1.3.1_02-b02, mixed mode)
   ```

   If you do not have JDK of required version in your system path, you can use the JDK supplied with WebLogic 6.1 SP2 server located at:

   WebLogic_Install_Dir`\bea\jdk131`

   Once this is done, the `uninstall_wlagent.bat` may be executed by typing the file name at the command prompt window in a directory where it is present, or by double clicking the file in Windows Explorer. For example:

   ```
   C:\Sun>uninstall_wlagent.bat
   ```

   The uninstallation program provides two types of interfaces—a graphical user interface (GUI) and a command line interface. By invoking the `uninstall_wlagent.bat` file from a command prompt window as shown above or by double clicking it in Windows Explorer, the uninstallation program is launched in the GUI mode. However, in a case where it is it is required that you use the command line uninstallation program for uninstalling the Agent, the uninstallation program may be launched by executing the `uninstall_wlagent.bat` file and passing in a command line argument `-nodisplay` as follows:

   ```
   C:\Sun>uninstall_wlagent.bat -nodisplay
   ```

   This can be done by typing the above command in a command prompt window in a directory where this file is available.

NOTE

- If the required JDK is configured to be in the system path, the uninstallation program can be launched from the Control Panel Add / Remove programs control. In the list of the installed programs on your system, select Sun ONE Identity Server Policy Agent for WebLogic and click the Change / Remove button. If your system path is not configured with an appropriate JDK version, the uninstall can fail.
- When using the Control Panel Add / Remove programs control to launch the uninstallation program for Agent, the uninstallation program will be launched in GUI mode only. To launch the uninstallation program in command line mode, use the provided uninstall script as mentioned previously.

Launching the Uninstallation Program on HP-UX 11

The uninstallation program for HP-UX platform may be launched by executing the generated uninstall script located in the installation directory. The following steps provide details on how to achieve this:

1. Login as root.
2. Go to the directory where the Agent is installed.
3. Set your `JAVA_HOME` environment variable to JDK version 1.3.1 or higher. If your system does not have JDK of required version, use the JDK supplied with WebLogic 6.1 SP2 server. This JDK is located under:

   WebLogic_Install_Dir`/bea/jdk131`
4. The uninstallation program provides two types of interfaces—a graphical user interface (GUI) and a command-line interface. In most cases, the GUI installer can be used for uninstalling the Agent. However, in cases when you are uninstalling the Agent over a telnet session on a remote server and do not have windowing capabilities, then the GUI uninstallation program cannot be used.

In such a case it is recommended that you use the command line uninstallation program for uninstalling the Agent. This can be launched by executing the uninstall script and passing in a command line argument `-nodisplay` as follows:

```
#./uninstall_wlagent -nodisplay
```

However, if you choose to use the GUI uninstallation program, then it is required that you set your `DISPLAY` environment variable to ensure that the GUI uninstallation program window appears on the correct console.

NOTE If you choose to use the command line based uninstallation program using the `-nodisplay` option, you may skip the next step and proceed directly to the next section, which details out the uninstallation procedure.

5. Launch the GUI uninstallation program by invoking the uninstall script as follows:

   ```
   # ./uninstall_wlagent
   ```

   The uninstallation program requires that you setup your `JAVA_HOME` variable correctly as pointed out in the step number 3. However, in case you have incorrectly set the `JAVA_HOME` variable, the uninstall script will prompt you for supplying the correct `JAVA_HOME` value:

   Enter JAVA_HOME location (Enter "." to abort):

   Specify the `JAVA_HOME` value to the uninstallation program by typing in the full path to the JDK installation directory that should be used for launching the installer. Otherwise, enter a period (`.`) to abort the uninstall.

   In order that the GUI uninstallation program be displayed on your console, the `DISPLAY` environment variable of your shell must be set correctly. In case your `DISPLAY` environment variable is not set at the time of invoking the uninstall script, the uninstallation program will prompt you for the `DISPLAY` environment variable value as follows:

   Please enter the value of DISPLAY variable (Enter "." to abort):

   Provide the `DISPLAY` value to the installer by typing in the exact value at the above prompt. Otherwise, enter a period (`.`) to abort the uninstallation.

Uninstalling the Agent Using GUI

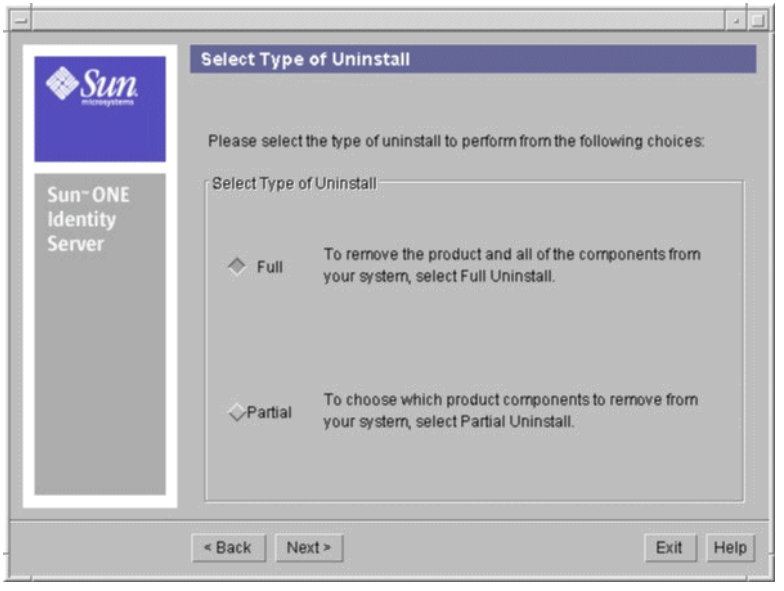
The uninstallation program begins with a Welcome screen. Click Next to step through the uninstallation screens, answering the questions.

1. In the Uninstall Type Selection Screen select the type and click Next.

   Full: Select this type to remove the product and all the components from your system.

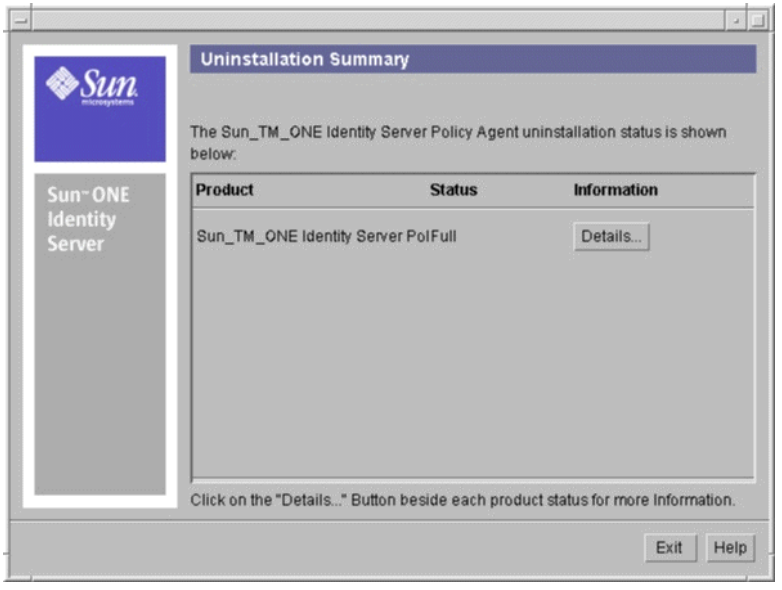
   Partial: Select this type if you want to remove only certain Agent components. Select the component you want to uninstall and click Next.

Figure 2-7 Uninstall Type Selection Screen



2. In the Ready to Uninstall screen, review the uninstallation information. If you need to make changes, click Back. Otherwise, click Uninstall Now.
3. The Uninstall Progress screen displays the progress of uninstall process.
4. In the Uninstallation Summary window, click Details for a detailed summary of the configuration information that was processed during uninstallation. Click Exit to end the program

Figure 2-8 Uninstall Summary Screen



NOTE If the status of the uninstall is “Failed”, you must view the log file for the uninstall by clicking on the Details button to identify which uninstall task failed. In certain situations these failures can be recovered from and the system can be restored to its original state. See the next section for details on how to restore the system to its original state.

Troubleshooting Uninstallation Problems

During the installation, the Agent Installer performs modifications for certain existing files on your system. In order that these modifications can be backed out completely, backups of these files are created before the modifications are made. In case you encounter any failures during the uninstall, it is possible to bring the system back to its original state manually by restoring the backed up files.

The following is a list of files that are backed up during installation and restored during the uninstall of the Agent:

- WebLogic Server Startup Script

- WebLogic config.xml
- java.security file

WebLogic Server Startup Script

The WebLogic Startup script is modified during installation and the backup is created in the same directory as the startup script. The name of the backup file is *WebLogic_Startup_Script_Name*-`preAgent`. For example, if the WebLogic startup script that you specified during installation was:

```
/bea/wlserver6.1/config/examples/startExamplesServer.sh
```

then, the backup file will be:

```
/bea/wlserver6.1/config/examples/startExamplesServer.sh-preAgent
```

WebLogic config.xml

The WebLogic Server's master configuration file, `config.xml`, although is not modified by the installer, is still backed up into the corresponding `config.xml-preAgent` file. This is done because after the Agent is installed, the WebLogic Server must be configured with the Agent Realm component, which results in modifications to the underlying `config.xml` file. If this file is not restored to its original state, the uninstall of Agent will leave the WebLogic Server configured to use the Agent Realm component which would no longer be available on your system. This may result in the WebLogic Server becoming unusable. Thus, when you remove the Agent, the `config.xml` is restored to its original state thereby bringing the WebLogic Server's configuration back to the state it was in before the installation of the Agent.

java.security file

The Agent installer installs the JCE and JSSE extensions to the JDK installation being used by WebLogic Server. These extensions require that the specific providers be entered into the JDK installations security file so that they are available at runtime. Before these providers are entered into the security file, a backup of the security file is created so that it can be restored during the uninstall of the Agent. The backup file is named `java.security-preAgent`, and is located in the JDK installation's `jre/lib/security` directory. For example, if during the install you entered the WebLogic `JAVA_HOME` value as `/bea/jdk131`, the `java.security` file will be:

```
/bea/jdk131/jre/lib/security/java.security
```

and the backup file will be:

```
/bea/jdk131/jre/lib/security/java.security-preAgent
```

Chapter 3

Policy Agent for IBM WebSphere 4.0.3 AE

This chapter provides a brief overview of Sun ONE Identity Server Policy Agent for IBM WebSphere Application Server 4.0.3 and describes how to install and configure the Policy Agent for WebSphere 4.0.3 running on Solaris 8 platform.

Topics include:

- Overview
- Guidelines
- Limitations
- Software Requirements
- Installing the Agent
- WebSphere Application Server Configuration
- Application Configuration
- Agent Configuration
- SSL Configuration
- Uninstalling the Agent
- Troubleshooting Information
- Known Problems

Overview

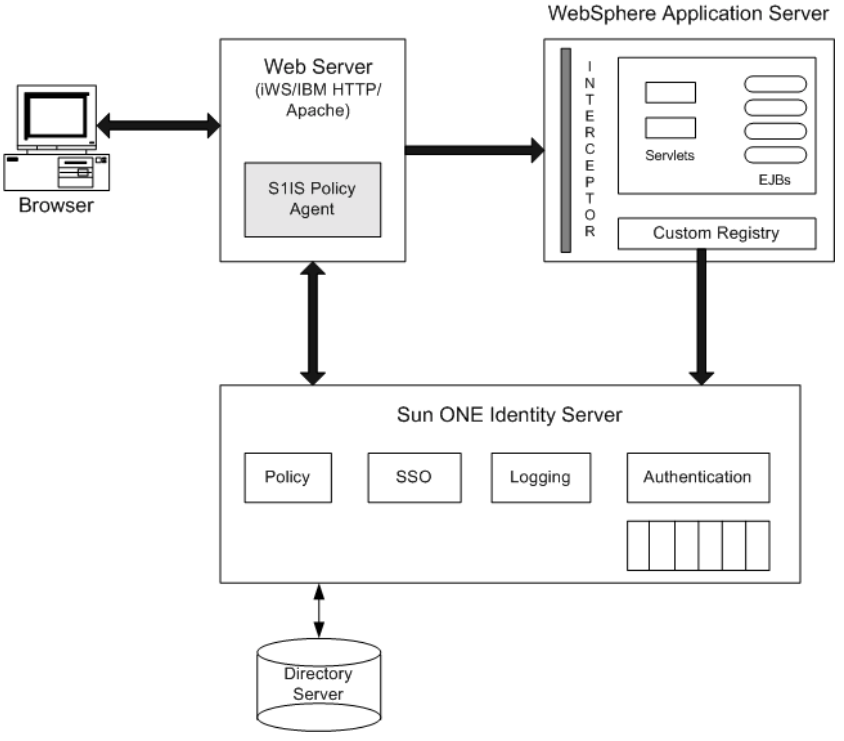
The Identity Server Policy Agent for WebSphere Application Server 4.0.3 is designed to enforce authentication and authorization for clients that interact with the application through the application server's web container. The resources deployed on the WebSphere are protected by a combination of a Web Agent and a J2EE Agent.

The Identity Server Web Agent is used as a Reverse Proxy Security Server (RPSS) to perform the authentication. The RPSS authenticates the HTTP clients and passes authenticated requests to the WebSphere Application Server. The Identity Server Policy Agent is installed on the Web server (IBM HTTP, iWS, or Apache) that authenticates clients for the WebSphere Application Server. The Web Agent is used to perform authentication by redirecting the request for protected resources to the Identity Server Authentication service. The Web Agent also performs coarse grained authorization based on the Allow or Deny URL Policies defined in the Identity Server.

The J2EE authorization model is based on the concept of security roles. A security role is a logical grouping of users defined by an Application Component Provider or Assembler. Application deployer maps roles to security identities (for example, principals and groups) in the user registry. Security roles are used with declarative security (through deployment descriptor) and programmatic security. The WebSphere policy Agent enables the mapping of Identity Server security identities such as Identity Server users and Identity Server roles, to the security roles used within the J2EE Application using the Custom User Registry implementation for WebSphere 4.0.3.

The J2EE role-based authorization is performed by the WebSphere application server. The WebSphere must trust the RPSS before accepting the authentication credentials. For the WebSphere to trust the RPSS, a Web Trust Association (WTA) between WebSphere and RPSS needs to be established. This can be achieved by implementing the Web Trust Association Interceptor (WTAI) for the RPSS. The WTAI implementation establishes the trust by validating SSO Token in the request. On a successful trust validation, the WebSphere runtime uses the WTAI to retrieve principal from the request. The WebSphere runtime uses the identity information obtained from the WTAI to communicate with the user registry. The user registry is the custom registry that communicates with the Identity Server. WebSphere runtime invokes the methods of the user registry to get the set of security identities applicable for the current user. The security identities are used to determine the security roles applicable for the user. This information is used by WebSphere to implement both declarative and programmatic security.

Figure 3-1 Identity Server Policy Agent for WebSphere 4.0.3 Architecture



Guidelines

The following guidelines will help you use the Identity Server Policy Agent for WebSphere most optimally:

- **Agent based Authentication and Authorization**

  After the Agent is installed and the application has been secured, the web agent enforces authentication for all web-based access to the protected application's enforced portions. Working in tandem with the Agent Realm (Custom Registry) component, the Agent Interceptor ensures that the J2EE policies defined for the protected application get evaluated correctly based on

the set role-to-principal mappings, at the same time offering other key services such as Single Sign-On. Therefore, it is recommended that protected applications do not use their own authentication mechanism or any container-based authentication mechanisms.

- **Coarse Grained and Fine Grained access-control**

  The resources deployed on WebSphere are protected by a combination of Web Agent and J2EE Agent. The Web Agent is primarily used to perform authentication by redirecting the request for protected resources to the Identity Server Authentication service. It also performs coarse grained authorization based on the Allow or Deny URL Policies defined in Identity Server.

  The J2EE Agent assists the WebSphere Application server to perform more fine grained access-control for individual servlets, EJBs, or EJB methods based on the security constraints and EJB method permissions defined in the deployment descriptors. The J2EE Agent maps the application defined security roles to the principals defined in Identity Server.

  The resources deployed on WebSphere Application server can be protected using either coarse grained, or fine grained access control, or a combination of both. For example, an employee portal wants to provide read access to any authenticated (regular) employee to its payroll web application, and modification or write access to only certain people. The first requirement can be achieved by defining a Allow URL Policy in the Identity Server for the application URL, applicable for the entire organization. This is coarse grained access-control. The second requirement can be met by defining the security constraint for the servlet or EJB method that performs the update or modification. The security role associated with the constraint is then mapped to privileged users in the Identity Server. Access to the EJBs or servlets is then restricted only to these users. This is fine grained access-control.

- **Create Enhanced Security Aware Applications**

  The Agent provides the rich APIs offered by Identity Server SDK libraries, which are available for use within the protected application. Using these APIs, the application architect can create enhanced security aware applications that are customized to work in the security framework offered by Identity Server. For more information on how to use the Identity Server SDK, refer *Sun ONE Identity Server Programmer's Guide.*

Limitations

- The Policy Agent for WebSphere Application Server 4.0.3 is designed to enforce authentication and authorization for clients that interact with the application through the application server's web container. Typically, such clients are thin clients like web browsers that communicate with the application server's web container using HTTP or HTTPS protocols. If however, the client does not access the application through the HTTP or HTTPS protocols, thereby bypassing the web container of the application server, the Agent Interceptor component will not be in a position to intercept the request. This is a likely case when the application is being accessed by a rich client using other protocols such as RMI over IIOP. Since the Agent Interceptor component is bypassed in such cases, the authentication cannot be enforced. Clients that do not get authenticated by the Agent Interceptor do not possess sufficient credentials for the Agent Realm component to evaluate the necessary J2EE security policies. Therefore, such clients will be denied access to all protected resources. Alternatively, for security aware applications that use the programmatic J2EE security APIs provided in the application server will be given negative results by such APIs.
- The resources deployed on the WebSphere are protected by a combination of Web agent and J2EE Agent. The agent functionality does not work properly (access to protected resources always get declined) when the resources are accessed directly using the web container's internal HTTP transports, thus bypassing the web server.
- WebSphere Server does not start properly when the Directory Server used by the Identity Server is configured for LDAPS.
- When trust association is configured between a WebSphere Application Server and a Reverse Proxy Security Server, access to all protected resources will go through trust association. The Trust Association is bypassed for access to unprotected resources. The WebSphere Application Server does not invoke the Trust Association Interceptor when access is made to unprotected resources. If an unprotected resource accesses a protected resource, the access will always fail, even if the resource was accessed from an authenticated session, which has the roles for accessing the protected resource.

Workaround

All unprotected resources, which access protected resources must be protected with a dummy security constraint. Access to these resources should be limited to any Authenticated User, by mapping the role associated with the dummy security constraint to the special subject “All Authenticated Users.” This will ensure that the Trust association is not bypassed when these resources are accessed. However, with this workaround the unprotected resource no longer remains truly unprotected.

Software Requirements

- Sun ONE Identity Server 5.1
- Web Server with an appropriate Identity Server Policy Agent installed
  - Apache Server 1.3.26
  - IBM HTTP Server 1.3.19
  - iPlanet Web Server, Enterprise Edition 4.1 SP8

Installing the Agent

The Identity Server Policy Agent for WebSphere 4.0.3 AE is supported on Solaris 8 platform.

Pre-Installation Tasks

The following tasks must be completed before the Identity Server Policy Agent for WebSphere can be installed:

1. Install the database for the WebSphere Application Server. For information on installing the database see the documentation at:
   `http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/was/nav_solaris.html`
2. Install one of the following Web Servers:
   - iPlanet Web Server
   - Apache

- IBM HTTP Server (Bundled with WebSphere)

3. Install WebSphere Application Server. See installation details at: `http://www-3.ibm.com/software/webservers/appserv/doc/v40/aeinfocenter/was/nav_solaris.html`

4. Install Fixpacks for WebSphere and IBM HTTP Server.

   - Version upgrade to 4.0.3

     Install Fixpack3 for WebSphere App Server, to upgrade the version from 4.0.1 to 4.0.3. Download Fixpack3 from: `ftp://ftp.software.ibm.com/software/websphere/appserv/support/fixpacks/was40/fixpack3/Sun/`

     When you apply the Fixpack, choose to apply the changes to all the components (IBM HTTP Server, bundled JDK, and WebSphere Application Server.)

   - Security Patch for IBM HTTP Server

     The security E-Fix for the IBM HTTP Server is available at: `ftp://ftp.software.ibm.com/software/websphere/ihs/support/fixes/PQ62369/SUN/1.3.19.2/`

   - Restart WebSphere and IBM HTTP Server.

5. Test your installation and configuration of WebSphere Application Server. Follow the instructions at: `http://www-3.ibm.com/software/webservers/appserv/doc/v40/aeinfocenter/was/022soltestgen.html`

6. Install the appropriate Identity Server Policy Agent for the Web Server. Refer *Sun ONE Identity Server Policy Agent Pack Guide*.

NOTE It is recommended to use IBM HTTP Server Agent, as you may encounter session synchronization problems with Apache and iPlanet Web Server Policy Agent. For more details see Step 8.

---

7. Add the unprotected resources to the Not-Enforced List of the Web Agent.

   The Web Agent intercepts the request to the web server. It checks for the presence of valid a SSO Token in the request. If a valid SSOToken is not found in the request, the user is redirected to the Identity Server for authentication. Requests to the resources deployed on WebSphere go through the Web Server and are intercepted by the Web Agent. Some applications deployed on the

WebSphere may not be protected and may be accessed by unauthenticated users. For such applications, the entries need to be created in the Not-Enforced List property of the Web agent configuration file. With this configuration, the web agent no longer forces authentication to the unprotected resources specified and access to these resources is allowed unconditionally. See the *Sun ONE Identity Server Policy Agent Pack Guide.*

8. Enable Cookie Reset Feature in the Web Agent

The LTPA authentication framework in WebSphere sets `LtpaToken` cookie in the browser session on successful authentication. This identifies the authenticated user to the WebSphere Application Server for the requests in that session. When Web Agent is installed, the authentication is managed by Identity Server. When a user session ends in Identity Server, the `LtpaToken` cookie in the request no longer identifies the authenticated user. Due to the continued presence of `LtpaToken`, WebSphere continues the execution in the context of the authenticated user represented by the `LtpaToken`. To clear the information in the `LtpaToken` cookie, enable the cookie reset feature. This can be done from the web agent by setting the following configuration parameters in the Web Agent's configuration file `AMAgent.properties` as shown below.

- Enable Cookie Reset if it is not already enabled.
  `com.iplanet.am.policy.agents.cookie.reset.enable=true`
- Append `LtpaToken` to the list of cookies to be reset
  `com.iplanet.am.policy.agents.cookie.reset.list=LtpaToken`

NOTE This feature is available only in IBM HTTP Server Agent.

If Apache or iPlanet Web Server Agent is used, you may encounter session synchronization problem. To overcome this problem, follow the steps given below:

a. Start the WebSphere Administrative console using the following command:

WAS_root_dir`/bin/adminclient.sh`

b. In the Administrative Console interface, choose Console > Security Center.

c. Click on Authentication tab.

d. Uncheck the check box "Enable Single Sign-On" and click OK.

e. Restart the WebSphere Server for changes to take effect.

9. Create URL Policies for the resources in Identity Server.

   To perform coarse grained access-control, define the Allow / Deny URL policies for them in the Identity Server. Create Allow Policies for resources, for which you don't want to perform any coarse grained access control, but only fine grained access control through security constraints and EJB method permissions in deployment descriptor. Assign these policies to the organization. For example, an Allow policy for URL `http://webserverhost.domain:port/*` will cause the Web Agent to allow all the requests to pass through. By default if no policy is created, access to all resources is blocked by the Web Agent.

10. Test the deployment of the application that you intend to protect.

    Before installing the WebSphere Agent, it is important that you deploy and test the application that must be protected for simple functionality. Once it is established then the application can be deployed successfully, you are ready to install the Agent.

11. Disable Security for the WebSphere Application Server, if already enabled.

    This can be done from the WebSphere Administrative Console as follows.

    a. Start the WebSphere Administrative Console.

    # *WAS_root_dir*`/bin/adminclient.sh`

    b. Choose Console > Security Center.

    c. In the Security Center Window, click on General tab.

    d. Ensure that the check box Enable Security is not selected.

    e. Click Apply.

    f. Click OK.

    Restart the WebSphere Administration Server using the following command:

    # *WAS_root_dir*`/bin/startupServers.sh`

NOTE The Agent Installer updates the Security Configuration, and installs the Agent Realm as the User Registry. If the WebSphere is configured to use the OS realm, LDAP realm, or any other custom registry, the WebSphere will not be able to authenticate to the current realm, subsequently the Agent Realm installation and Security Configuration will fail during the agent installation.

Launching the Installation Program

The binaries for Identity Server Policy Agent for WebSphere 4.0.3 AE for Solaris platform are provided as a tar-gzip archive. Copy this archive to the machine where WebSphere Application Server is installed. Perform the following steps to launch the installation program.

1. Login as root.
2. Unzip the binary archive using the following command:

   ```
   # /bin/gzcat WebSphere_4.0.3_agent_1.0_Solaris2.8.tar.gz |
   /bin/tar xvf -
   ```

3. Set your JAVA_HOME environment variable to the JDK supplied with WebSphere Server 4.0.3 server. This JDK is located at:

   WAS_root_dir/java

   The installation program provides two types of interfaces. A graphical user interface (GUI) and a command-line interface. In most cases, the GUI installation program can be used for installing the Agent. However, in cases when you are installing the Agent over a telnet session on a remote server, and do not have windowing capabilities, it is recommended that you use the command-line installation program for installing the Agent. You can launch this by executing the following command:

   ```
   # ./setup -nodisplay
   ```

   However, if you choose to use the GUI installation program, then it is required that you set your DISPLAY environment variable to ensure that the GUI installation program window appears on the correct console.

4. Launch the GUI installation program by invoking the setup script as follows:

   ```
   # ./setup
   ```

   The installation program requires that you set up your JAVA_HOME variable correctly as pointed out in the Step 3. However, if you have set the JAVA_HOME variable incorrectly, the setup script will prompt you for the correct JAVA_HOME value:

```
Enter JAVA_HOME location (Enter "." to abort):
```

Type the full path to the WebSphere JDK installation directory for launching the installation program. Otherwise, enter a period (.) to abort the installation.

In order that the GUI installation program be displayed on your console, the DISPLAY environment variable of your shell must be set correctly.

If your DISPLAY environment variable is not set at the time of invoking the setup script, the installation program will prompt you for the DISPLAY environment variable value as follows:

```
Please enter the value of DISPLAY variable (Enter "." to abort):
```

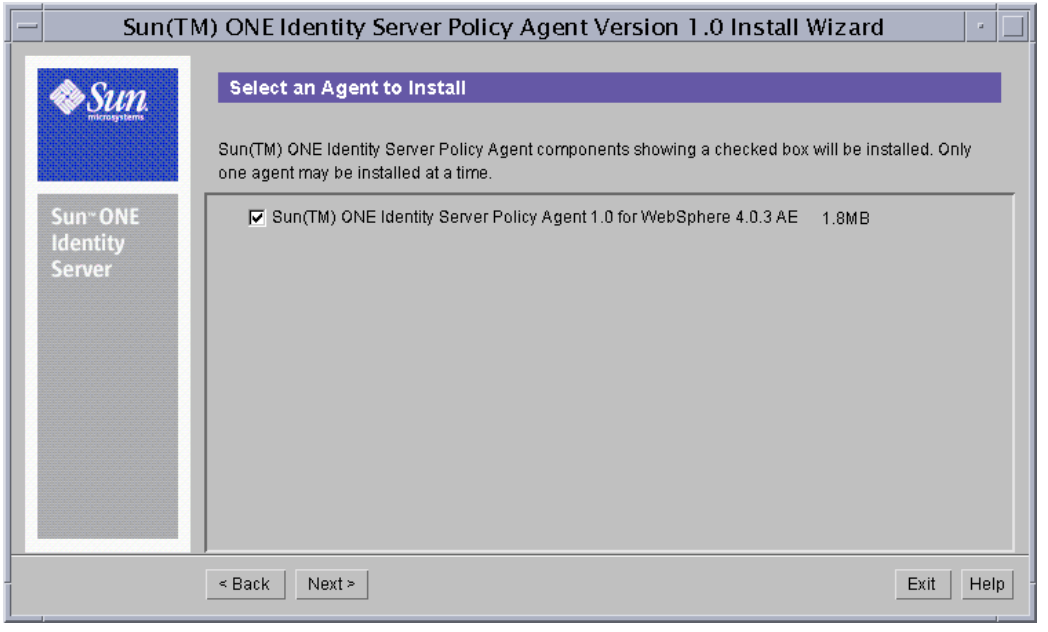
Provide the DISPLAY value to the installation program by typing the exact value at the above prompt. Otherwise, enter a period (.) to abort the installation.

Installing the Agent Using GUI

The installation program begins with a Welcome screen. Click Next to step through the installation screens, and answer the required questions.

1. Read the License Agreement. Click Yes (Accept License) to continue with the Installation.
2. In the Select Installation Directory screen, enter the path where you want to install.
3. In the Select an Agent screen, select the component you wish to install by selecting the check box against the component name. The only component available for installation is the Identity Server Policy Agent for WebSphere Server 4.0.3, which is selected by default.

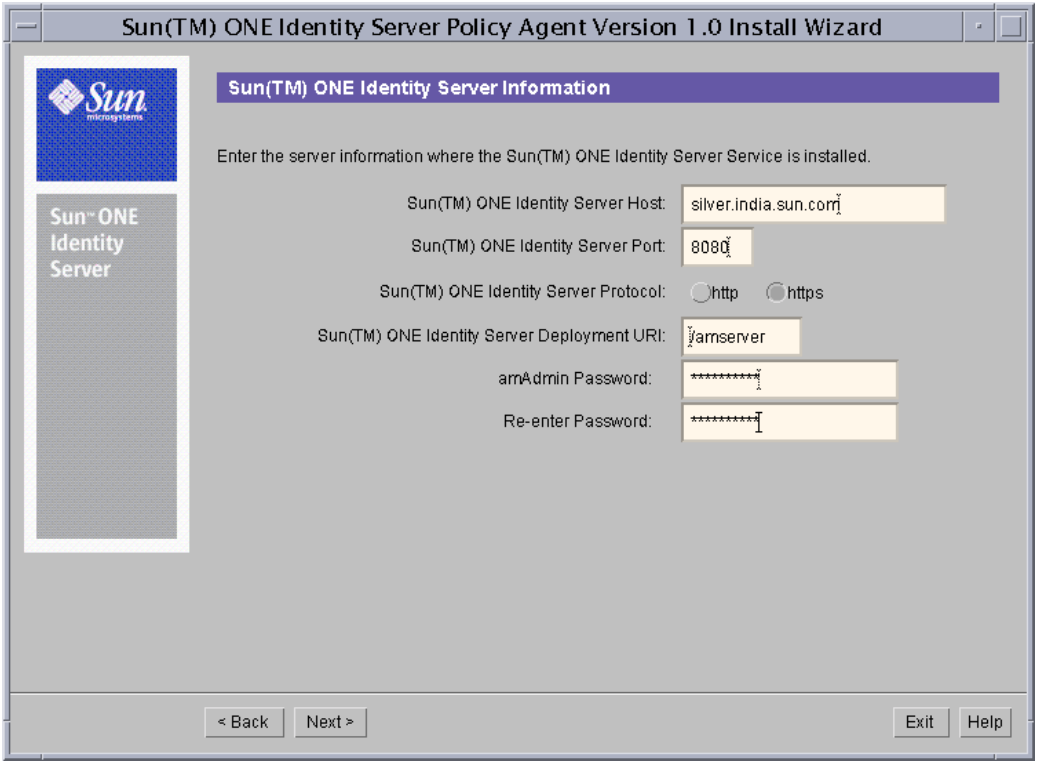
Figure 3-2 Component Selection Screen



NOTE For a given system, only one installation of Identity Server Policy Agent for WebSphere 4.0.3 is allowed at a time. If a previous installation of the Agent has not been removed completely from this system, the check box will be disabled and no selections can be made. In that case it is recommended that you exit the installation program by clicking the Exit button, and remove the old installation completely before starting the install again.

4. In the Sun ONE Identity Server Information screen, provide the following information about the Identity Server, and click Next.

Figure 3-3 Sun ONE Identity Server Information Screen



Sun ONE Identity Server Host: Enter the fully qualified host name of the system where Sun ONE Identity Server is installed.

Sun ONE Identity Server Port: Enter the port number for the Web Server that

runs Sun ONE Identity Server Services.

Sun ONE Identity Server Protocol: Select the protocol that will be used by the Agent to communicate with Identity Server services. This protocol may either be HTTP or HTTPS.

Sun ONE Identity Server Deployment URI: Enter the URI that should be used for accessing Identity Server Services.

amAdmin Password: Enter the password for the amAdmin user.

Re-enter Password: Re-enter the password for the amAdmin user for confirmation.

NOTE The password supplied during installation is recorded by the Agent in a secure manner. However, if in the future you change this password in Identity Server, you will have to update the password in the Agent as well. This can be done by using the `agentadmin` tool provided with the Agent. Once the Agent has been installed on the system, the `agentadmin` tool can be invoked from the location:

Agent_Install_Dir`/SUNWam/wasAgent/bin/agentadmin`

To change the password, invoke this tool as follows:

`#./agentadmin -password` *oldpassword newpassword*

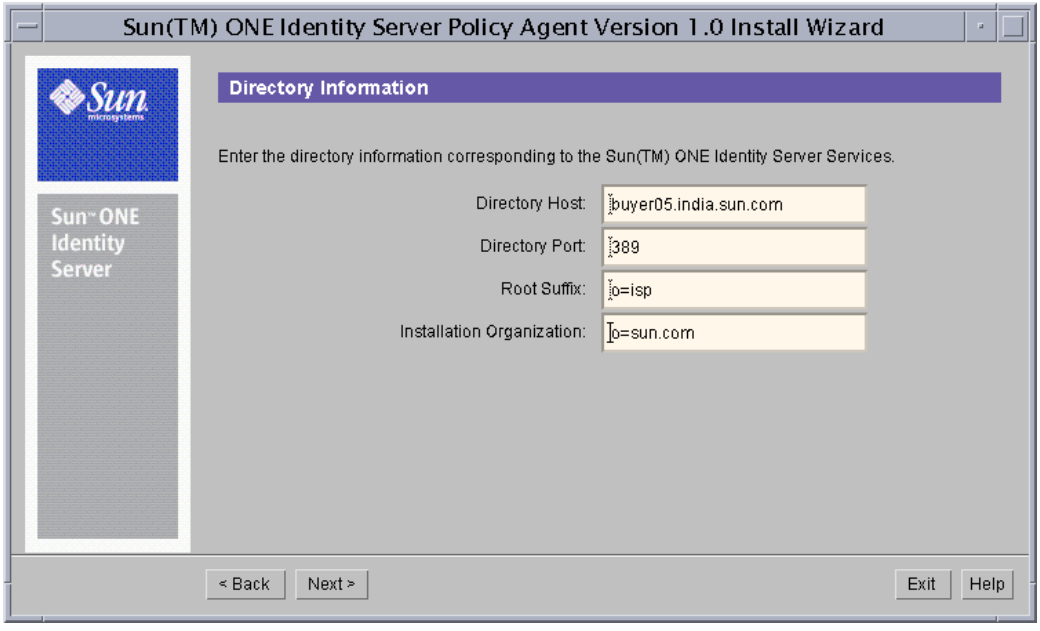
If the user amAdmin is used as the Realm Administrator (Security Server Password) perform the following steps:

1. Change the Security Server Password through WebSphere Administrative Console.
2. Change the Security Server Password by setting the property `com.ibm.CORBA.loginPassword=`*newpassword* in `sas.client.props` file. For more information refer WebSphere documentation.
3. Restart WebSphere for changes to take effect.

---

4. In the Directory Server Information screen, provide the following information about the Directory Server that is associated with Identity Server services.

Figure 3-4 Directory Information Screen



Directory Host: Enter the fully qualified host name of the system where the Directory Server is installed.

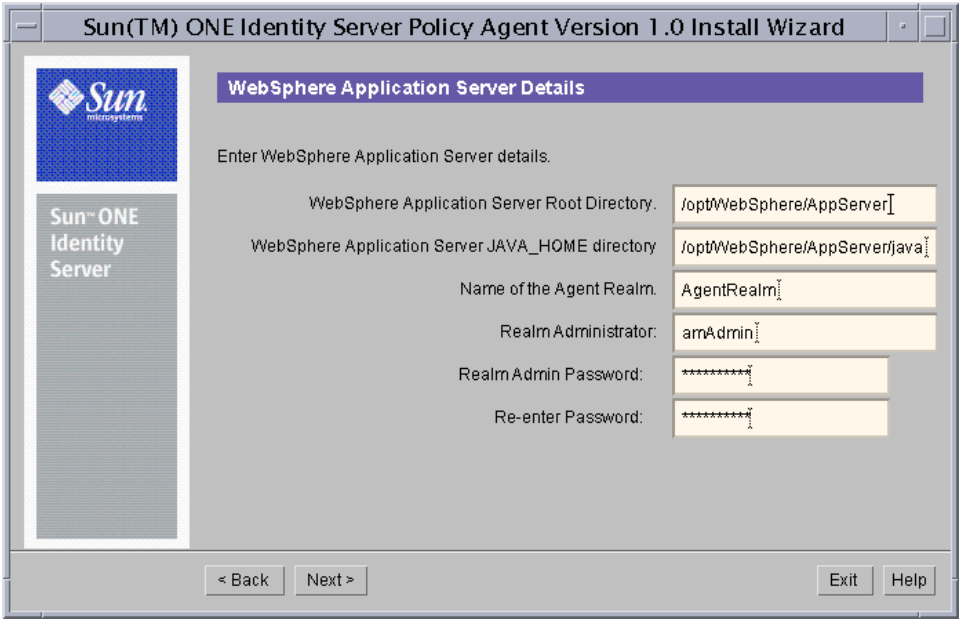
Directory Port: Enter the port number used by the Directory Server.

Root Suffix: Enter the root suffix to be used with this Directory Server.

Installation Organization: Enter the name of the installation organization as specified when installing the Sun ONE Identity Server.

5. In the WebSphere Application Server Details screen, provide the following information about the WebSphere Server on which the Agent is installed.

Figure 3-5 WebSphere Application Server Details Screen



WebSphere Application Server Root Directory: Enter the full path to the location of the WebSphere root directory.

WebSphere Application Server JAVA_HOME Directory: Enter the full path to the JDK installation directory used by the WebSphere Server. The WebSphere JAVA_HOME directory refers to the JDK installation that is used by the WebSphere Server. By default, this is *WAS_root_dir*/java

Name of the Agent Realm: Enter the name of the Realm that will be configured as the default realm to be used by the WebSphere Application Server.

Realm Administrator: Enter the userid of the Administrative user for the installed organization. The WebSphere Application Server requires administrative user credentials (userid and password) for configuring the Realm / User Registry. This is typically the user who does the top level user

management functions such as Creating and Deleting new roles and users. This user is used to check the validity of the realm by authenticating to the realm during the realm installation. This user is also used to call the Realm methods to get the list of all users and roles in the installed organization.

Realm Admin Password: Enter the password. This is used to perform initial authentication to the realm during the realm installation. This is also used to authenticate to the realm while getting the list of all users and roles in the installed organization.

NOTE It is very important that the Realm Administrator user is a valid user, and authentication credentials are accurate. Once the Realm is configured and security is enabled, this user is used to authenticate to the realm. If the user doesn't exist in Identity Server, or if the password is incorrect, then the realm initialization will fail, and may render the Application Server unusable.

Before entering these values in the installer, make sure you can login to the Identity Server console with this user and corresponding Password.

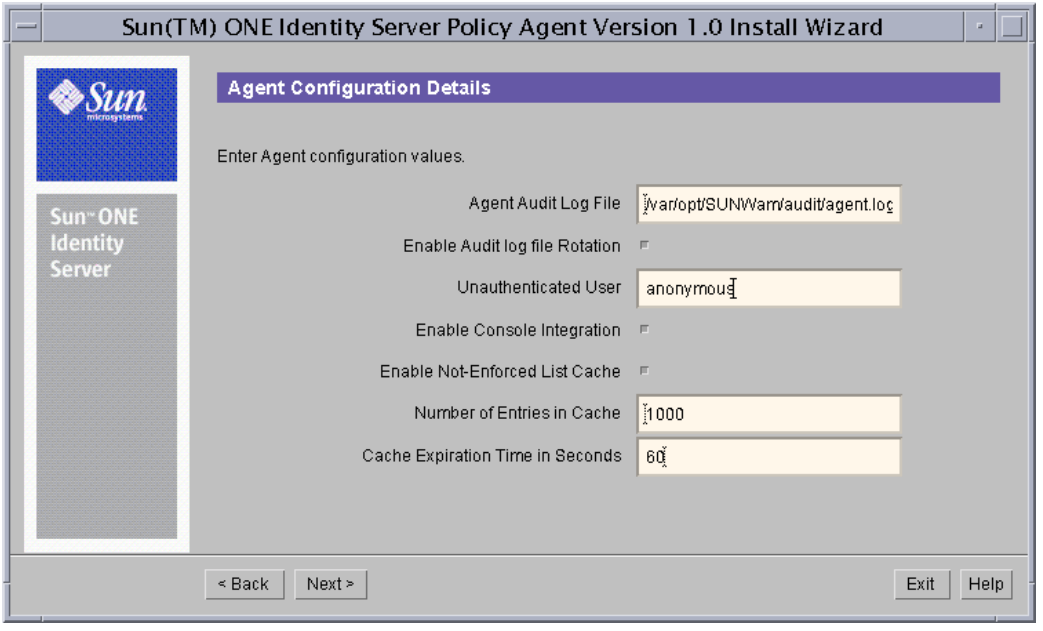
If you want to change the Realm Administrator (Security Server Password) perform the following steps:

1. Change the Security Server Password through WebSphere Administrative Console.
2. Change the Security Server Password by setting the property `com.ibm.CORBA.loginPassword=`*newpassword* in `sas.client.props` file. For more information refer WebSphere documentation.
3. Restart WebSphere for changes to take effect.

---

4. In the Agent Configuration Details screen, provide the key configuration information necessary for the Agent to function correctly.

Figure 3-6 Agent Configuration Details Screen



Audit Log File: Enter the complete path to the log file to be used by the agent to record Audit messages.

Enable Audit log file rotation: Select this to enable rotation of Audit Log files.

Unauthenticated User Name: Enter the User id of the unauthenticated user. This user will be used when protected resources are included in the Not-Enforced List. The Agent Interceptor returns this as the user to be used for accessing the protected resources included in the Not Enforced List.

Enable Console Integration: Select this to enable console level integration of Sun ONE Identity Server with WebSphere Administrative Console.

Enable Not-Enforced List Cache: Select this item to enable caching of Not-Enforced List evaluation results.

Number of Entries in Cache: Specify the number of entries that cache can hold at a given instance.

Cache Expiration Time: Specify the time in seconds to be used as the limit for entries that can exist in the Not-Enforced List cache.

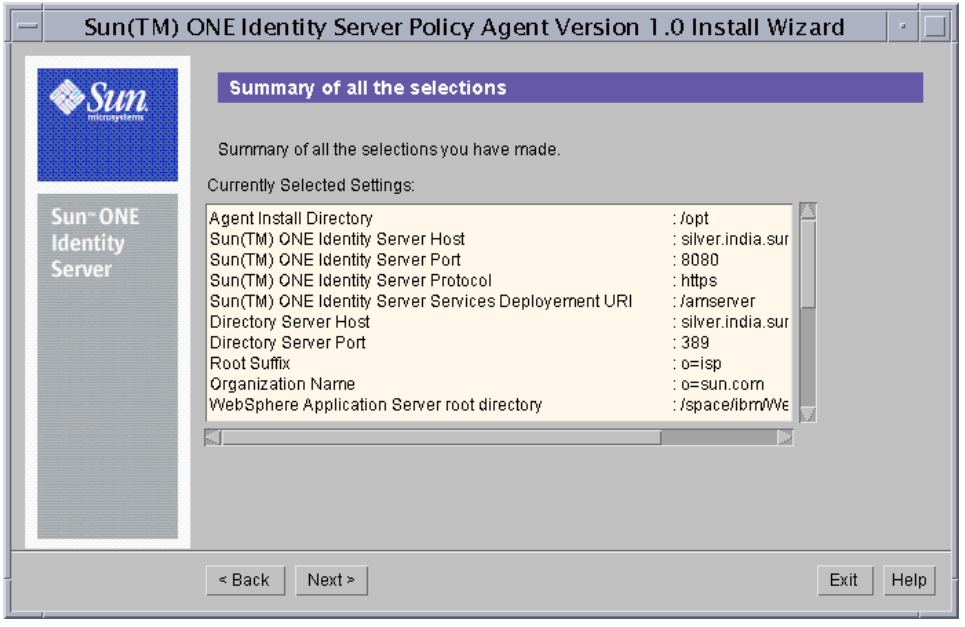
Important Tips

- The Audit Log file is a necessary requirement for the Agent. You may provide the name of a non-existing file on the system to be used as the Audit file. In which case, the Agent creates this file and the necessary directories in its path on first use. Alternatively, you can provide the filename of an existing file which can be used by the Agent. However, in either case, it is required that the specified file has write permissions for the WebSphere Server process because the Agent executes in the same process as the WebSphere Server. Providing an incorrect value for this will result in the malfunction of the Agent which can render the WebSphere Server unusable.
- Console Integration implies that the information regarding the names of all Users, Roles, and the Users who are assigned to these Roles in Identity Server will be visible on the WebSphere Server Administrative Console. Therefore, this feature must be enabled with caution since the User or Role information will then be accessible to administrators who access the WebSphere Server Console.
- The Agent Interceptor is bypassed for access to unprotected resources. The unprotected resources are the resources which are not protected by security constraints in the deployment descriptor. Since the Not-Enforced List is processed by the Agent Interceptor, this will not impact the unprotected resources. Therefore, it is not necessary to add any unprotected resources to the Not-Enforced List. The resources in the Not-Enforced List will always be accessed as the unauthenticated user—for example, anonymous, independent of the identity of the logged in user.
- The Application Server authorization enforcement takes precedence over the Not-Enforced List configuration specified in `AMAgentConfiguration.properties`. The access to the protected resource specified in Not-Enforced List will be successful, only if the unauthenticated user, has the appropriate roles to access the application based on the application's deployment descriptors.
- When the Agent must process a large list of Not-Enforced pattern rules specified in the configuration, every incoming request must be evaluated against every such rule to determine if the request can be allowed without authentication. In scenarios where the user load is high, the time spent in evaluating these rules can add up and degrade the overall performance of the system. To avoid this problem, it is recommended that Not-Enforced List Cache be enabled.

- Enabling the Not-Enforced List Cache may result in degraded performance if the values for Number of Entries in the Cache and Cache Expiration time are not set up appropriately. In cases when the cache expiration time duration is more than necessary, the cache will get filled up very fast and new requests will still be evaluated against all the specified pattern rules, resulting in no improvement in system performance. If the number of entries in Cache is set very high, it may result in excessive consumption of system memory, thereby leading to degraded performance. Therefore, these values must be set up after careful deliberation of the deployment scenario and should be changed as necessary to reflect the changing usage scenario of the system. It is recommended that the system be tested with various values of these two parameters in a controlled environment to identify the optimal values, and only then be deployed in production.
- The Agent maintains two caches in its memory, one for recording the URIs that were evaluated as enforced, and the other for recording the URIs that were evaluated as not-enforced. The specified values of Number of Entries in Cache and Cache Expiration time is equally applicable to both of these caches. This factor must be considered when setting the values for the size and expiration time of the cache.

5. In the Summary of all the selections screen, review and verify the installation options that you have specified for the Agent. If you need to make changes, click Back. Otherwise, click Next to proceed.

Figure 3-7 Summary of All the Selections Screen



6. In the Ready to Install screen, click Install Now to begin the installation.
7. The Install Progress screen displays the progress of the installation as the installation program makes changes to your system. If necessary, you may interrupt this process by clicking the Stop button.

NOTE It is strongly recommended that you do not interrupt this process, as this may lead to a partially installed product. This may also cause problems with uninstall, and may render the WebSphere Server unusable. This process should be interrupted only in cases where it is absolutely necessary to do so.

8. In the Installation Summary screen, click on Details for a detailed summary of the configuration information that was processed during installation. Click on Exit to end the program.

NOTE It is recommended that you check the install log file by clicking the Details button to identify errors. If you find errors, uninstall the Agent and try again after fixing the root cause of failure during this installation.

9. Restart WebSphere Application Server using the following command:

   ```
   # WAS_root_dir/bin/startupServers.sh
   ```

After the agent is installed, the next step is to configure the WebSphere Server as explained in the following sections.

Installing the Agent Using Command-Line

You must have root permissions when you run the agent installation program.

1. Unzip the Solaris tar file using the following command:

   ```
   # gzcat WebSphere_4.0.3_agent_1.0_Solaris2.8.tar.gz | tar -xvf -
   ```

2. Run the setup program. You'll find the program in the directory where you untarred the binaries. At the command-line, enter the following:

   ```
   # ./setup -nodisplay
   ```

3. When prompted, **Please enter path to pick up java:** Type the full path where the WebSphere JDK is located.

4. When prompted provide the following information:

   Have you read, and do you accept, all of the terms of the preceding Software License Agreement? Type Yes.

   Install Sun(TM) ONE Identity Server Agent in this directory: Enter the full path to the directory in which you want to install the Policy Agent.

5. The following text is displayed:

```
Specified directory is valid
1. Accept
2. Choose another Directory
Enter the number corresponding to your choice:
```

Type 1 to accept the specified directory. Otherwise, type 2 to choose another directory.

6. Provide the following information about the Web Server that runs Identity Server:
   - Sun(TM) ONE Identity Server Host
   - Sun(TM) ONE Identity Server Port
   - Sun(TM) ONE Identity Server Protocol
   - Sun(TM) ONE Identity Server Deployment URI
   - amAdmin Password
   - Re-enter Password

   For details on each of these items, see "Installing the Agent Using GUI."

7. Provide the following information about the Web Server that runs Identity Server Directory:
   - Directory Host
   - Directory Port
   - Root Suffix
   - Installation Organization

   For details on each of these items, see "Installing the Agent Using GUI."

8. Provide the following information about the WebSphere Application Server:
   - WebSphere Application Server Root Directory.
   - WebSphere Application Server JAVA_HOME directory
   - Name of the Agent Realm
   - Realm Administrator

- Realm Admin Password
- Re-enter Password

For details on each of these items, see “Installing the Agent Using GUI.”

9. Provide the configuration details for the installation of Sun ONE Identity Server Agent for WebSphere Application Server.

   - Agent Audit Log File
   - Enable Audit log file Rotation
   - Unauthenticated User
   - Enable Console Integration
   - Enable Not-Enforced List Cache
   - Number of Entries in Cache
   - Cache Expiration Time in Seconds

   For details on each of these items, see “Installing the Agent Using GUI.”

10. When displayed, review the summary of the installation information you have specified. Press Enter to continue.

    The following text is displayed:

```
Ready to Install

1. Install Now
2. Start Over
3. Exit Installation
```

When prompted, **What would you like to do?** Type `1` to start the installation.

11. The following text is displayed:

```
Installation details:

Product                                      Result     More Info

1.Sun_TM_ONE Identity Server Policy Agent  Installed  Available
2.Done

Enter the number corresponding to the desired selection for more
information, or enter 2 to continue [2] {"!" exits}:
```

To see log information, type `1`. To exit the Installation program, type `2`.

12. Restart WebSphere Application Server using the following command:

    # *WAS_root_dir*`/bin/startupServers.sh`

WebSphere Application Server Configuration

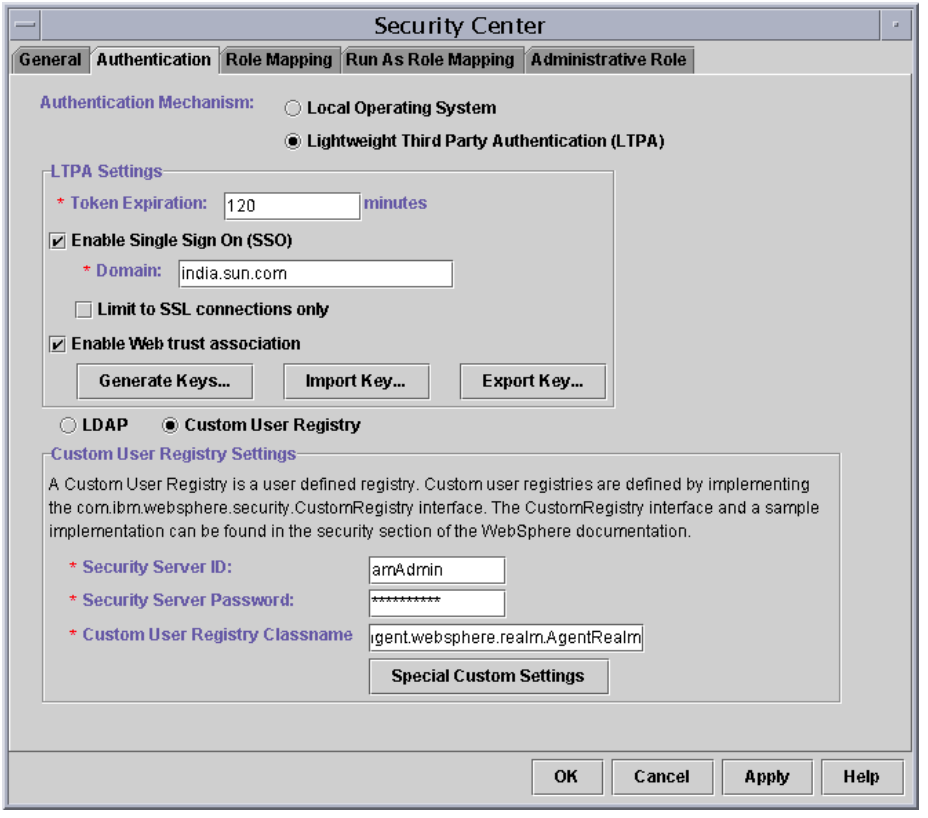
After the Sun ONE Identity Server Policy Agent for WebSphere has been installed on your system, Web Trust Association must be enabled for the Agent Interceptor to work appropriately. This can be done from the WebSphere Administrative Console.

1. Start the WebSphere Administrative console using the following command:

   # *WAS_root_dir*`/bin/adminclient.sh`

2. In the Administrative Console, choose Console > Security Center.
3. In the Security Center window, click on Authentication Tab.
4. Select the check box Enable Web trust association.

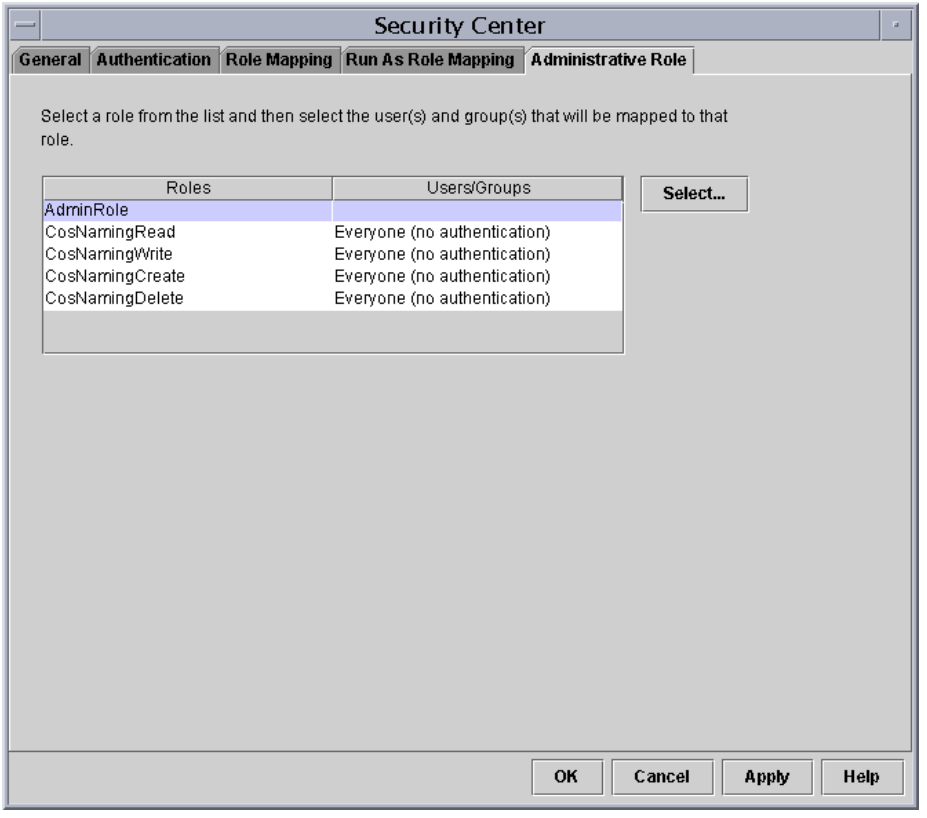
Figure 3-8 Enabling Web Trust Association



5. Click Apply. A message “Changes will take effect only after Administration Server is restarted” will be displayed.

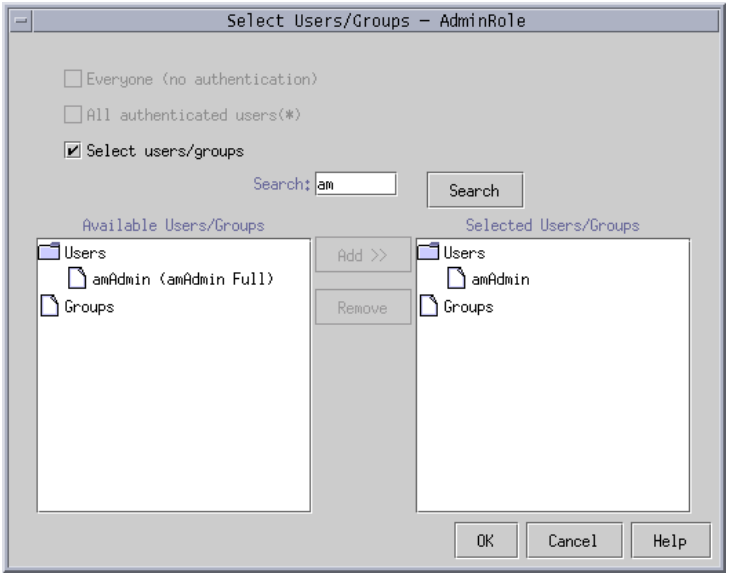
6. Click the Administrative Role tab and select AdminRole. Click Select.

Figure 3-9 Modifying Admin Role in Security Center



7. In the Select Users/Groups - AdminRole window, add the Realm Administrator User that was specified during installation for AdminRole. Click OK.

Figure 3-10 Adding the Realm Administrator to the Admin Role



8. In the Security Center window, click on Apply and then OK to apply the changes made.
9. Stop all the Application Server instances and also the Administration Server.
10. Restart Administration Server and Application Server instances for changes to take effect.

Application Configuration

The Agent Realm (Custom Registry) component of Identity Server Policy Agent for WebSphere provides runtime mapping of various principals in Identity Server.

Abstract security role names are used by the hosted application in order to determine if the currently authenticated user is authorized to access a particular resource, or is otherwise a member of a given role. This runtime evaluation can occur only if the user is authenticated as an Identity Server principal using Identity Server's authentication service. Without the user being authenticated appropriately, the results of such evaluations done by the Agent Realm will always be negative, resulting in access being denied to the user for the requested resource.

Creating Role-to-Principal Mappings

Once the application has been secured and the Web Trust Association has been enabled, the Trust Association Interceptor intercepts the requests for the application, validates SSOToken, and authenticates to the Security Service of the Application Server, thereby enabling the Agent Realm to successfully resolve the role-to-principal mappings. However, these mappings must first be created in order that the hosted application may use them during runtime.

The following are two ways to create such mappings:

- **Editing the WebSphere Server specific deployment descriptors**

  The WebSphere Server specific deployment descriptors may be edited to create the role to principal mappings. These descriptors are in `ibm-application-bnd.xmi` file. Refer WebSphere Server reference documentation for details on how these descriptors may be edited to create the role to principal mappings. Alternatively, you can refer the information provided in Appendix C for sample descriptors, to create such mappings.

- **Using the WebSphere Server Administrative Console**

  The WebSphere Server Administrative console allows you to create the role to principal mappings by editing the deployed application's deployment descriptors. In order to use this facility, the application must be deployed and the WebSphere Server must be up and running at that time. Refer WebSphere Server documentation on how to use the Administrative Console to create such mappings for deployed applications.

Application Specific Agent Configuration

Oftentimes, the deployed applications are partitioned into public and protected parts, which have varying access restrictions. In most cases, the public portions of the application are accessible to unauthenticated users, whereas the protected portions of the application are accessible only to registered users. The Agent can be configured to allow this type of access by letting unauthenticated users access the public portions of the application without requiring that they authenticate using Identity Server's authentication service. This information is provided as a part of the Agent's configuration properties file that is present in the following location:

Agent_Install_Dir`/SUNWam/wasAgent/amAgent/config/AMAgent.properties`

This file may be edited to provide general as well as application-specific configuration for the Agent.

NOTE

- The properties specified in the `AMAgent.properties` file are required for the Agent to function properly. Invalid values specified in this file can lead to malfunction of the Agent, application becoming inaccessible, or the entire system becomes unusable. It is recommended that you use extreme caution when modifying the values in this file. Always create backups before such modifications to ensure that you can back-out your changes to restore the system to its original state.
- The properties specified in the `AMAgent.properties` file are loaded during the WebSphere Server startup time. Any changes made to this file while the WebSphere Server is running will not take effect until the server has been restarted.

Providing Application-Specific Not-Enforced List

In order to allow unauthenticated users to access portions of the protected application, the `AMAgent.properties` file must include an entry for the specific application. Access to these entry points will be made as the unauthenticated user. The entry that specifies this property is called the application-specific not-enforced list and is identified by the string:

`com.iplanet.amagent.config.filter.`*AppName*`.notEnforcedList[`*index*`]=`*patter n*

This property requires that you format it correctly in order that it can be used by the Agent during runtime. The entries appearing in italics within this string should be replaced by their appropriate values as follows:

AppName: This string should be replaced by the deployed application's context path without its leading forward-slash character. The context path is the first URI segment that is used to identify which application the user is trying to access. For example, if the user accesses your application by typing the URL:

`http://myserver.mydomain.com/SomeApp/index.html` or

`http://myserver.mydomain.com/SomeApp/SomeModule/doSometing.jsp`

then in both these cases, the *AppNam*e is `SomeApp`.

index: This is an integer value starting from `0` for every deployed application and is unique for every entry in the application not-enforced list. For example, the following are two entries of application not-enforced list for an application with the context path `/SomeApp`:

```
com.iplanet.amagent.config.filter.SomeApp.notEnforcedList[0]=/SomeA
pp/public/*
```

```
com.iplanet.amagent.config.filter.SomeApp.notEnforcedList[1]=/SomeA
pp/images/*
```

pattern: This is a pattern string that will be matched with the incoming request to evaluate if the request should be allowed to pass without enforcing authentication or not. The pattern string could be a specific URI, for example `/SomeApp/public/RegistrationServlet`, or could be a generic pattern using the wild-card character '`*`' that can be used for denoting `0` or more characters in the request URI, for example `/SomeApp/public/*` will match with any URI that begins with `/SomeApp/public/`.

Using this property, you can specify none or many pattern strings and URIs that the Agent will treat as not-enforced. The user requests that match these particular patterns will be allowed to pass through without requiring a valid SSO token in the request.

Special Case: Default Web Application

A default web application on WebSphere Server is accessible without providing any context path in the request URI. For example, the following URL is that of a default web application:

```
http://myserver.mydomain.com/index.html
```

The above URL does not have an associated context path.

For such applications, the Agent provides a convenient means of identifying that an entry is specific to the default web application. This is done in two steps as follows:

1. The following property is set to a name that represents the default web application:

   ```
   com.iplanet.amagent.config.filter.defaultWebAppName=
   DefaultWebApp
   ```

2. This name is used to specify the application not-enforced list.

   ```
   com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList[
   0]=/index.html
   ```

   ```
   com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList[
   1]=/about.html
   ```

Using this scheme, the default web application that does not have a context path associated with it, may be configured just like any other application that has a context path. The same rules apply to the default web application for specifying the not-enforced list entries as applicable to the rest of the application. However, the

only difference is that the not-enforced list entries cannot begin with the `/DefaultWebApp/` path segment since such a path segment does not exist on the application server in reality. The *AppName* in this case where the actual context path is an empty string, is only provided as a convenience to specify the properties associated with the default web application and should not be used in specifying their values.

Global Agent Configuration

The `AMAgent.properties` file provides a way to specify a global not-enforced list, which will be applicable to all the protected applications that are deployed on the server. This list is specified using the following property:

`com.iplanet.amagent.config.filter.global.notEnforcedList[`*index*`]=` *pattern*

index is an integer value starting from 0 and unique for every entry.

pattern is either an exact URI or a pattern specified by using the wild-card character '*' that can be substituted for zero or more characters in the request URI.

Not-Enforced List Usage Considerations

Although the use of Not-Enforced List can be extremely helpful in partitioning your application for public and protected domains, it can also lead to undesirable effects if not used appropriately.

For example, if a request URI that represents a Servlet is matched by some not-enforced list pattern, then the Agent Interceptor will not perform authentication to the Application Server, for users who try to access that particular Servlet. However, consider the case where this Servlet accesses an Enterprise JavaBeans component that is protected by the Agent using role to principal mapping. In such a case, since the user is unauthenticated, the access to the protected component will result in a security violation exception being generated by the application server. Therefore, before an entry is added to the not-enforced list, it must be ensured that the entry does not in any way cover a resource to which the Unauthenticated user has no access, or may try to access such a resource.

Agent Configuration

The core configuration needed by the Sun ONE Identity Server Policy Agent for WebSphere is provided in the `AMAgent.properties` file located in the following directory:

Agent_Install_Dir`/wasAgent/amAgent/config`

This property file provides many configuration settings that can be modified in order to customize the Agent's operation for your deployment scenario.

NOTE Before proceeding, you must note that this file and the information within the file are critical for the operation of the Agent. It is strongly recommended that you always create a backup of this file before modifying it. Also, it is strongly recommended that you do not modify this file unless it is absolutely necessary. Any invalid data entries present in this file can lead to the malfunction of the Agent, malfunction of the deployed applications, and could render the entire system unusable.

The settings provided in this file can be classified into the following categories:

- Common Configuration
- Audit Configuration
- Realm Configuration
- Global Interceptor Configuration
- Application Interceptor Configuration
- Debug Engine Configuration

The following sections detail the settings for each of these classifications.

Common Configuration

Settings in this section are general settings that affect the behavior of the Agent as a whole.

Organization Name

Key: `com.iplanet.amagent.config.org`

Description: This property specifies the organization name to be used when searching for principals in Sun ONE Identity Server.

Valid Values: String representing the organization name in Sun ONE Identity Server. This property is set during Agent installation and must not be changed unless absolutely necessary.

Example: `com.iplanet.amagent.config.org=sun.com`

Root Suffix

Key: `com.iplanet.amagent.config.rootsuffix`

Description: This property specifies the root suffix to be used when searching for principals in Sun ONE Identity Server.

Valid Values: String representing the root suffix in Sun ONE Identity Server. This property is set during Agent installation and must not be changed unless absolutely necessary.

Example: `com.iplanet.amagent.config.rootsuffix=o=isp`

People Container Level

Key: `com.iplanet.amagent.config.realm.peopleContainerLevel`

Description: This property specifies the people container level to be used when searching for principals in Sun ONE Identity Server.

Valid Values: Non-zero unsigned integer representing the People Container Level in Identity Server, which may be used when searching for principals. This property is set during Agent installation and must not be changed unless absolutely necessary.

Example: `com.iplanet.amagent.config.realm.peopleContainerLevel=1`

Audit Configuration

These settings are exclusively used to configure the Audit Engine used by the Agent.

Language Code

Key: `com.iplanet.amagent.config.audit.localeLanguageCode`

Description: This property specifies the Locale for Audit log messages.

Valid Values: The `localeLanguageCode` must be a valid ISO Language Code. The default value of this property is `en`.

Example: `com.iplanet.amagent.config.audit.localeLanguageCode=en`

NOTE For more information, refer ISO 639 specification at `http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt`

Country Code

Key: `com.iplanet.amagent.config.audit.localeCountryCode`

Description: This property specifies the Locale for Audit log messages.

Valid Values: The `localeCountryCode` must be a valid ISO Country Code. The default value of this property is US.

Example: `com.iplanet.amagent.config.audit.localeCountryCode=US`

NOTE For more information, refer ISO 3166 specification:

`http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html`

Audit Log File

Key: `com.iplanet.amagent.config.audit.logfile.name`

Description: This property specifies the Audit log file to be used for recording Audit messages.

Valid Values: String representing the complete path name of the file to be used by the Agent to record Audit messages.

Example:
`com.iplanet.amagent.config.audit.logfile.name=/audit/agent.log`

NOTE

- Ensure that the WebSphere Server process has sufficient permissions to write to this file.
- Specifying an invalid value specified for this property may result in the failure of the system to start up correctly.

Audit Log File Rotation Flag

Key: `com.iplanet.amagent.config.audit.logfile.rotate`

Description: This property specifies if the Audit log file should be rotated by the Agent.

Valid Values: `true`/`false`. The default value of this property is false and should be changed as necessary.

Example: `com.iplanet.amagent.config.audit.logfile.rotate=false`

Audit Log File Rotation Size

Key: `com.iplanet.amagent.config.audit.logfile.rotate.size`

Description: This property specifies the approximate size of the Audit log file in bytes, which should be used to evaluate when the log file needs to be rotated.

Valid Values: Non-zero unsigned integer indicating the size in bytes to be used to evaluate when the log file needs to be rotated. The default value of this property is 52428800 bytes (~ 50 MB) and should be changed as required.

Example:
`com.iplanet.amagent.config.audit.logfile.rotate.size=52428800`

Realm Configuration

These settings are used to configure the Agent Realm component.

Realm Name

Key: `com.sun.amagent.config.websphere.realmName`

Description: Used by the Realm implementation to identify itself to the WebSphere Application Server. Any changes to this should be accompanied by the appropriate change in WebSphere property files `sas.server.props` and `sas.client.props`.

Allow Console Integration Flag

Key: `com.sun.amagent.config.websphere.allowConsoleIntegration`

Description: This property specifies if the Agent Realm should allow console level integration of Identity Server with WebSphere Server.

Valid Values: `true`/`false`

Example:
`com.sun.amagent.config.websphere.allowConsoleIntegration=true`

When set, the result of console level integration is that the WebSphere Server Administrator will be able to see the list of Identity Server principals in WebSphere Server console.

This setting enables the WebSphere Server Administrator to see the Identity Server principals in WebSphere Server console, it should be used with caution.

The default value of this setting is `false` and should be changed as necessary.

Global Interceptor Configuration

These settings are used to configure the Agent Filter component.

Is Alone Interceptor flag

Key: `com.sun.amagent.websphere.interceptor.isAlone`

Description: This property indicates if this is the lone interceptor configured in the system. By default, this is set to `true`. If additional interceptors need to be added to WebSphere, then this should be set to `false` and appropriate changes should be made in the file `trustedServers.properties`.

Valid Values: `true` / `false`. The default value is `true`.

If set to `true`, the absence of SSOToken in the request will cause the request to get denied. If set to `false`, the absence of SSOToken will result in control being passed on to the next interceptor that is installed.

Default User

Key: `com.sun.amagent.websphere.interceptor.defaultUser`

Description: This unauthenticated user is used to access the resources specified in the Not-Enforced List. Access to the resources in the Not-Enforced List will be successful, only if the user has the required roles for accessing them.

Valid Values: Any valid user in the Identity Server.

SSO Token Name

Key: `com.iplanet.amagent.config.filter.ssoTokenName`

Description: This property specifies the name of the cookie that represents the SSO Token.

Valid Values: A string that represents the name of SSO Token cookie issued by the Sun ONE Identity Server authentication service. This property is set during Agent installation and need not be changed unless absolutely necessary.

Example:
`com.iplanet.amagent.config.filter.ssoTokenName=iPlanetDirectoryPro`

Not-Enforced-List Cache Enable Flag

Key: `com.iplanet.amagent.config.filter.notEnforcedList.cache`

Description: This property specifies if the requested URIs that are evaluated as enforced or not-enforced may be cached to increase the performance of the system.

Valid Values: `true`/`false`. The default value of this property is `true`.

Example:
`com.iplanet.amagent.config.filter.notEnforcedList.cache=true`

Not-Enforced-List Cache Size

Key: `com.iplanet.amagent.config.filter.notEnforcedList.cacheSize`

Description: This property specifies the number of entries that will be kept in the cache of not-enforced URIs and enforced URIs by the Agent.

Valid Values: Non-zero unsigned integer indicating the number of enforced as not enforced request URIs to be cached during runtime.

NOTE

- Values that are valid but not suited for the deployment scenario may result in degradation of system performance.
- The value for optimal system performance will depend on the type of application deployed, the number of possible request URIs in the deployed application, the user load on the system, the expiration time set for cache entries and a host of other deployment specific factors.
- To determine the most optimal value of this property, the application should be load tested in a controlled test environment before being deployed for production.

Example:
`com.iplanet.amagent.config.filter.notEnforcedList.cacheSize=1000`

Not-Enforced-List Cache Expiration Time

Key: `com.iplanet.amagent.config.filter.notEnforcedList.cacheTime`

Description: This property specifies the amount of time in seconds that will be used to evaluate if a cached entry can be removed from the cache to free up resources for new cache entries.

Valid Values: Non-zero unsigned integer indicating the time in seconds that will be used as the cache expiration time for entries in the cache during a cleanup operation.

NOTE

- Values that are valid but not suited for the deployment scenario may result in degradation of system performance.
- The value for optimal system performance will depend on the type of application deployed, the number of possible request URIs in the deployed application, the user load on the system, the expiration time set for cache entries, and a host of other deployment specific factors.
- To determine the most optimal value of this property, the application should be load tested in a controlled test environment before being deployed for production.

Example:
`com.iplanet.amagent.config.filter.notEnforcedList.cacheTime=60`

SSO Token URL Decode Flag

Key: `com.iplanet.amagent.config.filter.urlDecodeSSOToken`

Description: This property indicates if the SSOToken needs to be URL Decoded by the Agent before it may be used.

Valid Values: `true` / `false`. The default value of this property is set to `true`.

NOTE Valid, but inappropriate value of this property may lead to the application being unreachable by the users.

Example: `com.iplanet.amagent.config.filter.urlDecodeSSOToken=true`

Default Web Application Name

Key: `com.iplanet.amagent.config.filter.defaultWebAppName`

Description: This property specifies a name for the Default Web Application deployed on the application server.

Valid Values: A string consisting of lower case and or upper case letters that can be used as the name for the default web application. The default value of this property is `DefaultWebApp`.

NOTE This property is necessary if the protected application is deployed as the default web application.

Example:
`com.iplanet.amagent.config.filter.defaultWebAppName=DefaultWebApp`

Global Not-Enforced List

Key: `com.iplanet.amagent.config.filter.global.notEnforcedList[index]`

Description: This property specifies a list of patterns that can be used to evaluate if the requested URI does not require the protection enforced by the Agent.

Valid Values: Valid values must comply with the syntax of this property. The valid values can be exact URIs or patterns consisting of wild-card character '*' to indicate zero or more characters. The syntax of this property is as follows:

`com.iplanet.amagent.config.filter.global.notEnforcedList[`*index*`]=`*pattern*

index is an integer that starts from `0` and increments for every entry in this property list

pattern is a string that represents request URIs that are not enforced by Agent.

The pattern string may consist of wild-card character '*', which may match zero or more characters.

The index must start from zero for the first entry and continue till the last in a sequence. Missing index values in this list will result in partial or complete loss of list entries.

NOTE No value for this property indicates an empty global not-enforced list.

Example:

```
com.iplanet.amagent.config.filter.global.notEnforcedList[0]=*.gif
com.iplanet.amagent.config.filter.global.notEnforcedList[1]=
public/*
```

```
com.iplanet.amagent.config.filter.global.notEnforcedList[2]=
/images/*
```

Application Interceptor Configuration

These settings are used to configure the Agent Interceptor for a particular application.

Application Not-Enforced-List

Key:
`com.iplanet.amagent.config.filter.AppName.notEnforcedList[index]`

Description: This property specifies a list of patterns that can be used to evaluate if the requested URI does not require the protection enforced by the Agent for a particular application. All the URIs that match any of these patterns will be accessed as Unauthenticated User.

Valid Values: Valid values must comply with the syntax of this property. The valid values can be exact URIs or patterns consisting of wild-card character '*' to indicate zero or more characters. The syntax of this property is as follows:

`com.iplanet.amagent.config.filter.`*AppName*`.notEnforcedList[`*index*`]=` *pattern*

AppName is the context path name without the leading forward-slash character (/) for the deployed application.

index is an integer that starts from `0` and increments for every specified property for the particular application.

pattern is a string that represents the URIs that are not enforced by the Agent.

NOTE

- The pattern string may consist of wild-card character '*', which may match zero or more characters.
- The index must start from zero for the first entry and continue till the last in a sequence. Missing index values in this list will result in partial or complete loss of list entries. The index value will be independent for different AppNames specified in this property list.
- If the protected application is the default web application, the AppName should be set to the same string as specified for the value of the property Default Web Application Name.
- No value for this property indicates an empty not enforced list.

Example:

```
com.iplanet.amagent.config.filter.Portal.notEnforcedList[0]=
/Portal/GuestPages/*

com.iplanet.amagent.config.filter.Portal.notEnforcedList[1]=
/Portal/Registration/*

com.iplanet.amagent.config.filter.Portal.notEnforcedList[2]=
/Portal/WebServices/PollServlet

com.iplanet.amagent.config.filter.BankApp.notEnforcedList[0]=
/BankApp/ModuleGuestTour/*

com.iplanet.amagent.config.filter.BankApp.notEnforcedList[1]=
/BankApp/index.html

com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList0]=/
index.html

com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList[1]=
/about.html
```

Debug Engine Configuration

These settings are used to configure the Debug Engine to generate diagnostic information.

Debug Level

Key: `com.iplanet.amagent.config.debug.level`

Description: This property specifies the amount of debug messages that will be generated by the Agent's Debug Engine.

Valid Values: Any of `0`, `1`, `3`, `7`, `15`, and `31`. These values indicate the following:

`0` = No debugging

`1` = Only Error messages

`3` = Error and Warning messages

`7` = Error, Warning and Brief Informational messages

`15`= Error, Warning and Verbose Informational messages

`31`= Error, Warning and Very Verbose Informational messages

NOTE For better performance of the system, this property should be set to a value `0`. Values other than `0` will affect the system performance depending upon the amount of information the Debug Engine has to generate.

Values other than the ones specified in the Valid Values list above will lead to invalid configuration of the Debug Engine, thereby affecting the volume of messages that will be generated.

Example: `com.iplanet.amagent.config.debug.level=7`

Debug Log File

Key: `com.iplanet.amagent.config.debug.logfile.name`

Description: This property specifies the Debug log file to be used for recording Debug messages.

Valid Values: String representing the complete path name of the file to be used by the Agent to record Debug messages.

NOTE
- Ensure that the WebSphere Application Server process has sufficient permissions to write to this file.
- Invalid or empty value of this property will lead to Debug messages not being logged in the log file.

Example:

```
com.iplanet.amagent.config.debug.logfile.name=/debug/agent_debug.log
```

Debug Log File Rotation Flag

Key: `com.iplanet.amagent.config.debug.logfile.rotate`

Description: This property specifies if the Debug log file should be rotated by the Agent.

Valid Values: `true`/`false`. The default value of this property is `false` and should be changed as necessary.

Example: `com.iplanet.amagent.config.debug.logfile.rotate=false`

Debug Log File Rotation Size

Key: `com.iplanet.amagent.config.debug.logfile.rotate.size`

Description: This property specifies the approximate size of the Debug log file in bytes, which should be used to evaluate when the log file needs to be rotated.

Valid Values: Non-zero unsigned integer indicating the size in bytes to be used to evaluate when the log file needs to be rotated. Default value of this property is 52428800 bytes (~ 50 MB) and should be changed as required.

NOTE This property is not used if the Debug Log File Rotation Flag is set to `false`.

Example:

```
com.iplanet.amagent.config.debug.logfile.rotate.size=52428800
```

Debug Time/Date Format String

Key: `com.iplanet.amagent.config.debug.date.format`

Description: This property specifies the format of time stamp that is used to mark the exact time when the Debug message was recorded.

Valid Values: `Valid java.text.SimpleDateFormat Time Format Syntax` string. For more information, refer URL:

```
http://java.sun.com/j2se/1.3/docs/api/java/text/SimpleDateFormat.html
```

NOTE The default value of this property is set to <MMM d, yyyy h:mm:ss a z> 'Agent' and should be changed as necessary.

Invalid value of this property will result in loss of time stamp data with debug messages.

Example:

```
com.iplanet.amagent.config.debug.date.format=[yyyy/MM/dd HH:mm:ss zzz]
```

SSL Configuration

The SSL Configuration involves the following tasks:

1. Configuring SSL for the Web Server

   Configuring SSL for the Web server depends on the Web server used. Refer to the appropriate Web Server documentation.

2. Configuring SSL for WebSphere plug-ins for Web servers. See the WebSphere documentation at:
   `http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/was/06061801a07.html#pi`

3. Configuring SSL for WebSphere Application Server. See the documentation at
   `http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/was/06061801a07.html#was`

4. Configuration Changes in `AmConfig.properties`

   This is the additional step to be performed when the Identity Server is running in SSL Mode.

   ```
   com.iplanet.am.server.protocol=https
   ```

   ```
   com.iplanet.am.naming.url=https://silver.india.sun.com:8080/amserver/namingservice
   ```

   ```
   com.iplanet.am.notification.url=https://silver.india.sun.com:8080/amserver/notificationservice
   ```

   ```
   com.iplanet.am.session.server.protocol=https
   ```

5. Importing the root CA cert into Application Server JVM Keystore

   a. Export the root CA certificate for the CA (who signed the certificate used by the Identity Server) into a Base64-encoded ASCII data file `/tmp/cacert.txt`

   b. Import this cert into AppServer JVM Keystore as follows:

   # *WAS_root_dir*`/java/bin/keytool -import -trustcacerts -alias cmscacert -keystore ../jre/lib/security/cacerts -file /tmp/cacert.txt`

   Enter keystore password: `changeit`

   Trust this certificate? [no]: `yes`

   `cmscacert: just a cert name`

   `keystore: all cert installed`

c. Verify that Cert is added

```
# WAS_root_dir/java/bin/keytool -list -keystore
WAS_root_dir/java/jre/lib/security/cacerts
```

Enter keystore password: `changeit`

Uninstalling the Agent

When you install the Identity Server Policy Agent for WebSphere Application Server software, an uninstallation program is created in the installation directory. Using this uninstallation program the Agent can be removed completely from your system. While the uninstallation program deletes all the installed files from your system, certain files such as audit log messages are not deleted. You can delete them manually.

Launching the Uninstallation Program

The uninstallation program for Solaris platform may be launched by executing the generated `uninstall_wasagent` script located in the installation directory. The following steps provide details on how to achieve this:

1. Login as root.
2. Go to the directory where the Agent is installed.
3. Set your JAVA_HOME environment variable to the JDK supplied with WebSphere Server 4.0.3 server:

   WAS_root_dir`/java`
4. The uninstallation program provides two types of interfaces, a graphical user interface (GUI) and a command-line interface. In most cases, the GUI installer can be used for uninstalling the Agent. However, in cases when you are uninstalling the Agent over a telnet session on a remote server and do not have windowing capabilities, then the GUI uninstallation program cannot be used. In such a case it is recommended that you use the command line uninstallation program for uninstalling the Agent. This can be launched by executing the uninstall script and passing in a command line argument `-nodisplay` as follows:

```
#./uninstall_wasagent -nodisplay
```

However, if you choose to use the GUI uninstallation program, then it is required that you set your DISPLAY environment variable to ensure that the GUI uninstallation program window appears on the correct console.

NOTE If you choose to use the command line uninstallation program using the `-nodisplay` option, you may skip the next step and proceed directly to the next section, which details out the uninstallation procedure.

5. Launch the GUI uninstallation program by invoking the uninstall script as follows:

```
# ./uninstall_wasagent
```

The uninstallation program requires that you setup your `JAVA_HOME` variable correctly as pointed out in the Step 3. However, in case you have set the `JAVA_HOME` variable incorrectly, the uninstall script will prompt you for supplying the correct `JAVA_HOME` value.

```
Enter JAVA_HOME location (Enter "." to abort):
```

Type the full path to the JDK installation directory for launching the installation program. Otherwise, enter a period (.) to abort the installation

In order that the GUI uninstallation program be displayed on your console, the DISPLAY environment variable of your shell must be set correctly. In case your DISPLAY environment variable is not set at the time of invoking the uninstall script, the uninstallation program will prompt you for the DISPLAY environment variable value as follows:

```
Please enter the value of DISPLAY variable (Enter "." to abort):
```

Provide the DISPLAY value to the installer by typing in the exact value at the above prompt. Otherwise, enter a period (.) to abort the installation at this time.

Uninstalling the Agent Using GUI

1. The uninstallation program begins with a Welcome screen. Click Next to step through the uninstallation screens, answering the questions.

2. In the Uninstall Type Selection screen select the type and click Next.

   Full: Select this type to remove the product and all the components from your system.

   Partial: Select this type if you want to remove only certain Agent components. Select the component you want to uninstall and click Next.

3. In the Ready to Uninstall screen, review the uninstallation information. If you need to make changes, click Back. Otherwise, click Uninstall Now.

4. The Uninstall Progress screen displays the progress of uninstall process.

5. In the Uninstallation Summary window, click Details for a detailed summary of the configuration information that was processed during uninstallation. Click Exit to end the program

NOTE It is recommended that you check the log file by clicking the Details button to identify errors. If you find errors, try to uninstall the Agent manually by removing the package `SUNWamwas` using the `pkgrm` command.

6. Remove the Realm Administrator from the AdminRole.

Uninstalling the Agent Using Command Line

1. From the directory where the agent is installed, enter the following command at the command line:

   ```
   # ./uninstall_wasagent -nodisplay
   ```

2. When prompted, **Please enter path to pick up java:** Type the full path where WebSphere JDK is located.

3. The following text displays:

```
Please select the type of uninstall to perform from the following
choices:
1. Full
2. Partial
```

To remove the product and all of the components from your system, type1 for Full. To remove some, but not all product components, type2 for Partial.

4. The following text displays:

```
Ready to Uninstall

1. Uninstall Now
2. Start Over
3. Exit Uninstallation
```

When prompted, **What would you like to do**? type 1 to begin uninstallation.

5. The following text displays:

```
Product                                      Result     More Info
1. Sun_TM_ONE Identity Server Policy Agent   Full       Available
2. Done
```

To see log information on the agent, enter 1. To exit the Uninstallation program, enter 2.

6. Remove the Realm Administrator from the AdminRole.

Removing Realm Administrator From AdminRole

1. Start the WebSphere Administrative Console using the following command:

   # *WAS_root_dir*/bin/adminclient.sh

2. In the WebSphere Advanced Administrative Console, choose Console > Security Center.
3. In the Security Center Window, click on the Administrative Role tab.
4. Select the AdminRole and click on Select.
5. In the Select Users / Groups - AdminRole screen, select the Realm Administrator and click on Remove. Click OK.
6. In the Security Centre window, click on Apply and then OK to apply the changes made.
7. Restart WebSphere for changes to take effect.

Troubleshooting Information

Installation Problems

Server Startup Error due to Agent Realm (Custom Registry) initialization failure, after the Agent installation

The realm initialization fails when the Realm is unable to authenticate to the Identity Server. This may be due to one of the following reasons.

- Incorrect values were given for Realm Admin User (Security Server Id) and Realm Admin User Password (Security Server Password).You can try giving the correct values in the Administrative Console Security Center. Apply the changes and restart the WebSphere.
- Installer fails to add System Property for Application Server Instances, for the Identity Server SDK config directory (`com.iplanet.coreservices.config`). You can add this property `com.iplanet.coreservices.config=`*Agent_Install_Dir*`/SUNWam/wasAgent/amSDK/config/ums` to each Application Server instance, under JVM Settings in the Administrative Console. Again save the changes and restart WebSphere.
- If the Identity Server is running in SSL Mode, then verify the following.

    a. The certificate for the CA that issued the Certificate for the Identity Server, is imported into the KeyStore of the JVM used by the WebSphere. See “SSL Configuration.”

b. The System property `java.protocol.handler.pkgs` was added by the Agent installer to each Application Server instances. This can be found under JVM Settings for each Application server instance, in the WebSphere Administrative console. This property must be set to the value: `com.ibm.net.ssl.internal.www.protocol`

c. Verify the Identity Server configuration parameters in the `AmAgent.properties` file.

Access to the resource fails even after Authenticating to the Identity Server.

- If you are accessing a protected resource from an unprotected resource, make sure that you have security constraint (dummy) defined for the unprotected resource with the role mapped to any principal other than the special subject EveryOne.
- Make sure that the Web trust association is enabled, and WebSphere was restarted after enabling Web Trust Association.
- Make sure that the following lines are present in the trusted *WAS_root_dir*`/properties/trustedservers.properties`

  ```
  com.ibm.websphere.security.trustassociation.enabled=true

  com.ibm.websphere.security.trustassociation.types=amagent

  com.ibm.websphere.security.trustassociation.amagent.interceptor=
  com.sun.amagent.websphere.interceptor.AgentInterceptor

  com.ibm.websphere.security.trustassociation.amagent.config=AMAge
  nt
  ```

Uninstallation Problems

During the installation, the Agent Installer performs modifications for certain existing files on your system. In order that these modifications can be backed out completely, backups of these files are created before the modifications are made. In case you encounter any failures during the uninstall, it is possible to bring the system back to its original state manually by restoring the backed up files. These backed up files are copied under *Agent_Install_Dir*`/SUNWam/wasAgent/backup` directory. This directory is deleted if the uninstallation was completed successfully.

The following files are backed up during installation and restored during the uninstallation of the Agent, along with their original location.

- WebSphere Server Startup Script file:

  WAS_root_dir`/bin/startupServers.sh`

- Command-Line Setup Script file:

  WAS_root_dir/bin/setupCmdLine.sh

- Admin Server Configuration file:

  WAS_root_dir/bin/admin.config

- Trusted Servers file:

  WAS_root_dir/properties/trustedservers.properties

- Client Configuration file:

  WAS_root_dir/properties/sas.client.props

Also, some of the files such as admin.config may have undergone changes after the Agent Installation. These changes could be unrelated to the Agent. For this reason, the Agent uninstallation also creates a backup of the version of the above files, just before the uninstallation under the directory *WAS_root_dir*/amAgentBackup. These can be used to manually reapply those changes to the current copy of these files.

Known Problems

- Silent installation is not supported.
- Expanding the installer panels does not resize the screen.
- Command-line installation does not allow user to go back to previous step.
- When a request is passed from Identity Server to Web Agent the LDAP Attribute header name containing the lower case characters change to upper case, the header is prefixed with “HTTP” key, and hyphen (-) is changed to underscore (_). These changes are undone when the request is passed from Web Server to WebSphere Application Server.
- The uninstaller does not remove the package SUNWamwas if the WebSphere is configured manually.

Chapter 4

Policy Agent for Sun ONE Application Server 7.0

This chapter describes how to install and configure Sun ONE Identity Server Policy Agent for Sun ONE Application Server 7.0 running on Solaris 8, Solaris 9 and Windows 2000 Server platforms. Topics include:

- Overview
- Limitations
- Installing the Agent
- Application Configuration
- Agent Configuration
- Uninstalling the Agent

Overview

The Sun ONE Identity Server Policy Agent for Sun ONE Application Server consists of two main components that affect the operation of Application Server as well as the behavior of the protected application. These components are:

- **Agent Realm.** The Agent Realm component provides the ability to Application Server to interact with Identity Server's user and role information. This acts as the core of the Agent and has to be configured correctly in order for the Agent to function.

- **Agent Filter.** The Agent Filter component provides the ability to the hosted application to enforce Sun ONE Identity Server based authentication and is responsible for creating the Security Principal associated with the logged on user. Every application that has to be protected by the Agent must have its Deployment Descriptors changed to reflect that it is now configured to use the Agent Filter component. Applications that do not have this setting will not be protected by the Agent and may malfunction or become unusable if deployed on a Application Server where the Agent Realm component is installed.

Together, the Agent Realm and Agent Filter components work in tandem with Identity Server and enforce authentication and authorization for clients trying to access protected J2EE Applications.

Limitations

The Sun ONE Identity Server Policy Agent for Sun ONE Application Server 7.0 is designed to enforce authentication and authorization for clients that interact with the application through the application server's web container. Typically, such clients are thin clients like web browsers that communicate with the application server's web container using the HTTP or HTTPS protocols.

When a client requests an application resource by accessing a URL of the application, the Agent Filter component intercepts such requests and enforces authentication as necessary. Once the client is authenticated, the Agent Realm component works with the application server to authorize the authenticated client for accessing a particular resource within the hosted application like an Enterprise JavaBeans component. Alternatively, for security aware applications that use the programmatic J2EE security APIs provided in the web container and Enterprise JavaBeans container, the Agent Realm component provides the underlying support necessary in order that such APIs respond to the caller correctly by evaluating the currently authenticated user's credentials.

If however, the client does not access the application via the HTTP or HTTPS protocol, thereby bypassing the web container of the application server, the Agent Filter component will not intercept the request. This is a likely case when the application is being accessed by a rich client using other protocols like RMI over IIOP etc. Since the Agent Filter component which resides in the web container is bypassed in such cases, the authentication cannot be enforced. Clients that do not get authenticated by the Agent Filter do not possess sufficient credentials for the Agent Realm component to evaluate the necessary J2EE security policies.

Therefore, such clients will be denied access to all protected resources. Alternatively, for security aware applications that use the programmatic J2EE security APIs provided in the application server will be given negative results by such APIs.

Installing the Agent

The Sun ONE Identity Server Agent for Sun ONE Application Server 7.0 may be installed on platforms—Solaris 8, Solaris 9, or Windows 2000 Server. The following sections provide the detailed steps necessary for installing the Agent on each of these platforms.

Pre-installation Tasks

The following tasks must be completed before the Agent can be installed:

- **Install Sun ONE Application Server 7.0**

  Refer Sun ONE Application Server documentation for the necessary details. When the server is installed, test the installation by using the provided sample Applications to ensure that the server is installed correctly.

- **Test the deployment of application that you intend to protect**

  Before installing the Agent, it is important that the application that must be protected be deployed and tested for simple functionality. Once it is established the application can be deployed successfully, you are ready to install the Agent.

- **Stop the Application Server instance on which you want to install the Agent.**

  Stop the Application Server instance using the following command: `/`*`S1AS_Install_Dir`*`/appserv/domains/`*`domain-instance`*`/`*`server-instance`*`/bin/stopserv`

Launching the Installer on Solaris Platform

The binaries for Sun ONE Identity Server Policy Agent for Sun ONE Application Server 7.0 for Solaris 8 and Solaris 9 platform is provided as a tar-gzip archive. Copy this archive on the machine where Sun ONE Application Server is installed. Perform the following steps to launch the installation program:

1. Login as root.
2. Unzip the binary archive in a convenient location using the following command:

   ```
   # gzip -dc as70agent-1.0-domestic-us.sparc-sun-solaris2.8.tar.gz
   | tar xvf -
   ```

3. Set your `JAVA_HOME` environment variable to JDK version 1.4.0 or higher. If your system does not have the required version of JDK, use the JDK supplied with Sun ONE Application Server 7.0. This JDK is located under:

   S1AS_Install_Dir`/jdk`

   The installer provides two types of interfaces—a graphical user interface (GUI) and a command line interface. In most cases, the GUI installer can be used for installing the Agent. However, in cases when you are installing the Agent over a telnet session on a remote server and do not have windowing capabilities, then the GUI installer cannot be used. In such a case it is recommended that you use the command line installer for installing the Agent. This can be launched by executing the `setup` script and passing in a command line argument `-nodisplay` as follows:

   ```
   # ./setup -nodisplay
   ```

   However, if you choose to use the GUI installer, then it is required that you set your `DISPLAY` environment variable to ensure that the GUI installer window appears on the correct console.

NOTE If you choose to use the command-line based installer using the `-nodisplay` option, you may skip the following step and proceed directly to the next section, which details out the installation procedure.

4. Launch the GUI installer by invoking the `setup` script as follows:

   ```
   # ./setup
   ```

   The installer requires that you setup your `JAVA_HOME` variable correctly as pointed out in the Step 3. However, in case you have incorrectly set the `JAVA_HOME` variable, the `setup` script will prompt you for supplying the correct `JAVA_HOME` value:

   ```
   Enter JAVA_HOME location (Enter "." to abort):
   ```

   Type the full path to the Sun ONE Application Server JDK installation directory for launching the installation program. Otherwise, enter a period (`.`) to abort the installation.

   In order that the GUI installer be displayed on your console, the DISPLAY environment variable of your shell must be set correctly. If your DISPLAY environment variable is not set at the time of invoking the `setup` script, the installer will prompt you for the `DISPLAY` environment variable value as follows:

   ```
   Please enter the value of DISPLAY variable  (Enter "." to abort):
   ```

   Provide the `DISPLAY` value by typing the exact value at the above prompt. Otherwise, enter a period (`.`) to abort the installation.

NOTE You can also use `agent_platform.class` file to install the agent. You can find this file in the directory where you have untarred the binaries.

Launching the Installer on Windows 2000 Server

The binaries for Sun ONE Identity Server Policy Agent for Sun ONE Application Server 7.0 for Windows platform are provided as a zip archive. Copy this archive on the machine where Application Server server is installed. Perform the following steps to launch the installation program:

1. You must have administrative privileges when you run the installation program. If you do not have administrative privileges, either log on as the "Administrator" user or request such privileges to be granted to your account by the system administrator of the machine or domain as applicable.

2. Unzip the Agent binaries in a convenient location. This may be done by using any Zip utility available on your system.

   The above operation results in two executable files `setup.bat` and `setup.exe`, which may be used to launch the installer. Each of these two files provide different features for launching the installer. You can choose either of these two files depending upon your installation requirements.

   Using setup.bat

   In order to use the `setup.bat` file to launch the installer, you must have JDK version 1.4.0 or higher. This can be verified by typing the following command in a command prompt window:

   ```
   C:\> java -version

   java version 1.4.0_01

   Java(TM) 2 Runtime Environment, Standard Edition (build
   1.4.0_01-b03)

   Java HotSpot(TM) Client VM (build 1.4.0_01-b03, mixed mode)
   ```

   If you do not have JDK of required version in your system path, you can use the JDK supplied with Application Server 7.0 located at:

   S1AS_Install_Dir`\jdk`

   The `setup.bat` can be executed by typing the file name at the command prompt window in a directory where it is present, or by double clicking the file in Windows Explorer. For example:

   ```
   C:\>setup.bat
   ```

   The installer provides two types of interfaces—a graphical user interface (GUI) and a command line interface. By invoking the `setup.bat` file from a command prompt window or by double clicking the file in Windows Explorer, the installer is launched in the GUI mode. To launch the installation program

with the command line argument `-nodisplay`, type the following:

```
C:\>setup.bat -nodisplay
```

Using setup.exe

You can invoke `setup.exe` either from the command prompt window or by double clicking the file from Windows Explorer. Unlike `setup.bat`, `setup.exe` does not take the `-nodisplay` command line argument and can launch only the GUI installer.

NOTE Since the command line installer cannot be launched using `setup.exe`, in cases where the command line installer is required, it is recommended that you use `setup.bat` to launch the command line installer.

Installing the Agent Using GUI

The installation program begins with a Welcome screen. Click Next to step through the installation screens, answering the questions.

1. Read the License Agreement. Click Yes (Accept License) to continue with the Installation.

   If you do not accept all the terms of the Software License Agreement, click No to end the Installation program without installing the product.

2. In the Select Installation Directory screen, enter the path where you want to install.

   If you wish to install the Agent in a directory different from the default directory, click the Browse button and choose the directory. Once you have selected the appropriate directory, click the Next button to proceed to the next screen.

NOTE If you select a directory that does not exist on your system, the installer will prompt you to specify if the new directory should be created. You can either choose to create this new directory by clicking on the Create Directory button or select a new directory by clicking on the Choose another Directory button.

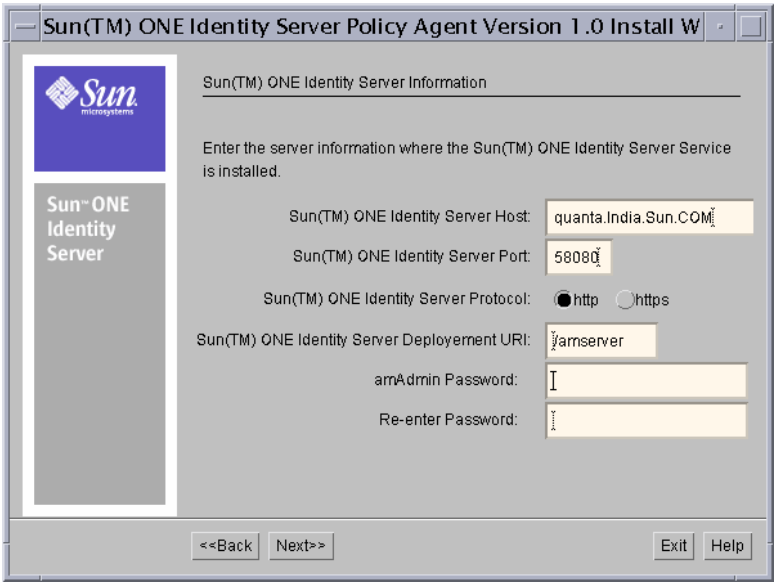
3. In the Select an Agent screen, select the component you wish to install in this by selecting the check box against the component name. The only component available for installation is the Sun ONE Identity Server Policy Agent for un ONE Application Server 7.0, which is selected by default.

NOTE For a given system, only one installation of Sun ONE Identity Server Policy Agent for Sun ONE Application Server 7.0 is allowed at a time. If a previous installation of the Agent has not been removed completely from this system, the selection box will be disabled and no selections can be made. In that case it is recommended that you exit the installer by clicking the Exit button and remove the previous installation completely before starting the install again.

Figure 4-1 Component Selection Screen

4. In the Sun ONE Identity Server Information screen, provide the following information about the Sun ONE Identity Server and click Next.

Figure 4-2 Sun ONE Identity Server Information Screen



Sun ONE Identity Server Host: Enter the fully qualified host name of the system where Sun ONE Identity Server is installed.

Sun ONE Identity Server Port: Enter the port number for the Web Server that runs Sun ONE Identity Server Services.

Sun ONE Identity Server Protocol: Select the protocol that will be used by the Agent to communicate with Sun ONE Identity Server services. This protocol may either be HTTP or HTTPS.

Sun ONE Identity Server Deployment URI: Enter the URI that should be used for accessing Sun ONE Identity Server Services.

amAdmin Password: Enter the password for amAdmin user.

Re-enter Password: Re-enter the password for amAdmin user for confirmation.

NOTE The password supplied during installation is recorded by the Agent in a secure manner. However, if in the future you change this password in Identity Server, you will have to update the password in the Agent as well. This can be done by using the `agentadmin` tool provided with the Agent. Once the Agent has been installed on the system, the `agentadmin` tool can be invoked from the location: *Agent_Install_Dir*`/SUNWam/asAgent/bin/agentadmin`

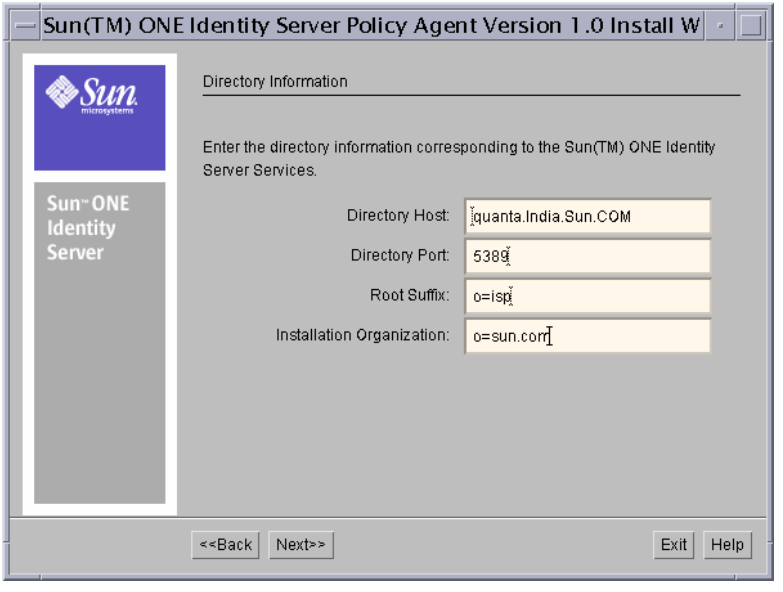
The `agentadmin` tool is available as `agentadmin.bat` on the Windows platform.

To change the password, invoke this tool as follows:

`#./agentadmin -password` *oldpassword newpassword*

5. In the Directory Information screen, provide the following information about the Directory Server that is associated with Sun ONE Identity Server services.

Figure 4-3 Directory Information Screen



Directory Host: Enter the fully qualified host name of the system where the Directory Server is installed.

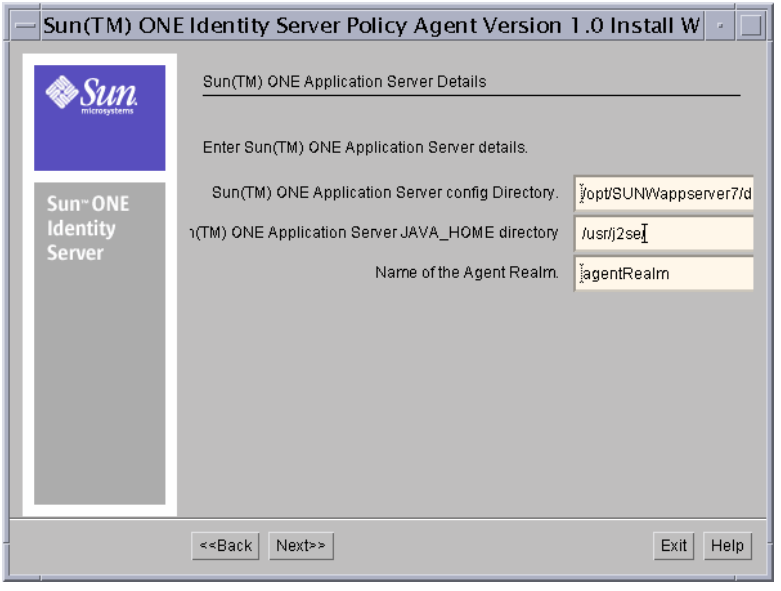
Directory Port: Enter the port number used by the Directory Server.

Root Suffix: Enter the root suffix to be used with this Directory Server.

Installation Organization: Enter the name of the installation organization as used when installing the Sun ONE Identity Server.

6. In the Sun ONE Application Server Details screen provide the following information about the Sun ONE Application Server on which the Agent is installed.

Figure 4-4 Sun ONE Application Server Details Screen



Sun ONE Application Server config Directory: Enter the full path to the location of the config directory of the Application Server instance.

Sun ONE Application Server JAVA_HOME directory: Enter the full path to the location of the JDK installation home used by the Sun ONE Application Server.

Name of the Agent Realm: Enter the name of the Realm that will be configured as the default realm to be used by Sun ONE Application Server.

The Sun ONE Application Server `config` directory is used to modify `server.xml`, `login.conf` and `server.policy` under this directory. Default location is the following directory:

S1AS_Install_Dir`/appsrv/domains/`*domain-instance*`/`*server-instance*`/config`

The Sun ONE Application Server JAVA_HOME directory refers to the JDK installation that is used by the Sun ONE Application Server. Typically the value of this is the full path to the following directory:

S1AS_Install_Dir`/jdk`

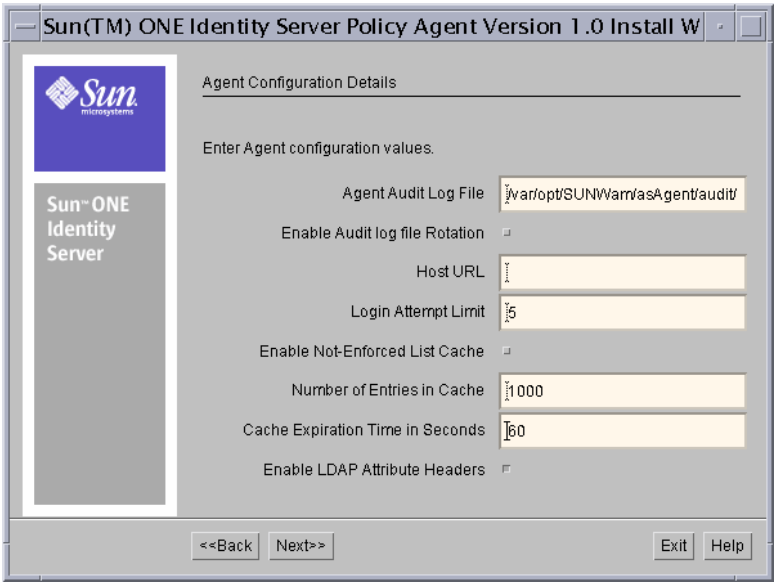
In case the value supplied is incorrect or is not the JDK used by the Application Server, it will result in malfunction of the Agent. To avoid this problem, ensure that the value you specify for Sun ONE Application Server `JAVA_HOME` is accurate.

CAUTION

- The installation program modifies the Application Server's `server.xml` file to include certain libraries in the Application Server `CLASSPATH`, as well as adds certain required parameters to the JVM options. Also it add the Agent Realm and makes it default Realm to be used. If you specify an incorrect value for the config directory, the necessary classes and parameters will not get added, resulting in malfunction of the Agent, which can render the Sun ONE Application Server unusable. To avoid this problem, ensure that the value you specify for Sun ONE Application Server config directory is accurate.
- The Installer modifies the Application Server's `login.conf` file to define `LoginModule` for the jaas-context.
- The Installer modifies the Application Server's `server.policy` to give `ProgramaticLogin` permission to `agent-filter.jar`.

7. In the Agent Configuration Details screen provide the key configuration information necessary for the Agent to function correctly.

Figure 4-5 Agent Configuration Details Screen



Audit Log File: Enter the complete path to the log file to be used by the agent to record Audit messages.

Enable Audit log file Rotation: Select this item to enable rotation of Audit Log files.

Host URL: Enter a valid URL to be used as the base URL by the Agent to redirect users as necessary. This value may be left blank.

Login Attempt Limit: Enter the number of unsuccessful access attempts in succession after which the user will not be allowed to access the requested URL temporarily for security purposes. Specify the value 0 to disable this feature.

Enable Not-Enforced List Cache: Select this item to enable caching of Not-Enforced List evaluation results.

Number of Entries in Cache: Specify the number of entries that cache can hold at a given instance.

Cache Expiration Time: Specify the time in seconds to be used as the limit for entries that can exist in the Not-Enforced List cache.

Enable LDAP Attribute Headers: Select this item to enable the passing LDAP attributes associated with the current user as HTTP Headers.

Important Tips

- The Audit Log file is a necessary requirement for the Agent. You may provide the name of a non-existing file on the system to be used as the Audit file. In which case, the Agent creates this file and the necessary directories in its path on first use. Alternatively, you can provide the file name of an existing file, which can be used by the Agent as well. However, in either case, it is required that the specified file has write permissions for the Application Server process. This is because the Agent executes in the same process as the Sun ONE Application Server. Providing an incorrect value for this will result in the malfunction of the Agent, which can render the Application Server unusable.

- When a request is intercepted by the Agent without sufficient credentials, the Agent redirects the user to the Identity Server's authentication service. Along with this redirect, the Agent also passes information regarding the original request to the authentication services, which is used to redirect the user back to the original requested destination. A part of this request is the Host URL that is used to identify the web container/server which the user was originally trying to access. This Host URL can be specified by setting the Host URL value on this screen. If the Host URL value is left blank, the Agent reconstructs the original Host URL from the request and uses that when redirecting the user to authentication service. The implication of having a Host URL is that no matter which web container/server that the user was originally trying to access, the user will be sent to the specified Host URL after successful authentication. While this configuration value can be used to override the default behavior, an incorrect value may lead to the application becoming inaccessible. It is therefore recommended that this value be left empty unless there is a specific need based on the deployment scenario, which calls for setting this value appropriately.

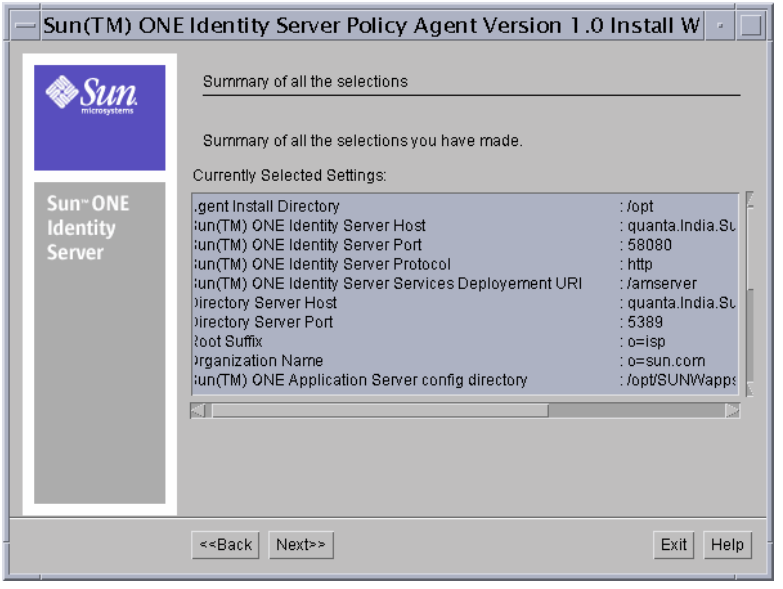
- The Login Attempt Limit feature can be used to guard the hosted application from Denial-Of-Service attacks where the end user can overload the application server by repeated authentication requests. By disabling this feature, the system remains vulnerable to such attacks. Therefore this feature should not be disabled unless there is a specific requirement necessitates the disabling of this feature.

- When the Agent must process a large list of Not-Enforced pattern rules specified in the configuration, every incoming request must be evaluated against every such rule to determine if the request can be allowed without authentication or not. In scenarios where the user load is high, the time spent in evaluating these rules can add up and degrade the overall performance of the system. To avoid this problem, it is recommended that Not-Enforced List Cache be enabled.

- Enabling the Not-Enforced List Cache may result in degraded performance if the values for Number of Entries in Cache and Cache Expiration time are not set up appropriately. In case when the cache expiration time duration is more than necessary, the cache will get filled up very fast and new requests will still be evaluated against all the specified pattern rules, resulting in no improvement in performance of the system. If the Number of Entries in Cache is set very high, it may result in excessive consumption of system memory, thereby leading to degraded performance. Therefore, these values must be set up after careful deliberation of the deployment scenario and should be changed as necessary to reflect changing usage scenario of the system. It is recommended that the system be tested with various values of these two parameters in a controlled environment to identify the optimal values, and only then be deployed in production.
- By enabling the LDAP Attribute Headers, for every incoming request, the Agent must retrieve the LDAP Attributes associated with the authenticated user and add them as Header values in the request. This feature should be used only in the case when the deployed application requires these header values for business logic implementation. Turning this feature on without an appropriate need will result in performance degradation of the system that can be easily avoided.

8. In the Summary screen review and verify the installation options that you have specified for the Agent. In case you find that certain values are in error, you may navigate back and correct them before proceeding further. Click the Next button to proceed with the installation.

Figure 4-6 Summary of all the selections Screen



9. In the Ready to Install screen, click the Install Now button to begin the installation.

10. The Install Progress screen displays the progress of the installation as the installer makes changes to your system. If necessary, you may interrupt this process by clicking the Stop button.

NOTE It is strongly recommended that you do not interrupt this process as this may lead to a partially installed product, which may also cause problems with uninstall, and may as well render the Application Server unusable. This process should be interrupted only in cases where it is absolutely necessary to do so.

11. In the Installation Summary window, click Details for a detailed summary of the configuration information that was processed during installation. Click Exit to end the program.

NOTE If the status of the installation is “Failed”, you must view the log file for the install by clicking on the Details button to identify which installation task failed. In this situation, you may uninstall the Agent and try again after fixing the root cause of failure during this installation.

The Agent installation program modifies the Application Server’s `server.xml` file to include certain libraries in the Application Server `CLASSPATH`, as well as adds certain required parameters to the JVM options. For the changes to take effect you must configure application server, which can be done by following either of these methods:

- To apply changes to application server instance using the Administration interface, perform the following steps:

  a. Access the Administration interface and click the name of the application server interface you want to reconfigure.

  b. Click the General tab.

  c. Click Apply Changes.

  When the changes are applied, the screen displays a message.

- To configure an application server instance using the command-line interface, execute the following command:

  `asadmin reconfig -u admin -w` *password* `-H` *hostname* `-p` *admin-port* `--keepmanualchanges=true` *server-instance*

NOTE `asadmin` is located at *S1AS_Install_Dir*`/bin` directory.

Installing the Agent Using Command-Line

You must have root permissions when you run the agent installation program.

1. Run the setup program with the command line argument `-nodisplay`. You’ll find the program in the directory where you untarred the binaries.

```
# ./setup -nodisplay
```

2. When prompted provide the following information:

   Have you read, and do you accept, all of the terms of the preceding Software License Agreement? Type Yes.

   Install Sun(TM) ONE Identity Server Agent in this directory: Enter the full path to the directory in which you want to install the Policy Agent.

3. The following text is displayed:

```
Sun(TM) ONE Identity Server Policy Agent components showing a
checked box will be installed. Only one agent may be installed at
a time.

[X]  1  Sun(TM) ONE Identity Server Policy Agent for Sun(TM) ONE
Application S

Select a component [0] {"<" goes back, "!" exits}:
```

   By default, 1 is already checked, type 0 or press enter key to install the Sun ONE Identity Server Policy Agent component.

4. Provide the following information about the Web Server that runs Identity Server:

   - Sun(TM) ONE Identity Server Host
   - Sun(TM) ONE Identity Server Port
   - Sun(TM) ONE Identity Server Protocol
   - Sun(TM) ONE Identity Server Deployment URI
   - amAdmin Password
   - Re-enter Password

   For details on each of these items, see “Installing the Agent Using GUI.”

5. Provide the following information about the Web Server that runs Identity Server Directory:

   - Directory Host
   - Directory Port
   - Root Suffix

- Installation Organization

For details on each of these items, see “Installing the Agent Using GUI.”

6. Provide the following information about the Sun ONE Application Server:
   - Sun(TM) ONE Application Server config Directory
   - Sun(TM) ONE Application Server JAVA_HOME directory
   - Name of the Agent Realm

   For details on each of these items, see “Installing the Agent Using GUI.”
7. Provide the configuration details for the installation of Sun ONE Identity Server Agent for Sun ONE Application Server.
   - Agent Audit Log File
   - Enable Audit log file Rotation
   - Host URL
   - Login Attempt Limit
   - Enable Not-Enforced List Cache
   - Enable LDAP Attribute Headers

   For details on each of these items, see “Installing the Agent Using GUI.”
8. When displayed, review the summary of the installation information you have specified. Press Enter to continue.

   The following text is displayed:

```
Ready to Install

1. Install Now
2. Start Over
3. Exit Installation
```

When prompted, **What would you like to do?** Type `1` to start the installation.

9. The following text is displayed:

```
Installation details:

Product                                      Result     More Info

1.Sun_TM_ONE Identity Server Policy Agent  Installed  Available
2.Done

Enter the number corresponding to the desired selection for more
information, or enter 2 to continue [2] {"!" exits}:
```

To see log information, type `1`. To exit the Installation program, type `2`.

Silent Installation

Silent installation provides a means for scripting the installation of Sun ONE Identity Server Policy Agent. When you perform a silent installation, you use a *StateFile*, to predefine all the answers that you would normally supply to the `setup` program interactively. This saves time and is useful when you want to install multiple instances of Sun ONE Identity Server Policy Agents using the same parameters in each instance.

Silent installation is a two-step process. First you generate a *StateFile* that contains all the parameter information the installation program needs. Then you run the silent installation program that automatically reads the parameters you've defined in the *StateFile*.

Silent Installation on Solaris

To Generate a StateFile

1. Run the installation program in the directory where the `setup` script is located. Enter the following command:

```
# ./setup -saveState StateFile
```

2. Proceed through the installation program, keeping in mind that your answers to the prompts are being recorded in the StateFile.

   Follow the instructions given in the section “Installing the Agent Using GUI,” on page 127.

   When installation is complete, the file *StateFile* is created in the same directory as `setup`.

To Run the Silent Installation

Enter the following command to run the silent installation:

```
# ./setup -nodisplay -noconsole -state StateFile
```

Silent Installation on Windows 2000

To Generate a StateFile

1. Run the installation program in the directory where the `setup` program is located. Open a DOS command prompt window and enter the following command:

   `setup.bat -saveState` *StateFile*

NOTE As an alternative, you can use the following command:

`java -classpath . agent_WINNT -saveState` *StateFile*

2. Proceed through the installation program, keeping in mind that your answers to the prompts are being recorded in the *StateFile*.

   Follow the instructions given in the section “Installing the Agent Using GUI,” on page 127.

   When installation is complete, the file StateFile is created in the same directory as `setup.exe`.

To Run the Silent Installation

Type the following command:

```
setup.bat -nodisplay -noconsole -state StateFile
```

NOTE As an alternative, you can use the following command:

```
java -classpath . agent_WINNT -nodisplay -noconsole
-state StateFile
```

Application Configuration

The Agent Realm component of Agent for Application Server 7.0 provides runtime mapping of various principals in Sun ONE Identity Server with abstract security role names used by the hosted application in order to determine if the currently authenticated user is authorized to access a particular resource or is otherwise a member of a given role. This runtime evaluation can occur only if the user is authenticated as a Sun ONE Identity Server principal by the means of Identity Server's authentication service. Without the user being authenticated appropriately, the results of such evaluations done by the Agent Realm will always be negative, resulting in access being denied to the user for the requested resource.

It is the Agent Filter component that enforces authentication for users who try to access particular application resources, thereby enabling the Agent Realm component to correctly evaluate the principal mappings as desired.

Unlike the Agent Realm component which is installed in the core of Application Server, the Agent Filter is installed in the deployed application, which must be protected by Identity Server. This is true for every application that must be protected on the Application Server using the Agent. It is also recommended that applications that are not protected using the Agent should not deployed on the Application Server on which the Agent Realm has been installed. This is to ensure that such applications can independently enforce their own security requirements as necessary. The presence of Agent Realm will interfere with the security evaluations done by such applications resulting in their malfunction.

Installing the Agent Filter Component in an Application

The Agent Filter can be installed by simply modifying the deployment descriptor of the application that needs to be protected. The following steps outline the process to install the Agent Filter component for a given application:

1. If the application is currently deployed on the Application Server, it must be removed using the Application Server's Administration Console or by the use of Application Server's deployment tools.
2. It is recommended that you create a backup of the deployment descriptor that will be edited in order to install the Agent Filter in this application.
3. Edit the application's `web.xml` deployment descriptor. Since the Filters were introduced in Servlet Specification 2.3, the `web.xml`'s `DOCTYPE` element must be changed to reflect that the deployment descriptor is a Servlet 2.3 compliant deployment descriptor. This can be done by setting the `DOCTYPE` element as:

   ```
   <!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
   Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
   ```

4. After the `DOCTYPE` element has been changed, add the Filter elements in the deployment descriptor. This can be done by specifying the Filter element and the Filter-mapping element in the `web.xml` deployment descriptor immediately following the description element of the `web-app` element. The following is a sample `web.xml` with the Filter and Filter-mapping elements.

```
<web-app>
  <display-name>...</display-name>
  <description>...</description>

  <filter>
  <filter-name>Agent</filter-name>
  <display-name>Agent</display-name>
  <description>SunTM ONE Idenitity Server Policy Agent for SunTM ONE
Application Server 7.0</description>
  <filter-class>com.sun.amagent.as.filter.AgentFilter</filter-class>
  </filter>
  <filter-mapping>
  <filter-name>Agent</filter-name>
  <url-pattern>/*</url-pattern>
  </filter-mapping>
  ...
  ...
</web-app>
```

5. Once the `web.xml` deployment descriptor has been modified to reflect the new `DOCTYPE` and filter elements, the Agent Filter has been added to the application.

Creating Role-to-Principal Mapping

Once the application has been configured to have the Agent Filter component in it, the Agent Filter will enforce authentication, thereby enabling the Agent Realm to successfully resolve the role-to-principal mappings. However, these mappings must first be created in order that the hosted application may use them during runtime.

To Create Role-to-Principal Mapping

- **Edit the Application Server specific deployment descriptors.**

  The Application Server specific deployment descriptors should be edited to create the role-to-principal mappings. These descriptors are in the files `sun-application.xml` and `sun-ejb-jar.xml` files. Refer Application Server reference documentation to learn the details on how these descriptors may be edited to create the role-to-principal mappings. Alternatively, you can refer Appendix C for sample descriptors which create such mappings.

Application-Specific Agent Configuration

Often the deployed applications are partitioned into public and protected parts, which have varying access restrictions. In most cases, the public portions of the application are accessible to anonymous users, whereas the protected portions of the application are accessible only to the registered users. The Agent can be configured to allow this type of access by letting anonymous users access the public portions of the application without requiring that they authenticate using Identity Server's authentication service. This information is provided as a part of the Agent's configuration properties file that is present in the following location:

Agent_Install_Dir`/SUNWam/asAgent/amAgent/config/AMAgent.properties`

This file may be edited to provide general as well as application-specific configuration for the Agent.

The properties specified in the `AMAgent.properties` file are required for the Agent to function properly. Invalid values specified in this file can lead to malfunction of the Agent, application becoming inaccessible, or the entire system to become unusable. It is recommended that you use extreme caution when modifying the values in this file and always create backups before such modifications to ensure that you can back out your changes to restore the system to its original state.

NOTE The properties specified in the `AMAgent.properties` file are loaded during the Application Server startup time. Any changes made to this file while the Sun ONE Application Server is running will not take effect until the server has been restarted.

Providing Application Specific Not-Enforced-List

In order to allow anonymous users to access portions of the application, the `AMAgent.properties` file must include an entry for the specific application. The entry that specifies this property is called the application not-enforced list and is identified by the string:

`com.iplanet.amagent.config.filter.`*`AppName`*`.notEnforcedList[`*`index`*`]=`*`pattern`*

This property requires that you format it correctly in order that it can be used by the Agent during runtime. The entries appearing in italics should be replaced by their appropriate values as follows:

AppName: This string should be replaced by the deployed application's context path. The context path is the first URI segment that is used to identify which application the user is trying to access. For example, if the user accesses your application by typing the URL:

`http://myserver.mydomain.com/SomeApp/index.html` or
`http://myserver.mydomain.com/SomeApp/SomeModule/doSometing.jsp`

then, in both these cases, the *AppName* is `SomeApp`.

index: This is an integer value starting from 0 for every deployed application and is unique for every entry in the application not-enforced list. For example, the following are two entries of application not-enforced list for an application called with context path `SomeApp`:

```
com.iplanet.amagent.config.filter.SomeApp.notEnforcedList[0]=/public/*
```

```
com.iplanet.amagent.config.filter.SomeApp.notEnforcedList[1]=/images/*
```

pattern: This is a pattern string that will be matched with the incoming request to evaluate if the request should be allowed to pass without enforcing authentication or not. The pattern string could be a specific URI, for example `/public/RegistrationServlet`, or could be a generic pattern using the wild card character '*' that can be used for denoting 0 or more characters in the request URI; for example `/public/*` will match with any URI that begins with `/public/`.

Using this property, you could specify pattern strings and URIs that the Agent will treat as not-enforced. In other words, user requests that match these particular patterns will be allowed to pass through without enforcing authentication.

Providing Application-Specific Access Denied URI

In cases when the Login Attempt Limit is enabled (refer to section “Installing the Agent” for details on how this feature can be configured, or refer “Agent Configuration” for details on this feature), the Agent is required to block the user's access. The default behavior of the Agent in this situation is to send an HTTP Status Code 403 Forbidden. In such a situation, the web container can display its preconfigured Forbidden page or simply send the status code in which case the user's browser displays the details of the error message in its own manner. While this is the default behavior of the Agent, it can be changed to suit the needs of the application by allowing the Agent to use an application-specific URI that will be used as the access denied error page.

This can be done by setting the following property in `AMAgent.properties` file:

`com.iplanet.amagent.config.filter.`*AppName*`.accessDeniedURI=/`*URI to use*

This property requires that you format it correctly in order that it can be used by the Agent during runtime. The entries appearing in italics should be replaced by their appropriate values as follows:

AppName: This string should be replaced by the deployed application's context path. The context path is the first URI segment that is used to identify which application the user is trying to access. For example, if the user accesses your application by typing the URL:

`http://myserver.mydomain.com/SomeApp/index.html` or

`http://myserver.mydomain.com/SomeApp/SomeModule/doSometing.jsp` then, in both these cases, the *AppName* is `SomeApp`.

URI to use: is an application specific URI that the Agent will use to locate the display page for blocking the user request. This URI can be a static HTML page, or a JSP or even a Servlet. However, this URI must be a part of the application itself. In other words, this URI must begin with */AppName/rest of the URI*.

Special Case: Default Web Application

A default web application in Application Server is accessible without providing any context path in the request URI. For example, the following URL is that of a default web application:

```
http://myserver.mydomain.com/index.html
```

Clearly, this URL does not have an associated context path.

For such applications, the Agent provides a convenient means of identifying that an entry is specific to the default web application. This is done in two steps as follows:

1. The following property is set to a name that represents the default web application:

   ```
   com.iplanet.amagent.config.filter.defaultWebAppName=DefaultWebAp
   p
   ```

2. This name is then used to specify the application not-enforced list as well as the application's access denied URI as follows:

   ```
   com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList
   [0]=/index.html
   ```

   ```
   com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList[
   1]=/about.html
   ```

   ```
   com.iplanet.amagent.config.filter.DefaultWebApp.accessDeniedURI=
   /URLAccessDenied.html
   ```

Using this scheme, the default web application that does not have a context path associated with it, may be configured just like any other application that has a context path. The same rules apply to the default web application for specifying the not-enforced list entries and access-denied URI as are applicable for the rest of the application. However, the only difference is that the access-denied URI of the default web application as well as the not-enforced list entries do not begin with the `/DefaultWebApp/` path segment since such a path segment does not exist on the application server in reality. It is only provided as a convenience to specify the properties associated with the default web application and not their values.

Global Agent Configuration

The `AMAgent.properties` file provides a way to specify a global not-enforced list, which will be applicable to all the deployed applications on the server. This list is specified by using the following property:

`com.iplanet.amagent.config.filter.global.notEnforcedList[0]=`*pattern*

where *pattern* is either an exact URI or a pattern specified by using the wild card character '*', which can be substituted for zero or more characters in the request URI.

Not-Enforced List Usage Considerations

Although the use of Not-Enforced List can be extremely helpful in partitioning your application for public and protected domains, it can also lead to undesirable effects if not used appropriately.

For example, if a request URI that represents a Servlet is matched by some not-enforced list pattern, then the Agent Filter will not enforce authentication for users who try to access that particular Servlet. However, consider the case where this Servlet access an Enterprise JavaBeans component that is protected by the Agent using role-to-principal mapping. In such a case, since the user is not authenticated, the access to the protected component will result in a an security violation exception being generated by the application server.

Therefore, before an entry is added to the not-enforced list, it must be ensured that it does in any way covers a resource that may be protected or may try to access a protected resource.

Another interesting aspect of the use of non-enforced list are the images. Typically, in a web page there are many images for various purposes like buttons, place holders, banners and logos. Every time the user accesses this page, the browser issues a request to the application server to get the images contained in this page. Each of such requests are treated as individual requests coming from the client and goes through the same evaluation mechanism for authentication and not-enforced list check as does any other request. This results in one client generating multiple calls to the server for displaying of a single page. Considering the overhead involved in enforcing authentication for every such request, it can impact the overall performance of the system. A solution to this problem is to have a global not-enforced list entry or entries that match all images. For example:

```
com.iplanet.amagent.config.filter.global.notEnforcedList[0]=*.gif
```

```
com.iplanet.amagent.config.filter.global.notEnforcedList[1]=/images
/*
```

This indicates that any request URI that ends with `.gif` will be not enforced and nor will be any URI that begins with `/images/`. In heavy user load situations, this can significantly increase the performance of the system.

Agent Configuration

The file `AMAgent.properites` located at: *Agent_Install_Dir*`/SUNWam/asAgent/amAgent/config` directory provides the core configuration needed by the Sun ONE Identity Server Policy Agent for Sun ONE Application Server 7.0 to function correctly. This property file provides many configuration settings that can be modified in order to customize the Agent's operation for your deployment scenario. This section provides a brief explanation of all the properties that are listed in the properties file.

Before proceeding, it is important to note that this file and the information within it are critical for the operation of the Agent. It is strongly recommended that you always create backup of this file before modifying it. Also it is strongly recommended that you do not modify this file unless you it is absolutely necessary. Please note that invalid data entries present in this file can lead to the malfunction of the Agent, the malfunction of the deployed applications, and could render the entire system unusable.

The settings provided in this file can be classified as follows:

- **Common Configuration:** Settings in this section are general settings that affect the behavior of the Agent as a whole.
- **Audit Configuration:** These settings are exclusively used to configure the Audit Engine used by the Agent.
- **Global Filter Configuration:** These settings are used to configure the Agent Filter component.
- **Application Filter Configuration:** These settings are used to configure the Agent Filter for a particular application.
- **Debug Engine Configuration:** These settings are used to configure the Debug Engine to generate diagnostic information.

Common Configuration

Organization Name

Key: `com.iplanet.amagent.config.org`

Description: This property specifies the organization name to be used when searching for principals in Identity Server.

Valid Values: A string that represents the organization name in Sun ONE Identity Server

NOTE	This property is set during Agent installation and need not be changed unless absolutely necessary.

Example: `com.iplanet.amagent.config.org=sun.com`

Root Suffix

Key: `com.iplanet.amagent.config.rootsuffix`

Description: This property specifies the root suffix to be used when searching for principals in Sun ONE Identity Server

Valid Values: A string that represents the root suffix in Sun ONE Identity Server

NOTE	This property is set during Agent installation and need not be changed unless absolutely necessary.

Example: `com.iplanet.amagent.config.rootsuffix=o=isp`

People Container Level

Key: `com.iplanet.amagent.config.realm.peopleContainerLevel`

Description: This property specifies the people container level to be used when searching for principals in Sun ONE Identity Server.

Valid Values: Non-zero unsigned integer representing the People Container Level in Identity Server, which may be used when searching for principals.

NOTE	This property is set during Agent installation and need not be changed unless absolutely necessary.

Example: `com.iplanet.amagent.config.realm.peopleContainerLevel=1`

Audit Configuration

Language Code

Key: com.iplanet.amagent.config.audit.localeLanguageCode

Description: This property specifies the Locale for Audit log messages.

Valid Values: The localeLanguageCode must be a valid ISO Language Code. Default value of this property is en

---

NOTE For more information, refer ISO 639 specification:

http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt

Example: com.iplanet.amagent.config.audit.localeLanguageCode=en

Country Code

Key: com.iplanet.amagent.config.audit.localeCountryCode

Description: This property specifies the Locale for Audit log messages.

Valid Values: The localeCountryCode must be a valid ISO Country Code. Default value of this property is US.

NOTE For more information, refer ISO 3166 specification:

http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

Example: com.iplanet.amagent.config.audit.localeCountryCode=US

Audit Log File

Key: com.iplanet.amagent.config.audit.logfile.name

Description: This property specifies the Audit log file to be used for recording Audit messages.

Valid Values: String representing the complete path name of the file to be used by Agent to record Audit messages.

NOTE Be sure the Application Server process has sufficient permissions to write to this file. Invalid value specified for this property may result in the failure of the system to start up correctly.

Example:
`com.iplanet.amagent.config.audit.logfile.name=/var/opt/SUNWam/asAgent/audit/agent.log`

Audit Log File Rotation Flag

Key: `com.iplanet.amagent.config.audit.logfile.rotate`

Description: This property specifies if the Audit log file should be rotated by the Agent.

Valid Values: `true/false`

NOTE Default value of this property is `false` and should be changed as necessary.

Example: `com.iplanet.amagent.config.audit.logfile.rotate=false`

Audit Log File Rotation Size

Key: `com.iplanet.amagent.config.audit.logfile.rotate.size`

Description: This property specifies the approximate size of the Audit log file in bytes, which should be used to evaluate when the log file needs to be rotated.

Valid Values: Non-zero unsigned integer indicating the size in bytes to be used to evaluate when the log file needs to be rotated.

NOTE Audit Log file rotation size is effective only when rotation flag is `true`.

Default value of this property is 52428800 bytes (~ 50 MB) and should be changed as necessary.

Example:
`com.iplanet.amagent.config.audit.logfile.rotate.size=52428800`

Global Filter Configuration

SSO Token Name

Key: `com.iplanet.amagent.config.filter.ssoTokenName`

Description: This property specifies the name of the Cookie that represents SSO Token.

Valid Values: A string that represents the name of SSO Token Cookie issued by Sun ONE Identity Server authentication service.

NOTE This property is set during Agent installation and need not be changed unless absolutely necessary.

Example:
`com.iplanet.amagent.config.filter.ssoTokenName=iPlanetDirectoryPro`

Login URL

Key: `com.iplanet.amagent.config.filter.loginURL`

Description: This property specifies the login URL to be used by the Agent to redirect incoming users without sufficient credentials to the Sun ONE Identity Server authentication service.

Valid Values: A string that represents the complete URL to be used as the redirect URL in order to send users without sufficient credentials to Sun ONE Identity Server authentication service.

NOTE This property is set during Agent installation and need not be changed unless absolutely necessary.

Example:
`com.iplanet.amagent.config.filter.loginURL=http://myserver.mydomain.com:58080/amserver/login`

Host URL

Key: `com.iplanet.amagent.config.filter.hostURL`

Description: This property specifies the host URL to be used by the Agent to reconstruct the request issued by the browser. This value is used by Sun ONE Identity Server Authentication service to redirect the user back to the original request destination after successful authentication.

Valid Values: A string value that represents the host URL that the user is expected to access in order that the Agent can intercept. This value must match the format of this property.

The format of this property value is as follows:

protocol://*hostname*.*optional-sub-domain*.*domain*:*port*

protocol can be `http` or `https`

hostname.*optional-sub-domain*.*domain* is the fully qualified host name that the user is expected to access in order that the Agent may intercept.

port is the port number on which the receiving web server is listening.

NOTE If left unspecified, the Agent will try to reconstruct the host URL from the request.

Example:

```
com.iplanet.amagent.config.filter.hostURL=http://www.iplanet.com:80
```

Login Attempt Limit

Key: `com.iplanet.amagent.config.filter.loginAttemptLimit`

Description: This property specifies the number of login attempts that a user can make without success using a single browser session, which will trigger the blocking of the user request.

Valid Values: Unsigned integer value, including 0, which indicates the number of unsuccessful login attempts that are allowed for any user trying to gain access to protected resources.

NOTE This option can be disabled by setting the value to `0`. The default value of this property is `5`.

Example: `com.iplanet.amagent.config.filter.loginAttemptLimit=5`

Login Counter Cookie Name

Key: `com.iplanet.amagent.config.filter.loginCounterCookieName`

Description: This property specifies the name of the cookie that will be used to track the number of unsuccessful login attempts made by the user.

Valid Values: A string that represents the name of the cookie to be issued by Agent in order to track the number of unsuccessful login attempts made by the user.

NOTE This property is set during Agent installation and need not be changed unless absolutely necessary.

Example:
`com.iplanet.amagent.config.filter.loginCounterCookieName=iPlanetLoginAttemptID`

Not-Enforced-List Cache Enable Flag

Key: `com.iplanet.amagent.config.filter.notEnforcedList.cache`

Description: This property specifies if the requests URIs that are evaluated as enforced or not-enforced may be cached to increase performance of the system.

Valid Values: `true/false`

NOTE The default value of this property is `true`.

Example:
`com.iplanet.amagent.config.filter.notEnforcedList.cache=true`

Not-Enforced-List Cache Size

Key: `com.iplanet.amagent.config.filter.notEnforcedList.cacheSize`

Description: This property specifies the number of entries that will be kept in the cache of not-enforced URIs and enforced URIs by the Agent.

Valid Values: Non-zero unsigned integer indicating the number of enforced as well as not enforced request URIs to be cached during runtime.

NOTE Values that are valid but not suited for the deployment scenario may result in degradation of system performance.

The value for optimal system performance will depend on the type of application deployed, the number of possible request URIs in the deployed application, the user load on the system, the expiration time set for cache entries and a host of other deployment specific factors.

To determine the most optimal value of this property, the application should be load tested in a controlled test environment before being deployed for production.

Example:
`com.iplanet.amagent.config.filter.notEnforcedList.cacheSize=1000`

Not-Enforced-List Cache Expiration Time

Key: `com.iplanet.amagent.config.filter.notEnforcedList.cacheTime`

Description: This property specifies the amount of time in seconds that will be used to evaluate if a cached entry can be removed from the cache to free up resources for new cache entries.

Valid Values: Non-zero unsigned integer indicating the time in seconds that will be used as the cache expiration time for entries in the cache during cleanup operation.

Values that are valid but not suited for the deployment scenario may result in degradation of system performance.

The value for optimal system performance will depend on the type of application deployed, the number of possible request URIs in the deployed application, the user load on the system, the expiration time set for cache entries and a host of other deployment specific factors.

To determine the most optimal value of this property, the application should be load tested in a controlled test environment before being deployed for production.

Example:
`com.iplanet.amagent.config.filter.notEnforcedList.cacheTime=60`

LDAP Attribute Header Enable Flag

Key: `com.iplanet.amagent.config.filter.enableLDAPAttributeHeaders`

Description: This property specifies if the Agent should populate the `HttpServletRequest` with LDAP Attributes associated with the currently authenticated user.

Valid Values: `true/false`

The default value of this property is `false` and should be changed as necessary.

Example:
`com.iplanet.amagent.config.filter.enableLDAPAttributeHeaders=true`

LDAP Attribute Header Map

Key: `com.iplanet.amagent.config.filter.ldapAttribute[`*attr-name*`]`

Description: This property specifies the LDAP Attributes to be populated under specific header names for the currently authenticated user.

Valid Values: Valid values must comply with the syntax of this property. The specified LDAP Attribute should be a valid attribute. The specified HTTP Header name should conform to HTTP Header name conventions.

The format for specifying this property is as follows:

`com.iplanet.amagent.config.filter.ldapAttribute[`*attr-name*`]=`*header-name*

attr-name is the name of the LDAP Attribute to be looked up for the authenticated user, and *header-name* is the name of the Header that will be used to store this value.

NOTE

- Be sure that the specified Header Names do not conflict with the existing Header names.
- Any number of such properties may be specified as long as they are valid properties conforming to the above stated requirements.

Example:

```
com.iplanet.amagent.config.filter.ldapAttribute[cn]=CUSTOM-Common-Name

com.iplanet.amagent.config.filter.ldapAttribute[ou]=CUSTOM-Organization-Unit

com.iplanet.amagent.config.filter.ldapAttribute[o]=CUSTOM-Organization

com.iplanet.amagent.config.filter.ldapAttribute[c]=CUSTOM-Country

com.iplanet.amagent.config.filter.ldapAttribute[mail]=CUSTOM-Email
```

```
com.iplanet.amagent.config.filter.ldapAttribute[employeenumber]=CUS
TOM-Employee-Number
```

LDAP Date Header Attribute Format String

Key: `com.iplanet.amagent.config.filter.ldapAttributeDateHeaderFormat`

Description: This property specifies the format of Date/Time value to be expected as a result of an attribute lookup. This is required when using the specialized get methods of the

`javax.servlet.http.HttpServletReqeust` interface that return Date values for headers.

Valid Values: Valid java.text.SimpleDateFormat Time Format Syntax string. For more information, see: documentation at

`http://java.sun.com/j2se/1.4/docs/api/java/text/SimpleDateFormat.ht
ml`

NOTE The default value of this property is set to `EEE, d MMM yyyy hh:mm:ss z` and should be changed as necessary.

Invalid value of this property may result in runtime exceptions in the application.

Example:

```
com.iplanet.amagent.config.filter.ldapAttributeDateHeaderFormat=EEE
, d MMM yyyy hh:mm:ss z
```

SSO Token URL Decode Flag

Key: `com.iplanet.amagent.config.filter.urlDecodeSSOToken`

Description: This property indicates if the SSO Token needs to be URL Decoded by the Agent before it may be used.

Valid Values: `true/false`

NOTE The default value of this property is set to true

Example: `com.iplanet.amagent.config.filter.urlDecodeSSOToken=true`

Default Web Application Name

Key: `com.iplanet.amagent.config.filter.defaultWebAppName`

Description: This property specifies a name for the Default Web Application deployed on the application server.

Valid Values: A string consisting of lower case and or upper case letters that can be used as the name for default web application.

NOTE

- The default value of this property is `DefaultWebApp`
- This property is necessary if the protected application is deployed as the default web application.

Example:

```
com.iplanet.amagent.config.filter.defaultWebAppName=DefaultWebApp
```

Global Not-Enforced List

Key: `com.iplanet.amagent.config.filter.global.notEnforcedList[`*index*`]`

Description: This property specifies a list of patterns that can be used to evaluate if the requested URI does not require the protection enforced by the Agent.

Valid Values: Valid values must comply with the syntax of this property. The valid values can be exact URIs or patterns consisting of wild-card character '*' to indicate zero or more characters.

The syntax of this property is as follows:
`com.iplanet.amagent.config.filter.global.notEnforcedList[`*index*`]=`*pattern*

index is an integer that starts from 0 and increments for every entry in this property list.

pattern is a string that represents request URIs that are not enforced by Agent.

NOTE

- The *pattern* string may consist of wild card character '*', which may match zero or more characters.
- The *index* must start from zero for the first entry and continue till the last in a sequence. Missing index values in this list will result in partial or complete loss of list entries.
- No value for this property indicates empty global not enforced list.

Example:

```
com.iplanet.amagent.config.filter.global.notEnforcedList[0]=*.gif
com.iplanet.amagent.config.filter.global.notEnforcedList[1]=/public
/*
com.iplanet.amagent.config.filter.global.notEnforcedList[2]=/images
/*
```

Application Filter Configuration

Access Denied URI

Key: `com.iplanet.amagent.config.filter.`*AppName*`.accessDeniedURI`

Description: This property specifies the application specific Access Denied URI for the protected application.

Valid Values: The URI within the deployed application that must be used as the access denied URI to block in coming requests when necessary.

This property is specific to the protected application. Therefore if there are more than one protected applications deployed on the system, there should be one property for each such application.

Note: This property must specify a URI that is within the application. Failing to do so can result in runtime internal server errors.

Note: The format for specifying this property is as follows:

`com.iplanet.amagent.config.filter.`*AppName*`.accessDeniedURI=`*URI*

AppName is the context path name for the deployed application and URI is the URI to be used.

NOTE

- The difference between *AppName* and context path of the application is that the context path has a leading / character.
- If the protected application is the default web application, the *AppName* should be set to the same string as specified in for the value of the property Default Web Application Name.
- If the property is not specified for a given application, the Agent uses HTTP Status Code 403 (Forbidden) to indicate a blocked access.

Example:

```
com.iplanet.amagent.config.filter.Portal.accessDeniedURI=/Portal/AccessDenied.html
```

```
com.iplanet.amagent.config.filter.BankApp.accessDeniedURI=/BankApp/Block.jsp
```

```
com.iplanet.amagent.config.filter.DefaultWebApp.accessDeniedURI=/URLAccessDenied.htm
```

Application Not-Enforced-List

Key:
`com.iplanet.amagent.config.filter.`*`AppName`*`.notEnforcedList[`*`index`*`]`

Description: This property specifies a list of patterns that can be used to evaluate if the requested URI does not require the protection enforced by the Agent for a particular application.

Valid Values: Valid values must comply with the syntax of this property. The valid values can be exact URIs or patterns consisting of wild-card character '*' to indicate zero or more characters.

The syntax of this property is as follows:

`com.iplanet.amagent.config.filter.`*`AppName`*`.notEnforcedList[`*`index`*`]=`*`pattern`*

AppName is the context path name without the leading '/' character for the deployed application.

index is an integer that starts from `0` and increments for every specified property for the particular application.

pattern is a string that represents the URIs that are not enforced by the Agent.

NOTE

- The *pattern* string may consist of wild card character '*', which may match zero or more characters.
- The *index* must start from zero for the first entry and continue till the last in a sequence. Missing index values in this list will result in partial or complete loss of list entries. The index value will be independent for different *AppName* specified in this property list.
- In case the protected application is the default web application, the *AppName* should be set to the same string as specified in for the value of the property Default Web Application Name.
- No value for this property indicates empty not-enforced list.

Example:

```
com.iplanet.amagent.config.filter.Portal.notEnforcedList[0]=/Portal
/GuestPages/*

com.iplanet.amagent.config.filter.Portal.notEnforcedList[1]=/Portal
/Registration/*

com.iplanet.amagent.config.filter.Portal.notEnforcedList[2]=/Portal
/WebServices/PollServlet

com.iplanet.amagent.config.filter.BankApp.notEnforcedList[0]=/BankA
pp/ModuleGuestTour/*

com.iplanet.amagent.config.filter.BankApp.notEnforcedList[1]=/BankA
pp/index.html

com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList[0]=
/index.html

com.iplanet.amagent.config.filter.DefaultWebApp.notEnforcedList[1]=
/about.html
```

Debug Engine Configuration

Debug Level

Key: `com.iplanet.amagent.config.debug.level`

Description: This property specifies the amount of debug messages that will be emitted by the Agent's Debug Engine.

Valid Values: Any of `0`, `1`, `3`, `7`, `15`, and `31`. These values indicate the following:

`0` : No debugging

`1` : Only Error messages

`3` : Error and Warning messages

`7` : Error, Warning and Brief Informational messages

`15`: Error, Warning and Verbose Informational messages

`31`: Error, Warning and Very Verbose Informational messages

NOTE

- For better performance of the system, this property should be set to a value `0`. Values other than that will affect the system performance depending upon the amount of information the Debug Engine has to emit.
- Values other than the ones specified in the Valid Values list above will lead to invalid configuration of the Debug Engine, thereby affecting the volume of messages that will be emitted.

Example: `com.iplanet.amagent.config.debug.level=7`

Debug Log File

Key: `com.iplanet.amagent.config.debug.logfile.name`

Description: This property specifies the Debug log file to be used for recording Debug messages.

Valid Values: String representing the complete path name of the file to be used by Agent to record Debug messages.

NOTE

- Be sure that the Application Server process has sufficient permissions to be able to write to this file.
- Invalid or empty value of this property will lead to Debug messages not being logged in the log file.

Example:

```
com.iplanet.amagent.config.debug.logfile.name=/debug/agent_debug.log
```

Debug Log File Rotation Flag

Key: `com.iplanet.amagent.config.debug.logfile.rotate`

Description: This property specifies if the Debug log file should be rotated by the Agent.

Valid Values: `true/false`

NOTE Default value of this property is false and should be changed as necessary.

Example: `com.iplanet.amagent.config.debug.logfile.rotate=false`

Debug Log File Rotation Size

Key: `com.iplanet.amagent.config.debug.logfile.rotate.size`

Description: This property specifies the approximate size of the Debug log file in bytes, which should be used to evaluate when the log file needs to be rotated.

Valid Values: Non-zero unsigned integer indicating the size in bytes to be used to evaluate when the log file needs to be rotated.

NOTE

- Default value of this property is 52428800 bytes (~ 50 MB) and should be changed as necessary.
- This property is not used if the Debug Log File Rotation Flag is set to false

Example:
`com.iplanet.amagent.config.debug.logfile.rotate.size=52428800`

Debug Time/Date Format String

Key: `com.iplanet.amagent.config.debug.date.format`

Description: This property specifies the format of time stamp that is used to mark the exact time when the Debug message was recorded.

Valid Values: Valid `java.text.SimpleDateFormat Time Format Syntax` string. For more information, see URL:

`http://java.sun.com/j2se/1.4/docs/api/java/text/SimpleDateFormat.html`

NOTE The default value of this property is set to:

MMM d, yyyy h:mm:ss a z 'Agent' and should be changed as necessary.

Invalid value of this property will result in loss of time stamp data with debug messages.

Example:

```
com.iplanet.amagent.config.debug.date.format=[yyyy/MM/dd HH:mm:ss
zzz]
```

Debug Print STDOUT Flag

Key: `com.iplanet.amagent.config.debug.print.stdout`

Description: This property specifies if the Debug Engine should print the debug messages on Standard Output stream.

Valid Values: `true/false`

NOTE

- The default value of this property is `true` and should be changed as necessary. When set to `true`, the Debug Engine prints all debug messages on the Standard output stream. This results in the debug messages being displayed on the console window where the Application Server startup script was executed from.
- This property has no affect on the ability of Debug Engine to write to debug log files.

Example: `com.iplanet.amagent.config.debug.print.stdout=true`

Uninstalling the Agent

When you install the Sun ONE Identity Server Policy Agent for Sun ONE Application Server software, an uninstallation program is created in the installation directory. Using this uninstallation program the Agent can be removed completely from your system. While the uninstallation program deletes all the installed files from your system, certain files such as audit log messages are not deleted. You can delete them manually.

The uninstallation program for Sun ONE Identity Server Policy Agent for Application Server should be launched according to the following steps for Solaris or Windows platform as applicable.

Launching the Uninstallation Program on Solaris

The uninstallation program for Solaris platform may be launched by executing the generated uninstall script located in the installation directory. The following steps provide details on how to achieve this:

1. Login as root.
2. Stop the Application Server instance using the following command:

   /S1AS_Install_Dir/`appserv/domains/`*domain-instance/server-instance/*`bin/stopserv`
3. Set your `JAVA_HOME` environment variable to JDK version 1.4.0 or higher. If your system does not have JDK of required version, use the JDK supplied with Sun ONE Application Server. This JDK is located under:

   S1AS_Install_Dir/`jdk`
4. The uninstallation program provides two types of interfaces—a graphical user interface (GUI) and a command-line. In most cases, the GUI installer can be used for uninstalling the Agent. However, in cases when you are uninstalling the Agent over a telnet session on a remote server and do not have windowing capabilities, then the GUI uninstallation program cannot be used. In such a case it is recommended that you use the command line uninstallation program for uninstalling the Agent. This can be launched by executing the uninstall script and passing in a command line argument `-nodisplay` as follows:

   ```
   #./uninstall_asagent -nodisplay
   ```

   However, if you choose to use the GUI uninstallation program, then it is required that you set your `DISPLAY` environment variable to ensure that the GUI uninstallation program window appears on the correct console.

NOTE If you choose to use the command-line uninstallation program using the `-nodisplay` option, you may skip the following step and proceed directly to the next section, which details out the uninstallation procedure.

---

5. Launch the GUI uninstallation program by invoking the uninstall script as follows:

   ```
   # ./uninstall_asagent
   ```

   The uninstallation program requires that you setup your `JAVA_HOME` variable correctly as pointed out in the Step 3. However, in case you have incorrectly set the `JAVA_HOME` variable, the uninstall script will prompt you for supplying the correct `JAVA_HOME` value:

   ```
   Enter JAVA_HOME location (Enter "." to abort):
   ```

   Type the full path to the JDK installation directory for launching the installation program. Otherwise, enter a period (`.`) to abort the uninstallation.

In order that the GUI uninstallation program be displayed on your console, the `DISPLAY` environment variable of your shell must be set correctly. In case your `DISPLAY` environment variable is not set at the time of invoking the uninstall script, the uninstallation program will prompt you for the `DISPLAY` environment variable value as follows:

```
Please enter the value of DISPLAY variable  (Enter "." to abort):
```

Provide the `DISPLAY` value to the installer by typing in the exact value at the above prompt. Otherwise, enter a period (`.`) to abort the installation.

Launching the Uninstallation Program on Windows 2000 Server

The uninstallation program for the Windows 2000 Server platform may be launched by executing the generated uninstall script located in the installation directory.

1. You must have administrative privileges when you run the uninstallation program. If you do not have administrative privileges, either login as “Administrator” or request such privileges to be granted to your account by the system administrator of the machine or domain as applicable.
2. Stop the Application Server instance using the following command:

   S1AS_Install_Dir`\appserv\domains\`*domain-instance*`\`*server-instance*`\bin\stopserv`
3. Go to the directory where Agent is installed.
4. The uninstall script `uninstall_asagent.bat` is located in this directory. In order to use the `uninstall_asagent.bat` script to launch the uninstallation program, you must have JDK version 1.4.0 or higher. This can be verified by typing the following command in a command prompt window:

   ```
   C:\> java -version

   java version “1.4.0_01”

   Java(TM) 2 Runtime Environment, Standard Edition (build
   1.4.0_01-b03)

   Java HotSpot(TM) Client VM (build 1.4.0_01-b03, mixed mode)
   ```

   If you do not have JDK of required version in your system path, you can use the JDK supplied with Application Server located at:

   S1AS_Install_Dir`\jdk`

The `uninstall_asagent.bat` may be executed by typing the file name at the command prompt window in a directory where it is present, or by double clicking the file in Windows Explorer. For example:

```
C:\Sun\uninstall_asagent.bat
```

The uninstallation program provides two types of interfaces—a graphical user interface (GUI) and a command line interface. By invoking the `uninstall_asagent.bat` file from a command prompt window as shown above or by double clicking it in Windows Explorer, the uninstallation program is launched in the GUI mode. However, in a case where it is it is required that you use the command line based uninstallation program for uninstalling the Agent, the uninstallation program may be launched by executing the `uninstall_asagent.bat` file and passing in a command line argument `-nodisplay` as follows:

Agent_Install_Dir`\uninstall_asagent.bat -nodisplay`

NOTE

- If the required JDK is configured to be in the system path, the uninstallation program can be launched from the Control Panel Add / Remove programs control. In the list of the installed programs on your system, select Sun ONE Identity Server Policy Agent for Application Server and click the Change / Remove button. If your system path is not configured with an appropriate JDK version, the uninstall can fail.
- When using the Control Panel Add / Remove programs to launch the uninstallation program for Agent, the uninstallation program will be launched in GUI mode only. To launch the uninstallation program in command line mode, use the provided uninstall script as mentioned previously.

Uninstalling the Agent Using GUI

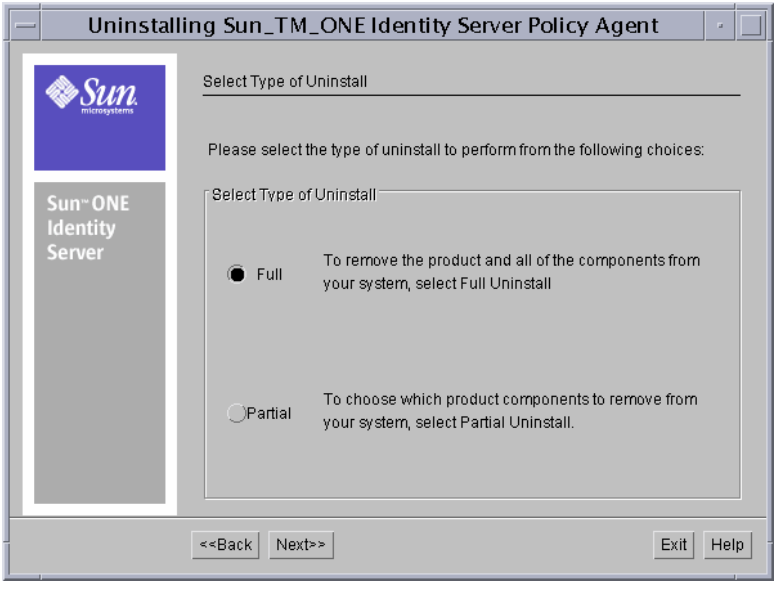
The uninstallation program begins with a Welcome screen. Click Next to step through the uninstallation screens, answering the questions.

1. In the Uninstall Type Selection screen select the type and click Next.

   Full: Select this type to remove the product and all the components from your system.

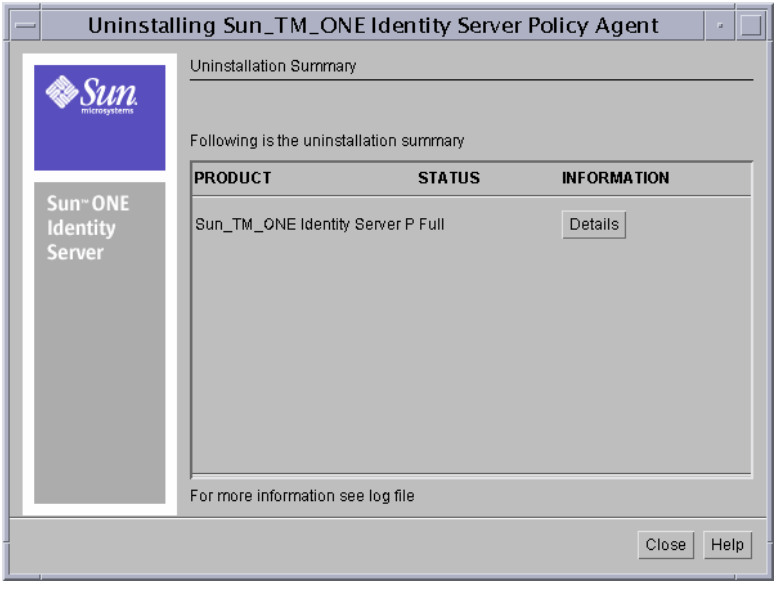
   Partial: Select this type if you want to remove only certain Agent components. Select the component you want to uninstall and click Next.

Figure 4-7 Uninstall Type Selection Screen



2. In the Ready to Uninstall screen, review the uninstallation information. If you need to make changes, click Back. Otherwise, click Uninstall Now.
3. The Uninstall Progress screen displays the progress of uninstall process.
4. In the Uninstallation Summary window, click Details for a detailed summary of the configuration information that was processed during uninstallation. Click Exit to end the program

Figure 4-8 Uninstall Summary Screen



NOTE If the status of the uninstall is “Failed”, you must view the log file for the uninstall by clicking on the Details button to identify which uninstall task failed. In certain situations these failures can be recovered from and the system can be restored to its original state. Refer to the next section for the detail of how to restore the system to its original state.

Uninstalling the Agent Using Command-Line

You must have root permissions when you run the agent installation program.

1. Run the uninstall program with the command line argument `-nodisplay`. You’ll find the program in the directory where you have installed the agent.

   ```
   # ./uninstall_asagent -nodisplay
   ```

2. The following text is displayed:

```
Please select the type of uninstall to perform from the following
choices:
1. Full
2. Partial

Choose one option from above [1]
```

Type `1` to uninstall full.

3. The following text is displayed:

```
Sun(TM) ONE Identity Server Policy Agent for Sun(TM) ONE
Application Server 7.0

Ready to Uninstall

1. Uninstall Now
2. Start Over
3. Exit
Choose one option from above [1] {"<" goes back, "!" exits}
```

Enter `1` to uninstall.

4. The following text is displayed:

```
Uninstallation Summary

Installation summary                    Summary Result More Details
1. Sun_TM_ONE Identity Server Policy   Full           Enter 1 to view the log

Done
```

To see log information, enter `1`. To exit the Uninstallation program, press Enter.

Chapter 5

Policy Agent for PeopleSoft 8.3

This chapter gives an overview of PeopleSoft Agent and describes how to install and configure Sun ONE Identity Server Policy Agent for PeopleSoft 8.3. Topics in this chapter include:

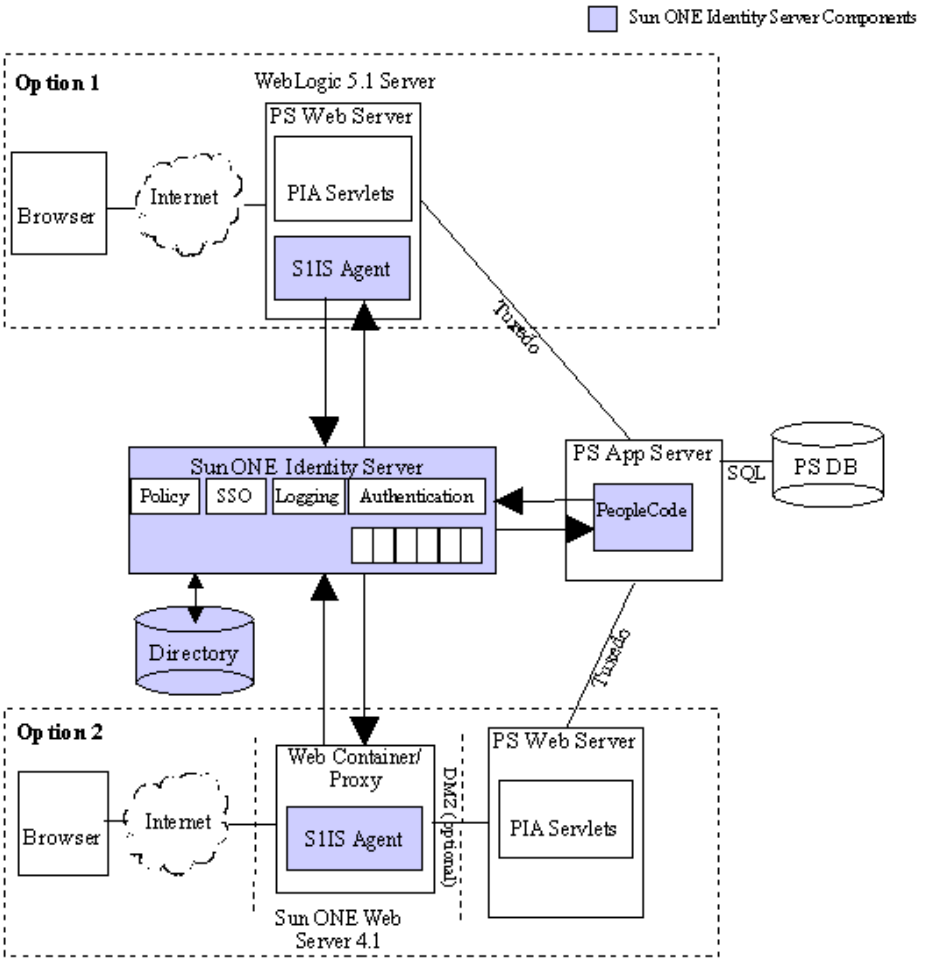
- PeopleSoft Agent Architecture
- Operating System and Software Required
- Installing the Agent
- Configuring Sun ONE Identity Server
- Verifying a Successful Installation
- Agent Configuration
- Troubleshooting Information
- Uninstalling the Agent

PeopleSoft Agent Architecture

Sun ONE Identity Server Policy Agent for PeopleSoft 8.3 protects web access to PeopleSoft 8.3 (HRMS Application) using Sun ONE Identity Server security features such as authentication, single sign-on and policy.

PeopleSoft applications are web-enabled through *PeopleSoft Internet Architecture* (PIA) that involves deploying PeopleSoft servlets on a web server (Apache or WebLogic) tier, which fronts the PeopleSoft proprietary application server. Sun ONE Identity Server Policy Agent can either be installed on the web server or on a web container that acts as proxy to the PeopleSoft web server.

Figure 5-1 PeopleSoft Agent Architecture



Sun ONE Identity Server Policy Agent is based on the Single Sign-on (SSO) infrastructure provided by PeopleSoft. For more information, see Chapter 7 of the *Security PeopleBook*.

The Sun ONE Identity Server Policy Agent code on the user's web server (either the proxy or the PeopleSoft provided web server) works like a regular Sun ONE Identity Server web agent. It intercepts all user-requests and is responsible for authenticating and authorizing web access to the PeopleSoft application based on the authentication schemes and policies defined in Sun ONE Identity Server.

The PeopleSoft configuration required for the agent to work is a three-step process:

1. Creating a default user in PeopleSoft.
2. Configuring the web server servlets to pass on the default user to the application server.
3. Configuring the application server to delegate the process of determining the user logged in to a special PeopleCode (PeopleSoft's proprietary language) provided by Sun ONE Identity Server.

   The rest of PeopleSoft configuration, specifically the existing PeopleSoft user repository, user profiles and PeopleSoft roles based access control do not need any changes, and work exactly like before.

The current implementation of identifying the user to PeopleSoft assumes that Sun ONE Identity Server and PeopleSoft separately maintain users, and a pre-determined LDAP attribute in the user's entry that contains the PeopleSoft userid. User synchronization between the two user repositories is beyond the scope of Sun ONE Identity Server or the Sun ONE Identity Server Policy Agents. While continuing to support this form of mapping, future implementations will provide support for installations that have a single user LDAP repository shared between Sun ONE Identity Server and PeopleSoft.

Deployment Options

The two deployment options are illustrated in the Figure 5-1.

Option 1) Agent on PeopleSoft Provided WebLogic 5.1 Server

This option requires redeploying the PeopleSoft application as a web application. The installer automatically makes necessary modifications. The AuthFilter implementation provided with the Agent gets called each time when a resource is accessed in this web application. The AuthFilter implementation is responsible for authenticating the user with Sun ONE Identity Server.

Option 2) Proxy Solution - Agent on a Separate Proxying Web Server

When you choose this option, the Sun ONE Web Server 4.1 (also referred as iPlanet Web Server or iWS) is configured as a proxy for WebLogic Server hosting PeopleSoft servlets. This option is recommended for the following reasons:

- More secure. Isolates the application business logic contained in PeopleSoft servlets and PeopleSoft Application server, thereby allowing the proxy server to be deployed in a Demilitarized Zone (DMZ) and hence minimizing attacks such as denial of service.
- Better Performance.
- Wider choice among Sun ONE Identity Server Policy agents that will act as the proxy. PeopleSoft provided web servers support only Apache and WebLogic 5.1. Sun ONE Identity Server web agents are available on a variety of web containers and hardware platforms.

Operating System and Software Required

Operating systems supported:

- Solaris 8
- HP-UX 11

Other required software:

- PeopleSoft-provided WebLogic 5.1 SP12 Server
- Sun ONE Identity Server 5.1
- Sun ONE Web Server 4.1 SP8
- Sun ONE Identity Server Policy Agent 1.1 for Sun ONE Web Server 4.1

Installing the Agent

The installation script installs the Sun ONE Identity Server Policy Agent. The following sections provide the steps necessary for installing the Agent on each of these platforms.

Pre-Installation Tasks

The following tasks must be completed before the Sun ONE Identity Server Policy Agent for PeopleSoft 8.3 can be installed:

- Determine which deployment option to choose by first identifying your scenario, and then follow the detailed instructions that correspond to that scenario.
- Install PeopleSoft 8.3 and configure the Web Server tier that will execute PeopleSoft servlets.
- If not already done, upgrade your WebLogic to sp12. To do this, you can download `weblogic510sp12.tar` from the following location:

  `ftp://ftp.peoplesoft.com/outgoing/GSC/jim/weblogic510`

  You can read the file `readme.html` available at this location for instructions on how to apply the service pack.
- Install the Sun ONE Web Server 4.1 in case you choose the proxy solution.

Overview of Installation and Configuration Tasks

In order to use the Sun ONE Identity Server Policy Agent for PeopleSoft 8.3, the following steps are necessary:

1. Setting up PeopleSoft for SSO
2. Installing the Sun ONE Identity Server Policy Agent
3. Installing Sun ONE Web Server 4.1 for Proxy Solution
4. Configuring Sun ONE Identity Server

Setting Up PeopleSoft for SSO

The following procedures include steps for setting up PeopleSoft for Single Sign-on (SSO).

To Create DEFAULT_USER

1. Invoke the web PeopleSoft application by entering the following URL:

   `http://`*`PS_web_server`*`:`*`port`*`/servlets/iclientservlet/peoplesoft8/?cmd=start`

2. Log on to PeopleTools as a privileged user. Example: `PS`
3. Navigate to Home > PeopleCode > Maintain Security > Use > UserProfiles screen then select "Add a New Value" and create a user.

4. In the User ID field, enter DEFAULT_USER.
5. In the Password field, enter the password.

   This user should not have any privileges / roles. Choose the default values or leave the non-mandatory fields blank.

NOTE You will need this password while configuring the Web server during setup.

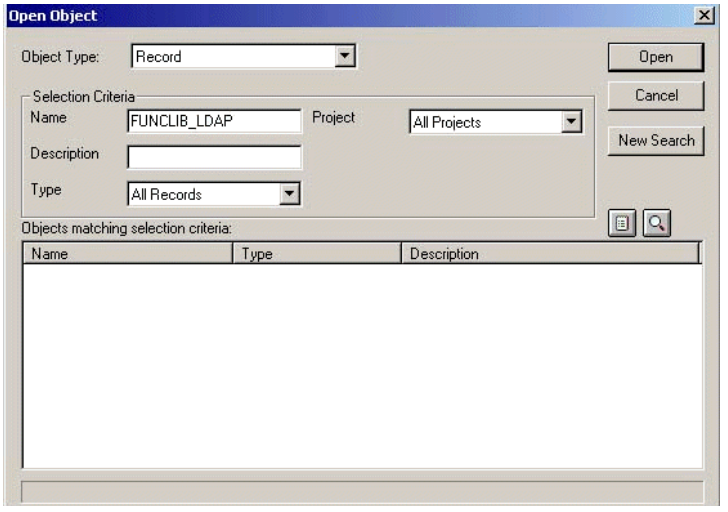
6. In the User Profiles page, open the ID tab and select the ID Type as None.

   DEFAULT_USER does the initial communication between PeopleSoft web server servlets and the PeopleSoft Application Server.
7. Click the Save button to save the changes.
8. Log out of the PeopleTools application.

To Install PeopleCode

1. Invoke Application Designer on the Windows NT machine where PeopleSoft application is installed. To do this, go to Start menu > Programs > PeopleSoft Installation > Application Designer.
2. Login as a privileged user. Example: PS
3. Navigate to the FUNCLIB_LDAP record. To do this, click File > Open.
4. In the Open Object window, select the Object Type Record.
5. In the name field, enter FUNCLIB_LDAP.
6. Leave the default values in the rest of the fields and click Open.

Figure 5-2 Open Object window



7. Select the LDAP Auth row and right-click the mouse button to select “View PeopleCode” option.
8. Append the contents of `ISPeopleCode.txt` to the end of the existing PeopleCode source. This file is provided with the Policy Agent binaries.
9. Search for Function getWWWAuthConfig()" in the PeopleCode:

```
Function getWWWAuthConfig()
&defaultUserId = "";
End-Function
```

10. Modify the variable &defaultUserId to return the User ID created in previous step as follows.

```
Function getWWWAuthConfig()
&defaultUserId = "DEFAULT_USER";
End-Function;
```

11. Click the Save button to save and exit from the Application Designer.

PeopleCode provided with the agent can be configured in four modes using the properties set in the `AMAgent.properties` file. For an explanation of these properties, see the section “Agent Configuration” on page 200.

- Validate SSO Token off, PeopleSoft userid retrieval off

This mode provides the least security. The Sun ONE Identity Server Policy Agent authenticates the user and a special HTTP header variable is set by the agent to identify the logged in user.

- Validate SSO Token off, PeopleSoft userid retrieval on

  This mode ignores the PeopleSoft userid in the HTTP header variable set by the agent, and retrieves the same from the Sun ONE Identity Server.

- Validate SSO Token on, PeopleSoft userid retrieval off

  This mode adds extra security by validating the SSO Token. If SSO Token is not valid, access is denied.

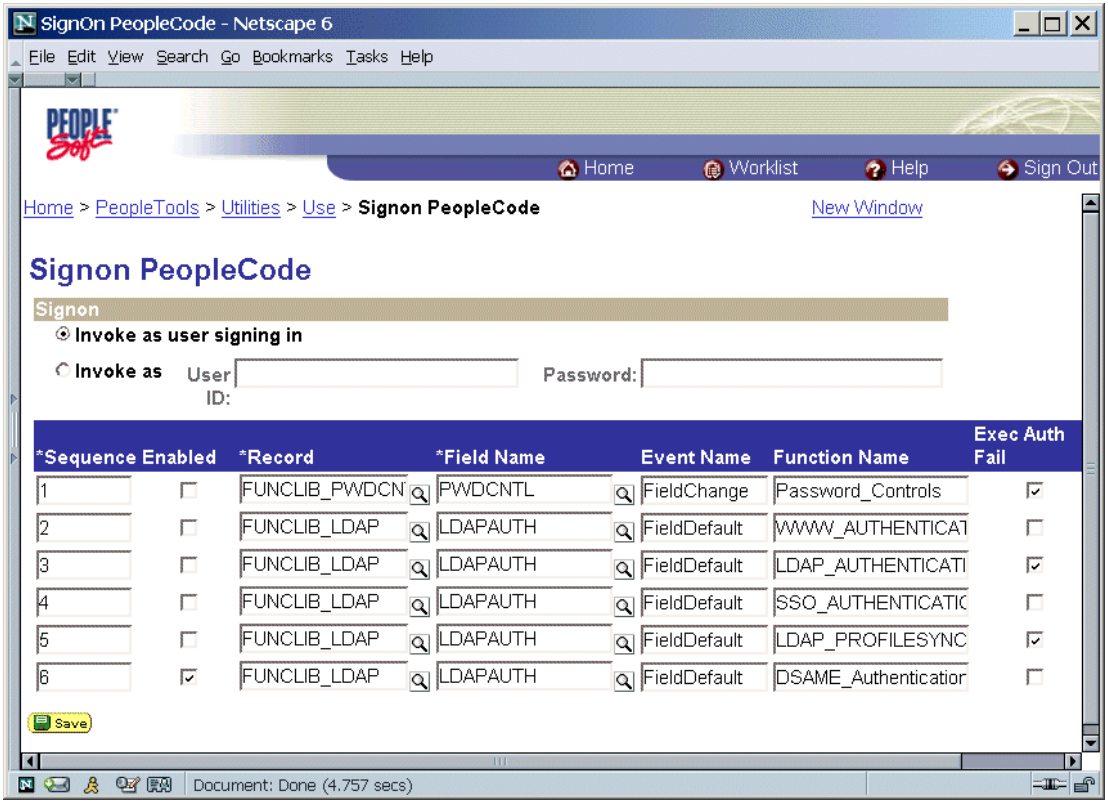
- Validate SSO Token on, PeopleSoft userid retrieval on

  This mode provides high security. The PeopleCode validates the SSO token and retrieves the mapped PeopleSoft userid ignoring the one specified in the HTTP header variable that is set by the agent.

To Register PeopleCode for Authentication

1. Navigate to Home > PeopleTools > Utilities > Use > SignOn PeopleCode window.
2. Add a new row, which should look like the 6th row shown in the Figure 5-3. To add a new row, click + on the last row.
3. Ensure that the FunctionName column shows DSAME_Authentication, Exec Auth Fail column is not selected, and the Enabled column is selected.

Figure 5-3 Sign-On PeopleCode Window



4. Click Save to save the changes.

Launching the Agent Installation Program

Launching the Agent Installation Program on Solaris Platform

Be sure the PeopleSoft application server and web server are shutdown before starting the installation program.

1. Login as a PeopleSoft user. Example: `psft`
2. Unpack the product binaries using the following command:

```
# gzip -dc ps83agent-1.0-domestic-us.sparc-sun-solaris2.8.tar.gz
| tar xvf -
```

3. Set your JAVA_HOME environment variable to JDK version 1.3.1 or higher. If your system does not have the required version of JDK, use the JDK supplied with PeopleSoft. This JDK is located at:

   PS_HOME/jre

4. Run the setup program. You'll find the program in the directory where you untarred the binaries. At the command line, type the following:

   ```
   # ./setup
   ```

   The installation program requires that you set up your JAVA_HOME variable correctly as pointed out in the Step 3. However, if you have set the JAVA_HOME variable incorrectly, the setup script will prompt you for the correct JAVA_HOME value:

   ```
   Enter JAVA_HOME location (Enter "." to abort):
   ```

   Specify the JAVA_HOME value by typing the full path to the JDK installation directory. Otherwise, enter a period (.) to abort the installation.

   In order that the GUI installation program be displayed on your console, the DISPLAY environment variable of your shell must be set correctly. If your DISPLAY environment variable is not set at the time of invoking the setup script, the installation program will prompt you for the DISPLAY environment variable value as follows:

   ```
   Please enter the value of DISPLAY variable (Enter "." to abort):
   ```

   Provide the DISPLAY value by typing the exact value at the above prompt. Otherwise, enter a period (.) to abort the installation.

Launching the Agent Installation Program on HP-UX 11 Platform

1. Login as a PeopleSoft User. Example: psft

2. Unpack the product binaries using the following the command:

   ```
   # gzip -dc ps83agent-1.0-domestic-us.hppa1.0-hp-hpux11.00.tar.gz
   | tar xvf -
   ```

3. Set your JAVA_HOME environment variable to JDK version 1.3.1 or higher. If your system does not have the required version of JDK, use the JDK supplied with PeopleSoft. This JDK is located at:

   PS_HOME/jre

4. Run the setup program. You'll find the program in the directory where you untarred the binaries file. At the command line, enter the following:

```
# ./setup
```

The installation program requires that you set up your JAVA_HOME variable correctly as pointed out in the Step 3. However, if you have incorrectly set the JAVA_HOME variable, the setup script will prompt you for the correct JAVA_HOME value:

```
Enter JAVA_HOME location (Enter "." to abort):
```

Specify the JAVA_HOME value by typing the full path to the JDK installation directory. For example, *PS_HOME*/jre. Otherwise, enter a period (.) to abort the installation.

In order that the GUI installation program be displayed on your console, the DISPLAY environment variable of your shell must be set correctly. If your DISPLAY environment variable is not set at the time of invoking the setup script, the installation program will prompt you for the DISPLAY environment variable value as follows:

```
Please enter the value of DISPLAY variable (Enter "." to abort):
```

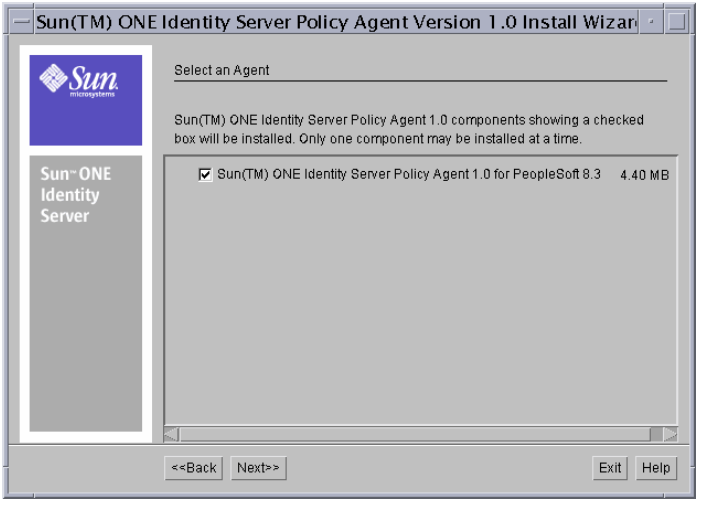
Provide the DISPLAY value by typing the exact value at the above prompt. For example, nila:0. Otherwise, enter a period (.) to abort the installation.

Installing the Agent Using GUI

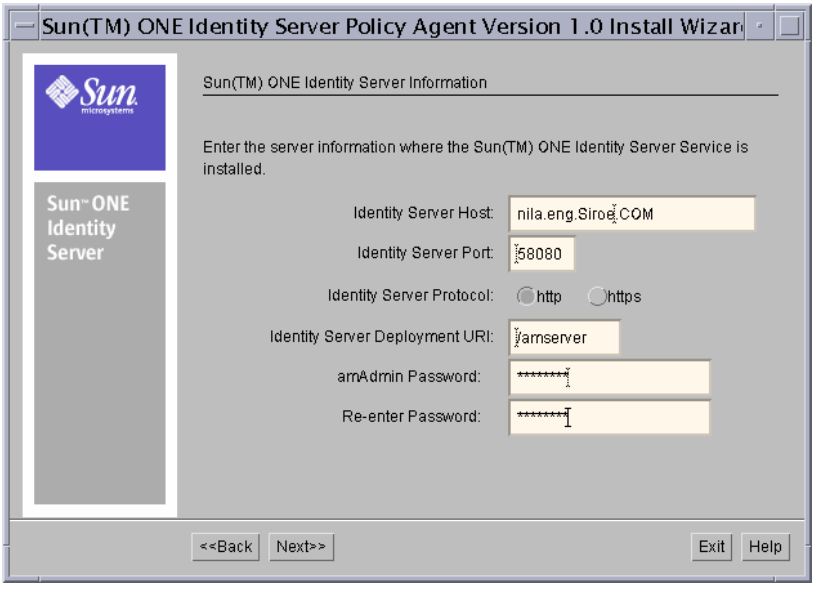
1. When you run the setup program, the installation GUI starts up. The opening panel informs you that you would need Administrative privileges in PeopleSoft to install the agent. Make sure that you have Administrative privileges in PeopleSoft before you proceed with installation.
2. Click Next to open the Welcome panel.
3. Click Next to open the Software License Agreement panel. Read the License Agreement and click Yes to agree to the license terms.
4. Click Next and in the Select Installation Directory panel, provide the following information.

Install Sun ONE Identity Server Policy Agent 1.0 in this directory: Enter the full path to the directory where you want this agent to be installed.

5. Click Next and in the Select an Agent panel, select **Sun ONE Identity Server Policy Agent 1.0 for PeopleSoft 8.3**.



6. In the Sun ONE Identity Server Information panel, provide information about the web server that runs Sun ONE Identity Server policy and management services. The URL Policy Agent will connect to this server.



Identity Server Host: Enter the fully qualified domain name of the system where the primary web server that runs Identity Server services is installed. Example: `myserver.siroe.com`.

Identity Server Port: Enter the port number for the web server that runs Identity Server services.

Identity Server Protocol: If the web server that runs Identity Server is SSL-enabled, select HTTPS; otherwise select HTTP.

Identity Server Deployment URI: Enter the location that was specified when Identity Server was installed. The default Universal Resource Identifier (URI) for Identity Server is `/amserver`.

amAdmin Password: Enter the password of the amAdmin user of Identity Server.

Re-enter Password: Enter the amAdmin user password again to confirm it.

7. In the Directory Server Information panel, provide the following information about the Directory Server that is associated with Sun ONE Identity Server services.

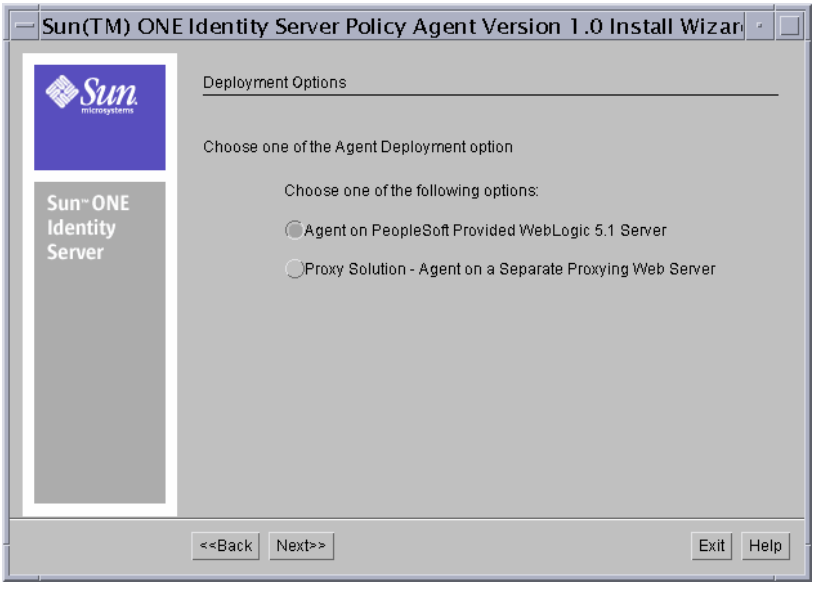
Directory Host: Enter the fully qualified host name of the system where the Directory Server is installed.

Directory Port: Enter the port number used by the Directory Server.

Root Suffix: Enter the root suffix to be used with this Directory Server.

Installation Organization: Enter the name of the installation organization used when installing the Sun ONE Identity Server.

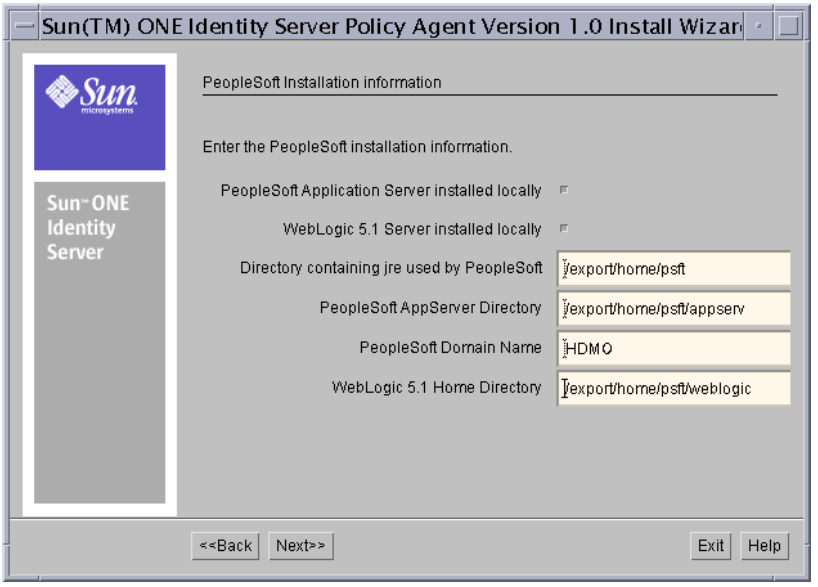
8. In the Deployment Options panel, select the mode for deploying the Policy Agent.



Agent on PeopleSoft Provided WebLogic 5.1 Server: Select this option to install the Agent on top of WebLogic 5.1 Server provided with PeopleSoft. This option requires redeploying the PeopleSoft application as a web application.

Proxy Solution - Agent on a Separate Proxying Web Server: Select this option if you want to configure Sun ONE Web Server 4.1 (also referred as iPlanet Web Server or iWS) as a proxy for WebLogic Server hosting PeopleSoft servlets.

9. In the PeopleSoft Installation Information panel, provide the following information about your existing PeopleSoft installation.



PeopleSoft Application Server Installed locally: Click this check box to specify that you have PeopleSoft Application Server installed locally.

WebLogic 5.1 Server Installed locally: Click this check box if you have WebLogic 5.1 Server installed locally.

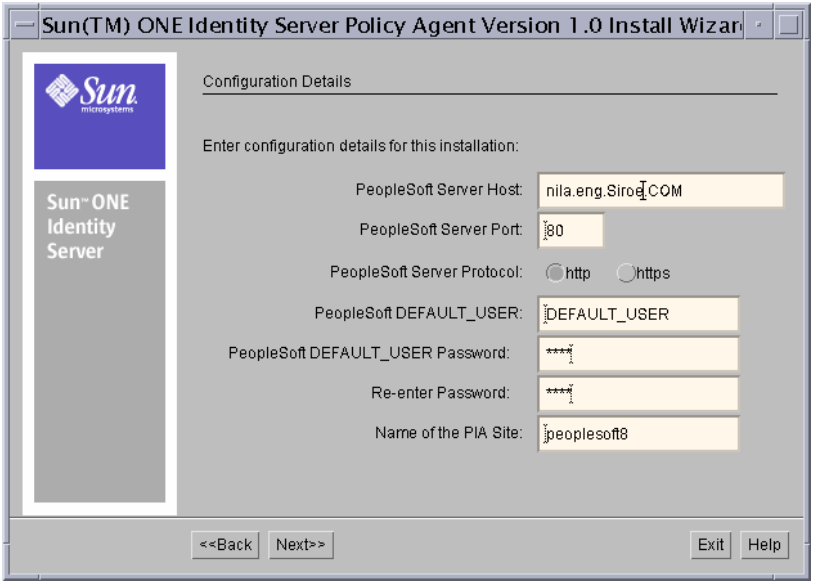
Directory containing JRE used by PeopleSoft: Enter the absolute path to the directory where the JRE used by PeopleSoft is available. For example, *PS_HOME*/jre.

PeopleSoft AppServer Directory: Enter the absolute path to the directory that contains the PeopleSoft AppServer. For example *PS_HOME*/appserv.

PeopleSoft Domain Name: Enter the HRMS domain that this agent will protect. Example: `HDMO`

WebLogic 5.1 Home Directory: Type the full patch to the directory where the application server is installed. Example: *PS_HOME*/`weblogic`

10. In the Configuration Details panel, provide the following information about how PeopleSoft is configured at your site. This panel is displayed only if, in the previous panel, you had checked the box WebLogic 5.1 Server Installed locally.



PeopleSoft Server Host: Enter the fully qualified host name of the PeopleSoft Web Server used to access the PeopleSoft Applications. Example: `peoplesoft.sun.com`

For Proxy Solution, enter the fully qualified host name of iWS 4.1.

PeopleSoft Server Port: Enter the port number used to access PeopleSoft Applications.

For Proxy Solution, enter the port number of iWS 4.1.

PeopleSoft Server Protocol: Enter the Web protocol used to access PeopleSoft Applications.

For Proxy Solution, use the web protocol of iWS 4.1.

PeopleSoft DEFAULT_USER: PeopleSoft SSO configuration parameter identifying a generic PeopleSoft userid that the web server uses to identify itself to the Application Server. Example: `DEFAULT_USER`.

Make sure it is same as the one configured for DEFAULT_USER. For more information, refer “To Create DEFAULT_USER.”

PeopleSoft DEFAULT_USER Password: Enter the password defined for the PeopleSoft DEFAULT_USER.

Re-enter Password: Enter the PeopleSoft DEFAULT_USER Password again to confirm the password.

Name of the PIA Site: Enter the name of the PIA site that was provided during PeopleSoft installation.

11. In the Summary of all the selections screen, review and verify the installation options that you have specified for the Agent. If you need to make changes, click Back. Otherwise, click Next to proceed.
12. In the Ready to Install screen, click Install Now to begin the installation.
13. In the Installation Summary panel, click Details for a detailed summary of the configuration information that was processed during installation. Click Exit to end the program.

NOTE A successful installation creates a file containing the environment variables necessary for the agent code installed PeopleSoft Application Server to execute correctly. This file is created under *PS_HOME*`/appserv/psdsameenv.sh`. You must add *PS_HOME*`/appserv/psdsameenv.sh` to the `$HOME/.profile` of the userid that executes PeopleSoft startup and shutdown.

Make sure that you execute `psdsameenv.sh` before starting the Application Server.

Installing from the Command Line

1. Run the command:

   ```
   ./setup -nodisplay
   ```

   The installation program starts. It provides instructions on the screen to help you proceed with the installation.
2. At the following prompt, type Yes or press Enter to continue.
3. At the following prompt, press Enter to display the Sun ONE Software License Agreement.
4. Read the agreement carefully and type Yes to accept it.
5. At the following prompt, provide the following information.

```
The Sun(TM) ONE Identity Server Policy Agent 1.0 components will
be installed in the following directory, which is referred to as
the "Installation Directory". To use this directory, press only
the Enter key. To use a different directory, type in the full path
of the directory to use followed by pressing
the Enter key.

Install Sun(TM) ONE Identity Server Policy Agent 1.0 in this
directory [/export/home/psft/agent] {"<" goes back, "!" exits}:
```

Install Sun ONE Identity Server Policy Agent 1.0 in this directory [export/home/psft/agent]: Press Enter if you want to install in the default directory prompted by the installation program. However, if you want install in a different directory, enter the full path to that directory.

If you enter a directory name that does not exist, the installation program asks whether you want to create the new directory and then creates it for you.

6. At the following prompt, press Enter to select Sun ONE Identity Server Policy Agent 1.0 for PeopleSoft 8.3.

```
Sun(TM) ONE Identity Server Policy Agent 1.0 components showing a
checked box will be installed. Only one component may be
installed at a time.

[X]  1  Sun(TM) ONE Identity Server Policy Agent 1.0 for
PeopleSoft 8.3
4.40 MB

Select a component" [0] {"<" goes back, "!" exits}
```

7. At the following prompts, provide information about the web server that runs Sun ONE Identity Server policy and management services. The URL Policy Agent will connect to this server.

```
Sun(TM) ONE Identity Server Services Details.
   Identity Server Host [nila.Eng.Siroe.COM] {"<" goes back, "!" exits}:
   Identity Server Port [58080] {"<" goes back, "!" exits}:
   Identity Server Protocol [http] {"<" goes back, "!" exits}:
   Identity Server Deployment URI [/amserver] {"<" goes back, "!" exits}:
   amAdmin Password [] {"<" goes back, "!" exits}:
Re-enter Password [] {"<" goes back, "!" exits}:
```

Identity Server Host: Enter the fully qualified domain name of the system where the primary web server that runs Identity Server services is installed. For example, `nila.Eng.siroe.com`.

Identity Server Port: Enter the port number for the web server that runs Identity Server services.

Identity Server Protocol: If the web server that runs Identity Server is SSL-enabled, select HTTPS; otherwise select HTTP.

Identity Server Deployment URI: Enter the location that was specified when Identity Server was installed. The default Universal Resource Identifier (URI) for Identity Server is `/amserver`.

amAdmin Password: Enter the password of the amAdmin user of Identity Server.

Re-enter Password: Enter the amAdmin user's password.

8. At the following prompts, provide the following information about the Directory Server that is associated with Sun ONE Identity Server services.

```
Sun(TM) ONE Identity Server Directory Details
   Directory Host [nila.Eng.Siroe.COM] {"<" goes back, "!" exits}:
   Directory Port [389] {"<" goes back, "!" exits}:
   Root Suffix [o=isp] {"<" goes back, "!" exits}:
Installation Organization [o=sun.com] {"<" goes back, "!" exits}:
```

Directory Host: Enter the fully qualified host name of the system where the Directory Server is installed.

Directory Port: Enter the port number used by the Directory Server.

Root Suffix: Enter the root suffix to be used with this Directory Server.

Installation Organization: Enter the name of the installation organization used when installing the Sun ONE Identity Server.

9. At the following prompt, select the mode for deploying the Policy Agent. Enter the number corresponding to an option to select it.

```
Deployment Options
1. Agent on PeopleSoft Provided WebLogic 5.1 Server
2. Proxy Solution - Agent on a Separate Proxying Web Server
Choose one of the options from above [1] {"<" goes back, "!" exits}
```

Agent on PeopleSoft Provided WebLogic 5.1 Server: Select this option to install the Agent on top of WebLogic 5.1 Server provided with PeopleSoft. This option requires redeploying the PeopleSoft application as a web application.

Proxy Solution - Agent on a Separate Proxying Web Server: Select this option if you want to configure Sun ONE Web Server 4.1 (also referred as iPlanet Web Server or iWS) as a proxy for WebLogic Server hosting PeopleSoft servlets.

10. At the following prompts, provide information about your existing PeopleSoft installation.

```
PeopleSoft installation Details
PeopleSoft Application Server installed locally [true] {"<" goes back,
"!" exits}
WebLogic 5.1 Server installed locally [true] {"<" goes back, "!" exits}
Directory containing jre used by PeopleSoft [/export/home/psft] {"<"
goes back, "!" exits}
PeopleSoft AppServer Directory [/export/home/psft/appserv] {"<" goes
back,"!" exits}
PeopleSoft Domain Name [HDMO] {"<" goes back, "!" exits} HIND
WebLogic 5.1 Home Directory [/export/home/psft/weblogic] {"<" goes
back, "!" exits}
```

PeopleSoft Application Server Installed locally: Press Enter to specify that you have PeopleSoft Application Server installed locally. Otherwise, enter false.

WebLogic 5.1 Server Installed locally: Press Enter if you have WebLogic 5.1 Server installed locally. Otherwise, enter false.

Directory containing JRE used by PeopleSoft: Enter the absolute path to the directory where the JRE used by PeopleSoft is available. For example *PS_HOME*/jre.

PeopleSoft AppServer Directory: Enter the path to the directory where the Application Server is installed. For example, *PS_HOME*/appserv

PeopleSoft Domain Name: Enter the HRMS domain that this agent will protect. Example: `HDMO`

WebLogic 5.1 Home Directory: Type the full patch to the directory where the application server is installed. Example: *PS_HOME*/`weblogic`

11. In the Configuration Details panel, provide the following information about how your PeopleSoft installation is configured. These prompts appear only if you had specified that WebLogic 5.1 Server is installed locally.

```
Configuration details for this installation.
   PeopleSoft Server Host [nila.Eng.Siroe.COM] {"<" goes back, "!" exits}:
   PeopleSoft Server Port [80] {"<" goes back, "!" exits}:
   PeopleSoft Server Protocol [http] {"<" goes back, "!" exits}:
   PeopleSoft DEFAULT_USER [DEFAULT_USER] {"<" goes back, "!" exits}:
   PeopleSoft DEFAULT_USER Password [] {"<" goes back, "!" exits}: 1
   Re-enter Password [] {"<" goes back, "!" exits}: 1
   Name of the PIA Site [peoplesoft8] {"<" goes back, "!" exits}:
```

PeopleSoft Server Host: Enter the fully qualified host name of the PeopleSoft Web Server used to access the PeopleSoft Applications. Example: `peoplesoft.siroe.com`

If you are deploying the agent as a Proxy Solution, enter the fully qualified host name of iWS 4.1.

PeopleSoft Server Port: Enter the port number used to access PeopleSoft Applications.

If you are deploying the agent as a Proxy Solution, enter the port number of iWS 4.1.

PeopleSoft Server Protocol: Enter the Web protocol used to access PeopleSoft Applications.

If you are deploying the agent as a Proxy Solution, use the web protocol of iWS 4.1.

PeopleSoft DEFAULT_USER: PeopleSoft SSO configuration parameter identifying a generic PeopleSoft userid that the web server uses to identify itself to the Application Server. Example: `DEFAULT_USER`

PeopleSoft DEFAULT_USER Password: Enter the password defined for PeopleSoft DEFAULT_USER.

Re-enter Password: Reenter the DEFAULT_USER password to confirm it.

Name of the PIA Site: Enter the name of the PIA site provided during PeopleSoft installation.

The installation program displays a summary of all your selections at the previous prompts.

12. Review the summary and at the prompt, press Enter to proceed.

13. At the following prompt, press Enter to install the Agent.

```
Following components will be installed:
PRODUCT: Sun_TM_ONE Identity Server Policy Agent
LOCATION: /export/home/psft/agent
SIZE: 4.40 MB
-----------------------------------------------
Sun(TM) ONE Identity Server Policy Agent 1.0 for PeopleSoft 8.3
Ready to install
1. Install Now
2. Start Over
3. Exit
   Choose one option from above [1] {"<" goes back, "!" exits}
```

14. The next prompt informs you that you have successfully installed the agent. At this prompt, press Enter to exit the installation program.

NOTE A successful installation creates a file containing the environment variables necessary for the agent code installed PeopleSoft Application Server to execute correctly. This file is created under *PS_HOME*/appserv/psdsameenv.sh. You must add *PS_HOME*/appserv/psdsameenv.sh to the $HOME/.profile of the userid that executes PeopleSoft startup and shutdown.

Make sure that you execute psdsameenv.sh before starting the Application Server.

Installing Sun ONE Web Server 4.1 for Proxy Solution

The following procedure includes steps for configuring the Sun ONE Web Server 4.1 as proxy for WebLogic Server hosting PeopleSoft servlets.

1. Install Sun ONE Web Server 4.1 and the supporting Sun ONE Identity Server Policy Agent. For more information on installing Sun ONE Policy Agent for Sun ONE Web Server 4.1 access the documentation at: http://docs.sun.com/source/816-5428-10/index.html

2. Download and install WebLogic plug-in for Sun ONE Web Server 4.1. To do this, make a directory *libproxy_Dir*, for example *web_server_dir*/*server_instance*/lib.

3. Go to *WebLogic_Dir* / lib / solaris.
4. Copy `libproxy.so` to *libproxy_Dir*.

   For detailed information see:
   `http://www.weblogic.com/docs51/admindocs/nsapi.html`
5. Make following modifications to the `obj.conf`, which is located at:

   Web_Server_Dir/*server_instance*/`config`

   a. Append the following lines at the beginning of the `obj.conf` file.

Code Example 5-1

```
Init fn="load-modules" funcs="wl-proxy,wl-init"\
    shlib=_libproxy_Dir/libproxy.so
Init fn="wl-init"
```

   b. Append the following lines at the end of the `obj.conf` file

Code Example 5-2

```
<Object name="weblogic" ppath="*">
Service fn=wl-proxy WebLogicHost=myserver.com\
   WebLogicPort=7001
</Object>
```

where:

Web_Server_Dir: The directory where Sun ONE Web Server is installed.

server_instance: The identity of the web server instance you want to configure.

Install_Dir: The installation directory of the WebLogic proxy libraries.

`myserver.com`: WebLogic hostname.

Object is given a name `weblogic`. This object instructs the web server to redirect all the URLs that match the pattern defined as the value of `ppath`. The library method invoked when such a pattern occurs, is defined as the value of the *fn* variable. The rest of the parameters are input parameters to the `wl-proxy` function itself.

The above example shows that all the requests to this server are serviced by the `wl-proxy` function. If you modify the `ppath=*/weblogic/*` then all URLs that matches the pattern `*/weblogic/*` are serviced by the `wl-proxy` function.

It is recommended that only the proxy web server be allowed to access the WebLogic 5.1 server hosting PeopleSoft servlets and all other accesses be blocked. You can do this either by:

- Installing a firewall on the machine executing the WebLogic server with appropriate firewall rules. For example, allowing http access only from the proxy web server machine, and denying all other http access.
- Enabling the `SimpleConnectionFilter` class on the WebLogic server. For more information refer the documentation at:

  `http://www.weblogic.com/docs51/examples/security/net/package-examples.security.net.html`

Configuring Sun ONE Identity Server

The following procedures help you configure the Sun ONE Identity Server.

Configuring Sun ONE Identity Server Policy and Authentication Server

Option 1

1. Log on to Sun ONE Identity Server Console as Administrator.
2. Select the LDAP attribute to be used to map Sun ONE Identity Server userids to PeopleSoft userids. Update the attribute for each Sun ONE Identity Server user that needs access to PeopleSoft.

Option 2

1. Log on to Sun ONE Identity Server Console as Administrator.
2. Set up appropriate URL policies and roles for access to PeopleSoft Application. Example:

```
Allow http://Web_server:port/servlets/*
```

3. Assign PeopleSoft policies and roles to users.
4. Select the LDAP attribute to be used to map Sun ONE Identity Server userids to PeopleSoft userids. Update the attribute for each Sun ONE Identity Server user that needs access to PeopleSoft.

NOTE PeopleSoft uses the mapped userid while performing its own authorization. No changes are necessary to PeopleSoft policy.

Verifying a Successful Installation

To verify a successful installation of Sun ONE Identity Server Policy Agent do the following:

- Try to access PeopleSoft Application by entering one of the following links depending on your deployment option. You should see the Sun ONE Identity Server login page.
  - If you have deployed the Agent on PeopleSoft-provided WebLogic 5.1 Server, use this link:

  ```
  http://PS_WebLogic_server:port/servlets/iclientservlet/peoplesoft8/?cmd=start
  ```

  - If you have deployed the Agent as a Proxy Solution, use this link:

  ```
  http://Web_Server:port/servlets/iclientservlet/peoplesoft8/?cmd=start
  ```

- Successful login to Sun ONE Identity Server should display the PeopleSoft Application with the appropriately mapped PeopleSoft user logged in. Unsuccessful login should not allow access to the PeopleSoft Application.
- Logout from the PeopleSoft Application should result in complete logout from PeopleSoft Application as well as Sun ONE Identity Server, and should return back to the Sun ONE Identity Server authentication page.

Agent Configuration

The file `AMAgent.properties` located at *Agent_Install_Dir*`/SUNWam/agent/config` directory provides the core configuration needed by the Sun ONE Identity Server Policy Agent for PeopleSoft 8.3 to function correctly. This property file provides many configuration settings that can be modified in order to customize the Agent's operation for your deployment scenario. This section provides a brief explanation of all the properties that are listed in the properties file.

Before proceeding, it is important to note that this file and the information within it are critical for the operation of the Agent. It is strongly recommended that you always create a backup of this file before modifying it. Also it is strongly recommended that you do not modify this file unless it is absolutely necessary. Please note that invalid data entries present in this file can lead to the malfunction of the Agent, the malfunction of the deployed applications, and could render the entire system unusable.

After you make the changes to `AMAgent.properties`, you must perform the following steps:

1. Restart WebLogic server.
2. Reboot PeopleSoft Application Server domain.

If WebLogic server and PeopleSoft Application Server are installed on different systems, make sure that the Agent Core Configuration properties in the `AMAgent.properties` file on both the systems are the same.

The settings provided in this file can be classified as follows:

Agent Core Configuration: These are general settings that impact the behavior of the Agent at runtime.

Audit Configuration: These settings are used to configure the Audit Engine used by the Agent.

Debug Engine Configuration: These settings help configure the Debug Engine to generate diagnostic information.

Agent Core Configuration

SSO Token Name

Key: com.sun.am.policy.config.ssoTokenName

Description: This property specifies the name of the Cookie that represents SSO Token.

Valid Values: A String that represents the name of the SSO Token Cookie issued by Sun ONE Identity Server authentication service.

NOTE This property is set during Agent installation and need not be changed unless absolutely necessary.

Example: com.sun.am.policy.config.ssoTokenName=iPlanetDirectoryPro

Get LDAP Attribute Flag

Key: com.sun.am.policy.config.getLDAPAttribute

Description: This property specifies whether the LDAP attribute representing a PeopleSoft user should be got from Identity Server.

Valid Values: This property can have one of these two values:

true, indicates that PeopleCode will retrieve the attribute.

false, indicates that it will rely on the HTTP header for the PeopleSoft user value. This value is used only if you have installed the agent as a proxy solution.

NOTE Default value of this property is "true" and should be changed as necessary. It is mandatory to set the value of this property to true if you have deployed the Agent on PeopleSoft-provided WebLogic Server. If you are deploying the Agent as a Proxy Solution, setting the value to true will provide extra security.

Example: com.sun.am.policy.config.getLDAPAttribute=true

LDAP Attribute Name

Key: com.sun.am.policy.config.LDAPAttribute

Description: This property specifies the LDAP Attribute, in Identity Server, to be mapped to a PeopleSoft user. This is used only if the property com.sun.am.policy.config.getLDAPAttribute is set to true.

Valid Values: String representing the valid LDAP Attribute in Identity Server.

NOTE This property is set during Agent installation and need not be changed unless absolutely necessary.

Example: com.sun.am.policy.config.LDAPAttribute=employeenumber

HTTP Header Attribute Name

Key: com.sun.am.policy.config.headerAttribute

Description: This property specifies the HTTP Header Attribute in the http request representing the PeopleSoft user. This is used if the property com.sun.am.policy.config.getLDAPAttribute is set to false. In this case, you need to edit iWS Agent's `AMAgent.properties` available in /etc/opt/SUNWam/confIWS4.1/*_pathinstancename* to ensure that com.iplanet.am.policy.agents.forward_ldapattr_in_http_headers.enable=true and that com.iplanet.am.policy.agents.ldapattr contains the value of HTTP Header Attribute Name as one of the value pairs. You can get more information on this at http://docs.sun.com/source/816-5428-10/iws60.html#35377.

Valid Values: String representing the HTTP Header Attribute

NOTE This property is set during Agent installation and need not be changed unless absolutely necessary.

Example: com.sun.am.policy.config.headerAttribute=employee-number

Validate SSO Token Flag

Key: com.sun.am.policy.config.validateSSO

Description: This property specifies if validation of Identity Server SSO token should be done in PeopleSoft Application server for extra security.

Valid Values: This property should have one of these values:

true: indicates that the validation will be done.

false: indicates that no validation will be performed.

NOTE Default value of this property is "true" and should be changed as necessary.

Example: com.sun.am.policy.config.validateSSO=true

Simple Policy Flag

Key: com.sun.am.policy.config.simplePolicy

Description: This property specifies if Simple Policy should be enforced by Agent. This property is applicable only if you deploy Agent on PeopleSoft Provided WebLogic 5.1 Server (Deployment Option 1).

Valid Values: This property should have one of these values:

true: indicates that Simple Policy will be enforced. This ensures that a user can access only the URI starting with /servlets or /peoplesoft8 (PIA site name). If a user tries to access any other URI, it will invalidate the SSOToken and show the Identity Server Login page. This means that the user will not get WebLogic's index page when she tries to access a URI with just http://*domainname*:*port*/.

false: indicates that Simple Policy will not be enforced. For example, the user will be able to access WebLogic's index.html page even with a URI such as http://*domainname*:*port*/.

NOTE	The default value of this property is "true" and is mandatory for security reasons.

Example: com.sun.am.policy.config.simplePolicy=true

Audit Configuration

Language Code

Key: com.sun.am.policy.config.audit.localeLanguageCode

Description: This property specifies the Locale for Audit log messages.

Valid Values: The localeLanguageCode must be a valid ISO Language Code. Default value of this property is 'en'.

NOTE	For more information, refer to ISO 639 specification: http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt

Example: com.sun.am.policy.config.audit.localeLanguageCode=en

Country Code

Key: com.sun.am.policy.config.audit.localeCountryCode

Description: This property specifies the Locale for Audit log messages.

Valid Values: The localeCountryCode must be a valid ISO Country Code. Default value of this property is 'US'.

NOTE For more information, refer to ISO 3166 specification: http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

---

Example: com.sun.am.policy.config.audit.localeCountryCode=US

Audit Log File

Key: com.sun.am.policy.config.audit.logfile.name.

Description: This property specifies the Audit log file to be used for recording Audit messages.

Valid Values: String representing the complete path name of the file to be used by Agent to record Audit messages.

No default value is set for this property. A valid value must be specified before the Application Server is started.

NOTE Be sure the Application Server process has sufficient permissions to write to this file. Invalid value specified for this property may result in the failure of the system to start up correctly.

Example:com.sun.am.policy.config.audit.logfile.name=/var/opt/SUNWam/asAgent/audit/agent.log

Audit Log File Rotation Flag

Key: com.sun.am.policy.config.audit.logfile.rotate

Description: This property specifies if the Audit log file should be rotated by the Agent.

Valid Values: true/false

NOTE Default value of this property is false and should be changed as necessary.

Example: com.sun.am.policy.config.audit.logfile.rotate=false

Audit Log File Rotation Size

Key: com.sun.am.policy.config.audit.logfile.rotate.size

Description: This property specifies the approximate size of the Audit log file in bytes, which should be used to evaluate when the log file needs to be rotated.

Valid Values: Non-zero unsigned integer indicating the size in bytes to be used to evaluate when the log file needs to be rotated.

NOTE Default value of this property is 52428800 bytes (~ 50 MB) and should be changed as necessary. This property is not used if the Audit Log File Rotation Flag is set to false.

Example: com.sun.am.policy.config.audit.logfile.rotate.size=52428800

Debug Engine Configuration

Debug Level

Key: com.sun.am.policy.config.debug.level

Description: This property specifies the amount of debug messages that will be emitted by the Agent's Debug Engine.

Valid Values: Any of 0, 1, 3, 7, 15, and 31. These values indicate the following:

0 : No debugging

1 : Only Error messages

3 : Error and Warning messages

7 : Error, Warning and Brief Informational messages

15: Error, Warning and Verbose Informational messages

31: Error, Warning and Very Verbose Informational messages

NOTE For better performance of the system, this property should be set to a value 0. Values other than that will affect the system performance depending upon the amount of information the Debug Engine has to emit.

Values other than the ones specified in the Valid Values list above will lead to invalid configuration of the Debug Engine, thereby affecting the volume of messages that will be emitted.

Example: com.sun.am.policy.config.debug.level=7

Debug Log File

Key: com.sun.am.policy.config.debug.logfile.name

Description: This property specifies the Debug log file to be used for recording Debug messages.

Valid Values: String representing the complete path name of the file to be used by Agent to record Debug messages.

NOTE Be sure that the Application Server process has sufficient permissions to be able to write to this file.

Invalid or empty value of this property will lead to Debug messages not being logged in the log file.

Example: com.sun.am.policy.config.debug.logfile.name=/debug/agent_debug.log

Debug Log File Rotation Flag

Key: com.sun.am.policy.config.debug.logfile.rotate

Description: This property specifies if the Debug log file should be rotated by the Agent.

Valid Values: true/false

NOTE Default value of this property is false and should be changed as necessary.

Example: com.sun.am.policy.config.debug.logfile.rotate=false

Debug Log File Rotation Size

Key: com.sun.am.policy.config.debug.logfile.rotate.size

Description: This property specifies the approximate size of the Debug log file in bytes, which should be used to evaluate when the log file needs to be rotated.

Valid Values: Non-zero unsigned integer indicating the size in bytes to be used to evaluate when the log file needs to be rotated.

NOTE Default value of this property is 52428800 bytes (~ 50 MB) and should be changed as necessary.

This property is not used if the Debug Log File Rotation Flag is set to false.

Example: com.sun.am.policy.config.debug.logfile.rotate.size=52428800

Debug Time/Date Format String

Key: com.sun.am.policy.config.debug.date.format

Description: This property specifies the format of time stamp that is used to mark the exact time when the Debug message was recorded.

Valid Values: Valid java.text.SimpleDateFormat Time Format Syntax string. For more information, refer to:

http://java.sun.com/j2se/1.4/docs/api/java/text/SimpleDateFormat.html

---

NOTE The default value of this property is set to: [yyyy/MM/dd HH:mm:ss zzz] and should be changed as necessary.

Invalid value of this property will result in loss of time stamp data with debug messages.

Example: com.sun.am.policy.config.debug.date.format=[yyyy/MM/dd HH:mm:ss zzz]

Debug Print STDOUT Flag

Key: com.sun.am.policy.config.debug.print.stdout

Description: This property specifies if the Debug Engine should print the debug messages on Standard Output stream.

Valid Values: This property can have one of the following values:

true, indicates that the debug messages should be printed on STDOUT.

false, indicates that the debug messages should not be printed on STDOUT.

NOTE The default value of this property is true and should be changed as necessary. When set to true, the Debug Engine prints all debug messages on the Standard output stream. This results in the debug messages being displayed on the console window from where the WebLogic Server startup script was executed.

This property has no effect on the ability of Debug Engine to write to debug log files.

Example: com.sun.am.policy.config.debug.print.stdout=true

Troubleshooting Information

- After installing the Agent, if you encounter any problems, the following could be the reasons:
  - Not supplied the fully qualified hostname while accessing PeopleSoft applications.
  - File Permission problems while creating Debug files.
  - The policy was not configured correctly on Sun ONE Identity Server. For information on configuring policy, refer *Sun ONE Identity Server Administration Guide.*
  - Sun ONE Identity Server user profile not updated with PeopleSoft userid.
  - Not used the same password while creating the DEFAULT_USER user that was supplied during setup. See “To Create DEFAULT_USER.”
  - Incorrect Sun ONE Identity Server shared password supplied for `agentadmin`.
  - PeopleSoft application URL and Sun ONE Identity Server server are not in same DNS domain.

To enable this, install CDSSO component on Sun ONE Identity Server. Refer *Sun ONE Identity Server Installation and Configuration Guide*.

- After installing the Agent, if the PeopleSoft Application does not start up generate a debug file to determine the problem.
  - Enable PeopleCode tracing on the Application Server using `psadmin`. The file `LOGS/DEFAULT_USER_192.18.123.173.tracesqlis` located at *PS_HOME*`/appserv/`*domain*`/` is generated with the following trace/error message:

    ```
    "..Java Exception..." messages indicates Sun ONE Identity
    Server jar setup problems.
    ```

  - Check the SSO Token validation invoked by PeopleCode:

    ```
    /agent_install_dir/SUNWam/debug/PSAuthFilter

    /agent_install_dir/SUNWam/debug/amPeopleCode file
    ```

  - Set the Sun ONE Identity Server Debug to identify and fix the Sun ONE Identity Server problems.

Uninstalling the Agent

When you install the Sun ONE Identity Server Policy Agent for PeopleSoft 8.3 software, an uninstallation program is created in the installation directory. Using this uninstallation program, the Agent can be removed completely from your system. While the uninstallation program deletes all the installed files from your system, certain files such as audit log messages are not deleted. You can delete them manually.

The uninstallation program for Sun ONE Identity Server Policy Agent for PeopleSoft 8.3 should be launched according to the steps detailed for Solaris or HP-UX platforms as applicable.

NOTE Make sure that WebLogic server and the PeopleSoft Application server are shut down before you uninstall the agent.

Launching the Uninstallation Program on Solaris 8

The uninstallation program for Solaris platform may be launched by executing the generated uninstall script located in the installation directory. The following steps provide details on how to achieve this:

1. Login as a PeopleSoft user. Example: `psft`
2. Go to the directory where the Agent is installed.
3. Set your `JAVA_HOME` environment variable to JDK version 1.3.1 or higher. If your system does not have the required version of JDK, use the JDK supplied with PeopleSoft. This JDK is located at:

   PS_HOME`/jre`
4. Run the uninstall script. You'll find the script in the directory where you have installed the Agent. At the command line, enter the following command:

   ```
   ./uninstall_psagent
   ```

   The installation program requires that you set up your `JAVA_HOME` variable correctly as pointed out in the Step 3. However, if you have set the `JAVA_HOME` variable incorrectly, the `setup` script will prompt you for the correct `JAVA_HOME` value:

   ```
   Enter JAVA_HOME location (Enter "." to abort):
   ```

   Specify the `JAVA_HOME` value to the installation program by typing the full path to the JDK installation directory. Otherwise, enter a period (`.`) to abort the installation.

   In order that the GUI uninstallation program be displayed on your console, the `DISPLAY` environment variable of your shell must be set correctly. If your `DISPLAY` environment variable is not set at the time of invoking the `setup` script, the installation program will prompt you for the `DISPLAY` environment variable value as follows:

   ```
   Please enter the value of DISPLAY variable  (Enter "." to abort):
   ```

   Provide the `DISPLAY` value to the installation program by typing the exact value at the above prompt. Otherwise, enter a period (`.`) to abort the installation.

Launching the Uninstallation Program on HP-UX 11

The uninstallation program for HP-UX platform may be launched by executing the generated uninstall script located in the installation directory. The following steps provide details on how to achieve this:

1. Login as PeopleSoft user. Example: `psft`
2. Go to the directory where the Agent is installed.
3. Set your `JAVA_HOME` environment variable to JDK version 1.3.1 or higher. If your system does not have the required version of JDK, use the JDK supplied with PeopleSoft. This JDK is located at:

   PS_HOME`/jre`
4. Run the uninstall script. You'll find the script in the directory where you have installed the Agent. At the command line, enter the following command:

   ```
   ./uninstall_psagent
   ```

   The installation program requires that you set up your `JAVA_HOME` variable correctly as pointed out in the Step 3. However, if you have incorrectly set the `JAVA_HOME` variable, the `setup` script will prompt you for the correct `JAVA_HOME` value:

   ```
   Enter JAVA_HOME location (Enter "." to abort):
   ```

   Provide the `JAVA_HOME` value to the installation program by typing the full path to the JDK installation directory. Otherwise, enter a period (`.`) to abort the installation.

   In order that the GUI installation program be displayed on your console, the `DISPLAY` environment variable of your shell must be set correctly. If your `DISPLAY` environment variable is not set at the time of invoking the `setup` script, the installation program will prompt you for the `DISPLAY` environment variable value as follows:

   ```
   Please enter the value of DISPLAY variable  (Enter "." to abort):
   ```

   Provide the `DISPLAY` value to the installation program by typing the exact value at the above prompt. Otherwise, enter a period (`.`) to abort the installation.

Uninstalling the Agent Using GUI

1. When you run the uninstallation script, the uninstallation program starts up. It alerts you that you must have Administrative privileges to run the uninstallation program.
2. Click Next to open the Welcome panel.
3. Click Next to open the Ready to Uninstall panel. This panel displays the product you are going to uninstall.
4. Click Uninstall Now to start uninstalling the Agent. The Agent is uninstalled. In the Uninstallation Summary panel, you can click Details for detailed information about the uninstallation. Click Close to end the program.

Uninstalling from the Command Line

To run the uninstallation program from the Command Line, you must run the setup program in the 'nodisplay' mode. To do this:

1. Change to the directory where you have installed the agent.
2. Run the following command:

   ```
   ./uninstall_psagent -nodisplay
   ```

3. Follow the instructions on the screen to proceed with uninstallation.
4. At the following prompt, press Enter to uninstall the agent.

```
PRODUCT: Sun_TM_ONE Identity Server Policy Agent
LOCATION: /export/home/psft/agent
SIZE: 4.40 MB
-------------------------------------------------
Sun(TM) ONE Identity Server Policy Agent 1.0 for PeopleSoft 8.3
Ready to Uninstall
1. Uninstall Now
2. Start Over
3. Exit
   Choose one option from above [1] {"<" goes back, "!" exits}
```

Appendix A

Configuration Tasks Performed by Installer

The Sun ONE Identity Server Policy Agent Installer performs certain configuration tasks. Depending on your system configuration and various other factors, these tasks can occasionally fail resulting in a unusable installation. Fortunately, such failures in most cases are recoverable by manually performing these tasks.

This appendix explains how to configure your application server to recover from installation failure.

WebLogic 6.1 SP2

The following sections describe the configuration tasks performed by Sun ONE Identity Server Policy Agent Installer.

WebLogic Server Startup Script Modifications

The Installer modifies the WebLogic Server startup script in order to add the newly installed libraries to the `CLASSPATH` as well as to add certain startup properties for the Java Virtual Machine.

Solaris and HP-UX CLASSPATH Modifications

The following new lines are added to the WebLogic Server startup script, above the line where `CLASSPATH` variable is defined:

```
AM_INSTALL_DIR=/opt/SUNWam/wlAgent

AM_SDK_DIR=$AM_INSTALL_DIR/amSDK
AM_SDK_LIB_DIR=$AM_SDK_DIR/lib
```

```
AM_SDK_LOCALE_DIR=$AM_SDK_DIR/locale
AM_SDK_JAR1=$AM_SDK_LIB_DIR/am_sdk.jar
AM_SDK_JAR2=$AM_SDK_LIB_DIR/am_services.jar
AM_SDK_JAR3=$AM_SDK_LIB_DIR/am_sso_provider.jar
AM_SDK_JARS=$AM_SDK_JAR1:$AM_SDK_JAR2:$AM_SDK_JAR3
AM_SDK_PATH1=$AM_SDK_DIR:$AM_SDK_LIB_DIR
AM_SDK_PATH2=$AM_SDK_LOCALE_DIR:$AM_SDK_JARS
AM_SDK_CLASSPATH=$AM_SDK_PATH1:$AM_SDK_PATH2

AM_AGT_DIR=$AM_INSTALL_DIR/amAgent
AM_AGT_CONFIG_DIR=$AM_AGT_DIR/config
AM_AGT_LOCALE_DIR=$AM_AGT_DIR/locale
AM_AGT_LIB_DIR=$AM_AGT_DIR/lib
AM_AGT_JAR1=$AM_AGT_LIB_DIR/amagent_core.jar
AM_AGT_JAR2=$AM_AGT_LIB_DIR/amagent_weblogic.jar
AM_AGT_JAR3=$AM_AGT_LIB_DIR/amagent_filter.jar
AM_AGT_JAR4=$AM_AGT_LIB_DIR/amagent_tools.jar
AM_AGT_JARS12=$AM_AGT_JAR1:$AM_AGT_JAR2
AM_AGT_JARS34=$AM_AGT_JAR3:$AM_AGT_JAR4
AM_AGT_JARS=$AM_AGT_JARS12:$AM_AGT_JARS34
AM_AGT_PATH1=$AM_AGT_DIR:$AM_AGT_CONFIG_DIR
AM_AGT_PATH2=$AM_AGT_LOCALE_DIR:$AM_AGT_LIB_DIR:$AM_AGT_JARS
AM_AGT_CLASSPATH=$AM_AGT_PATH1:$AM_AGT_PATH2

AM_CLASSPATH=$AM_SDK_CLASSPATH:$AM_AGT_CLASSPATH
```

Once these entries have been added, the `AM_CLASSPATH` is appended to the `CLASSPATH` variable.

```
CLASSPATH=$WL_HOME:$WL_HOME/lib/weblogic_sp.jar:$WL_HOME/lib/web
logic.jar:$WL_HOME/samples/eval/cloudscape/lib/cloudscape.jar:./
config/examples/serverclasses:$AM_CLASSPATH
```

Windows CLASSPATH Modifications

The `CLASSPATH` is modified by adding the following lines to the WebLogic Server startup script immediately after the definition of the `CLASSPATH` variable.

```
set AM_INSTALL_DIR=C:\Sun\SUNWam\wlAgent

set AM_SDK_DIR=%AM_INSTALL_DIR%\amSDK
set AM_SDK_LIB_DIR=%AM_SDK_DIR%\lib
set AM_SDK_LOCALE_DIR=%AM_SDK_DIR%\locale
set AM_SDK_JAR1=%AM_SDK_LIB_DIR%\am_sdk.jar
set AM_SDK_JAR2=%AM_SDK_LIB_DIR%\am_services.jar
set AM_SDK_JAR3=%AM_SDK_LIB_DIR%\am_sso_provider.jar
set AM_SDK_JARS=%AM_SDK_JAR1%;%AM_SDK_JAR2%;%AM_SDK_JAR3%
set AM_SDK_PATH1=%AM_SDK_DIR%;%AM_SDK_LIB_DIR%
```

```
set AM_INSTALL_DIR=C:\Sun\SUNWam\wlAgent
set AM_SDK_PATH2=%AM_SDK_LOCALE_DIR%;%AM_SDK_JARS%
set AM_SDK_CLASSPATH=%AM_SDK_PATH1%;%AM_SDK_PATH2%

set AM_AGT_DIR=%AM_INSTALL_DIR%\amAgent
set AM_AGT_CONFIG_DIR=%AM_AGT_DIR%\config
set AM_AGT_LOCALE_DIR=%AM_AGT_DIR%\locale
set AM_AGT_LIB_DIR=%AM_AGT_DIR%\lib
set AM_AGT_JAR1=%AM_AGT_LIB_DIR%\amagent_core.jar
set AM_AGT_JAR2=%AM_AGT_LIB_DIR%\amagent_weblogic.jar
set AM_AGT_JAR3=%AM_AGT_LIB_DIR%\amagent_filter.jar
set AM_AGT_JAR4=%AM_AGT_LIB_DIR%\amagent_tools.jar
set AM_AGT_JARS12=%AM_AGT_JAR1%;%AM_AGT_JAR2%
set AM_AGT_JARS34=%AM_AGT_JAR3%;%AM_AGT_JAR4%
set AM_AGT_JARS=%AM_AGT_JARS12%;%AM_AGT_JARS34%
set AM_AGT_PATH1=%AM_AGT_DIR%;%AM_AGT_CONFIG_DIR%
set
AM_AGT_PATH2=%AM_AGT_LOCALE_DIR%;%AM_AGT_LIB_DIR%;%AM_AGT_JARS%
set AM_AGT_CLASSPATH=%AM_AGT_PATH1%;%AM_AGT_PATH2%

set AM_CLASSPATH=%AM_SDK_CLASSPATH%;%AM_AGT_CLASSPATH%

set CLASSPATH=%CLASSPATH%;%AM_CLASSPATH%
```

The last line in the added text modifies the `CLASSPATH` to include the libraries provided by the Agent.

Adding Parameters to Java Virtual Machine

For the installation platforms Solaris 8, Windows 2000, and HP-UX 11 the following parameters are added to the Java Virtual Machine (JVM) invocation command that loads the WebLogic Server:

```
-D"com.iplanet.coreservices.configpath=/opt/SUNWam/wlAgent/amSDK
/config/ums"
-D"max_conn_pool=10"
-D"min_conn_pool=1"
```

The resulting command on Solaris and HP-UX:

```
java $JAVA_OPTIONS -classpath $CLASSPATH
-Dweblogic.Domain=examples -Dweblogic.Name=examplesServer
-Dweblogic.management.password=$WLS_PW -Dbea.home=/bea
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy==$WL_HOME/lib/weblogic.policy
-D"com.iplanet.coreservices.configpath=/opt/SUNWam/wlAgent/amSDK
/config/ums" -D"max_conn_pool=10" -D"min_conn_pool=1"
weblogic.Server
```

On Windows:

```
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -classpath
"%CLASSPATH%" -Dweblogic.Domain=examples
-Dweblogic.Name=examplesServer
-Dweblogic.management.password=%WLS_PW% -Dbea.home="C:\bea"
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="C:\bea\wlserver6.1/lib/weblogic.policy"
-D"com.iplanet.coreservices.configpath=C:/Sun/SUNWam/wlAgent/amS
DK/config/ums" -D"max_conn_pool=10" -D"min_conn_pool=1"
weblogic.Server
```

Installation of JCE 1.2.1 and JSSE 1.0.2 Extensions

The Installer also performs the installation of JCE 1.2.1 and JSSE 1.0.2 extensions which result in the modification of the file `JAVA_HOME/jre/lib/secrurity/java.security` and the copying of various jar files in the `JAVA_HOME/jre/lib/ext` directory. If the installation of any of these extensions fails, you can manually install them. For obtaining these extensions and documentation on how to install them, refer to the product website:

`http://java.sun.com/products/jce` and

`http://java.sun.com/products/jsse`

WebSphere 4.0.3 AE

The following sections describe the configuration tasks performed by Sun ONE Identity Server Policy Agent Installer.

Modifications to WebSphere Command-Line Setup Script

- Append the following lines to WebSphere command-line setup script *WAS_root_dir*/bin/setupCmdLine.sh.

```
BASEDIR=Agent_Install_Dir
PRODUCT_DIR=SUNWam
AM_INSTALL_DIR=$BASEDIR/$PRODUCT_DIR/wasAgent
AM_SDK_DIR=$AM_INSTALL_DIR/amSDK
AM_AGT_DIR=$AM_INSTALL_DIR/amAgent
SDK_CLASSPATH=$AM_SDK_DIR:$AM_SDK_DIR/locale:$AM_SDK_DIR/lib:$AM
_SDK_DIR/lib/am_sdk.jar:$AM_SDK_DIR/lib/am_services.jar:  \
$AM_SDK_DIR/lib/am_sso_provider.jar:$AM_SDK_DIR/lib/jaas.jar
AGENT_CLASSPATH=$AM_AGT_DIR:$AM_AGT_DIR/config:$AM_AGT_DIR/local
e:$AM_AGT_DIR/lib:$AM_AGT_DIR/lib/amagent_core.jar:  \
$AM_AGT_DIR/lib/amagent_tools.jar:$AM_AGT_DIR/lib/amagent_was.ja
r: $AM_AGT_DIR/lib/amagent_interceptor.jar
```

- Update the WAS_EXT_DIRS and WAS _CLASSPATH by adding SDK_CLASSPATH and AGENT_CLASSPATH

```
WAS_EXT_DIRS=$JAVA_HOME/lib:$WAS_HOME/classes:$WAS_HOME/lib:$WAS
_HOME/lib/ext:$WAS_HOME/web/help:  \

$DBDRIVER_JARS:$SDK_CLASSPATH:$AGENT_CLASSPATH
WAS_CLASSPATH=$WAS_HOME/properties:$WAS_HOME/lib/bootstrap.jar:$
SDK_CLASSPATH:$AGENT_CLASSPATH
```

Modifications to WebSphere Server Startup Script

- Add the System property for the Identity Server SDK `config` directory to WebSphere Server startup script *WAS_root_dir*`/bin/startupServers.sh`

```
BASEDIR=Agent_Install_Dir
PRODUCT_DIR=SUNWam
AM_INSTALL_DIR=$BASEDIR/$PRODUCT_DIR/wasAgent
AM_SDK_DIR=$AM_INSTALL_DIR/amSDK
```

- Add the property `-Dcom.iplanet.coreservices.configpath=$AM_SDK_DIR/config/ums` to the JAVA command line that invokes `com.ibm.ws.bootstrap.WSLauncher`, to the appropriate section depending on the configuration database used by the WebSphere.

Modifications to Admin Server Configuration File

- Update the property `com.ibm.ejs.sm.adminserver.classpath` and add all the SDK and AGENT related directories that contain property files.
- Update the `-Dws.ext.dirs` value for `com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs`, to include SDK and agent related directories and jars.
- Add the following arguments to `com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs` property:

  `-Djava.protocol.handler.pkgs=com.ibm.net.ssl.internal.www.protocol`

  `-Dcom.iplanet.coreservices.configpath=`*Agent_Install_Dir*`/SUNWam/wasAgent/amSDK/config/ums`

Modifications to trustedserver.properties

- Comment out all the lines that are not commented, and add the following lines to *WAS_root_dir*`/properties/trustedservers.properties` file

  `com.ibm.websphere.security.trustassociation.enabled=true`

```
com.ibm.websphere.security.trustassociation.types=amagent

com.ibm.websphere.security.trustassociation.amagent.interceptor=
com.sun.amagent.websphere.interceptor.AgentInterceptor

com.ibm.websphere.security.trustassociation.amagent.config=AMAge
nt
```

- Restart the Administration Server and the Application Server instance.

NOTE Before performing Agent Realm configuration, the Administration Server and Application server instance must be restarted, so that the above modifications take effect. During the Agent Realm installation, the authenticate( ) method of Realm is invoked with the given Realm Administrator credentials. The SDK and the Agent related classes must be in the classpath. The realm is installed only when the authentication is successful.

Configurations Through Administrative Console

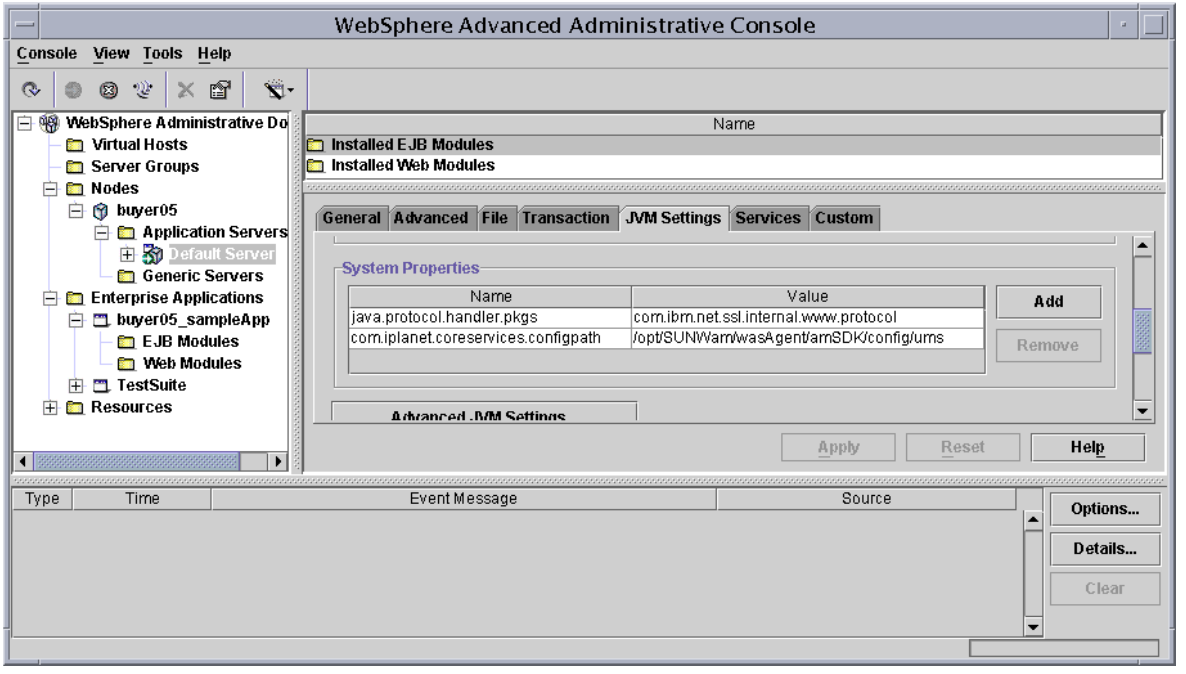
Setting System Properties for the Application Server Instance

The following property must be added to each Application server instance, through the Administrative console as shown in the Figure A-1.

```
-Djava.protocol.handler.pkgs=com.ibm.net.ssl.internal.www.protocol

-Dcom.iplanet.coreservices.configpath=
Agent_Install_Dir/SUNWam/wasAgent/amSDK/config/ums
```

Figure A-1 Setting System Properties



Agent Realm Configuration

The Agent Realm can be configured from the Security Center of Administrative Console. Perform the following steps to configure the Agent Realm:

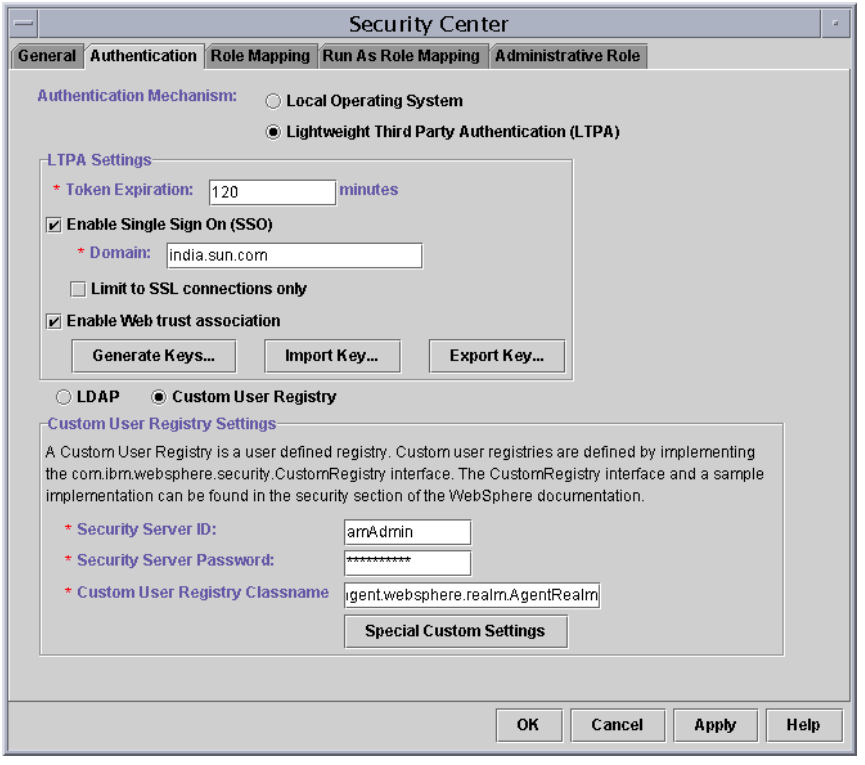
1. Start the WebSphere Administrative console using the following command:

   # *WAS_roor_dir*/bin/adminclient.sh

2. In the Administrative Console window, choose Console > Security Center.
3. In the Security Center window, click on Authentication tab. See Figure A-2.
4. Choose “Lightweight Third Party Authentication (LTPA)” for authentication mechanism.
5. Choose “Custom Registry Option.”
6. Enter the Administrator's user ID for “Security Server Id.” For example, amAdmin for o=sun.com.
7. Enter the password for the Administrative User for “Security Server Password.”

8. Enter the value `com.sun.amagent.websphere.realm.AgentRealm` for Custom Registry Class.
9. Click on check box “Enable Web trust association.”
10. Click Apply. A message “Changes will take effect only after Administration Server is restarted” will be displayed. Click OK
11. Click on General tab and enable the check box for “Enable Security.”
12. Stop all Application Server instances.
13. Stop the Administration Server.
14. Restart Administration Server and Application Server instances for changes to take effect.

Figure A-2 Agent Realm Configuration



Sun ONE Application Server 7.0

The following sections describe the configuration tasks performed by Sun ONE Identity Server Policy Agent Installer.

Application Server Config Files

The following configuration files are modified by the Installer:

S1AS_Install_Dir/appserv/domains/domain1/*server-instance*/config/server.xml

S1AS_Install_Dir/appserv/domains/domain1/*server-instance*/config/login.conf

S1AS_Install_Dir/appserv/domains/domain1/*server-instance*/config/server.policy

Modifications in server.xml

Installer modifies `server.xml` in order to add all the newly installed libraries to the `classpath` as well as to add certain properties for the Java Virtual Machine. Also, it adds Agent Realm and makes it default to be used.

classpath Modifications

The `classpath` is modified by adding the following lines to the Application Server `startup` script immediately after the definition of the `classpath` variable.

```
Agent_Install_Dir/SUNWam/asAgent/amSDK;
Agent_Install_Dir/SUNWam/asAgent/amSDK/lib;
Agent_Install_Dir/SUNWam/asAgent/amSDK/locale;
Agent_Install_Dir/SUNWam/asAgent/amSDK/lib/am_sdk.jar;
Agent_Install_Dir/SUNWam/asAgent/amSDK/lib/am_services.jar;
Agent_Install_Dir/SUNWam/asAgent/amSDK/lib/am_sso_provider.jar;
Agent_Install_Dir/SUNWam/asAgent/amAgent;
Agent_Install_Dir/SUNWam/asAgent/amAgent/config;
Agent_Install_Dir/SUNWam/asAgent/amAgent/locale;
Agent_Install_Dir/SUNWam/asAgent/amAgent/lib;
Agent_Install_Dir/SUNWam/asAgent/amAgent/lib/amagent_core.jar;
Agent_Install_Dir/SUNWam/asAgent/amAgent/lib/amagent_as.jar;
Agent_Install_Dir/SUNWam/asAgent/amAgent/lib/amagent_filter.jar;
Agent_Install_Dir/SUNWam/asAgent/amAgent/lib/amagent_tools.jar
```

You can verify the modifications by invoking the Admin Console:

1. Click the *server-instance* under Application Server Instances.
2. On the right pane, click JVM Settings > Path Settings.
3. Check the values of Classpath Suffix, which includes the above values.

Adding Parameters to Java Virtual Machine

The following parameters are added to the Java Virtual Machine invocation command that loads the Application Server:

```
-Dcom.iplanet.coreservices.configpath=Agent_Install_Dir/SUNWam/asAgen
t/amSDK/config/ums
-Dmax_conn_pool=10
-Dmin_conn_pool=1
```

Adding Agent Realm

The following lines are added under `<security-service>` element:

```
<auth-realm name="agentRealm"
classname="com.sun.amagent.as.realm.AgentRealm">
<property name="jaas-context" value="agentRealm"/>
</auth-realm>
```

Here `name=agnetRealm` is value provided at installation time.

You can verify this using the Admin Console.

- Choose Security > Realms and verify the Agent Realm.

Making Agent Realm as Default

`default-realm` attribute of `<security-service>` element is modified as follows:

```
<security-service default-realm="agentRealm" anonymous-role="ANYONE"
audit-enabled="false">
```

You can verify this using the Admin Console.

- Click Security and verify the value of Default Realm.

Modifications in login.conf

Installer modifies `login.conf` to define `LoginModule` for the jaas-context as follows:

```
agentRealm {
    com.sun.amagent.as.realm.AgentLoginModule required;
};
```

Modifications in server.policy

Installer modifies `server.policy` to give Programmatic Login permission to `agent-filter.jar` as follows:

```
// ProgrammaticLoginPermission to Sun(TM) ONE Identity Server
Policy Agent Filter
grant codeBase
"file:Agent_Install_Dir/SUNWam/asAgent/amAgent/lib/amagent_filter.jar
" {
    permission
com.sun.appserv.security.ProgrammaticLoginPermission "login";
};
```

PeopleSoft 8.3

The following sections describe the configuration tasks performed based on the selections you made during the installation of Sun ONE Identity Server Policy Agent for PeopleSoft 8.3.

When PeopleSoft Application Server is installed locally

These tasks are performed for both the deployment scenarios: Agent deployed on PeopleSoft-provided WebLogic Server or Agent deployed on a separate proxying web server.

On successful installation, a file `psdsameenv.sh` containing the environment is created under $PS_HOME/appserv/. The PS_CLASSPATH environment variable is required by PeopleCode to set up the appropriate CLASSPATH when authentication-related PeopleCode is invoked.

NOTE It is recommended that `$PS_HOME/appserv/psdsameenv.sh` be added to the `$HOME/.profile` of the userid that executes PeopleSoft startup and shutdown.

In the file `$PS_HOME/appserv/`*`DOMAIN`*`/psappsrv.cfg`, the following JVM option is added:

```
JavaVM
Options=-Dcom.iplanet.coreservices.configpath=Agent_Install_Dir/
SUNWam/amSDK/config/ums
```

When WebLogic 5.1 Server installed locally

The following tasks are common to both deployment options: Agent deployed on PeopleSoft-provided WebLogic Server or Agent deployed on a separate proxying web server.

The files `dsamesignin.html`, `dsamesignin1.html`, and `dsamesimplesignin.html` are installed under:

`WEBLOGIC-HOME/myserver/myserver/public_html/peoplesoft8`

The file `WEBLOGIC-HOME/myserver/psftdocs/peoplesoft8/configuration.properties` is updated with the following changes.

- PeopleSoft configured for SSO:

  ```
  byPassSignOn=true
  defaultUSERID=DEFAULT_USER
  defaultPWD=some-long-password
  ```

- Single Signout:

  ```
  signon_page=dsamesignin.html
  signonError_page=dsamesignin.html
  logout_page=dsamesignin.html
  expire_page=dsamesignin.html
  ```

The following symbolic links are created:

```
$ ln -s WEBLOGIC-HOME/myserver/psftdocsWEBLOGIC-HOME
$ ln -s WEBLOGIC-HOME/myserver/public_html/peoplesoft8WEBLOGIC
HOME/myserver
```

```
$ ln -s WEBLOGIC-HOME/myserver/public_html/index.htmlWEBLOGIC
HOME/myserver
```

Configuration Tasks when Deploying Agent on PeopleSoft Provided WebLogic 5.1 Server

- In the file `WEBLOGIC-HOME/setEnv.sh`, the CLASSPATH is appended at the end of the file.

```
INSTALL_DIR=Agent_Install_Dir
WL_DIR= WEBLOGIC-HOME
SDK_DIR=$INSTALL_DIR/amSDK
SDK_LIB_DIR=$SDK_DIR/lib
SDK_LOCALE_DIR=$SDK_DIR/locale
SDK_JAR1=$SDK_LIB_DIR/am_sdk.jar
SDK_JAR2=$SDK_LIB_DIR/am_services.jar
SDK_JAR3=$SDK_LIB_DIR/am_sso_provider.jar
SDK_JARS=$SDK_JAR1:$SDK_JAR2:$SDK_JAR3
SDK_PATH1=$SDK_DIR:$SDK_LIB_DIR
SDK_PATH2=$SDK_LOCALE_DIR:$SDK_JARS
SDK_CLASSPATH=$SDK_PATH1:$SDK_PATH2
AGT_DIR=$INSTALL_DIR/agent
AGT_CONFIG_DIR=$AGT_DIR/config
AGT_LOCALE_DIR=$AGT_DIR/locale
AGT_LIB_DIR=$AGT_DIR/lib
AGT_JAR=$AGT_LIB_DIR/am_ext.jar
AGT_PATH1=$AGT_DIR:$AGT_CONFIG_DIR
AGT_PATH2=$AGT_LOCALE_DIR:$AGT_LIB_DIR:$AGT_JAR
AGT_CLASSPATH=$AGT_PATH1:$AGT_PATH2
EXT_LIB_DIR=$INSTALL_DIR/lib
EXT_JARS=$EXT_LIB_DIR/servlet.jar:$EXT_LIB_DIR/crimson.jar:$EXT_
LIB_DIR/jaxp.jar:$EXT_LIB_DIR/jaas.jar:$EXT_LIB_DIR/xercesImpl.j
ar:$EXT_LIB_DIR/xmlParserAPIs.jar
EXT_CLASSPATH=$EXT_LIB_DIR:$EXT_JARS
PS_WEB_DIR=$INSTALL_DIR/PS
PS_WEB_CLASSPATH=$PS_WEB_DIR:$PS_WEB_DIR/classes:$WL_DIR/myserve
r/WEB-INF
AGENT_CLASSPATH=$SDK_CLASSPATH:$AGT_CLASSPATH:$EXT_CLASSPATH:$PS
_WEB_CLASSPATH
CLASSPATH=$CLASSPATH:.:$AGENT_CLASSPATH
export CLASSPATH
```

- In the file WEBLOGIC-HOME/startWebLogic.sh, the above $CLASSPATH gets appended in the PRE-CLASSPATH variable after weblogic510sp.jar and the following JVM option in the Java invocation command:

  -Dcom.iplanet.coreservices.configpath=Agent_Install_Dir/SUNWam/amSDK/config/ums

  -Dcom.sun.am.policy.config.PIASiteName=*PIA_SITE_NAME*

Also, the following is added to the variable JAVACLASSPATH:

Agent_Install_Dir/SUNWam/agent/config

- In the file WEBLOGIC-HOME/weblogic.policy,the following are added to grant property read / write permission for the agent.

  ```
  // Grant property read/write permission for the agent

  grant { permission java.util.PropertyPermission
  "com.sun.am.policy.config.*", "read,write";};
  ```

- The following files are installed. These files represent redeployment of PeopleSoft application as a Web application. This is necessary for agent code executing the WebLogic server to work correctly.

  WEBLOGIC-HOME/myserver/WEB-INF

  WEBLOGIC-HOME/myserver/WEB-INF/web.xml

  WEBLOGIC-HOME/myserver/WEB-INF/weblogic.xml

- The file *WEBLOGIC-HOME/weblogic.properties* is updated to register PeopleSoft application as a Web application. The following line is appended at the end:

  weblogic.httpd.defaultWebApp=WEBLOGIC-HOME/myserver

JCE/JSSE

The Installer also performs the installation of JCE 1.2.1 and JSSE 1.0.2 extensions. While doing this, it modifies the file JAVA_HOME/jre/lib/secrurity/java.security and the copies various jar files to the JAVA_HOME / jre / lib / ext directory. If the installation of any of these extensions fails, you can manually install them. For obtaining these extensions and the documentation on how to install them, refer to the product website at:

http://java.sun.com/products/jce and

http://java.sun.com/products/jsse

Appendix B

Using the Agent Debug Engine

Sun ONE Identity Server Policy Agent for Application Server is equipped with a sophisticated Debug Engine that can be used to gather statistics about your deployment, oversee the execution of the Agent as it protects various applications, and troubleshoot hard to locate problems with your installation.

To use this engine, set the properties correctly for the Debug Engine configuration.

A few things worth noting are:

- When the Debug Engine is operational, that is the debug level is set to a value other than 0, the Agent reports varying set of information to the Debug Engine. The Debug Engine has to process this information in order to store or display it appropriately. This process results in a overhead that should be avoided if the performance of the system is of utmost significance. Therefore, the Debug Engine should not be used when the system is in production. It should be used during testing of the deployed application under controlled environment.
- If the Debug Engine is enabled, that is the debug level is set to a value other than 0, it is possible that `Print STDOUT` flag is disabled as well as the debug file is not specified correctly. In such a situation, all the debug messages are lost, yet the performance of the system suffers. Therefore, to ensure that the debug engine is disabled, set the debug level to 0.
- The Debug Engine's date and time format is configurable and can be used to generate specific strings which may help distinguish between Agent Messages from other messages being generated on the Application Server console.

Appendix C

Sample Scenarios for Role-to-Principal Mapping

This appendix provides sample scenarios on creating role-to-principal mappings. Topics include:

- WebLogic 6.1 SP2 Application Server
- WebSphere 4.0.3 AE Application Server
- Sun ONE Application Server 7.0

WebLogic 6.1 SP2 Application Server

Declarative Security

Consider an Enterprise JavaBeans component that must have protected access for one of its methods. This can be enforced by adding a security-role and method-permission element in the assembly-descriptor element of the `ejb-jar.xml` deployment descriptor.

```
<?xml version="1.0"?>

<!DOCTYPE ejb-jar PUBLIC
'-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN'
'http://java.sun.com/dtd/ejb-jar_2_0.dtd'>
<ejb-jar>
   ...
   <assembly-descriptor>
      <security-role>
         <role-name>FOO</role-name>
      </security-role>
```

```
      <method-permission>
         <role-name>FOO</role-name>
      <method>
            <ejb-name>WebProxy</ejb-name>
            <method-intf>Remote</method-intf>
            <method-name>getWebPage</method-name>
            <method-params>
               <method-param>java.lang.String</method-param>
               <method-param>java.lang.String</method-param>
            </method-params>
       </method>
       </method-permission>
   </assembly-descriptor>
</ejb-jar>
```

This security role `FOO` can be mapped to a real principal using the `weblogic-ejb-jar.xml` deployment descriptor.

```
<?xml version="1.0"?>
<!DOCTYPE weblogic-ejb-jar PUBLIC '-//BEA Systems, Inc.//DTD
WebLogic 6.0.0 EJB//EN'
    'http://www.bea
.com/servers/wls6000/dtd/weblogic-ejb-jar.dtd'>
<weblogic-ejb-jar>
   <weblogic-enterprise-bean>
      <ejb-name>WebProxy</ejb-name>
      <jndi-name>ejb.WebProxy</jndi-name>
   </weblogic-enterprise-bean>
   <security-role-assignment>
    <role-name>FOO</role-name>
    <principal-name>amAdmin</principal-name>
   </security-role-assignment>
</weblogic-ejb-jar>
```

Programmatic Security

Consider a sample application in which one of the Servlets uses the programmatic security API such as `HttpServletRequest.isUserInRole(String)`. Assuming that the role name used within the Servlet code is `SAMPLE-ROLE`, its deployment descriptor.

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN" "http://java.sun.c
om/dtd/web-app_2_3.dtd">
<web-app>
  <display-name>Sample Security Aware application</display-name>
```

```
    <filter>
      <filter-name>Agent</filter-name>
      <display-name>Agent</display-name>
      <description>Identity Server Policy Agent for WebLogic
Server 6.1</description>
      <filter-class>
         com.iplanet.amagent.weblogic.filter.AgentFilter
      </filter-class>
    </filter>
    <filter-mapping>
      <filter-name>Agent</filter-name>
      <url-pattern>/*</url-pattern>
    </filter-mapping>
  <servlet>
    <servlet-name>SampleServlet</servlet-name>
    <servlet-class>
       com.iplanet.sample.SampleServlet
    </servlet-class>
    <security-role-ref>
       <description>
          This role is a sample test role for the
          security aware servlet
       </description>
       <role-name>SAMPLE-ROLE</role-name>
       <role-link>SAMPLE-ROLE-LINK</role-link>
    </security-role-ref>
  </servlet>
  <servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>/Test</url-pattern>
  </servlet-mapping>
  <security-role>
    <description>Some description</description>
    <role-name>SAMPLE-ROLE-LINK</role-name>
  </security-role>
</web-app>
```

The `SAMPLE-ROLE-LINK` is mapped to an actual principal using the `weblogic.xml` deployment descriptor.

```
<!DOCTYPE weblogic-web-app PUBLIC "-//BEA Systems, Inc.//DTD Web
Application 6.1//EN" "http://www.be
a.com/servers/wls610/dtd/weblogic-web-jar.dtd">
<weblogic-web-app>
   <security-role-assignment>
     <role-name>SAMPLE-ROLE-LINK</role-name>
     <principal-name>Employee</principal-name>
   </security-role-assignment>
</weblogic-web-app>
```

WebSphere 4.0.3 AE Application Server

Web Authorization

In `web.xml`, there is an element `security-constraint` which is used to define constraints placed on various parts of this web application.

```
?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
  <web-app id="WebApp_1">
   <display-name>Test Application</display-name>
   <description>A Web Application for Testing Various Test Cases
for WebSphere Agent</description>
   ..............
   ...........
      <servlet id="Servlet_10">
         <servlet-name>ProtectedServlet</servlet-name>
         <display-name>ProtectedServlet</display-name>
         <description>A Protected servlet that accesses an
unprotected EJB</description>
          <servlet-class>com.agent.servlet.ProtectedServlet</ser
vlet-class>
      </servlet>

      <servlet-mapping id="ServletMapping_10">
         <servlet-name>ProtectedServlet</servlet-name>
         <url-pattern>/ProtectedServlet</url-pattern>
      </servlet-mapping>

      <security-constraint id="SecurityConstraint_2">
         <web-resource-collection id="WebResourceCollection_2">
            <web-resource-name>Protected</web-resource-name>
           <url-pattern>/ProtectedServlet</url-pattern>
           <http-method>GET</http-method>
           <http-method>POST</http-method>
         </web-resource-collection>
         <auth-constraint id="AuthConstraint_2">
           <description>Protected-tc4:+:</description>
           <role-name>Protected-tc3</role-name>
         </auth-constraint>
         <user-data-constraint id="UserDataConstraint_2">
           <transport-guarantee>NONE</transport-guarantee>
         </user-data-constraint>
      </security-constraint>

      <security-role id="SecurityRole_5">
         <role-name>Protected-tc3</role-name>
      </security-role>
       ................
```

```
        ........
  </web-app>
```

In the `ibm-application-bnd.xmi` file, define role mappings for the role `Protected-tc3` as:

```
<applicationbnd:ApplicationBinding xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:applicationbnd="applicationbnd.xmi"
   xmlns:application="application.xmi" xmlns:common="common.xmi"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmi:id="Application_ID_Bnd">

   <appName xsi:nil="true"/>
   <application href="META-INF/application.xml#Application_ID"/>
   <authorizationTable xmi:id="AuthorizationTable_1">
    ............
    .........
    <authorizations xmi:id="RoleAssignment_4">
        <role href="META-INF/application.xml#SecurityRole_5"/>
        <users xmi:id="User_1" name="amAdmin"/>
        <groups xmi:id="Group_4" name="manager"/>
    </authorizations>
    ..........
    .........
   </authorizationTable>
   <runAsMap xmi:id="RunAsMap_1"/>
</applicationbnd:ApplicationBinding>
```

EJB Authorization

Consider an Enterprise JavaBeans component that must have protected access for one of its methods. This can be enforced by adding a security-role and method-permission element in the assembly-descriptor element of the `ejb-jar.xml` deployment descriptor.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD
Enterprise JavaBeans 1.1//EN"
  "http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd">
  <ejb-jar id="ejb-jar_ID">
    <enterprise-beans>
        ............
        .........
        <session id="Session_3">
           <ejb-name>ProtectedEJB</ejb-name>
```

```
            <home>com.agent.ejb.ProtectedEJBHome</home>
            <remote>com.agent.ejb.ProtectedEJB</remote>
             <ejb-class>com.agent.ejb.ProtectedEJBImpl</ejb-class
>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
         </session>
         ............
         ........
    </enterprise-beans>

    <assembly-descriptor id="AssemblyDescriptor_ID">
      ...........
      ........
      <security-role id="SecurityRole_4">
         <role-name>Protected-tc2</role-name>
      </security-role>
      ..........
      .......
      <method-permission id="MethodPermission_2">
         <description>tc2:+:</description>
         <role-name>Protected-tc2</role-name>
         <method id="MethodElement_2">
            <ejb-name>ProtectedEJB</ejb-name>
            <method-intf>Remote</method-intf>
            <method-name>protectedMethod</method-name>
            <method-params></method-params>
         </method>
      </method-permission>

      .........
      ....
      </assembly-descriptor>
  </ejb-jar>
```

In the `ibm-application-bnd.xmi` file, define role mappings for the role `Protected-tc2`:

```
<applicationbnd:ApplicationBinding xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:applicationbnd="applicationbnd.xmi"
  xmlns:application="application.xmi" xmlns:common="common.xmi"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmi:id="Application_ID_Bnd">

  <appName xsi:nil="true"/>
  <application href="META-INF/application.xml#Application_ID"/>
  <authorizationTable xmi:id="AuthorizationTable_1">
    ............
    .........
    <authorizations xmi:id="RoleAssignment_3">
       <role href="META-INF/application.xml#SecurityRole_4"/>
```

```
        <groups xmi:id="Group_2" name="manager"/>
        <groups xmi:id="Group_3" name="employee"/>
    </authorizations>
    ..........
    ..........
  </authorizationTable>
  <runAsMap xmi:id="RunAsMap_1"/>
</applicationbnd:ApplicationBinding>
```

Sun ONE Application Server 7.0

Declarative Security

Consider an Enterprise JavaBeans component that must have protected access for one of its methods. This can be enforced by adding a security-role and method-permission element in the assembly-descriptor element of the `ejb-jar.xml` deployment descriptor.

```
<?xml version="1.0"?>

<!DOCTYPE ejb-jar PUBLIC-//Sun Microsystems, Inc.//DTD Enterprise
JavaBeans 2.0//EN'
'http://java.sun.com/dtd/ejb-jar_2_0.dtd'>
<ejb-jar>
   ...
   <assembly-descriptor>
    <security-role>
      <role-name>staffRole</role-name>
    </security-role>
    <method-permission>
        <role-name>staffRole</role-name>
        <method>
            <ejb-name>ConverterEJB</ejb-name>
            <method-name>dollarToYen</method-name>
        </method>
    </method-permission>
</assembly-descriptor>
    ...
</ejb-jar>
```

This security role staffRole can then be mapped to a real principal in sun-application.xml.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sun-application PUBLIC '-//Sun Microsystems, Inc.//DTD
Sun ONE Application Server 7.0 J2EE Application 1.3//EN'
'http://www.sun.com/software/sunone/appserver/dtds/sun-applicati
on_1_3-0.dtd'>

<sun-application>

  <security-role-mapping>
    <role-name>staffRole</role-name>
    <group-name>cn=staff,o=sun.com,o=isp</group-name>
    <principal-name>amAdmin</principal-name>
  </security-role-mapping>

</sun-application>
```

NOTE For stand-alone ejb modules there is no sun-application.xml, in that case this data goes into sun-ejb-jar.xml.

Programmatic Security

Consider a sample application in which one of the Servlets uses the programmatic security API such as HttpServletRequest.isUserInRole(String). Assuming that the role name used within the Servlet code is SAMPLE-ROLE, its deployment descriptor.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE web-app PUBLIC '-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN' 'http://java.sun.com/dtd/web-app_2_3.dtd'>
<web-app>
  ...
  <servlet>
    <servlet-name>ProtectedServlet</servlet-name>
    <servlet-class>
       com.sun.sample.ProtectedServlet
    </servlet-class>
    <security-role-ref>
       <description>
          This role is a sample test role for the
          security aware servlet
```

```
        </description>
        <role-name>SAMPLE-ROLE</role-name>
        <role-link>SAMPLE-ROLE-LINK</role-link>
    </security-role-ref>
  </servlet>
  <servlet-mapping>
    <servlet-name>ProtectedServlet</servlet-name>
    <url-pattern>/ProtectedServlet</url-pattern>
  </servlet-mapping>
  ...
<security-constraint>
    <web-resource-collection>
      <web-resource-name>basic secuity test</web-resource-name>
      <url-pattern>/ProtectedServlet</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>SAMPLE-ROLE-LINK</role-name>
    </auth-constraint>
</security-constraint>
   ...
  <security-role>
    <description>Some description</description>
    <role-name>SAMPLE-ROLE-LINK</role-name>
  </security-role>
   ...
</web-app>
```

In the `sun-application.xml` file, define role mappings as:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sun-application PUBLIC '-//Sun Microsystems, Inc.//DTD
Sun ONE Application Server 7.0 J2EE Application 1.3//EN'
'http://www.sun.com/software/sunone/appserver/dtds/sun-applicati
on_1_3-0.dtd'>

<sun-application>

  <security-role-mapping>
    <role-name>SAMPLE-ROLE-LINK</role-name>
    <group-name>cn=staff,o=sun.com,o=isp</group-name>
    <principal-name>amAdmin</principal-name>
  </security-role-mapping>

</sun-application>
```