Sun Java System Access Manager Policy Agent 2.2 Guide for Apache HTTP Server 2.2

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–3288–11
September 14, 2008

080913@20795

Contents

Preface

The *Sun Java System Access Manager Policy Agent 2.2 Guide for Apache HTTP Server 2.2* provides specific information about the Apache HTTP Server 2.2 web agent and general information about web agents in the Policy Agent 2.2 software set.

Who Should Use This Guide

This guide is intended for IT professionals who are responsible for installing and managing Sun Java System Access Manager and policy agents. Administrators should be familiar with the following technologies:

- Directory technologies
- JavaServer Pages™ (JSP) technology
- HyperText Transfer Protocol (HTTP)
- HyperText Markup Language (HTML)
- eXtensible Markup Language (XML)
- Web Services
- Web Technologies

Before You Read This Guide

Sun Java System Policy Agent software works with Sun Java™ System Access Manager. Both products work with Sun Java Enterprise System, a software infrastructure that supports enterprise applications distributed across a network or Internet environment. Furthermore, Sun Java System Directory Server is a necessary component in a new Access Manager deployment since it is used as the data store. To understand how these products interact and to understand this book, you should be familiar with the following documentation:

- Sun Java Enterprise System documentation set, which can be accessed online at `http://docs.sun.com`.

  You can browse the documentation archive or search for a specific book title, part number, or subject.
- Sun Java System Directory Server documentation set.
- Sun Java System Access Manager documentation set, which is explained in more detail subsequently in this chapter.

- Sun Java System Access Manager Policy Agent 2.2 documentation set, which is explained in more detail subsequently in this chapter.

Related Documentation

Sun Microsystems documentation is available at `http://docs.sun.com`, including:

- "Access Manager Documentation Set" on page 8
- "Policy Agent 2.2 Documentation Set" on page 9
- "Sun Java Enterprise System Product Documentation" on page 10

Access Manager Documentation Set

The following table describes the Access Manager 7.1 documentation set, which is available at the following location:

`http://docs.sun.com/app/docs/coll/1292.2`

TABLE P–1 Access Manager 7.1 Documentation Set

Title	Description
Sun Java System Access Manager 7.1 Documentation Center	Contains links to commonly referenced information in the Access Manager documentation collection.
Sun Java System Access Manager 7.1 Release Notes	Describes new features, problems fixed, installation notes, and known issues and limitations. The Release Notes are updated periodically after the initial release to describe any new features or problems.
Sun Java System Access Manager 7.1 Technical Overview	Provides an overview of how Access Manager components work together to consolidate access control functions, and to protect enterprise assets and web-based applications. It also explains basic Access Manager concepts and terminology.
Sun Java System Access Manager 7.1 Deployment Planning Guide	Provides planning and deployment solutions for Access Manager based on the solution life cycle.
Sun Java System Access Manager 7.1 Postinstallation Guide	Provides information about configuring Access Manager after installation. Usually, you perform postinstallation tasks only a few times. For example, you might want to deploy an additional instance of Access Manager or configure Access Manager for session failover.
Sun Java System Access Manager 7.1 Administration Guide	Describes how to use the Access Manager console as well as manage user and service data via the command line interface.

TABLE P–1 Access Manager 7.1 Documentation Set *(Continued)*

Title	Description
Sun Java System Access Manager 7.1 Administration Reference	Provides reference information for the Access Manager command-line interface (CLI), configuration attributes, `AMConfig.properties` attributes, `serverconfig.xml` file attributes, log files, and error codes.
Sun Java System Access Manager 7.1 Federation and SAML Administration Guide	Provides information about the Federation module based on the Liberty Alliance Project specifications. It includes information on the integrated services based on these specifications, instructions for enabling a Liberty-based environment, and summaries of the application programming interface (API) for extending the framework.
Sun Java System Access Manager 7.1 Developer's Guide	Provides information about customizing Access Manager and integrating its functionality into an organization's current technical infrastructure. It also contains details about the programmatic aspects of the product and its API.
Sun Java System Access Manager 7.1 C API Reference	Provides summaries of data types, structures, and functions that make up the public Access Manager C APIs.
Sun Java System Access Manager 7.1 Java API Reference	Provides information about the implementation of Java packages in Access Manager.
Sun Java System Access Manager 7.1 Performance Tuning and Troubleshooting Guide	Provides information about how to tune Access Manager and its related components for optimal performance.
Sun Java System Access Manager Policy Agent 2.2 User's Guide	Provides an overview of Policy Agent software, including the web agents and J2EE agents that are currently available. To view the Access Manager Policy Agent 2.2 documentation collection, see: `http://docs.sun.com/coll/1322.1`

Policy Agent 2.2 Documentation Set

Other Policy Agent guides, besides this guide, are available as described in the following sections:

- "Sun Java System Access Manager Policy Agent 2.2 User's Guide" on page 9
- "Other Individual Agent Guides" on page 10
- "Release Notes" on page 10

Sun Java System Access Manager Policy Agent 2.2 User's Guide

The *Sun Java System Access Manager Policy Agent 2.2 User's Guide* is available in two documentation sets: the Access Manager documentation set as described in Table P–2 and in the Policy Agent 2.2 documentation set as described in this section.

Other Individual Agent Guides

The individual agents in the Policy Agent 2.2 software set, of which this book is an example, are available on a different schedule than Access Manager itself. Therefore, documentation for Access Manager and Policy Agent are available in separate sets, except for the *Sun Java System Access Manager Policy Agent 2.2 User's Guide*, which is available in both documentation sets.

The documentation for the individual agents is divided into two subsets: a web Policy Agent subset and a J2EE Policy Agent subset.

Each web Policy Agent 2.2 guide provides general information about web agents and installation, configuration, and uninstallation information for a specific web agent.

Each J2EE Policy Agent 2.2 guide provides general information about J2EE agents and installation, configuration, and uninstallation information for a specific J2EE agent.

The individual agent guides are listed along with supported server information in the following chapters of the *Sun Java System Access Manager Policy Agent 2.2 User's Guide*:

Web Agents	Chapter 2, "Access Manager Policy Agent 2.2 Web Agents: Compatibility, Supported Servers, and Documentation," in *Sun Java System Access Manager Policy Agent 2.2 User's Guide*
J2EE Agents	Chapter 3, "Access Manager Policy Agent 2.2 J2EE Agents: Compatibility, Supported Servers, and Documentation," in *Sun Java System Access Manager Policy Agent 2.2 User's Guide*

Release Notes

The *Sun Java System Access Manager Policy Agent 2.2 Release Notes* are available online after an agent or set of agents is released. The release notes include a description of what is new in the current release, known problems and limitations, installation notes, and how to report issues with the software or the documentation.

Sun Java Enterprise System Product Documentation

Policy Agent 2.2 was first introduced with Sun Java Enterprise System (Java ES) 2005Q4 but now also supports the following Java ES releases:

Java ES 5: `http://docs.sun.com/coll/1286.2`

Java ES 5 Update 1: `http://docs.sun.com/coll/1286.3`

TABLE P–2 Sun Java Enterprise System 5 Documentation Collections

Title	Location
Sun Java System Directory Server:	http://docs.sun.com/coll/1224.3
Sun Java System Web Server:	http://docs.sun.com/coll/1308.3
Sun Java System Application Server:	http://docs.sun.com/coll/1310.3
Sun Java System Message Queue:	http://docs.sun.com/coll/1307.4
Sun Java System Web Proxy Server:	http://docs.sun.com/coll/1311.5

Accessing Sun Resources Online

For product downloads, professional services, patches and support, and additional developer information, go to the following:

Download Center
http://wwws.sun.com/software/download

Sun Java System Services Suite
http://www.sun.com/service/sunps/sunone/index.html

Sun Enterprise Services, Solaris Patches, and Support
http://sunsolve.sun.com/

Developer Information
http://developers.sun.com/prodtech/index.html

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in the product documentation, go to:

http://www.sun.com/service/contacting

Documentation, Support, and Training

Sun Function	URL	Description
Documentation	http://www.sun.com/documentation/	Download PDF and HTML documents, and order printed documents
Support and Training	http://www.sun.com/training/	Obtain technical support, download patches, and learn about Sun courses

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P–3 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. Perform a *patch analysis*. Do *not* save the file. [Note that some emphasized items appear bold online.]

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–4 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

Related Third-Party Web Site References

Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to `http://docs.sun.com` and click Send Comments. In the online form, provide the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the guide or at the top of the document.

For example, the title of this guide is *Sun Java System Access Manager Policy Agent 2.2 Guide for Apache HTTP Server 2.2*, and the part number is 820-3288.

◆ ◆ ◆ CHAPTER 1

Introduction to Web Agents for Policy Agent 2.2

The Sun Java™ System Access Manager Policy Agent 2.2 software set includes both J2EE agents and web agents. This guide primarily covers web agents, the functionality of which has increased for this release. This chapter provides a brief overview of web agents in the version 2.2 release as well as some concepts you need to understand before proceeding with a web agent deployment. For a general introduction of agents, both J2EE agents and web agents, see *Sun Java System Access Manager Policy Agent 2.2 User's Guide*.

Topics in this chapter include:

Uses of Web Agents

Web agents function with Sun Java System Access Manager to protect content on web servers and web proxy servers from unauthorized intrusions. They control access to services and web resources based on the policies configured by an administrator. Web agents perform these tasks while providing single sign-on (SSO) and cross domain single sign-on (CDSSO) capabilities as well as URL protection.

Web agents are installed on deployment containers for a variety of reasons. Here are three examples:

- A web agent on a human resources server prevents non-human resources personnel from viewing confidential salary information and other sensitive data.
- A web agent on an operations deployment container allows only network administrators to view network status reports or to modify network administration records.

- A web agent on an engineering deployment container allows authorized personnel from many internal segments of a company to publish and share research and development information. At the same time, the web agent restricts external partners from gaining access to the proprietary information.

In each of these situations, a system administrator must set up policies that allow or deny users access to content on a deployment container. For information on setting policies and for assigning roles and policies to users, see the *Sun Java System Access Manager 7.1 Administration Guide*.

How Web Agents Work

When a user points a browser to a particular URL on a protected deployment container, a variety of interactions take place as explained in the following numbered list. See the terminology list immediately following this numbered list for a description of terms.

1. The web agent intercepts the request and checks information in the request against not-enforced lists. If specific criteria are met, the authentication process is by passed and access is granted to the resource.
2. If authentication is required, the web agent validates the existing authentication credentials. If the existing authentication level is insufficient, the appropriate Access Manager Authentication Service will present a login page. The login page prompts the user for credentials such as username and password.
3. The authentication service verifies that the user credentials are valid. For example, the default LDAP authentication service verifies that the username and password are stored in Sun Java System Directory Server. You might use other authentication modules such as RADIUS and Certificate modules. In such cases, credentials are not verified by Directory Server but are verified by the appropriate authentication module.
4. If the user's credentials are properly authenticated, the web agent checks if the users is authorized to access the resource.
5. Based on the aggregate of all policies assigned to the user, the individual is either allowed or denied access to the URL.

Terminology: How Web Agents Work

Authentication Level	The ability to access resources can be divided into levels. Therefore, different resources on a deployment container (such as a web server or proxy server) might require different levels of authentication
Service	Access Manager is made of many components. A service is a certain type of component that performs specific tasks. Some of the Access Manager services available are Authentication Service, Naming Service, Session Service, Logging Service, and Policy Service.

Authentication Module	An authentication interface, also referred to as an authentication module, is used to authenticate a user on Access Manager.
Roles	Roles are a Directory Server entry mechanism. A role's members are LDAP entries that possess the role.
Policy	A policy defines rules that specify access privileges to protected resources on a deployment container, such as a web server.

What's New About Version 2.2 Web Agents

Several important features have been added to the web agents in the version 2.2 release as follows:

Policy Agents Developed in the OpenSSO Project

Some policy agents such as the Apache HTTP Server 2.2 and Sun Java System Web Server 7.0 web agents have been developed as part of the OpenSSO project. These agents have features that are unique from other version 2.2 web agents that were developed independently of the OpenSSO project.

The features of web agents that are unique to the OpenSSO project are described in Chapter 2, "Vital Installation Information for a Web Agent in Policy Agent 2.2." Some of the feature differences, such as the use of the `agentadmin` program, originated with the J2EE agents and have been part of the J2EE agents since Policy Agent 2.2 was first released. To learn more about J2EE agents and the OpenSSO project, see the following resources:

J2EE agents	Chapter 3, "Access Manager Policy Agent 2.2 J2EE Agents: Compatibility, Supported Servers, and Documentation," in *Sun Java System Access Manager Policy Agent 2.2 User's Guide*
OpenSSO Project	`http://opensso.dev.java.net/`

Support for Fetching User Session Attributes

Before this release of web agents, header and cookie information was retrieved, or *sourced*, solely from user profile properties. Now, header and cookie information can also be sourced from session properties.

Use the following property to choose how you want session attributes retrieved:

```
com.sun.am.policy.agents.config.session.attribute.fetch.mode
```

For the preceding property, the following modes are available as retrieval methods:

- NONE
- HTTP_HEADER
- HTTP_COOKIE

The following example illustrates this property with the retrieval method set to `HTTP_HEADER`:

```
com.sun.am.policy.agents.config.session.attribute.fetch.mode = HTTP_HEADER
```

The source of header and cookie information is controlled by the following configuration property in the web agent `AMAgent.properties` configuration file:

```
com.sun.am.policy.agents.config.session.attribute.map
```

This configuration property has the same format as an LDAP header property. The following is an example of how this configuration property can be set:

```
com.sun.am.policy.agents.config.session.attribute.map =
name-of-session-attribute1|name-of-header-attribute1,
name-of-session-attribute2|name-of-header-attribute2
```

Where *name-of-session-attribute1* and other similarly named properties, or *attributes*, in the preceding code represent actual property names.

Benefit - Support for Fetching User Session Attributes: The benefit of this feature is that session properties can be more effective for transferring information, especially dynamic information. Prior to this release, agents could only fetch users' profile attributes, which tend to be static attributes. However, session attributes allow applications to obtain dynamic user information when necessary. Since this feature allows you to fetch non-user profile attributes, you can fetch attributes such as SAML assertion.

A good use case for fetching user session attributes presents itself when a post authentication plug-in is involved. A post authentication plug-in performs some tasks right after user authentication. You can configure a post authentication plug-in to fetch data from an external

data repository and then set this data as session attributes for that user. These session attributes can be retrieved by the web container and made available to the application.

Policy-Based Response Attributes

Starting with this release of web agents, a new method is available for retrieving LDAP user attributes based on Access Manager policy configurations.

Policy-based response attributes take advantage of functionality now available in Access Manager that involves querying policy decisions. In previous versions of Access Manager, header attributes could only be determined by the list of attribute-value pairs in the agent configuration. Now, header attributes can also be determined by Access Manager policy configurations. With policy-based response attributes you can define attribute-value pairs at each policy definition as opposed to the method used in prior versions of Access Manager, which only allowed pairs to be defined globally in the agent configuration. For more information on policy-based response attributes, see "Providing Personalization With Policy-Based Response Attributes" on page 66

Benefit - Policy-Based Response Attributes: The benefit of policy-based response attributes is that they allow for personalization, improve the deployment process, allow greater flexibility in terms of customization, and provide central and hierarchical control of attribute values.

Personalization is provided in that an application can retrieve specific user information, such as a name, from a cookie or HTTP header and present it to the user in the browser.

Defining attribute-value pairs at each policy definition instead of at the root level allows an attribute value to be distributed only to the applications that need it. Furthermore, you can customize attribute names allowing the same attribute name to have entirely different property values for two different applications.

For example, you can gray out a button for certain users on certain applications depending on the polices and the attributes defined. You can have fine-grained customizations using this feature.

Composite Advice

Starting with this release, web agents provide a composite advice feature. This feature allows the policy and authentication services of Access Manager to decouple the advice handling mechanism of the agents. This allows you to introduce and manage custom advices by solely writing Access Manager side plug-ins. Starting with this release, you are not required to make changes on the agent side. Such advices are honored automatically by the composite advice handling mechanism.

Benefit - Composite Advice: A benefit of composite advice is that you can incorporate a custom advice type without having to make changes to an agent deployment. Prior to the 2.2 release of web agents, no interface existed on the client side to write client-side plug-ins.

Additional Method for Fetching the REMOTE_USER Server Variable

Prior to this release of web agents, the only method for fetching the value of the `REMOTE_USER` variable set by an agent was from session properties. Starting with the 2.2 release, the value can also be fetched from user profiles. This fetching process uses LDAP.

By default the value for the `REMOTE_USER` is fetched from the session. If the value needs to be fetched from LDAP, the following property needs to be defined in the web agent `AMAgent.properties` configuration file:

```
com.sun.am.policy.am.userid.param.type = LDAP
```

The following property can still be used to configure the key (*key* refers to the value assigned to this property) that needs to be searched. In addition to setting the preceding property, you need to give the correct LDAP attribute name for the following property.

```
com.sun.am.policy.am.userid.param
```

For example the property will be set as follows:

```
com.sun.am.policy.am.userid.param = ldap-attribute-name
```

where *ldap-attribute-name* represents the name of an LDAP attribute.

To enable the `REMOTE_USER` setting for a globally not-enforced URL as specified in the web agent `AMAgent.properties` configuration file (this is a URL that can be accessed by unauthenticated users) you must set the following property in the web agent `AMAgent.properties` configuration file to `true`. While the following example, has the value is set to `true`, the default value is `false`:

```
com.sun.am.policy.agents.config.anonymous_user.enable = true
```

When you set this property value to `true`, the value of `REMOTE_USER` will be set to the value contained in the following property in the web agent `AMAgent.properties` configuration file. In the following example the value is set to `anonymous`, which is the default:

```
com.sun.am.policy.agents.config.anonymous_user = anonymous
```

Benefit - Additional Method for Fetching the REMOTE_USER Server Variable: The benefit of this feature is that it gives better customization for end users since the `REMOTE_USER` server variable can now be obtained from either session attributes or user profile attributes.

Also, you do not need to write server-side plug-in code in order to add session attributes after authentication, which is necessary when this value is fetched from session properties.

Malicious Header Attributes Automatically Cleared by Agents

Starting with this release of web agents, malicious header attributes are automatically cleared.

Benefit - Header Attributes Set by Agents Automatically Cleared: The benefit of this automatic clean up is that security is improved. Header information that is *not* automatically cleared has greater risk of being accessed.

Load Balancing Enablement

Starting with this release of web agents, the default agent host port and protocol settings can be overridden to enable load balancing. For more information, see "Enabling Load Balancing" on page 75.

Benefit - Load Balancing Enablement: The benefit of this override capability is that you do not need to manually change the hostname, port, and protocol settings to enable load balancing.

Support for Heterogeneous Agent Types on the Same Machine

Starting with this release of web agents, you can install different types of agents on the same machine. Prior to this release, you could not install web agents from different product groups on the same machine. For example, previously, an agent instance for Sun Java System Web Server 6.1 and an agent instance for Apache 2.0.52 could not be installed on the same machine. Now, they can.

Benefit - Support for Heterogeneous Agent Types on Same Machine: The benefit of this feature is that a deployment that has agents in a multi-server scenario requires fewer hardware sources.

Support for Turning Off FQDN Mapping

Starting with this release, fully qualified domain name (FQDN) mapping of HTTP requests can be disabled. In prior web agent releases, the methods employed for checking if a user is using a valid URL could not be turned off.

This checking capability is controlled by the FQDN default and the FQDN map properties in the web agent `AMAgent.properties` configuration file as follows:

- `com.sun.am.policy.agents.config.fqdn.default`
- `com.sun.am.policy.agents.config.fqdn.map`

A toggling capability has been introduced that allows FQDN checking to be turned off. The following property allows for this toggling:

```
com.sun.am.policy.agents.config.fqdn.check.enable
```

The following property specifies whether the request URLs that are present in user requests are checked against the FQDN default and the FQDN map properties by the web agent:

```
com.sun.am.policy.agents.config.fqdn.check.enable
```

The valid values are `true` and `false`.

`true` The request URLs that are present in user requests are checked against FQDN values.

`false` No checking occurs against FQDN values.

The default value is `true`. If no value is specified, then the default value, `true`, is used.

Benefit - Support for Turning Off FQDN Mapping: This feature allows you to turn off or on FQDN mapping comparison. This feature can be beneficial when a deployment includes a number of virtual servers for which the agent is configured using FQDN mapping.

Backward Compatibility With Access Manager 6.3

Policy Agent 2.2 is backward compatible with Access Manager 6.3 Patch 1 or greater.

Note – Policy Agent 2.2 is only compatible with Access Manager 6.3 when the Access Manager patch has been applied.

Be aware that Policy Agent 2.2 takes advantage of certain features that exist in Access Manager 7.1 that do not exist in Access Manager 6.3, such as “composite advices,” “policy-based response attributes,” and others.

◆ ◆ ◆ CHAPTER 2

Vital Installation Information for a Web Agent in Policy Agent 2.2

This chapter applies to Policy Agent 2.2 web agents developed through the OpenSSO project. These web agents differ from other web agents as described in the following sections:

Although this chapter applies to all web agents developed as part of the OpenSSO project, occasionally a specific web agent is used in examples. These examples show only the general format; you will need to replace the information for the specific web agent you are deploying.

Distribution File Format for Web Agents

The distribution files for agents in the Policy Agent 2.2 software set are different, depending upon whether the agent was developed as part of the OpenSSO project. Since the Apache HTTP Server 2.2 agent was developed in the OpenSSO project, it is available only as a `.zip` file, regardless of the platform where it will be deployed.

Introduction of the `agentadmin` Program for Web Agents

Web agents developed as part of the OpenSSO project, including the Apache HTTP Server 2.2 agent, project now use the `agentadmin` program for installation, uninstallation, and other tasks.

The location of the `agentadmin` program is:

PolicyAgent-base`/bin`

For more information about *PolicyAgent-base*, see Example 2–15.

Consideration for the `agentadmin` program for web agents include:

- Web agent installation and uninstallation is achieved with the agentadmin command.
- All tasks performed by the agentadmin program, except those involving uninstallation, require the acceptance of a license agreement. This agreement is presented only the first time you use the program.
- The following table describes the agentadmin command options that can be used with web agents.

  A detailed explanation of each option follows the table.

TABLE 2–1 agentadmin Program: Supported Options for Web Agents

Option	Task Performed
--install	Installs a new agent instance
--uninstall	Uninstalls an existing agent instance
--listAgents	Displays details of all configured agents
--agentInfo	Displays details of the agent corresponding to the specified agent IDs
--version	Displays the version information
--usage	Displays the usage message
--help	Displays a brief help message

agentadmin --install

This section demonstrates the format and use of the agentadmin command with the --install option.

EXAMPLE 2–1 Command Format: agentadmin --install

The following example illustrates the format of the agentadmin command with the --install option:

```
./agentadmin --install
[--useResponse] [--saveResponse] filename
```

The following arguments are supported with the agentadmin command when using the --install option:

--saveResponse — Use this argument to save all supplied responses to a state file, or response file, represented as *filename* in command examples. The response file, or state file, can then be used for silent installations.

`--useResponse`	Use this argument to install a web agent in silent mode as all installer prompts are answered with responses previously saved to a response file, represented as *filename* in command examples. When this argument is used, the installer runs in non-interactive mode. At which time, user interaction is not required.
filename	Use this argument to specify the name of a file that will be created as part of the processing of this command. This file stores your responses when this argument is used in conjunction with the `--saveResponse` argument and provides your responses when this argument is used in conjunction with the `--useResponse` argument.

EXAMPLE 2–2 Command Usage: `agentadmin --install`

When you issue the `agentadmin` command, you can choose the `--install` option. With the `--install` option, you can choose the `--saveResponse` argument, which requires a file name be provided. The following example illustrates this command when the file name is `myfile`:

```
./agentadmin --install --saveResponse myfile
```

Once the installer has executed the preceding command successfully, the responses are stored in a state file that can be used for later runs of the installer.

If desired, you can modify the state file and configure the second installation with a different set of configuration parameters.

Then you can issue another command that uses the `./agentadmin --install` command and the name of the file that you just created with the `--saveResponse` argument. The difference between the previous command and this command is that this command uses the `--useResponse` argument instead of the `--saveResponse` argument. The following example illustrates this command:

```
./agentadmin --install --useResponse myfile
```

With this command, the installation prompts run the installer in silent mode, registering all debug messages in the install logs directory.

agentadmin --uninstall

This section demonstrates the format and use of the `agentadmin` command with the `--uninstall` option.

EXAMPLE 2–3 Command Format: `agentadmin --uninstall`

The following example illustrates the format of the `agentadmin` command with the `--uninstall` option:

```
./agentadmin --uninstall [--useResponse] [--saveResponse] filename
```

The following arguments are supported with the `agentadmin` command when using the `--uninstall` option:

`--saveResponse`	Use this argument to save all supplied responses to a state file, or response file, represented as *filename* in command examples. The response file, or state file, can then be used for silent uninstallations.
`--useResponse`	Use this argument to uninstall a web agent in silent mode as all uninstaller prompts are answered with responses previously saved to a response file, represented as *filename* in command examples. When this argument is used, the uninstaller runs in non-interactive mode. At which time, user interaction is not required.
filename	Use this argument to specify the name of a file that will be created as part of the processing of this command. This file stores your responses when this argument is used in conjunction with the `--saveResponse` argument and provides your responses when this argument is used in conjunction with the `--useResponse` argument.

EXAMPLE 2–4 Command Usage: `agentadmin --uninstall`

When you issue the `agentadmin` command, you can choose the `--uninstall` option. With the `--uninstall` option, you can choose the `--saveResponse` argument, which requires a file name be provided. The following example illustrates this command where the file name is `myfile`:

```
./agentadmin --uninstall --saveResponse myfile
```

Once the uninstaller has executed the preceding command successfully, the responses are stored in a state file that can be used for later runs of the uninstaller.

If desired, you can modify the state file and configure the second uninstallation with a different set of configuration parameters.

Then you can issue another command that uses the `./agentadmin --uninstall` command and the name of the file that you just created with the `--saveResponse` argument. The difference between the previous command and this command is that this command uses the `--useResponse` argument instead of the `--saveResponse` argument. The following example illustrates this command:

EXAMPLE 2–4 Command Usage: agentadmin --uninstall *(Continued)*

```
./agentadmin --uninstall --useResponse myfile
```

With this command, the uninstallation prompts run the uninstaller in silent mode, registering all debug messages in the install logs directory.

agentadmin --listAgents

This section demonstrates the format and use of the agentadmin command with the --listAgents option.

EXAMPLE 2–5 Command Format: agentadmin --listAgents

The following example illustrates the format of the agentadmin command with the --listAgents option:

```
./agentadmin --listAgents
```

No arguments are currently supported with the agentadmin command when using the --listAgents option.

EXAMPLE 2–6 Command Usage: agentadmin --listAgents

Issuing the agentadmin command with the --listAgents option provides you with information about all the configured web agents on that machine. For example:

```
The following are the details for agent Agent_001 :-
Apache 2.2 Web Server Config Directory: /usr/local/apache2/conf
```

Notice that the agentadmin program provides unique names, such as Agent_001, for all web agents that protect the same instance of a deployment container, in this case Apache HTTP Server 2.2. Each name uniquely identifies the web agent instance.

Note – The string "Agent" in Agent_00x is configurable. You can change this string by editing the following file: *PolicyAgent-base*/config/AMToolsConfig.properties

---

agentadmin --agentInfo

This section demonstrates the format and use of the agentadmin command with the --agentInfo option.

EXAMPLE 2–7 Command Format: agentadmin --agentInfo

The following example illustrates the format of the agentadmin command with the --agentInfo option:

```
./agentadmin --agentInfo AgentInstance-Dir
```

The following argument is supported with the agentadmin command when using the --agentInfo option:

AgentInstance-Dir Use this option to specify which agent instance directory, therefore which agent instance such as Agent_002, you are requesting information about.

EXAMPLE 2–8 Command Usage: agentadmin --agentInfo

Issuing the agentadmin command with the --agentInfo option provides you with information about the web agent instance that you name in the command. For example, if you want information about a web agent instance named Agent_001 configured on Apache HTTP Server 2.2, issue the command shown in the following example:

```
./agentadmin --agentInfo Agent_001

The following are the details for agent Agent_001:-
Apache 2.2 Web Server Config Directory:
/usr/local/apache2/conf
```

agentadmin --version

This section demonstrates the format and use of the agentadmin command with the --version option.

EXAMPLE 2–9 Command Format: agentadmin --version

The following example illustrates the format of the agentadmin command with the --version option:

```
./agentadmin --version
```

No arguments are currently supported with the agentadmin command when using the --version option.

EXAMPLE 2–10 Command Usage: `agentadmin --version`

Issuing the `agentadmin` command with the `--version` option provides you with version information for the configured web agents on that machine.

agentadmin --usage

This section demonstrates the format and use of the `agentadmin` command with the `--usage` option.

EXAMPLE 2–11 Command Format: `agentadmin --usage`

The following example illustrates the format of the `agentadmin` command with the `--usage` option:

```
./agentadmin --usage
```

No arguments are currently supported with the `agentadmin` command when using the `--usage` option.

EXAMPLE 2–12 Command Usage: `agentadmin --usage`

Issuing the `agentadmin` command with the `--usage` option provides you with a list of the options available with the `agentadmin` program and a short explanation of each option. The following text is the output you receive after issuing this command:

```
./agentadmin --usage

Usage: agentadmin <option> [<arguments>]

The available options are:
--install: Installs a new Agent instance.
--uninstall: Uninstalls an existing Agent instance.
--version: Displays the version information.
--listAgents: Displays details of all the configured agents.
--agentInfo: Displays details of the agent corresponding to the specified
agent ID.
--usage: Display the usage message.
--help: Displays a brief help message.
```

The preceding output serves as the content for the table of `agentadmin` options, introduced at the beginning of this section.

agentadmin --help

This section demonstrates the format and use of the `agentadmin` command with the `--help` option.

EXAMPLE 2–13 Command Format: `agentadmin --help`

The following example illustrates the format of the `agentadmin` command with the `--help` option:

```
./agentadmin --help
```

No arguments are currently supported with the `agentadmin` command when using the `--help` option.

EXAMPLE 2–14 Command Usage: `agentadmin --help`

Issuing the `agentadmin` command with the `--help` option provides similar results to issuing the `agentadmin` command with the `--usage` option. Both commands provide the same explanations for the options they list. With the `--usage` option, all `agentadmin` command options are explained. With the `--help` option, explanations are not provided for the `--usage` option or for the `--help` option itself.

A another difference is that the `--help` option also provides information about the format of each option while the `--usage` option does not.

Web Agent Directory Structure

The Policy Agent installation directory is referred to as the Policy Agent base directory (or *PolicyAgent-base* in examples). The location of this directory and its internal structure are important facts that are described in this section. While the specifics of the agent directory structure described in this section apply to Policy Agent 2.2 for Apache HTTP Server 2.2, they do not apply to web agents that were not developed through the OpenSSO project.

Location of the Web Agent Base Directory in Policy Agent 2.2

Unzipping the web agent binaries creates a directory named `web_agents`, within which an agent-specific directory is created. The Apache HTTP Server 2.2 agent directory is `apache22_agent`.

This agent-specific directory is the Policy Agent base directory, referred to throughout this guide as the *PolicyAgent-base* directory. For the full path to the *PolicyAgent-base* directory, see Example 2–15, which follows.

EXAMPLE 2–15 Policy Agent Base Directory

The directory you choose in which to unzip the web agent binaries is referred to here as *Agent-HomeDirectory*. The following is the path to the *PolicyAgent-base* directory of Policy Agent 2.2 for Apache HTTP Server 2.2:

Agent-HomeDirectory`/web_agents/apache22_agent`

References in this book to the *PolicyAgent-base* directory are references to the preceding path.

Inside the Web Agent Base Directory

After you finish installing an agent by issuing the `agentadmin ---install` command and interacting with the installer, you will need to access web agent files in order to configure and otherwise work with the product. Within the Policy Agent base directory are various subdirectories that contain all agent configuration and log files. The structure of the Policy Agent base directory for a web agent developed through the OpenSSO project is illustrated in Table 2–2.

The list that follows the table provides information about many of the items in the example Policy Agent base directory. The Policy Agent base directory is represented in code examples as *PolicyAgent-base*. The full path to any item in this directory is as follows:

PolicyAgent-base/*item-name*

where *item-name* represents the name of a file or subdirectory. For example, the full path to the `bin` directory is as follows:

PolicyAgent-base`/bin`

TABLE 2–2 Example of Policy Agent Base Directory for a Web Agent

Directory Contents: Files and Subdirectories	
`license.txt`	`etc`
`README`	`lib`
`bin`	`locale`
`config`	`logs`

TABLE 2–2 Example of Policy Agent Base Directory for a Web Agent *(Continued)*

Directory Contents: Files and Subdirectories	
`data`	`Agent_001`

The preceding example of *PolicyAgent-base* lists files and directories you are likely to find in this directory. The notable items in this directory are summarized in the list that follows:

`bin` This directory contains the `agentadmin` script for the agent bits. You will use this script a great deal. For details about the tasks performed with this script, see "Introduction of the `agentadmin` Program for Web Agents" on page 23.

`logs` This directory contains installation-related log files, for example log files created after you issue the `agentadmin` command.

Log information is stored in the installation log file after you install a web agent instance. The following is the location of this log file:

PolicyAgent-base`/logs/audit/install.log`

`lib` The `lib` directory has a list of all the agent libraries that are used by the installer as well as the agent run time.

`locale` This directory has all the agent installer information as well as agent run time specific locale information pertaining to the specific agent.

`data` This directory has all the installer specific data.

Caution – Do not edit any of the files in the `data` directory under any circumstance. If this directory or any of its content loses data integrity, the `agentadmin` program cannot function normally.

`Agent_001` The full path for this directory is as follows:

PolicyAgent-base`/`*AgentInstance-Dir*

where *AgentInstance-Dir* refers to an agent instance directory, which in this case is `Agent_001`.

Note – This directory does not exist until you successfully install the first instance of a web agent. Once you have successfully executed one run of the `agentadmin --install` command, an agent specific directory, `Agent_00x` is created in the Policy Agent base directory. This directory is uniquely tied to an instance of the deployment container, such as a web server instance. Depending on the number of times the `agentadmin --install` command is run, the number that replaces the `x` in the `Agent_00x` directory name will vary.

Furthermore, the string "Agent" in `Agent_00x` is configurable. You can change this string by editing the following file:
PolicyAgent-base`/config/AMToolsConfig.properties`

After you successfully install the first instance of a web agent, an agent instance directory named `Agent_001` appears in the Policy Agent base directory. The path to this directory is as follows:

PolicyAgent-base`/Agent_001`

The next installation of the agent creates an agent instance directory named `Agent_002`. The directories for uninstalled agents are not automatically removed. Therefore, if `Agent_001` and `Agent_002` are uninstalled, the next agent instance directory is `Agent_003`.

Agent instance directories contain directories named `config` and `logs`.

Note – When a web agent is uninstalled, the `config` directory is removed from the agent instance directory but the `logs` directory still exists.

The following table is an example of the contents of an agent instance, such as `Agent_001`, directory.

Example of an Agent Instance (`Agent_001`) Directory
`logs`
`config`

`logs` Two subdirectories exist within this directory as follows:

`audit` This directory contains the local audit trail for the agent instance.

`debug` This directory has all the agent-specific debug information. When the agent runs in full debug mode, this directory stores all the debug files that are generated by the agent code.

`config` This directory contains the web agent `AMAgent.properties` configuration file that is specific to the agent instance. Each web agent can be configured by a unique instance of the web agent `AMAgent.properties` configuration file. This file holds the key to the agent behavior at runtime.

◆ ◆ ◆ CHAPTER 3

About the Apache HTTP Server 2.2 Policy Agent

While the individual web agents are similar in terms of installation and configuration, they have unique characteristics that allow them to interact with the underlying deployment container. This chapter describes the characteristics that are unique to the Apache HTTP Server 2.2 policy agent, including:

Supported Platforms and Compatibility for the Apache HTTP Server 2.2 Policy Agent

Supported Platforms for the Apache HTTP Server 2.2 Policy Agent

TABLE 3–1 Supported Platforms for the Apache HTTP Server 2.2

Agent for	Supported Platforms
Apache HTTP Server 2.2	Solaris™ Operating System (OS) for the SPARC® platform, versions 8, 9, and 10
	Solaris (OS) for x86 platforms, versions 8, 9, and 10
	Red Hat Enterprise Linux Advanced Server, version 3.0, 32–bit and 64–bit
	Red Hat Enterprise Linux Advanced Server, version 4.0, 32–bit and 64–bit
	SUSE Linux 10.1
	Windows 2003, Enterprise Edition
	Windows 2003, Standard Edition

Note – The Apache HTTP Server 2.2 agent is verified with the Apache default Multi-Processing Module (MPM) on supported platforms, as described in: `http://httpd.apache.org/docs/2.2/mpm.html`.

Compatibility of the Apache HTTP Server 2.2 Agent With Sun Java System Access Manager

Compatibility With Access Manager 7.1 and Access Manager 7 2005Q4

The Apache HTTP Server 2.2 agent is compatible with Access Manager 7.1 (both Realm Mode and Legacy Mode) and Access Manager 7 2005Q4 (also both Realm Mode and Legacy Mode).

To ensure that all Access Manager enhancements and fixes are applied, install the latest Access Manager patches . For the list of patches that can be installed, see the compatibility information in the *Sun Java System Access Manager Policy Agent 2.2 Release Notes*.

Compatibility With Access Manager 6 2005Q1 (6.3)

The Apache HTTP Server 2.2 agent is compatible with Access Manager 6 2005Q1 (6.3) Patch 1 or greater. However, certain limitations can apply. For more information about these limitations, see "Backward Compatibility With Access Manager 6.3" on page 22.

Information Specific to the Apache HTTP Server 2.2 Agent

This section describes characteristics that are unique about to the Apache HTTP Server 2.2 agent, including "Development Through the OpenSSO Project" on page 37

The Apache HTTP Server 2.2 agent is unique in that it does not support the following features:

- "Notifications" on page 37
- "Log Rotation" on page 38

Development Through the OpenSSO Project

The Apache HTTP Server 2.2 agent was developed as part of the OpenSSO project and therefore differs from other web agents developed outside of this project. For more information about the differences, see Chapter 2, "Vital Installation Information for a Web Agent in Policy Agent 2.2."

For information about the OpenSSO project, see `http://opensso.dev.java.net/`

Notifications

The Apache HTTP Server 2.2 agent does not support notifications. Therefore, updating the cache through a notification mechanism is not an available feature. However, since the notification mechanism is available for other agents in the Policy Agent 2.2 software set, a property exists in the web agent `AMAgent.properties` configuration file. The property that controls the notification mechanism, `com.sun.am.notification.enable`, is set to `false` for this agent. Do not set this property to `true` for this agent, because it might result in unexpected behavior.

The two following properties can also affect notifications for most agents:

```
com.sun.am.notification.url
override_notification.url
```

However, you can ignore these properties for this agent.

Log Rotation

While most web agents have log rotation turned on by default, this agent has log rotation turned off by default. Log rotation is controlled by a property in the web agent `AMAgent.properties` configuration file. The following example illustrates the default setting of this property for this agent:

```
com.sun.am.policy.agents.config.local.log.rotate = false
```

Do not change the value of this property to `true` for this agent. A setting of `true` results in log rotation that is inconsistent and unpredictable.

◆ ◆ ◆ CHAPTER 4

Installing the Apache HTTP Server 2.2 Policy Agent

This chapter describes how to install the Apache HTTP Server 2.2 agent, including:

Note – Because the Apache HTTP Server 2.2 agent was developed as part of the OpenSSO project, the distribution files are available only in `.zip` file format. Also, the installation is similar for all platforms, so this chapter is not divided into platform-specific sections

After you have successfully installed the agent, as described in this chapter, complete the post-installation tasks described in Chapter 6, "Post-Installation Tasks for the Apache HTTP Server 2.2 Agent."

Preparing to Install the Apache HTTP Server 2.2 Agent

Follow the specific steps in this section before you install the web agent to reduce the chance of complications occurring during and after the installation.

▼ To Prepare to Install the Apache HTTP Server 2.2 Agent

1 **Ensure that the Apache HTTP Server 2.2 agent is supported on the desired platform, as listed in "Supported Platforms and Compatibility for the Apache HTTP Server 2.2 Policy Agent" on page 35.**

2 **If necessary, install and configure the Apache HTTP Server 2.2 web container.**

Also, check that the Apache HTTP Server 2.2 has the latest patches.

For more information, refer to the Apache HTTP Server 2.2 documentation:
`http://httpd.apache.org/docs/2.2/`

3 **Set your `JAVA_HOME` environment variable to a JDK version 1.5.0 or higher.**

The installation program requires that your `JAVA_HOME` variable be set correctly. If you have incorrectly set the `JAVA_HOME` variable, the `setup` script will prompt you to supply the correct path:

```
Please enter JAVA_HOME path to pick up java:
```

4 **(Conditional) Create a valid agent profile in the Access Manager Console, if one has not already been created.**

Web agents can function using the default agent profile (`UrlAccessAgent`), but creating a different agent profile provides greater security. You must also create a different agent profile if Access manager is configured for cross domain single sign-on (CDSSO).

For information about how to create an agent profile, see Chapter 5, "Relationship Between the Agent Profile and Web Agents."

To avoid configuration problems for the agent, you must know the agent profile ID and password used to create the agent profile. You must specify the agent profile password in the next step, and you must enter the agent profile ID when you install the agent.

5 **Create an agent profile password word file.**

An agent profile password file is a text file with one line that contains the agent profile password. You will need to provide the path to this file during the agent installation process. By using an agent profile password file, you do not need to enter the password during the agent installation. Set the security permissions for this file as required for your specific deployment.

6 **Unzip the web agent `.zip` file. For example:**

```
# unzip apache_v22_platform_agent.zip
```

where *platform* identities the specific platform where you are installing the agent:

`SunOS` — Solaris SPARC systems

`SunOS_x86` — Solaris x86 systems

`Linux` — Linux systems

`WINNT` — Windows systems

7 **On UNIX-based systems, ensure that the following programs have executable permissions:**

- `agentadmin`
- `crypt_util`
- `certutil`

These programs are located in the *PolicyAgent-base*`/bin` directory. For example, to secure these programs on Solaris systems:

```
# chmod +x agentadmin crypt_util certutil
```

Installing the Apache HTTP Server 2.2 Agent

The agent installation program (`agentadmin`) performs the following operations:

- Creates the Apache HTTP Server 2.2 agent instance directory
- Sets values (tag swapping) in the `AMAgent.properties` file
- Updates agent information in the Apache HTTP Server 2.2 `httpd.conf` file

▼ To Install the Apache HTTP Server 2.2 Agent

1 **Change to the** *PolicyAgent-base*`/bin` **directory.**

PolicyAgent-base`/bin`

For information about the *PolicyAgent-base* directory, see"Location of the Web Agent Base Directory in Policy Agent 2.2" on page 30.

2 **Issue the following command:**

```
./agentadmin --install
```

3 **If you receive license agreement information, accept or reject the agreement. If you reject any portion of the agreement, the installation program will end.**

The license agreement is displayed only during the first run of the `agentadmin` program.

4 **After you accept the license agreement (if necessary), provide the following information when requested by the installation program (or accept the default values):**

- Path to the Apache HTTP Server 2.2 configuration directory
- Access Manager services host name, port, and protocol

- Access Manager services deployment URI
- Agent host name, port, and protocol
- Agent profile name and password file

The prompts are shown in "Example of the Installation Program Interaction for the Apache HTTP Server 2.2 Agent" on page 43.

Key points about the installation program to consider include:

- Each step in the installation program includes an explanation that is followed by a more succinct prompt.
- For most of the steps you can type any of the following characters to get the results described:

  ? Type the question mark to display Help information for that specific step.

  < Type the left arrow symbol to go back to the previous interaction.

  ! Type the exclamation point to exit the program.
- Most of the steps provide a default value that can be accepted or replaced. If a default value is correct for your site, accept it. If it is not correct, enter the correct value.

5 After you entered all values, the installation program displays a summary of your responses.

Note the agent instance name, such as `Agent_001`. You might be prompted for this name during the configuration process.

- If you are satisfied with the summary, choose 1 (the default).
- If you want to edit input from the last interaction, choose 2.
- If you want to edit input starting at the beginning of the installation program, choose 3.
- If you want to exit the installation program without installing, choose 4.

Edit your responses if needed. When you are satisfied with your responses, choose option 1 to continue with the installation.

About the Installation Prompts for the Apache HTTP Server 2.2 Agent

The following list provides information about specific prompts in the installation.

Apache HTTP Server 2.2 configuration directory path
: Enter the path to the Apache HTTP Server 2.2 configuration directory. The default is `/usr/local/apache2/conf`.

Access Manager services host name, port, and protocol
: Enter the fully qualified host name, port, and protocol for the server where Access Manager is installed. The default port is 80, and the default protocol is `http`.

Access Manager services deployment URI
: Enter the URI that will be used to for Access Manager. The default value is `/amserver`.

Agent profile name
: To use an agent profile, you must create the profile as a pre-installation step, as described in "Preparing to Install the Apache HTTP Server 2.2 Agent" on page 39. For more information about creating an agent profile, see also Chapter 5, "Relationship Between the Agent Profile and Web Agents." The default is `UrlAccessAgent`.

  Web agents can function using the default agent profile (`UrlAccessAgent`), but creating a different agent profile provides greater security. You must also create a different agent profile if Access manager is configured for cross domain single sign-on (CDSSO).

Agent profile password file
: You should create the agent profile password file as a pre-installation step as described in "Preparing to Install the Apache HTTP Server 2.2 Agent" on page 39.

  When the installation program prompts you for the password for the agent, enter the fully qualified path to this password file.

Example of the Installation Program Interaction for the Apache HTTP Server 2.2 Agent

The following example shows a sample installation for the Apache HTTP Server 2.2 agent.

```
********************************************************************************
Welcome to the Access Manager Policy Agent for Apache Server If the Policy
Agent is used with Federation Manager services, User needs to enter
information relevant to Federation Manager.

********************************************************************************


Do you completely agree with all the terms and conditions of this License
Agreement (yes/no): [no]: yes


Enter the complete path to the directory which is used by Apache Server to
store its configuration Files. This directory uniquely identifies the
Apache Server instance that is secured by this Agent.
[ ? : Help, ! : Exit ]
Enter the Apache Server Config Directory Path [/usr/local/opt/apache2/conf]:
```

```
/usr/local/opt/apache2/conf


Enter the fully qualified host name of the server where Access Manager
Services are installed.
[ ? : Help, < : Back, ! : Exit ]
Access Manager Services Host: amhost.example.com


Enter the port number of the Server that runs Access Manager Services.
[ ? : Help, < : Back, ! : Exit ]
Access Manager Services port [80]: 8080


Enter http/https to specify the protocol used by the Server that runs Access
Manager services.
[ ? : Help, < : Back, ! : Exit ]
Access Manager Services Protocol [http]:


Enter the Deployment URI for Access Manager Services.
[ ? : Help, < : Back, ! : Exit ]
Access Manager Services Deployment URI [/amserver]:


Enter the fully qualified host name on which the Web Server protected by the
agent is installed.
[ ? : Help, < : Back, ! : Exit ]
Enter the Agent Host name: agenthost.example.com


Enter the preferred port number on which the Web Server provides its
services.
[ ? : Help, < : Back, ! : Exit ]
Enter the port number for Web Server instance [80]: 7000


Select http or https to specify the protocol used by the Web server instance
that will be protected by Access Manager Policy Agent.
[ ? : Help, < : Back, ! : Exit ]
Enter the Preferred Protocol for Web Server instance [http]:


Enter a valid Agent profile name. Before proceeding with the agent
installation, please ensure that a valid Agent profile exists in Access
Manager.
[ ? : Help, < : Back, ! : Exit ]
Enter the Agent Profile name [UrlAccessAgent]:
```

```
Enter the path to a file that contains the password to be used for identifying
the Agent.
[ ? : Help, < : Back, ! : Exit ]
Enter the path to the password file: /opt/agent-profile-password-file


---------------------------------------------
SUMMARY OF YOUR RESPONSES
---------------------------------------------
Apache Server Config Directory : /usr/local/opt/apache2/conf
Access Manager Services Host : amhost.example.com
Access Manager Services Port : 8080
Access Manager Services Protocol : http
Access Manager Services Deployment URI : /amserver
Agent Host name : agenthost.example.com
Web Server Instance Port number : 7000
Protocol for Web Server instance : http
Agent Profile name : UrlAccessAgent
Agent Profile Password file name : /opt/agent-profile-password-file

Verify your settings above and decide from the choices below.
1. Continue with Installation
2. Back to the last interaction
3. Start Over
4. Exit
Please make your selection [1]:
```

Summary of an Agent Installation

At the end of the installation process, the installation program prints the status of the installation along with the installed agent information. The information that the program displays can be very useful. The program also displays the location of specific files, which can be of great importance.

You might want to view the installation log file after the installation is complete, before performing the post-installation steps as described in Chapter 6, "Post-Installation Tasks for the Apache HTTP Server 2.2 Agent."

The location of directories displayed by the installer are specific. However, throughout this guide and specifically in the summary of the agent installation shown in this section, *PolicyAgent-base* represents the directory where the distribution files are stored for a specific web agent:

Agent-HomeDirectory`/web_agents/apache22_agent`

where *Agent-HomeDirectory* is the directory where you unzipped the web agent distribution file.

Information regarding the location of the web agent base directory is also described in "Location of the Web Agent Base Directory in Policy Agent 2.2" on page 30.

The installation program prints the following information:

```
SUMMARY OF AGENT INSTALLATION
---------------------------
Agent instance name: Agent_001
Agent Configuration file location:
PolicyAgent-base/Agent_001/config/AMAgent.properties
Agent Audit directory location:
PolicyAgent-base/Agent_001/logs/audit
Agent Debug directory location:
PolicyAgent-base/Agent_001/logs/debug

Install log file location:
PolicyAgent-base/logs/audit/install.log

Thank you for using Access Manager Policy Agent
```

After the agent is installed, the directories shown in the previous example are created in the `Agent_00`*x* directory, which for this example is `Agent_001`. Those directories and files are described in the following paragraphs.

PolicyAgent-base`/Agent_001/config/AMAgent.properties`
: Location of the web agent `AMAgent.properties` configuration file for the agent instance. Every instance of a web agent has a unique copy of this file. You can configure this file to meet your site's requirements. For more information, see the following sections:

  - Appendix C, "Web Agent `AMAgent.properties` Configuration File"
  - Chapter 7, "Managing the Apache HTTP Server 2.2 Web Agent"

PolicyAgent-base`/Agent_001/logs/audit`
: Location of the web agent local audit trail.

PolicyAgent-base`/Agent_001/logs/debug`
: Location of all debug files required to debug an agent installation or configuration issue.

PolicyAgent-base`/logs/audit/install.log`
: Location of the file that has the agent install file location. If the installation failed for any reason, you can look at this file to diagnose the issue.

Implications of Specific Deployment Scenarios for the Apache HTTP Server 2.2 Agent

The following sections refer to specific deployment scenarios involving the Apache HTTP Server 2.2 agent. These scenarios can affect how you respond to prompts during the installation process. You might also need to perform additional configuration operations.

Configuring the Apache HTTP Server 2.2 Agent for Multiple Apache HTTP Server Virtual Hosts

Consider the scenario where the Apache HTTP Server 2.2 has two virtual hosts: `http://site1.example.com/` and `http://site2.example.com/`.

▼ To Enforce Access to the Individual Virtual Hosts

1. **Define the FQDN map property in the `AMAgent.properties` file as:**

```
com.sun.am.policy.agents.config.fqdn.map =
     valid1|site1.example.com,valid2|site2.example.com
```

2. **Define policies in Access Manager with virtual host names in the policy rules.**

▼ To Protect Only `http://site1.example.com/` and Not `http://site2.example.com/`

1. **Define the FQDN map property in the `AMAgent.properties` file as:**

```
com.sun.am.policy.agents.config.fqdn.map =
   valid1|site1.example.com,valid2|site2.example.com
```

2. **Define the `site2` URLs in the not-enforced URL list.**

Installing the Apache HTTP Server 2.2 Agent on the Access Manager Host

Note – Installing the Apache HTTP Server 2.2 agent on the Access Manager host is not recommended for production deployments because performance can be degraded.

However, if you want to install the agent on the Access Manager host on the same Apache HTTP Server 2.2 instance, add all of the URLs related to Access Manager to the not enforced URL list. Configuring the not-enforced URL list is described in "Configuring the Not-Enforced URL List" on page 62. If you are installing the agent on a different Apache HTTP Server 2.2 instance, configuration of the not-enforced URL list is not required.

Verifying a Successful Installation for the Apache HTTP Server 2.2 Agent

After installing the Apache HTTP Server 2.2 agent, ensure that it is installed successfully by using either or both of these methods:

- Attempt to access a resource on the Apache HTTP Server 2.2 deployment container where the agent is installed.

  If the web agent is installed correctly, accessing any resource should take you to the Access Manager login page. After a successful authentication, if the policy is properly defined, you should be able to access the resource, provided the policy definition in Access Manager allows it. The default is access denied.

- Check the web agent `AMAgent.properties` configuration file.

  Make sure that each property is set properly. For information about the properties in this file, see Appendix C, "Web Agent `AMAgent.properties` Configuration File."

◆ ◆ ◆ CHAPTER 5

Relationship Between the Agent Profile and Web Agents

This section describes how to create or update an agent profile in the Access Manager Console, including:

- "Overview of a Web Agent Profile" on page 49
- "Creating or Updating a Web Agent Profile" on page 50
- "Updating the Agent Profile Name and the Agent Profile Password in Web Agents" on page 51

Note – If you are interested only in resetting the shared secret in the web agent and not the agent profile name, see "Resetting the Shared Secret Password" on page 73. However, first read the introductory paragraphs that follow in this section to become acquainted with the process and terminology related to the credentials used by web agents to authenticate with Access Manager. A common reason to reset only the shared secret is that it was entered incorrectly when prompted for during the installation of the web agent.

Overview of a Web Agent Profile

A web agent uses a user name and password as credentials to authenticate with Access Manager. You can use the default values for these credentials or you can create an agent profile in Access Manager Console and use those credentials. In web agents, the term for the default user name is agent user name. The default value of the agent user name is `UrlAccessAgent`. The term for the default password is shared secret. The default value of the shared secret is the password of the Access Manager internal LDAP authentication user. This user is commonly referred to as `amldapuser`.

Web agents can function using the default agent profile (`UrlAccessAgent`), but creating a different agent profile in the Access Manager Console provides greater security. You must also create a different agent profile if Access manager is configured for cross domain single sign-on (CDSSO).

The terms used for the credentials are different once you create them in the agent profile. Agent user name is then called agent profile name. Shared secret is then called agent profile password. After you create the agent profile, you must assign the values of the agent profile name and the agent profile password to the correct properties in the web agent `AMAgent.properties` configuration file.

Creating or Updating a Web Agent Profile

The instructions that follow in this section explain how to change both the agent profile name and the agent profile password on the Access Manager side.

Since the agent profile is created and updated in Access Manager Console, tasks related to the agent profile are discussed in Access Manager documentation. Nonetheless, tasks related to the agent profile are also described in this Policy Agent guide, specifically in this chapter. For related information about defining the Policy Agent profile in Access Manager Console, see the following section of the respective document: "Agents Profile" in *Sun Java System Access Manager 7.1 Administration Guide*.

▼ To Create or Update an Agent Profile in Access Manager

Perform the following tasks in Access Manager Console. The key steps of this task involve creating an agent ID (agent profile name) and an agent profile password.

1. **With the Access Control tab selected click the name of the realm for which you would like to create an agent profile.**
2. **Select the Subjects tab.**
3. **Select the Agent tab.**
4. **Click New.**
5. **Enter values for the following fields:**

   ID. Enter the agent profile name or identity of the agent.

   This is the agent profile name, which is the name the agent uses to log into Access Manager. Multi-byte names are not accepted. Do not use the web agent default value of `UrlAccessAgent`.

   Password. Enter the agent profile password.

   Do not use the web agent default value of this password. The web agent default value of this password is the password of the internal LDAP authentication user, commonly referred to as `amldapuser`.

Password (confirm). Confirm the password.

Device Status. Select the device status of the agent. The default status is Active. If set to Active, the agent will be able to authenticate to and communicate with Access Manager. If set to Inactive, the agent will not be able to authenticate to Access Manager.

6 **Click Create.**

The list of agents appears.

7 **(Optional) If you desire, add a description to your newly created agent profile:**

a. **Click the name of your newly created agent profile in the agent list.**

b. **In the Description field, enter a brief description of the agent.**

For example, you can enter the agent instance name or the name of the application it is protecting.

c. **Click Save.**

Updating the Agent Profile Name and the Agent Profile Password in Web Agents

After you have changed the agent profile in Access Manager Console, assign the values for the agent profile name and the agent profile password to the corresponding properties in the web agent `AMAgent.properties` configuration file. This process involves the following:

- Adding the agent profile name to the following property in the web agent `AMAgent.properties` configuration file: `com.sun.am.policy.am.username`
- Encrypting the agent profile password (shared secret) using the encryption utility
- Adding the encrypted agent profile password (shared secret) to the following property in the web agent `AMAgent.properties` configuration file: `com.sun.am.policy.am.password`

The procedures specified in the preceding list are detailed in the platform-specific task descriptions that follow. Implement the steps according to the platform on which the web agent is installed.

▼ To Update the Agent Profile Name and Agent Profile Password

1 **Update the following property in the web agent `AMAgent.properties` configuration file:**

`com.sun.am.policy.am.username`

Replace the value of this property with the agent profile name you just updated in the Access Manager Console.

2 **Go to the following directory:**

PolicyAgent-base/bin

3 **Encrypt the new password. For example, on Solaris systems:**

./crypt_util *agent-profile-password*

where *agent-profile-password* represents the agent profile password you just updated in the Access Manager Console.

Windows systems: Use the cryptit script to encrypt the password.

4 **Copy the output from the** crypt_util **command and paste it as the value for the following property:**

com.sun.am.policy.am.password

5 **Restart the Apache HTTP Server 2.2 web container and try to access a resource protected by the agent.**

If the agent is redirected to Access Manager, this indicates the above steps were executed properly.

◆ ◆ ◆ CHAPTER 6

Post-Installation Tasks for the Apache HTTP Server 2.2 Agent

This chapter describes the "Using SSL With the Apache HTTP Server 2.2 Agent" on page 53 post-installation configuration procedure.

After completing this step, perform the tasks to configure the web agent to your site's specific needs, as described in Chapter 7, "Managing the Apache HTTP Server 2.2 Web Agent."

Using SSL With the Apache HTTP Server 2.2 Agent

During installation, if you choose the HTTPS protocol, theApache HTTP Server 2.2 agent is automatically configured and ready to communicate over Secure Sockets Layer (SSL). Before proceeding with tasks in this section, ensure that the Apache HTTP Server 2.2 instance is configured for SSL.

Caution – You should have an understanding of SSL concepts and the security certificates required to enable communication over the HTTPS protocol. For information. see the Apache HTTP Server 2.2 documentation: `http://httpd.apache.org/docs/2.2/`

▼ To Configure Notifications on the Apache HTTP Server 2.2 Agent for SSL

If Apache HTTP Server 2.2 is running in SSL mode and is receiving notifications, first perform the following broadly defined steps:

1. **Add the Apache HTTP Server 2.2 certificate's root CA certificate to the Access Manager certificate database.**
2. **Mark the CA root certificate as trusted to enable Access Manager to successfully send notifications to Agent for Apache HTTP Server 2.2.**

Default Trust Behavior of Agent for Apache HTTP Server 2.2

This section applies only when Access Manager itself is using SSL. By default, the web agent installed on a remote Apache HTTP Server 2.2 instance trusts any server certificate presented over SSL by the Access Manager host. The web agent does not check the root Certificate Authority (CA) certificate. If the Access Manager host is SSL-enabled and you want the agent to perform certificate checking, adhere to the guidelines as described in the following subsections:

- "Disabling the Default Trust Behavior of Agent for Apache HTTP Server 2.2" on page 54
- "Installing the Access Manager Root CA Certificate for a Remote Apache HTTP Server 2.2 Instance" on page 55

Disabling the Default Trust Behavior of Agent for Apache HTTP Server 2.2

The following property in the web agent `AMAgent.properties` configuration file controls the agent's trust behavior, and by default it is set to `true`:

```
com.sun.am.trust_server_certs
```

▼ To Disable the Default Trust Behavior of Agent for Apache HTTP Server 2.2

With the property `com.sun.am.trust_server_certs` set to `true`, the web agent does not perform certificate checking. Setting this property to false is one of the steps involved in enabling the web agent to perform certificate checking as illustrated in the following task.

1. **Set the following property in the web agent `AMAgent.properties` configuration file to `false` as follows:**

   ```
   com.sun.am.trust_server_certs = false
   ```

2. **Set the directory Cert DB as described in the substeps that follow:**

   a. **Create a directory named `cert`.**

      The best practice is to create this folder in the following directory:

      PolicyAgent-base/AgentInstance-Dir/

      The following is a feasible example of the full path to the `cert` directory:

      ```
      /usr/local/webagents/apache22_agent/Agent_001/cert
      ```

      For more information about the directory structure, see "Inside the Web Agent Base Directory" on page 31.

b. In the web agent `AMAgent.properties` configuration file, set the path to the `cert` directory.

The following example, includes the property, `com.sun.am.sslcert.dir`, and the value:

`com.sun.am.sslcert.dir =` *PolicyAgent-base*`/AgentInstance-Dir/cert`

3 Set the Cert DB Prefix, if required.

In cases where the specified Cert DB directory has multiple certificate databases, the following property must be set to the prefix of the certificate database to be used:

`com.sun.am.certdb.prefix`

Set the property as follows:

`com.sun.am.certdb.prefix = https-`*host.domain.com.host*`-`

4 Save and close the web agent `AMAgent.properties` configuration file.

Installing the Access Manager Root CA Certificate for a Remote Apache HTTP Server 2.2 Instance

The root CA certificate that you install on the remote instance of Apache HTTP Server 2.2 must be the same one that is installed on the Access Manager host.

▼ To Install the Access Manager Root CA Certificate on Apache HTTP Server 2.2

1 Change to the `cert` directory.

The following example illustrates the location of the `cert` directory:

PolicyAgent-base`/AgentInstance-Dir/cert`

The following is a feasible example of the full path to the `cert` directory without the *PolicyAgent-base* placeholder:

```
/usr/local/webagents/apache22_agent/Agent_001/cert
```

2 Set the proper environment by issuing the following command:

`setenv LD_LIBRARY_PATH` *PolicyAgent-base*`/lib:/usr/lib/mps`

3 (Conditional) If you have not already created the necessary certificate database, create that database now by issuing the following command:

PolicyAgent-base`/bin/certutil -N -d .`

The following is a feasible example of how this command might look without the *PolicyAgent-base* placeholder:

```
/usr/local/webagents/apache22_agent/bin/certutil -N -d .
```

For more information about the directory structure, see "Inside the Web Agent Base Directory" on page 31.

4 Install root CA certificate by issuing the following command:

PolicyAgent-base`/bin/certutil -A -n` *cert-name* `-t "C,C,C" -d` *cert-dir* `-i` *cert-file*

cert-name The name for this root CA certificate

cert-dir The directory where the certificate and key stores are located

cert-file The base-64 encoded root CA certificate file

For example, if the Root CA certificate of the Access Manager host is present in the directory *PolicyAgent-base*`/Agent_001/cert` and if the name of this certificate file is `root_ca.crt`, then the following command would be appropriate:

```
PolicyAgent-base/bin/certutil -A -n am_root_ca_cert
-t "C,C,C" -d . -i root_ca.crt
```

The following is a feasible example of how this command might look without the *PolicyAgent-base* placeholder:

```
/usr/local/webagents/apache22_agent/bin/certutil -A -n
am_root_ca_cert -t "C,C,C" -d . -i root_ca.crt
```

5 To verify that the certificate is properly installed, using the command line, issue the following command:

PolicyAgent-base`/bin/certutil -L -d .`

The root CA certificate is then listed in the output of the `certutil -L` command as illustrated in the following code example:

```
Certificate Name                              Trust Attrubutes


cert-name                                      C,C,C

p    Valid peer
P    Trusted peer (implies c)
c    Valid CA
T    Trusted CA to issue client certs (implies c)
C    Trusted CA to certs(only server certs for ssl) (implies c)
u    User cert
w    Send warning
```

6 Restart Apache HTTP Server.

◆ ◆ ◆ CHAPTER 7

Managing the Apache HTTP Server 2.2 Web Agent

The web agent `AMAgent.properties` configuration file is a text file of configuration properties that you can modify to change the web agent behavior. The content of this file, however, is very sensitive, and changes made can result in changes to how the agent works. Errors made can cause the agent to malfunction.

This chapter specifically describes the key features and tasks performed with the web agent `AMAgent.properties` configuration file, including:

For a list and description of every property in the web agent `AMAgent.properties` configuration file, access the configuration file itself. The location of the configuration file is described in Table 2–2. Also a list of the properties is available in this guide, at Appendix C, "Web Agent `AMAgent.properties` Configuration File."

In addition to editing the web agent AMAgent.properties configuration file, you can also perform other functions using the agentadmin command, as explained in "Introduction of the agentadmin Program for Web Agents" on page 23.

Using the Web Agent AMAgent.properties Configuration File

The web agent AMAgent.properties configuration file is available at the following location:

PolicyAgent-base/AgentInstance-Dir/config

For more information about the directory structure, see "Inside the Web Agent Base Directory" on page 31.

Changing the web agent AMAgent.properties configuration file can have serious and far-reaching effects. When you make changes, keep the following in mind:

- Make a backup copy of this file before you make changes.
- Trailing spaces are significant; use them judiciously.
- Use a forward slash (/) to separate directories, not a backslash (\) or double backslashes (\\). This holds true even on Windows systems.
- Spaces in the Windows file names are allowed.

Note – If you make changes to the web agent AMAgent.properties configuration file, restart the deployment container to make your changes take effect.

The web agent AMAgent.properties configuration file includes information for a variety of configurations, including the following:

- debugging
- fully qualified domain name (FQDN) map
- Access Manager services
- service and agent deployment descriptors
- session failover

The configuration file also contains configuration information on advanced features, such as forwarding LDAP user attributes through HTTP headers.

Providing Failover Protection for a Web Agent

When you install a web agent, you can specify a failover or backup deployment container, such as a web server, for running Access Manager. This is essentially a high availability option. It ensures that if the deployment container that runs Access Manager service becomes unavailable, the web agent still processes access requests through a secondary, or failover, deployment container running Access Manager service.

Setting up failover protection for the web agent, requires modifying the web agent `AMAgent.properties` configuration file. However, you must first install two different instances of Access Manager on two separate deployment containers.

Then follow the instructions in this guide to about installing the web agent. The web agent installation program prompts you for the host name and port number of the failover deployment container that you have configured to work with Access Manager. The following property in the web agent `AMAgent.properties` configuration file, stores the failover deployment container name:

```
com.sun.am.policy.am.login.url
```

Set this property in order to store failover server information. Given the values in the following list, the property would be set as shown in Example 7–1.

`host1` Name of the primary Access Manager host.

`host2` Name of the first failoverAccess Manager host.

`host3` Name of the second failoverAccess Manager host.

example Name of the domain.

`58080` Default port number

EXAMPLE 7–1 Configuration Property Setting for Failover Protection of a Web Agent

```
com.sun.am.policy.am.login.url = http://host1.example.com:58080/
amserver/UI/Login http://host2.example.com:58080/amserver/UI/Login
http://host3.example.com:58080/amserver/UI/Login
```

A failover server name is configurable after it has been set during installation. When configuring this property, note that a space is required between each Access Manager login URL.

Changing the Web Agent Caching Behavior

Each web agent maintains a cache that stores the policies for every user's session. The cache can be updated by a cache polling mechanism and a cache notification mechanism.

Cache Updates

A web agent maintains a cache of all active sessions involving content that the agent protects. Once an entry is added to an agent's cache, it remains valid for a period of time after which the entry is considered expired and later purged.

The `com.sun.am.policy.am.polling.interval` property in the web agent `AMAgent.properties` configuration file determines the number of minutes an entry will remain in the web agent cache. Once the interval specified by this property has elapsed, the entry is dropped from the cache. By default, the expiration time is set to three minutes.

In a normal deployment situation, policy changes on the server are frequent, which requires sites to accept a certain amount of latency for web agents to reflect policy changes. Each site decides the amount of latency time that is acceptable for the site's specific needs. When setting the `com.sun.am.policy.am.polling.interval` property, set it to the lower of the following two values:

- Session idle timeout period
- Your site's accepted latency time for policy changes

Note – The Apache HTTP Server 2.2 agent does not support notifications. Therefore, updating the cache through a notification mechanism is not an available feature. However, since the notification mechanism is available for other agents in the Policy Agent 2.2 software set, a property exists in the web agent `AMAgent.properties` configuration file to control this feature. The property that controls the notification mechanism, `com.sun.am.notification.enable`, is set to `false` for this agent. Do not set this property to `true` for this agent as it might result in unexpected behavior.

Configuring the Not-Enforced URL List

The not-enforced URL list defines the resources that should not have any policies (neither allow nor deny) associated with them.

By default, the web agent denies access to all resources on the deployment container that it protects. However, various resources (such as a web site or an application) available through a deployment container might not need to have any policy enforced. Common examples of such

resources include the HTML pages and `.gif` images found in the home pages of web sites and the cascading style sheets (CSS) that apply to these home pages. The user should be able to browse such pages without authenticating. For the home page example, all these resources need to be on the not-enforced URL list or the page will not be displayed properly. The property `com.sun.am.policy.agents.config.notenforced_list` is used for this purpose. Wild cards can be used to define a pattern of URLs. Space is the separator between the URLs mentioned in the list.

There can be a reverse, or "inverted", scenario when all the resources on the deployment container, except a list of URLs, are open to any user. In that case, the property `com.sun.am.policy.agents.config.notenforced_list.invert` would be used to reverse the meaning of `com.sun.am.policy.agents.config.notenforced_list`. If it is set to `true` (by default it is set to `false`), then the not-enforced URL list would become the enforced list.

EXAMPLE 7–2 Configuration Property Settings for Not-Enforced URL List

The following are examples:

Scenario 1: Not-Enforced URL List

```
com.sun.am.policy.agents.config.notenforced_list.invert = false

com.sun.am.policy.agents.config.notenforced_list =
http://host1.example.com:80/welcome.html
http://host1.example.com:80/banner.html
```

In this case, authentication and policies will not be enforced on the two URLs listed in the `notenforcedList`. All other resources will be protected by the web agent.

Scenario 2: Inverted Not-Enforced URL List

```
com.sun.am.policy.agents.config.notenforced_list.invert = true

com.sun.am.policy.agents.config.notenforced_list =
 http://host1.example.com:80/welcome.html
 http://host1.example.com:80/banner.html
```

In this case, authentication and policies will be enforced by the web agent on the two URLs mentioned in the `notenforcedList`. All other resources will be accessible to any user.

Caution – If feasible, keep this property set to `false` as such:

```
com.sun.am.policy.agents.config.notenforced_list.invert = false
```

A value of `false` reduces the chance of unintentionally allowing access to resources.

Configuring the Not-Enforced IP Address List

The `com.sun.am.policy.agents.config.notenforced_client_ip_list` property is used to specify a list of IP addresses. No authentication is required for the requests coming from these client IP addresses.

In other words, the web agent will not enforce policies for the requests originating from the IP addresses in the Not-Enforced IP Address list.

Disabling the Apache HTTP Server 2.2 Agent

You disable a web agent by resetting the `com.sun.am.policy.agents.config.notenforced_list` property, which controls the not-enforced URI list, in the web agent `AMAgent.properties` configuration file.

▼ To Disable the Apache HTTP Server 2.2 Agent

1. **Reset the property to an asterisk (`*`):**

   ```
   com.sun.am.policy.agents.config.notenforced_list = *
   ```

2. **Restart the Apache HTTP Server 2.2 instance.**

Enforcing Authentication Only

The property `com.sun.am.policy.agents.config.do_sso_only` is used to specify if only authentication is enforced for URLs protected by the web agent. If this property is set to `true` (by default it is set to `false`), it indicates that the web agent enforces authentication only, without enforcing policies. After a user logs onto Access Manager successfully, the web agent will not check for policies related to the user and the accessed URLs.

Providing Personalization Capabilities

Web agents can personalize page content for users in three distinct ways, as described in the following subsections:

Providing Personalization With Session Attributes

Web agents support a feature where a user's session attributes are fetched and set as headers or cookies. The following property responsible for this task:

```
com.sun.am.policy.agents.config.session.attribute.fetch.mode
```

This property can be set to one of the following values:

- `NONE`
- `HTTP_HEADER`
- `HTTP_COOKIE`

When set to `NONE`, no session attributes are fetched and the `com.sun.am.policy.agents.config.session.attribute.map` property is ignored. With this property set to either `HTTP_HEADER` or `HTTP_COOKIE`, the web agent fetches session attributes. Use the following property to configure attributes that are to be forwarded as HTTP headers or cookies: `com.sun.am.policy.agents.config.session.attribute.map`.

The following content is from the web agent `AMAgent.properties` configuration file. The text has been reformatted for this section. This section illustrates how the `com.sun.am.policy.agents.config.session.attribute.map` property maps session attributes to headers or cookies.

Session attributes are added to an HTTP header following this format:

```
session_attribute_name|http_header_name[,...]
```

The value of the attribute being fetched in session is `session_attribute_name`. This value gets mapped to a header value as follows: `http_header_name`.

Note – In most cases, in a destination application where `http_header_name` appears as a request header, it is prefixed with `HTTP_` and the following type of conversion takes place:

Lower case letters	convert to upper case letters.
Hyphen "-"	converts to underscore "_"
"`common-name`"	as an example, converts to "`HTTP_COMMON_NAME`."

```
com.sun.am.policy.agents.config.session.attribute.map =
successURL | success-url, contextId | context-id
```

The session attribute is forwarded as a header or a cookie as determined by the end-user applications on the web container that the web agent is protecting. These applications can be considered the consumers of the forwarded header values. The forwarded information is used for the customization and personalization of web pages. You can also write server side plug-ins to put any user session attribute and define the corresponding attribute name and mapping in the preceding property to retrieve the value.

Providing Personalization With Policy-Based Response Attributes

Header attributes can also be determined by Access Manager policy configurations. With policy-based response attributes you can define attribute-value pairs at each policy.

Web agents in this release set policy-based response attributes as headers or cookies based on configuration. All subjects that match this attribute set obtain this attribute.

The following is a new property that has been added to the web agent `AMAgent.properties` configuration file to control this functionality:

```
com.sun.am.policy.agents.config.response.attribute.fetch.mode
```

This property can be set to one of the following values:

- `NONE`
- `HTTP_HEADER`
- `HTTP_COOKIE`

The following example shows this configuration property with the default setting, which is `HTTP_HEADER`:

```
com.sun.am.policy.agents.config.response.attribute.fetch.mode = HTTP_HEADER
```

Attribute mapping is available for response attributes. Therefore, the format of policy information can be mapped to the format of a header or a cookie. The below property is used for this type of mapping:

```
com.sun.am.policy.agents.config.response.attribute.map
```

Unlike profile attributes and session attributes, where only the mapped attributes are displayed as headers or cookies, by default, response attributes are set by the agent as headers or cookies based on the setting of this property:

```
com.sun.am.policy.agents.config.response.attribute.fetch.mode
```

If a response attribute map is specified, then the corresponding attribute mapped name is fetched from the map and its corresponding value is displayed as either a header or a cookie based on the setting of the above property.

Providing Personalization With User Profile Attributes Globally

Web agents have the ability to forward user profile attribute values via HTTP headers to end-web applications. The user profile attribute values come from the server side of Access Manager. The web agent behaves like a broker to obtain and relay user attribute values to the destination servlets, CGI scripts, or ASP pages. These applications can in turn use the attribute values to personalize page content.

This feature is configurable through two properties in the web agent `AMAgent.properties` configuration file. To turn this feature on and off, edit the following property in the web agent `AMAgent.properties` configuration file:

```
com.sun.am.policy.agents.config.profile.attribute.fetch.mode
```

This property can be set to one of the following values:

- `NONE`
- `HTTP_HEADER`
- `HTTP_COOKIE`

When set to `NONE`, the web agent does not fetch LDAP attributes from the server and ignores the `com.sun.am.policy.agents.config.profile.attribute.map` property. In the other two cases, the web agent fetches the attribute.

To configure the attributes that are to be forwarded in the HTTP headers, use the following property:

`com.sun.am.policy.agents.config.profile.attribute.map`

Below is an example section from the web agent `AMAgent.properties` configuration file, which describes how this feature is used:

```
#
# The policy attributes to be added to the HTTP header.  The
# specification is of the format
# ldap_attribute_name|http_header_name[,...]. ldap_attribute_name
# is the attribute in data store to be fetched and
# http_header_name is the name of the header to which the value
# needs to be assigned.
#
# NOTE: In most cases, in a destination application where a
# "http_header_name" shows up as a request header, it will be
# prefixed by HTTP_, and all lower case letters will become upper
# case, and any - will become _; For example, "common-name" would
# become "HTTP_COMMON_NAME"
#
com.sun.am.policy.agents.config.profile.attribute.map = cn|common-name,ou|
organizational-unit,
o|organization,mail|email,employeenumber|employee-number,c|country
```

By default, some LDAP user attribute names and HTTP header names are set to sample values.

To find the appropriate LDAP user attribute names, check the following XML file on the machine where Access Manager is installed:

AccessManager-base`/SUNWam/config/xml/amUser.xml`

The attributes in this file could be either Access Manager user attributes or Access Manager dynamic attributes. For an explanation of these two types of user attributes, see *Sun Java System Access Manager 7.1 Administration Guide*.

The attribute and HTTP header names that need to be forwarded must be determined by the end-user applications on the deployment container that the web agent is protecting. Basically, these applications are the consumers of the forwarded header values (the forwarded information is used for the customization and personalization of web pages).

Setting the Fully Qualified Domain Name

To ensure appropriate user experience, it is necessary that the users access resources protected by the web agent using valid URLs. The configuration property `com.sun.am.policy.agents.config.fqdn.default` provides the necessary information needed by the web agent to identify if the user is using a valid URL to access the protected resource. If the web agent determines that the incoming request does not have a valid hostname in the URL, it redirects the user to the corresponding URL with a valid hostname. The difference between the redirect URL and the URL originally used by the user is only the hostname, which is changed by the web agent to a fully qualified domain name (FQDN) as per the value specified in this property.

This is a required configuration property without which the deployment container may not start up correctly. This property is set during the web agent installation and must not be modified unless absolutely necessary to accommodate deployment requirements. An invalid value for this property can result in the deployment container becoming unusable or the resources becoming inaccessible.

The property `com.sun.am.policy.agents.config.fqdn.map` provides another way by which the web agent can resolve partial or malformed access URLs and take corrective action. The web agent gives precedence to the entries defined in this property over the value defined in the `com.sun.am.policy.agents.config.fqdn.default` property. If none of the entries in this property matches the hostname specified in the user request, the agent uses the value specified for `com.sun.am.policy.agents.config.fqdn.default` property.

The `com.sun.am.policy.agents.config.fqdn.map` property can be used for creating a mapping for more than one hostname. This may be the case when the deployment container protected by this agent is accessible by more than one hostname. However, this feature must be used with caution as it can lead to the deployment container resources becoming inaccessible.

This property can also be used to override the behavior of the web agent in cases where necessary. The format for specifying the property `com.sun.am.policy.agents.config.fqdn.map` is:

```
com.sun.am.policy.agents.config.fqdn.map =
[invalid_hostname|valid_hostname][,...]
```

where:

`invalid_hostname` is a possible invalid hostname such as partial hostname or an IP address that the user may provide .

`valid_hostname` is the corresponding valid hostname that is fully qualified. For example, the following is a possible value specified for hostname `xyz.domain1.com`:

```
com.sun.am.policy.agents.config.fqdn.map = xyz|xyz.domain1.com,
xyz.domain1|xyz.domain1.com
```

This value maps `xyz` and `xyz.domain1` to the FQDN `xyz.domain1.com`.

This property can also be used in such a way that the web agent uses the name specified in this map instead of the deployment container's actual name.

If you want your server to be addressed as *xyz.hostname*`.com` whereas the actual name of the server is *abc.hostname*`.com`. The browser only knows *xyz.hostname*`.com` and you have specified policies using *xyz.hostname*`.com` in the Access Manager Console. In this file, set the mapping as `com.sun.am.policy.agents.config.fqdn.map = valid|`*xyz.hostname*`.com`.

Resetting Cookies

The cookie reset feature enables the web agent to reset some cookies in the browser session while redirecting to Access Manager for authentication.

This feature is configurable through two properties in the web agent `AMAgent.properties` configuration file.

- Enable Cookie Reset

  ```
  com.sun.am.policy.agents.config.cookie.reset.enable = true
  ```

  This property must be set to `true` if this web agent needs to reset cookies in the response while redirecting to Access Manager for authentication. By default, this is set to `false`.

- Cookie List

  This property gives the comma-separated list of cookies that need to be reset in the response while redirecting to Access Manager for authentication. This property is used only if the Cookie Reset feature is enabled.

  Cookie details must be specified in the following format:

  ```
  name[=value][;Domain=value]
  ```

  For example,

  ```
  com.sun.am.policy.agents.config.cookie.reset.list = LtpaToken, cookie1=value1,
  cookie2=value2;Domain=example.com
  ```

Configuring Cross Domain Single Sign-on (CDSSO)

The cross domain single sign-on (CDSSO) feature is configurable through three properties in the web agent `AMAgent.properties` configuration file. To turn this feature on or off, use the following property:

```
com.sun.am.policy.agents.config.cdsso.enable = true
```

By default, this property is set to `false`, and the feature is turned off. To turn on CDSSO, set this property to `true`.

Set the URL where CDC controller is installed by specifying the URL in the following property:

```
com.sun.am.policy.agents.config.cdcservlet.url
```

The following is an example of how this property could be set:

```
com.sun.am.policy.agents.config.cdcservlet.url =
http://host1.eng.example.com:58080/amserver/cdcservlet
```

The third property, `com.sun.am.policy.agents.config.cookie.domain.list` allows you to specify a list of domains in which cookies have to be set in a CDSSO scenario. This property is used only if CDSSO is enabled. If you leave this property blank, then the fully qualified cookie domain for the web agent server will be used for setting the cookie domain. In such a case, it is a host cookie and not a domain cookie.

For more information on CDSSO, see *Sun Java System Access Manager 7.1 Technical Overview*

Setting the REMOTE_USER Server Variable

The property `com.sun.am.policy.am.userid.param` allows you to configure the user ID parameter passed by the session or user profile information from Access Manager. The user ID value is used by the agent to set the value of the `REMOTE_USER` server variable. By default, this parameter is set to `UserToken` and is fetched from session attributes.

It can be set to any other session attribute. Another property determines where to retrieve the value, from user profiles or from session properties.

Example 1: This example demonstrates how to set the user ID parameter with session attributes:

`com.sun.am.policy.am.userid.param.type=SESSION` (this is default)

`com.sun.am.policy.am.userid.param=UserToken` (UserId, Principal, or any other session attribute)

Example 2: This example demonstrates how to set the user ID parameter with LDAP user profile attributes:

`com.sun.am.policy.am.userid.param.type=LDAP`

`com.sun.am.policy.am.userid.param=cn` (any profile attribute)

Setting the Anonymous User

For resources on the not-enforced list, the default configuration does not allow the `REMOTE_USER` variable to be set. To enable the `REMOTE_USER` variable to be set for not-enforced URLs, you must set the following property in the web agent `AMAgent.properties` configuration file to `TRUE` (by default the value is `FALSE`):

```
com.sun.am.policy.agents.config.anonymous_user.enable = TRUE
```

When you set the value of this property to `TRUE`, the value of `REMOTE_USER` will be set to the value contained in the following property in the web agent `AMAgent.properties` configuration file:

```
com.sun.am.policy.agents.config.anonymous_user
```

By default, the value of this property is set to `anonymous` as follows:

```
com.sun.am.policy.agents.config.anonymous_user = anonymous
```

Validating Client IP Addresses

This feature can be used to enhance security by preventing the stealing or *hijacking* of SSO tokens.

The web agent `AMAgent.properties` configuration file contains a property titled `com.sun.am.policy.agents.config.client_ip_validation.enable`, which by default, is set to `false`.

If you set this property value to `true`, client IP address validation will be enabled for each incoming request that contains an SSO token. If the IP address from which the request was generated does not match the IP address issued for the SSO token, the request will be denied. This is essentially the same as enforcing a deny policy.

This feature should not be used, however, if the client browser uses a web proxy or if there is a load balancer somewhere between the client browser and the agent-protected deployment container. In such cases, the IP address appearing in the request will not reflect the real IP address on which the client browser runs.

Resetting the Shared Secret Password

This section describes how to reset the shared secret. The web agent stores the shared secret in the web agent `AMAgent.properties` configuration file.

If you are only interested in resetting the shared secret, not the agent profile name, continue reading this section. If you are interested in creating or updating the agent profile in Access Manager Console and then updating the same credential information in the web agent, see Chapter 5, "Relationship Between the Agent Profile and Web Agents." The steps described in that chapter are comprehensive, integrating the simpler steps described in this section.

The chapter mentioned in the preceding paragraph also provides a useful explanation of the process and terminology related to the credentials used by web agents to authenticate with Access Manager. Refer to that chapter for more information.

This section specifically describes how to change the shared secret in web agents. The following situations might require you to reset the shared secret:

- You entered the shared secret incorrectly during web agent installation.
- You have been using the default shared secret, which is the `amldapuser` password, but this password has since been changed.

The value for the property `com.sun.am.policy.am.password` in the web agent `AMAgent.properties` configuration file is set with the encrypted shared secret during web agent installation. Therefore, if the shared secret is entered incorrectly during installation, the preceding property is assigned an incorrect value, preventing the web agent from authenticating with Access Manager.

To reset or change the shared secret, use the encryption utility to encrypt the shared secret and then set the value in the property as described in the following platform-specific tasks (follow the steps according to the platform on which the agent is installed).

▼ To Reset the Shared Secret on Solaris Systems

1. **Go to the following directory:**

   PolicyAgent-base`/bin`

2. **Execute the following script in the command line:**

   `# ./crypt_util` *shared-secret*

where *shared-secret* represents the password, that along with the agent user name, allows the web agent to authenticate with Access Manager. The default value of the shared secret is the password of the Access Manager internal LDAP authentication user. This user is commonly referred to as `amldapuser`.

3 **Copy the output obtained after issuing the `# ./crypt_util` *shared-secret* command and paste it as the value for the following property:**

`com.sun.am.policy.am.password`

4 **Restart the deployment container and try accessing any resource protected by the agent.**

If the agent gets redirected to Access Manager, this indicates the above steps were executed properly.

▼ To Reset the Shared Secret on Windows Systems

1 **Go to the following directory:**

PolicyAgent-base`\\bin`

2 **Execute the following script in the command line**

`cryptit` *shared-secret*

where *shared-secret* represents the password, that along with the agent user name, allows the web agent to authenticate with Access Manager. The default value of the shared secret is the password of the Access Manager internal LDAP authentication user. This user is commonly referred to as `amldapuser`.

3 **Copy the output obtained after issuing the `cryptit` *shared-secret* command and paste it as the value for the following property:**

`com.sun.am.policy.am.password`

4 **Restart the deployment container and try accessing any resource protected by the agent.**

If the agent gets redirected to Access Manager, this indicates the above steps were executed properly.

▼ To Reset the Shared Secret on Linux Systems

1 **Go to the following directory:**

PolicyAgent-base`/bin`

2 **Execute the following script in the command line:**

`crypt_util` *shared-secret*

where *shared-secret* represents the password, that along with the agent user name, allows the web agent to authenticate with Access Manager. The default value of the shared secret is the password of the Access Manager internal LDAP authentication user. This user is commonly referred to as `amldapuser`.

3 **Copy the output obtained after issuing the `crypt_util` *shared-secret* command and paste it as the value for the following property:**

`com.sun.am.policy.am.password`

4 **Restart the deployment container and try accessing any resource protected by the agent.**

If the agent gets redirected to Access Manager, this indicates the above steps were executed properly.

Enabling Load Balancing

Various properties in the web agent `AMAgent.properties` configuration file can be used to enable load balancing. Edit the properties that apply, according to the location of the load balancer or load balancers in your deployment, as follows:

- "Load Balancer in Front of Access Manager" on page 75
- "Load Balancer in Front of the Web Agent" on page 76
- "Load Balancers in Front of Both the Web Agent and Access Manager" on page 77

Load Balancer in Front of Access Manager

When a load balancer is deployed in front of Access Manager and a web agent interacts with the load balancer, the following properties must be edited:

```
com.sun.am.naming.url
com.sun.am.policy.am.login.url
com.sun.am.load_balancer.enable
```

EXAMPLE 7–3 Property Settings: Load Balancer in Front of Access Manager

This example illustrates property settings in the web agent `AMAgent.properties` configuration file that can be used to enable load balancing:

EXAMPLE 7–3 Property Settings: Load Balancer in Front of Access Manager *(Continued)*

```
com.sun.am.naming.url = LB-url/amserver/namingservice
com.sun.am.policy.am.login.url = LB-url/amserver/UI/Login
com.sun.am.load_balancer.enable = true
```

where *LB-url* represents the load balancer URL. The following example is a conceivable load balancer URL:

```
http://hostname.example.com:8080
```

Load Balancer in Front of the Web Agent

In many cases, when a load balancer is deployed in front of the web agent only the following property must be set:

```
com.sun.am.policy.agents.config.fqdn.map
```

EXAMPLE 7–4 Property Settings: Load Balancer in Front of the Web Agent

```
com.sun.am.policy.agents.config.fqdn.map = valid|LB-hostname
```

where *LB-hostname* represents the name of the machine on which the load balancer is located.

However, if SSL-termination or a proxy server is used in the deployment, all the following properties in the web agent `AMAgent.properties` configuration file should be set in addition to the preceding property:

```
com.sun.am.policy.agents.config.override_protocol
com.sun.am.policy.agents.config.override_host
com.sun.am.policy.agents.config.override_port
com.sun.am.policy.agents.config.agenturi.prefix
```

This example illustrates how properties can be set to enable load balancing when the protocol, hostname, and port number of the load balancer differ from that of the web agent. However, if the load balancer and the web agent share one of these characteristics, such as the protocol or hostname, then the respective property would be left blank instead of being assigned a value of *true*.

```
com.sun.am.policy.agents.config.override_protocol = true
com.sun.am.policy.agents.config.override_host = true
com.sun.am.policy.agents.config.override_port = true
com.sun.am.policy.agents.config.agenturi.prefix = LB-url/amagent
```

where *LB-url* represents the load balancer URL. The following example is a conceivable load balancer URL:

```
http://hostname.example.com:8080
```

Load Balancers in Front of Both the Web Agent and Access Manager

This scenario is simply a combination of the scenarios described in the preceding sections. See "Load Balancer in Front of Access Manager" on page 75 and "Load Balancer in Front of the Web Agent" on page 76.

◆ ◆ ◆ CHAPTER 8

Uninstalling the Apache HTTP Server 2.2 Policy Agent

This chapter describes how to uninstall the Apache HTTP Server 2.2 agent using the `agentadmin` program.

Uninstalling the Apache HTTP Server 2.2 Agent

The uninstallation program (`agentadmin`) performs the following functions:

- Unconfigures the agent in the Apache HTTP Server 2.2 `httpd.conf` file
- Removes the Apache HTTP Server 2.2 agent configuration directory

▼ To Uninstall the Apache HTTP Server 2.2 Agent

1 **Change to the *PolicyAgent-base*`/bin` directory.**

PolicyAgent-base`/bin`

This directory contains the `agentadmin` program, which is used for uninstalling an agent (as well as for performing other tasks). For more information about the `agentadmin` program, see "Introduction of the `agentadmin` Program for Web Agents" on page 23.

2 **Issue the following command:**

```
./agentadmin --uninstall
```

3 **The `uninstall` program requests the path to the Apache HTTP Server 2.2 configuration directory.**

Either accept the displayed value or enter a different path.

4 **The `uninstall` program then asks if you want to continue with the uninstallation.**

To continue, select 1 (the default).

Example 8–1 Uninstallation Sample Run for the Apache HTTP Server 2.2 Agent

```
************************************************************************
Welcome to the Access Manager Policy Agent for Apache Server If the Policy
Agent is used with Federation Manager services, User needs to enter
information relevant to Federation Manager.
************************************************************************


Enter the complete path to the directory which is used by Apache Server to
store its configuration Files. This directory uniquely identifies the
Apache Server instance that is secured by this Agent.
[ ? : Help, ! : Exit ]
Enter the Apache Server Config Directory Path [/opt/apache/conf]:
/opt/apache2/conf


---------------------------------------------
SUMMARY OF YOUR RESPONSES
---------------------------------------------
Apache Server Config Directory : /opt/apache2/conf

Verify your settings above and decide from the choices below.
1. Continue with Uninstallation
2. Back to the last interaction
3. Start Over
4. Exit
Please make your selection [1]:
```

Next Steps To completely remove the Apache HTTP Server 2.2 agent installation files, go to the `web_agents` directory and remove the `apache_agent` directory (and subdirectories).

◆ ◆ ◆ APPENDIX A

Silent Installation and Uninstallation of a Web Agent in Policy Agent 2.2

This appendix describes how to use the silent option for the installation and uninstallation of web agents that were developed as part of the OpenSSO project, including:

About the Silent Installation and Uninstallation of a Web Agent

A silent installation or uninstallation refers to installing or uninstalling a program by implementing a script. The script is part of a state file. The script provides all the answers that you would normally supply to the installation or uninstallation program interactively. Running the script saves time and is useful when you want to install or uninstall multiple instances of a policy agent using the same parameters in each instance.

Silent installation is a simple two-step process of generating a state file and then using that state file. To generate a state file, you record the installation or uninstallation process, entering all the required information that you would enter during a standard installation or uninstallation. Then you run the installation or uninstallation program with the state file as the input source.

Preforming a Silent Installation of a Web Agent

Generating a State File for a Silent Web Agent Installation

This section describes how to generate a state file for installing a web agent. This task requires you to issue a command that records the information you will enter as you follow the agent installation steps. Enter all the necessary installation information in order to create a complete state file.

▼ To Generate a State File for a Silent Web Agent Installation

To generate a state file for a silent web agent installation , perform the following:

1 **Change to the following directory:**

PolicyAgent-base`/bin`

This directory contains the `agentadmin` program, which is used for installing a web agent and for performing other tasks. For more information on the `agentadmin` program, see "Introduction of the `agentadmin` Program for Web Agents" on page 23.

2 **Issue the following command:**

```
./agentadmin --install --saveResponse filename
```

`-saveResponse`	An option that saves all of your responses to installation prompts in a state file.
filename	Represents the name that you choose for the state file.

3 **Perform the installation as described in Chapter 4, "Installing the Apache HTTP Server 2.2 Policy Agent"**

Your answers to the prompts are recorded in the state file. When the installation is complete, the state file is created in the same directory where the installation program is located.

Note – When generated, a state file will have `read` permissions for all users. However, because the state file contains clear text passwords, it is recommended that you change the file permissions to restrict `read` and `write` access to the user `root`.

Using a State File for a Silent Web Agent Installation

The installation program does not validate inputs or the state in the silent installation. Ensure that the proper environment exists before performing a silent installation.

▼ To Install a Web Agent Using a State File

To perform a silent installation of a web agent using a state file, perform the following:

1 Change to the following directory:

PolicyAgent-base/bin

At this point, this bin directory should contain the agentadmin program and the web agent installation state file.

2 Issue the following command:

```
./agentadmin --install --useResponse filename
```

-useResponse	An option that directs the installer to run in non-interactive mode as it obtains all responses to prompts from the named state file.
filename	Represents the name of the state file from which the installer obtains all responses.

The installation takes place hidden from view. After completion, the program exits automatically and displays the prompt.

Performing a Silent Uninstallation of a Web Agent

Generating a State File for a Web Agent Silent Uninstallation

This section describes how to generate a state file for uninstalling a web agent. This task requires you to issue a command that records the information you will enter as you follow the agent uninstallation steps. Enter all the necessary uninstallation information in order to create a complete state file.

▼ To Generate a State File for a Web Agent Silent Uninstallation

To generate a state file for uninstallation of a web agent, perform the following:

1 Change to the following directory:

PolicyAgent-base/bin

This directory contains the `agentadmin` program, which is used for uninstalling a web agent and for performing other tasks. For more information on the `agentadmin` program, see "Introduction of the `agentadmin` Program for Web Agents" on page 23.

2 Issue the following command:

```
./agentadmin --uninstall --saveResponse filename
```

`-saveResponse`	An option that saves all of your responses to uninstallation prompts in a state file.
filename	Represents the name that you choose for the state file.

3 Perform the uninstallation as explained in Chapter 8, "Uninstalling the Apache HTTP Server 2.2 Policy Agent."

Your answers to the prompts are recorded in the state file. When uninstallation is complete, the state file is created in the same directory where the uninstallation program is located.

Note – When generated, a state file will have `read` permissions for all users. However, because the state file contains clear text passwords, it is recommended that you change the file permissions to restrict `read` and `write`access to the user `root`.

Using a State File for a Web Agent Silent Uninstallation

The uninstallation program does not validate inputs or the state in the silent installation. Ensure that the proper environment exists before performing a silent uninstallation.

▼ To Uninstall a Web Agent Using a State File

To perform a silent uninstallation of a Web agent using a state file, perform the following:

1 Change to the following directory:

PolicyAgent-base`/bin`

At this point, this `bin` directory should contain the `agentadmin` program and the Web uninstallation state file.

2 Issue the following command:

```
./agentadmin --uninstall --useResponse filename
```

`-useResponse`	An option that runs the uninstallation process in non-interactive mode as all responses to prompts are obtained from the named state file.
filename	Represents the name of the state file from which the installer obtains all responses.

The uninstallation takes place hidden from view. After completion, the program exits automatically and displays the prompt.

◆ ◆ ◆ APPENDIX B

Troubleshooting the Apache HTTP Server 2.2 Policy Agent

Many of the following symptoms and solutions apply to all web agents, but some solutions are adapted specifically to the Apache HTTP Server 2.2 agent:

To determine if a problem is a known limitation of the web agent, check the *Sun Java System Access Manager Policy Agent 2.2 Release Notes*. For some limitations, a workaround might also be available.

On UNIX, retrieving and encrypting information from the password file returns an error

Symptom: On UNIX-based systems, during the installation process, retrieving and encrypting information from the password file results in an error such as the following:

```
Reading data from file path-of-password-file
 and encrypting it ... ***ERROR: Installation failed due to the
following error - (Invalid empty password specified.).
```

Where *path-of-password-file* is a placeholder representing the path to a file from which the system is attempting to retrieve the password.

Possible Causes: The `crypt_util` program does not have executable permissions. Ensuring that this program has executable permissions is a step that should be performed prior to installation.

Possible Solution:

1. Add executable permissions to the `crypt_util` program as described in "To Prepare to Install the Apache HTTP Server 2.2 Agent" on page 40.
2. Remove the `Agent_00x` directory, presumably `Agent_001`.
3. Install the agent.

Browser loops before displaying an access-denied page

Symptom: The browser goes into a loop for approximately a minute before displaying an access-denied page.

Possible Cause: The user tries to access a resource for which a policy with a time condition has been set and the time on the web agent host and the Access Manager host are not in sync.

Possible Solution: Login as `root` and run the command `rdate` *hostname* to synchronize the time on both the hosts.

Using Internet Explorer, access is denied when you access a resource

Symptom: When a user attempts to access a resource using Internet Explorer as the browser, access is denied.

Possible Cause: Internet Explorer overrides the port number of the web agent with the Access Manager port number. In such cases, the agent log file lists the URL that is being evaluated. The port number for that URL is incorrect.

Possible Solution: You can ensure this problem does not occur by setting the following property in the web agent `AMAgent.properties` configuration file to `true` as shown:

```
com.sun.am.policy.agents.config.override_port = true
```

◆ ◆ ◆ APPENDIX C

Web Agent `AMAgent.properties` Configuration File

The web agent `AMAgent.properties` configuration file contains the necessary configuration properties needed for the web agent to function properly. It also contains the necessary information needed for the Sun Java System Access Manager SDK to function properly in a client installation mode as used by the web agent.

Properties in the Web Agent `AMAgent.properties` Configuration File

The web agent `AMAgent.properties` configuration file is located as described in Table 2–2. For a more detailed discussion of the key tasks you can perform using this configuration file, see Chapter 7, "Managing the Apache HTTP Server 2.2 Web Agent."

For detailed information about every property, see the actual web agent `AMAgent.properties` configuration file in the product itself for a description of each property.

Most property names in the web agent `AMAgent.properties` configuration file have changed for Policy Agent 2.2. The following list highlights the change in property names by presenting the current property name in the release paired with the former property name from the 2.1 release. You can use this information to map the former property name to the current property name. Most properties apply to all web agents in the 2.2 release. A few properties are specific to one or a few web agents.

TABLE C–1 Changes in the Web Agent `AMAgent.properties` Configuration File for Policy Agent 2.2

Version 2.2 Property Name	Former Property Name: Version 2.1 and Prior
`com.sun.am.cookie.name`	`com.sun.am.cookieName`
`com.sun.am.cookie.encode`	`com.sun.am.cookieEncoded`

TABLE C–1 Changes in the Web Agent AMAgent.properties Configuration File for Policy Agent 2.2 *(Continued)*

Version 2.2 Property Name	Former Property Name: Version 2.1 and Prior
com.sun.am.log.level	com.sun.am.logLevels
com.sun.am.naming.url	com.sun.am.namingURL
com.sun.am.sslcert.dir	com.sun.am.sslCertDir
com.sun.am.certdb.prefix	com.sun.am.certDbPrefix
com.sun.am.trust_server_certs	com.sun.am.trustServerCerts
com.sun.am.notification.enable	com.sun.am.notificationEnabled
com.sun.am.notification.url	com.sun.am.notificationURL
com.sun.am.load_balancer.enable	com.sun.am.loadBalancer_enable
com.sun.am.policy.am.login.url	com.sun.am.policy.am.loginURL
com.sun.am.policy.am.username (unchanged)	com.sun.am.policy.am.username
com.sun.am.policy.am.password (unchanged)	com.sun.am.policy.am.password
com.sun.am.policy.am.url_comparison.case_ignore	com.sun.am.policy.am.urlComparison.caseIgnore
com.sun.am.policy.am.polling.interval	com.sun.am.policy.am.cacheEntryLifeTime
com.sun.am.policy.am.userid.param	com.sun.am.policy.am.userIdParam
com.sun.am.policy.am.lb.cookie.name	com.sun.am.policy.am.ias_SLB_cookie_name
com.sun.am.policy.am.fetch_from_root_resource	com.sun.am.policy.am.fetchFromRootResource
com.sun.am.policy.agents.config.local.log.file	com.sun.am.logFile
com.sun.am.policy.agents.config.local.log.rotate	Not applicable: New property for version 2.2
com.sun.am.policy.agents.config.local.log.size	Not applicable: New property for version 2.2
com.sun.am.policy.agents.config.remote.log	com.sun.am.serverLogFile
com.sun.am.policy.agents.config.profile.attribute.fetch.mode	com.sun.am.policy.am.ldapattribute.mode
com.sun.am.policy.agents.config.profile.attribute.map	com.sun.am.policy.am.headerAttributes

TABLE C–1 Changes in the Web Agent AMAgent.properties Configuration File for Policy Agent 2.2 *(Continued)*

Version 2.2 Property Name	Former Property Name: Version 2.1 and Prior
com.sun.am.policy.agents.config. profile.attribute.cookie.prefix	com.sun.am.policy.am.ldapattribute. cookiePrefix
com.sun.am.policy.agents.config. profile.attribute.cookie.maxage	com.sun.am.policy.am.ldapattribute. cookieMaxAge
com.sun.am.policy.agents.config. session.attribute.fetch.mode	Not applicable: New property for version 2.2
com.sun.am.policy.agents.config. session.attribute.map	Not applicable: New property for version 2.2
com.sun.am.policy.agents.config. response.attribute.fetch.mode	Not applicable: New property for version 2.2
com.sun.am.policy.agents.config. add_response_attrs	Not applicable: New property for version 2.2
com.sun.am.policy.agents.config.version	com.sun.am.policy.agents.version
com.sun.am.policy.agents.config. audit.accesstype	com.sun.am.policy.agents.logAccessType
com.sun.am.policy.agents.config. agenturi.prefix	com.sun.am.policy.agents.agenturiprefix
com.sun.am.policy.agents.config.locale	com.sun.am.policy.agents.locale
com.sun.am.policy.agents.config. instance.name	com.sun.am.policy.agents.instanceName
com.sun.am.policy.agents.config. do_sso_only	com.sun.am.policy.agents.do_sso_only
com.sun.am.policy.agents.config. accessdenied.url	com.sun.am.policy.agents.accessDeniedURL
com.sun.am.policy.agents.config. url.redirect.param	com.sun.am.policy.agents.urlRedirectParam
com.sun.am.policy.agents.config. fqdn.default	com.sun.am.policy.agents.fqdnDefault
com.sun.am.policy.agents.config.fqdn.map	com.sun.am.policy.agents.fqdnMap
com.sun.am.policy.agents.config. cookie.reset.enable	com.sun.am.policy.agents. cookie_reset_enabled
com.sun.am.policy.agents.config. cookie.reset.list	com.sun.am.policy.agents.cookie_reset_list

TABLE C–1 Changes in the Web Agent AMAgent.properties Configuration File for Policy Agent 2.2 *(Continued)*

Version 2.2 Property Name	Former Property Name: Version 2.1 and Prior
com.sun.am.policy.agents.config. cookie.domain.list	com.sun.am.policy.agents.cookieDomainList
com.sun.am.policy.agents.config. anonymous_user	com.sun.am.policy.agents. unauthenticatedUser
com.sun.am.policy.agents.config. anonymous_user.enable	com.sun.am.policy.agents. anonRemoteUserEnabled
com.sun.am.policy.agents.config. notenforced_list	com.sun.am.policy.agents. notenforcedList
com.sun.am.policy.agents.config. notenforced_list.invert	com.sun.am.policy.agents. reverse_the_meaning_of_notenforcedList
com.sun.am.policy.agents.config. notenforced_client_ip_list	com.sun.am.policy.agents. notenforced_client_IP_address_list
com.sun.am.policy.agents.config. postdata.preserve.enable	com.sun.am.policy.agents. is_postdatapreserve_enabled
com.sun.am.policy.agents.config. postcache.entry.lifetime	com.sun.am.policy.agents. postcacheentrylifetime
com.sun.am.policy.agents.config. cdsso.enable	com.sun.am.policy.agents.cdsso-enabled
com.sun.am.policy.agents.config. cdcservlet.url	com.sun.am.policy.agents.cdcservletURL
com.sun.am.policy.agents.config. client_ip_validation.enable	com.sun.am.policy.agents. client_ip_validation_enable
com.sun.am.policy.agents.config. logout.url	com.sun.am.policy.agents.logout.url
com.sun.am.policy.agents.config. logout.cookie.reset.list	com.sun.am.policy.agents.logout. cookie_reset_list
com.sun.am.policy.agents.config. get_client_host_name	com.sun.am.policy.agents.getClientHostname
com.sun.am.policy.agents.config. convert_mbyte.enable	com.sun.am.policy.agents. convertMbyteEnabled
com.sun.am.policy.agents.config. ignore_path_info	com.sun.am.ignore_path_info
com.sun.am.policy.agents.config. override_protocol	com.sun.am.policy.agents.overrideProtocol

TABLE C–1 Changes in the Web Agent AMAgent.properties Configuration File for Policy Agent 2.2 *(Continued)*

Version 2.2 Property Name	Former Property Name: Version 2.1 and Prior
com.sun.am.policy.agents.config.override_host	com.sun.am.policy.agents.overrideHost
com.sun.am.policy.agents.config.override_port	com.sun.am.policy.agents.overridePort
com.sun.am.policy.agents.config.override_notification.url	com.sun.policy.agents.overrideNotificationUrl
com.sun.am.policy.agents.config.connection_timeout	Not applicable: New property for version 2.2
com.sun.am.policy.agents.config.iis6.basicAuthentication.username	Not applicable: New property for version 2.2
com.sun.am.policy.agents.config.iis6.basicAuthentication.password	Not applicable: New property for version 2.2
com.sun.am.policy.agents.config.iis6.basicAuthentication.logFile	Not applicable: New property for version 2.2

◆ ◆ ◆ APPENDIX D

Error Codes

This appendix lists the error codes you might encounter while installing and configuring a web agent. It also provides explanations for the each code item.

Error Code List

This list of error codes includes locations that are reserved for error codes that do not currently exist.

0. AM_SUCCESS	The operation completed successfully.
1. AM_FAILURE	The operation did not complete successfully. Please refer to the log file for more details.
2. AM_INIT_FAILURE	The C SDK initialization routine did not complete successfully. All the other APIs may be used only if the initialization went through successfully.
3. AM_AUTH_FAILURE	The authentication did not go through successfully. This error is returned either by the Authentication API or the Policy Initialization API, which tries to authenticate itself as a client to Access Manager.
4. AM_NAMING_FAILURE	The naming query failed. Please look at the log file for further information.
5. AM_SESSION_FAILURE	The session operation did not succeed. The operation may be any of the operations provided by the session API.
6. AM_POLICY_FAILURE	The policy operation failed. Details of policy failure may be found in the log file.

7. This is a reserved error code.	Currently, no error code exists at this location.
8. AM_INVALID_ARGUMENT	The API was invoked with one or more invalid parameters. Check the input provided to the function.
9. This is a reserved error code.	Currently, no error code exists at this location.
10. This is a reserved error code.	Currently, no error code exists at this location.
11. AM_NO_MEMORY	The operation failed because of a memory allocation problem.
12. AM_NSPR_ERROR	The underlying NSPR layer failed. Please check log for further details.
13. This is a reserved error code.	Currently, no error code exists at this location.
14. AM_BUFFER_TOO_SMALL	The web agent does not have memory allocated to receive data from Access Manager.
15. AM_NO_SUCH_SERVICE_TYPE	The service type input by the user does not exist. This is a more specific version of `AM_INVALID_ARGUMENT` error code. The error can occur in any of the API that take `am_policy_t` as a parameter.
16. AM_SERVICE_NOT_AVAILABLE	Currently, no error code exists at this location.
17. AM_ERROR_PARSING_XML	During communication with Access Manager, there was an error while parsing the incoming XML data.
18. AM_INVALID_SESSION	The session token provided to the API was invalid. The session may have timed out or the token is corrupted.
19. AM_INVALID_ACTION_TYPE	This exception occurs during policy evaluation, if such an action type does not exist for a given policy decision appropriately found for the resource.
20. AM_ACCESS_DENIED	The user is denied access to the resource for the kind of action requested.
21. AM_HTTP_ERROR	There was an HTTP protocol error while contacting Access Manager.
22. AM_INVALID_FQDN_ACCESS	The resource provided by the user is not a fully qualified domain name. This is a web container

	specific error and may be returned by the `am_web_is_access_allowed` function only.
23. AM_FEATURE_UNSUPPORTED	The feature being invoked is not implemented as of now. Only the interfaces have been defined.
24. AM_AUTH_CTX_INIT_FAILURE	The Auth context creation failed. This error is thrown by `am_auth_create_auth_context`.
25. AM_SERVICE_NOT_INITIALIZED	The service is not initialized. This error is thrown by `am_policy` functions if the provided service was not initialized previously using `am_policy_service_init`.
26. AM_INVALID_RESOURCE_FORMAT	This is a plug-in interface error. Implementors of the new resource format may throw this error if the input string does not meet their specified format. This error is thrown by the `am_web` layer, if the resource passed as parameter does not follow the standard URL format.
27. AM_NOTIF_NOT_ENABLED	This error is thrown if the notification registration API is invoked when the notification feature is disabled in the configuration file.
28. AM_ERROR_DISPATCH_LISTENER	Error during notification registration.
29. AM_REMOTE_LOG_FAILURE	This error code indicates that the service that logs messages to Access Manager has failed. The details of this error can be found in the web agent's log file.

Index

R

S

T

U

V

W