OpenAM Policy Agent 3.1.0 Installation Guide

Mark Craig
Vanessa Richie

Software release date: February 20, 2013

Publication date: February 21, 2013

Abstract

Guide to installing OpenAM policy agents. OpenAM provides open source Authentication, Authorization, Entitlement and Federation software.

Table of Contents

Preface

This guide shows you how to install OpenAM web server and Java EE policy agents, as well as how to integrate with other access management software. Unless you are planning a throwaway evaluation or test installation, read the *Release Notes* before you get started.

1. Who Should Use this Guide

This guide is written for anyone installing OpenAM policy agents to interface with supported web servers and Java EE application containers.

This guide covers procedures that you theoretically perform only once per version. This guide aims to provide you with at least some idea of what happens behind the scenes when you perform the steps.

You do not need to be an OpenAM wizard to learn something from this guide, though a background in access management and maintaining web application software can help. You do need some background in managing services on your operating systems and in your application servers. You can nevertheless get started with this guide, and then learn more as you go along.

2. Formatting Conventions

Some items are formatted differently from other text, like `filenames`, **commands**, and `literal values`.

```
$ echo Command line sessions are formatted with lines folded for easier reading.
 In HTML documents click the [-] image for a flat, copy-paste version. Click
 the [+] image for an expanded, line-wrapped version. > /dev/null
```

In many cases, sections pertaining to UNIX, GNU/Linux, Mac OS X, BSD, and so forth are marked (UNIX). Sections pertaining to Microsoft Windows might be marked (Windows). To avoid repetition, however, file system directory names are often given only in UNIX format as in `/path/to/OpenAM`, even if the text applies to `C:\path\to\OpenAM` as well.

Absolute path names usually begin with the placeholder `/path/to/`, which might translate to `/opt/`, `C:\Program Files\`, or somewhere else on your system. Unless you install from native packages, you create this location before you install.

```
class Test
{
    public static void main(String [] args)
    {
        System.out.println("This is a program listing.");
    }
}
```

3. Accessing OpenAM Documentation Online

Core documentation, such as what you are now reading, aims to be technically accurate and complete with respect to the software documented. Core documentation therefore follows a three-phase review process designed to eliminate errors. The review process should slow authors down enough that documentation you get with a stable release has had time to bake fully.

Fully baked core documentation is available at docs.forgerock.org.

The OpenAM Wiki regularly brings you more, fresh content. In addition, you are welcome to sign up and then edit the Wiki if you notice an error, or if you have something to share.

4. Joining the OpenAM Community

After you sign up at ForgeRock, you can also login to the Wiki and the issue database to follow what is happening with the project.

If you have questions regarding OpenAM which are not answered by the documentation, there is a mailing list which can be found at https://lists.forgerock.org/mailman/listinfo/openam where you are likely to find an answer. You can also make suggestions regarding updates at the documentation mailing list (https://lists.forgerock.org/mailman/listinfo/docs).

The Wiki has information on how to check out OpenAM source code. There is also a mailing list for OpenAM development that can be found at https://lists.forgerock.org/mailman/listinfo/openam-dev. Should you want to contribute a patch, test, or feature, or want to author part of the core documentation, first have a look on the ForgeRock site at how to get involved.

Chapter 1. About OpenAM Web Policy Agents

OpenAM web policy agents provide light touch integration for web applications running on supported web servers. This chapter covers what web policy agents do and how they work.

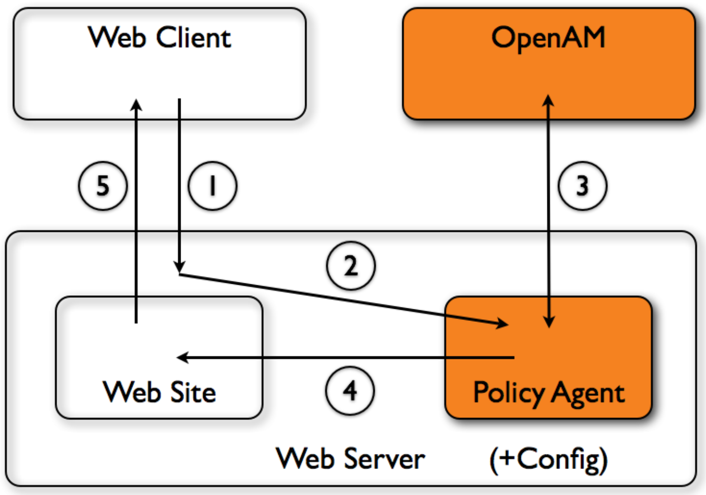
A *policy agent* enforces policy for OpenAM. A *web policy agent* installed in a web server intercepts requests from users trying to access a protected web resource, and denies access until the user has authorization from OpenAM to access the resource.

1.1. How the User, Web Policy Agent, & OpenAM Interact

Imagine that a user attempts to access a protected resource before having authenticated by pointing her browser to a web page. Assume that you have configured OpenAM to protect the web page. Then the web policy agent intercepting her browser's request finds no session token in the request, and so redirects the user's browser to the OpenAM login page for authentication. After the user has successfully authenticated, OpenAM sets a session token in a browser cookie, and redirects her browser back to the page she tried to access initially.

When the user's browser reiterates the request, the policy agent again checks that the request has a session token, finds a session token this time, and validates the session token with OpenAM. Given the valid session token, the policy agent gets a policy decision from OpenAM concerning whether the user can access the page. If OpenAM's Policy Service determines that the user is allowed to access the page, OpenAM responds to the policy agent that access should be granted. The web policy agent then permits the web page to be returned to the user's browser.

The following diagram shows how the pieces fit together when a web client accesses a web page protected by a policy agent. This diagram is simplified to show only the essential principals rather than to describe every possible case.



A web policy agent is a library installed in the web server and configured to be called by the web server when a client requests access to a protected resource in a web site.

1. The web client requests access to a protected resource.
2. The web server runs the request through the policy agent that protects the resource according to OpenAM policy. The policy agent acts to enforce policy, whereas the policy configuration and decisions are handled by OpenAM.
3. The policy agent communicates with OpenAM to get the policy decision to enforce.
4. For a resource to which OpenAM approves access, the policy agent allows access.
5. The web server returns the requested access to the web client.

1.2. How Web Policy Agents are Configured

You install web policy agents in the web servers holding web resources that you want to protect. By default, the web policy agent has only enough configuration at installation time to connect to OpenAM in order to get the rest of its configuration from the OpenAM configuration store. With nearly all configuration stored centrally, you can manage policy agents centrally from the OpenAM console.

You can opt to store the agent configuration locally if necessary. If you store the configuration locally, then avoid issues with the configuration by making sure you provide valid values for configuration properties ending in the following.

- `.cookie.name`
- `.fqdn.default`
- `.agenturi.prefix`
- `.naming.url`
- `.login.url`
- `.instance.name`
- `.username`
- `.password`
- `.connection_timeout`
- `.policy_clock_skew`

You configure web policy agents per realm. Thus to access centralized configuration, you select Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`*. Web policy agent configuration is distinct from policy configuration. The only policy-like configuration that you apply to web policy agents is indicating which URLs in the web server can be ignored (*not enforced URLs*) and which client IP address are exempt from policy enforcement (*not enforced IPs*).

For each aspect of web policy agent configuration, you can configure the policy agent through the OpenAM console during testing, and then export the resulting configuration in order to script configuration in your production environment.

Chapter 2. Installing the Apache 2.0.x Policy Agent

This chapter covers installation of the policy agent for Apache Web Server 2.0.x.

2.1. Before You Install

Make sure OpenAM is installed, running, that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install Apache HTTP Server before you install the policy agent, and you must stop the server during installation.

You must install a Java 6 runtime environment, and set the `JAVA_HOME` environment variable. The policy agent installer requires Java.

```
$ echo $JAVA_HOME
/path/to/java1.6
$ which java
/usr/bin/java
```

Download the Apache 2.0 policy agent for your platform from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Unzip the file in the directory where you plan to install the web policy agent. The agent you install stores its configuration and logs under this directory.

When you unzip the policy agent .zip download, you find the following directories under the `web_agents/apache_agent` directory.

`bin`
: Contains the installation and configuration program, **agentadmin**; the certificate management tool **certutil** and the password hashing tool **crypt_util**.

`config`
: Configuration templates used by the **agentadmin** command during installation

`data`
: Not used

`etc`
: Apache configuration template used during installation

`installer-logs`
: Location for log files written during installation

`lib`
: Shared libraries used by the web policy agent

`locale`
: Property files used by the installation program

2.2. Installing Apache 2.0 Web Policy Agent

Complete the following procedures to install the policy agent.

Procedure 2.1. To Create the Apache 2.0 Web Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *`Realm Name`*> Agents > Web, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

   Password
   : Password the agent uses to authenticate to OpenAM

   Configuration
   : Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

   Server URL
   : The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

     In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

   Agent URL
   : The web server URL that the agent protects

In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 2.2. To Create the Password File

1. Create a text file containing only the password.

   ```
   $ echo password > /tmp/pwd.txt
   ```

2. Protect the password file you create as appropriate for your operating system.

   ```
   $ chmod 400 /tmp/pwd.txt
   ```

Procedure 2.3. To Install the Policy Agent into Apache 2.0

1. Shut down the Apache 2.0 server where you plan to install the agent.

   ```
   $ /path/to/apache20/bin/apachectl -k stop
   ```

2. Make sure OpenAM is running.

3. Run **./agentadmin --install** to install the agent.

   ```
   $ cd /path/to/web_agents/apache_agent/bin/
   $ ./agentadmin --install
   ...
   -----------------------------------------------
   SUMMARY OF YOUR RESPONSES
   -----------------------------------------------
   Apache Server Config Directory : /path/to/apache20/conf
   OpenAM server URL : http://openam.example.com:8080/openam
   Agent URL : http://www.example.com:80
   Agent Profile name : Apache 2.0 Agent
   Agent Profile Password file name : /tmp/pwd.txt

   ...
   SUMMARY OF AGENT INSTALLATION
   -----------------------------
   Agent instance name: Agent_001
   Agent Bootstrap file location:
   /path/to/web_agents/apache_agent/Agent_001/config/
    OpenSSOAgentBootstrap.properties
   Agent Configuration Tag file location
   /path/to/web_agents/apache_agent/Agent_001/config/
    OpenSSOAgentConfiguration.properties
   Agent Audit directory location:
   /path/to/web_agents/apache_agent/Agent_001/logs/audit
   Agent Debug directory location:
   /path/to/web_agents/apache_agent/Agent_001/logs/debug


   Install log file location:
   ```

```
/path/to/web_agents/apache_agent/installer-logs/audit/install.log
...
```

Upon successful completion, the installer has added the agent as a module to the Apache 2.0 configuration, and also set up configuration and log directories for the agent.

Note

If the agent is in a different domain than the server, refer to the *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

4. Take note of the configuration files and log locations.

   Each agent instance that you install on the system has its own numbered configuration and logs directory. The first agent's configuration and logs are thus located under the directory `web_agents/apache_agent/Agent_001/`.

   `config/OpenSSOAgentBootstrap.properties`
   : Used to bootstrap the web policy agent, allowing the agent to connect to OpenAM and download its configuration

   `config/OpenSSOAgentConfiguration.properties`
   : Only used if you configured the web policy agent to use local configuration

   `logs/audit/`
   : Operational audit log directory, only used if remote logging to OpenAM is disabled

   `logs/debug/`
   : Debug directory where the `amAgent` debug file resides. Useful in troubleshooting policy agent issues.

5. If your policy agent configuration is not in the top-level realm (/), then you must edit config/OpenSSOAgentBootstrap.properties to identify the sub-realm that has your policy agent configuration. Find com.sun.identity.agents.config.organization.name and change the / to the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

6. Start the Apache 2.0 server where you installed the agent.

```
$ /path/to/apache20/bin/apachectl -k start
```

Procedure 2.4. To Check the Policy Agent Installation

1. Check the Apache 2.0 error log after you start the server to make sure startup completed successfully.

```
$ tail -n 1 /path/to/apache20/logs/error_log
[Fri Sep 09 17:01:37 2011] [notice] Apache/2.0.64 (Unix) configured
 -- resuming normal operations
```

2. Check the `amAgent` debug log to verify that no errors occurred on startup.

```
$ tail /path/to/web_agents/apache_agent/Agent_001/logs/debug/amAgent
2011-09-09 17:01:37.179    -1 14516:94c6e58 all: ==============...=====
2011-09-09 17:01:37.179    -1 14516:94c6e58 all: Version: ...
2011-09-09 17:01:37.179    -1 14516:94c6e58 all:
2011-09-09 17:01:37.179    -1 14516:94c6e58 all: Build Date: ...
2011-09-09 17:01:37.179    -1 14516:94c6e58 all: Build Machine: ..forgerock.com
2011-09-09 17:01:37.179    -1 14516:94c6e58 all: ==============...=====
```

3. If you have a policy configured, you can test your policy agent. For example, try to browse to a resource that your policy agent protects. You should be redirected to OpenAM to authenticate, for example as user `demo`, password `changeit`. After you authenticate, OpenAM then redirects you back to the resource you tried to access.

2.3. Custom Apache 2.0 Web Policy Agent Installation

When running multiple Apache 2.0 servers on the same host, use **./agentadmin --custom-install**.

When performing a scripted, silent installation, use **./agentadmin --install --saveResponse** ***`response-file`*** to create a response file for scripted installation. Then install silently using **./agentadmin --install --useResponse** ***`response-file`***.

With **./agentadmin --custom-install**, you can opt to create the policy agent profile during installation. The OpenAM administrator must first create an agent administrator user, as described in *Delegating Agent Profile Creation*, and provide you with the agent administrator user name and password. Before running the **./agentadmin --custom-install** command, put the password alone in a read-only file only the user installing can access, as for the agent password. When the **agentadmin** command prompts you to create the profile during installation, enter `true`, and then respond to the **agentadmin** prompts for the agent administrator user name and password file.

2.4. Remove Apache 2.0 Web Policy Agent Software

Shut down the Apache 2.0 server before you uninstall the policy agent.

```
$ /path/to/apache20/bin/apachectl -k stop
```

To remove the web policy agent, use **./agentadmin --uninstall**.

```
$ ./agentadmin --uninstall
...
-----------------------------------------------
SUMMARY OF YOUR RESPONSES
-----------------------------------------------
Apache Server Config Directory : /path/to/apache20/conf

...
Deleting the config directory
/path/to/web_agents/apache_agent/Agent_001/config ...DONE.

Removing Agent parameters from /path/to/apache20/conf/httpd.conf file
...DONE.


Uninstall log file location:
/path/to/web_agents/apache_agent/installer-logs/audit/uninstall.log
...
```

Chapter 3. Installing the Apache 2.2 Policy Agent

This chapter covers installation of the policy agent for Apache HTTP Server 2.2.x.

3.1. Before You Install

Make sure OpenAM is installed, running, that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install Apache HTTP Server before you install the policy agent, and you must stop the server during installation.

You must install a Java 6 runtime environment, and set the `JAVA_HOME` environment variable. The policy agent installer requires Java.

```
$ echo $JAVA_HOME
/path/to/java1.6
$ which java
/usr/bin/java
```

Download the Apache 2.2 policy agent for your platform from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Unzip the file in the directory where you plan to install the web policy agent. The agent you install stores its configuration and logs under this directory.

When you unzip the policy agent .zip download, you find the following directories under the `web_agents/apache22_agent` directory.

`bin`
: Contains the installation and configuration program, **agentadmin**; the certificate management tool **certutil** and the password hashing tool **crypt_util**.

`config`
: Configuration templates used by the **agentadmin** command during installation

`data`
: Not used

`etc`
: Apache configuration template used during installation

`installer-logs`
: Location for log files written during installation

`lib`
: Shared libraries used by the web policy agent

`locale`
: Property files used by the installation program

3.2. Installing Apache 2.2 Web Policy Agent

Complete the following procedures to install the policy agent.

Procedure 3.1. To Create the Apache 2.2 Web Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *`Realm Name`*> Agents > Web, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

   Password
   : Password the agent uses to authenticate to OpenAM

   Configuration
   : Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

   Server URL
   : The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

     In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

   Agent URL
   : The web server URL that the agent protects

In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 3.2. To Create the Password File

1. Create a text file containing only the password.

```
$ echo password > /tmp/pwd.txt
```

2. Protect the password file you create as appropriate for your operating system.

```
$ chmod 400 /tmp/pwd.txt
```

Procedure 3.3. To Install the Policy Agent into Apache 2.2

1. Shut down the Apache 2.2 server where you plan to install the agent.

```
$ /path/to/apache22/bin/apachectl -k stop
```

2. Make sure OpenAM is running.

3. Run **./agentadmin --install** to install the agent.

```
$ cd /path/to/web_agents/apache22_agent/bin/
$ ./agentadmin --install
...
-----------------------------------------------
SUMMARY OF YOUR RESPONSES
-----------------------------------------------
Apache Server Config Directory : /path/to/apache22/conf
OpenAM server URL : http://openam.example.com:8080/openam
Agent URL : http://www.example.com:80
Agent Profile name : Apache Web Agent
Agent Profile Password file name : /tmp/pwd.txt

...
SUMMARY OF AGENT INSTALLATION
-----------------------------
Agent instance name: Agent_001
Agent Bootstrap file location:
/path/to/web_agents/apache22_agent/Agent_001/config/
 OpenSSOAgentBootstrap.properties
Agent Configuration Tag file location
/path/to/web_agents/apache22_agent/Agent_001/config/
 OpenSSOAgentConfiguration.properties
Agent Audit directory location:
/path/to/web_agents/apache22_agent/Agent_001/logs/audit
Agent Debug directory location:
/path/to/web_agents/apache22_agent/Agent_001/logs/debug


Install log file location:
```

```
/path/to/web_agents/apache22_agent/installer-logs/audit/install.log
...
```

Upon successful completion, the installer has added the agent as a module to the Apache 2.2 configuration, and also set up configuration and log directories for the agent.

Note

If the agent is in a different domain than the server, refer to *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

4. Take note of the configuration files and log locations.

   Each agent instance that you install on the system has its own numbered configuration and logs directory. The first agent's configuration and logs are thus located under the directory `web_agents/apache22_agent/Agent_001/`.

   `config/OpenSSOAgentBootstrap.properties`
   : Used to bootstrap the web policy agent, allowing the agent to connect to OpenAM and download its configuration

   `config/OpenSSOAgentConfiguration.properties`
   : Only used if you configured the web policy agent to use local configuration

   `logs/audit/`
   : Operational audit log directory, only used if remote logging to OpenAM is disabled

   `logs/debug/`
   : Debug directory where the `amAgent` debug file resides. Useful in troubleshooting policy agent issues.

5. If your policy agent configuration is not in the top-level realm (/), then you must edit config/OpenSSOAgentBootstrap.properties to identify the sub-realm that has your policy agent configuration. Find com.sun.identity.agents.config.organization.name and change the / to the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

6. Start the Apache 2.2 server where you installed the agent.

```
$ /path/to/apache22/bin/apachectl -k start
```

Procedure 3.4. To Check the Policy Agent Installation

1. Check the Apache 2.2 error log after you start the server to make sure startup completed successfully.

```
$ tail -n 2 /path/to/apache22/logs/error_log
[Sat Sep 03 13:28:16 2011] [notice] Policy web agent shared memory conf...
[Sat Sep 03 13:28:16 2011] [notice] Apache/2.2.19 (Unix) DSAME/3.0 configured
 -- resuming normal operations
```

2. Check the `amAgent` debug log to verify that no errors occurred on startup.

```
$ tail /path/to/web_agents/apache22_agent/Agent_001/logs/debug/amAgent
2011-09-03 13:28:16.971    -1 32686:9daae60 all: ==============...=====
2011-09-03 13:28:16.972    -1 32686:9daae60 all: Version: ...
2011-09-03 13:28:16.972    -1 32686:9daae60 all:
2011-09-03 13:28:16.972    -1 32686:9daae60 all: Build Date: ...
2011-09-03 13:28:16.972    -1 32686:9daae60 all: Build Machine: ..forgerock.com
2011-09-03 13:28:16.972    -1 32686:9daae60 all: ==============...=====
```

3. If you have a policy configured, you can test your policy agent. For example, try to browse to a resource that your policy agent protects. You should be redirected to OpenAM to authenticate, for example as user `demo`, password `changeit`. After you authenticate, OpenAM then redirects you back to the resource you tried to access.

3.3. Custom Apache 2.2 Web Policy Agent Installation

When running multiple Apache 2.2 servers on the same host, use **./agentadmin --custom-install**.

When performing a scripted, silent installation, use **./agentadmin --install --saveResponse** ***`response-file`*** to create a response file for scripted installation. Then install silently using **./agentadmin --install --useResponse** ***`response-file`***.

With **./agentadmin --custom-install**, you can opt to create the policy agent profile during installation. The OpenAM administrator must first create an agent administrator user, as described in *Delegating Agent Profile Creation*, and provide you with the agent administrator user name and password. Before running the **./agentadmin --custom-install** command, put the password alone in a read-only file only the user installing can access, as for the agent password. When the **agentadmin** command prompts you to create the profile during installation, enter `true`, and then respond to the **agentadmin** prompts for the agent administrator user name and password file.

3.4. Remove Apache 2.2 Web Policy Agent Software

Shut down the Apache 2.2 server before you uninstall the policy agent.

```
$ /path/to/apache22/bin/apachectl -k stop
```

To remove the web policy agent, use **./agentadmin --uninstall**.

```
$ ./agentadmin --uninstall
...
-----------------------------------------------
SUMMARY OF YOUR RESPONSES
-----------------------------------------------
Apache Server Config Directory : /path/to/apache22/conf

...
Deleting the config directory
/path/to/web_agents/apache22_agent/Agent_001/config
...DONE.

Removing Agent parameters from /path/to/apache22/conf/httpd.conf file
...DONE.


Uninstall log file location:
/path/to/web_agents/apache22_agent/installer-logs/audit/uninstall.log
...
```

Chapter 4. Installing the Apache 2.4 Policy Agent

This chapter covers installation of the policy agent for Apache HTTP Server 2.4.x.

4.1. Before You Install

Make sure OpenAM is installed, running, that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install Apache HTTP Server before you install the policy agent, and you must stop the server during installation.

You must install a Java 6 runtime environment, and set the `JAVA_HOME` environment variable. The policy agent installer requires Java.

Download the Apache 2.4 policy agent for your platform from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Unzip the file in the directory where you plan to install the web policy agent. The agent you install stores its configuration and logs under this directory.

When you unzip the policy agent .zip download, you find the following directories under the `web_agents/apache24_agent` directory.

`bin`
: Contains the installation and configuration program, **agentadmin**; the certificate management tool **certutil** and the password hashing tool **crypt_util**.

`config`
: Configuration templates used by the **agentadmin** command during installation

`data`
: Not used

`etc`
: Apache configuration template used during installation

`installer-logs`
: Location for log files written during installation

`lib`
: Shared libraries used by the web policy agent

`locale`
: Property files used by the installation program

4.2. Installing Apache 2.4 Web Policy Agent

Complete the following procedures to install the policy agent.

Procedure 4.1. To Create the Apache 2.4 Web Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *`Realm Name`*> Agents > Web, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

   Password
   : Password the agent uses to authenticate to OpenAM

   Configuration
   : Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

   Server URL
   : The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

     In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

   Agent URL
   : The web server URL that the agent protects

     In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 4.2. To Create the Password File

1. Create a text file containing only the password.

```
$ echo password > /tmp/pwd.txt
```

2. Protect the password file you create as appropriate for your operating system.

```
$ chmod 400 /tmp/pwd.txt
```

Procedure 4.3. To Install the Policy Agent into Apache 2.4

1. Shut down the Apache 2.4 server where you plan to install the agent.

```
$ /path/to/apache24/bin/apachectl -k stop
```

2. Make sure OpenAM is running.

3. Run **./agentadmin --install** to install the agent.

```
$ cd /path/to/web_agents/apache24_agent/bin/
$ ./agentadmin --install
...
-----------------------------------------------
SUMMARY OF YOUR RESPONSES
-----------------------------------------------
Apache Server Config Directory : /path/to/apache24/conf
OpenAM server URL : http://openam.example.com:8080/openam
Agent URL : http://www.example.com:80
Agent Profile name : Apache Web Agent
Agent Profile Password file name : /tmp/pwd.txt

...
SUMMARY OF AGENT INSTALLATION
-----------------------------
Agent instance name: Agent_001
Agent Bootstrap file location:
/path/to/web_agents/apache24_agent/Agent_001/config/
 OpenSSOAgentBootstrap.properties
Agent Configuration Tag file location
/path/to/web_agents/apache24_agent/Agent_001/config/
 OpenSSOAgentConfiguration.properties
Agent Audit directory location:
/path/to/web_agents/apache24_agent/Agent_001/logs/audit
Agent Debug directory location:
/path/to/web_agents/apache24_agent/Agent_001/logs/debug


Install log file location:
/path/to/web_agents/apache24_agent/installer-logs/audit/install.log
...
```

Upon successful completion, the installer has added the agent as a module to the Apache 2.4 configuration, and also set up configuration and log directories for the agent. You can find a backup Apache HTTPD configuration file, `http.conf-preAmAgent-*`, in the Apache HTTPD configuration directory.

Note

If the agent is in a different domain than the OpenAM server, refer to the *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

4. Take note of the configuration files and log locations.

   Each agent instance that you install on the system has its own numbered configuration and logs directory. The first agent's configuration and logs are thus located under the directory `web_agents/apache24_agent/Agent_001/`.

   `config/OpenSSOAgentBootstrap.properties`
   : Used to bootstrap the web policy agent, allowing the agent to connect to OpenAM and download its configuration

   `config/OpenSSOAgentConfiguration.properties`
   : Only used if you configured the web policy agent to use local configuration

   `logs/audit/`
   : Operational audit log directory, only used if remote logging to OpenAM is disabled

   `logs/debug/`
   : Debug directory where the `amAgent` debug file resides. Useful in troubleshooting policy agent issues.

5. If your policy agent configuration is not in the top-level realm (/), then you must edit `config/OpenSSOAgentBootstrap.properties` to indentify the sub-realm that has your policy agent configuration. Find com.sun.identity.agents.config.organization.name and change the `/` to the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

6. Start the Apache 2.4 server where you installed the agent.

   ```
   $ /path/to/apache24/bin/apachectl -k start
   ```

Procedure 4.4. To Check the Policy Agent Installation

1. Check the Apache 2.4 error log after you start the server to make sure startup completed successfully.

```
$ tail -n 2 /path/to/apache24/logs/error_log
[Fri Sep 14 12:48:55.765192 2012] [dsame:notice] [pid 18991:tid 3075335872]
 Policy web agent shared memory configuration: notif_shm_size[2099200],
 pdp_shm_size[3213312], max_pid_count[256], max_pdp_count[256]
[Fri Sep 14 12:48:55.774790 2012] [mpm_event:notice] [pid 18991:tid 3075335872]
 AH00489: Apache/2.4.3 (Unix) DSAME/3.0 configured
 -- resuming normal operations
```

2. Check the `amAgent` debug log to verify that no errors occurred on startup.

```
$ tail /path/to/web_agents/apache24_agent/Agent_001/logs/debug/amAgent
2012-09-14 12:48:55.613      -1 18991:85fdd48 all: ==============...=====
2012-09-14 12:48:55.614      -1 18991:85fdd48 all: Version: ...
2012-09-14 12:48:55.614      -1 18991:85fdd48 all: Revision: ...
2012-09-14 12:48:55.614      -1 18991:85fdd48 all: Build Date: ...
2012-09-14 12:48:55.614      -1 18991:85fdd48 all: Build Machine: ...
2012-09-14 12:48:55.614      -1 18991:85fdd48 all: ==============...=====
```

3. If you have a policy configured, you can test your policy agent. For example, try to browse to a resource that your policy agent protects. You should be redirected to OpenAM to authenticate, for example as user `demo`, password `changeit`. After you authenticate, OpenAM then redirects you back to the resource you tried to access.

4.3. Custom Apache 2.4 Web Policy Agent Installation

When running multiple Apache 2.4 servers on the same host, use **./agentadmin --custom-install**.

When performing a scripted, silent installation, use **./agentadmin --install --saveResponse *`response-file`*** to create a response file for scripted installation. Then install silently using **./agentadmin --install --useResponse *`response-file`***.

With **./agentadmin --custom-install**, you can opt to create the policy agent profile during installation. The OpenAM administrator must first create an agent administrator user, as described in *Delegating Agent Profile Creation*, and provide you with the agent administrator user name and password. Before running the **./agentadmin --custom-install** command, put the password alone in a read-only file only the user installing can access, as for the agent password. When the **agentadmin** command prompts you to create the profile during installation, enter `true`, and then respond to the **agentadmin** prompts for the agent administrator user name and password file.

4.4. Remove Apache 2.4 Web Policy Agent Software

Shut down the Apache 2.4 server before you uninstall the policy agent.

```
$ /path/to/apache24/bin/apachectl -k stop
```

To remove the web policy agent, use **./agentadmin --uninstall**.

```
$ ./agentadmin --uninstall
...
-----------------------------------------------
SUMMARY OF YOUR RESPONSES
-----------------------------------------------
Apache Server Config Directory : /path/to/apache24/conf

...
Deleting the config directory
/path/to/web_agents/apache24_agent/Agent_001/config
...DONE.

Removing Agent parameters from /path/to/apache24/conf/httpd.conf file
...DONE.


Uninstall log file location:
/path/to/web_agents/apache24_agent/installer-logs/audit/uninstall.log
...
```

Chapter 5. Installing the Microsoft IIS 6 Policy Agent

This chapter covers installation of the policy agent for Microsoft Internet Information Services 6.

5.1. Before You Install

Make sure OpenAM is installed, running, that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install Microsoft IIS 6 before you install the policy agent, and make sure that IIS 6 allows anonymous authentication. Make sure that IIS 6 listens on the URL used during the web policy agent installation, such as `http://win2003.example.com:80/`. Furthermore, you must reset IIS 6 after installing the policy agent.

Download the IIS 6 policy agent for 32 or 64-bit Windows from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Unpack the file in the directory where you plan to install the web policy agent. The agent you install stores its configuration and logs under this directory.

When you unpack the policy agent you download, you find the following directories under the `web_agents\iis6_agent\` directory.

`bin`
: Contains the configuration creation script, **IIS6CreateConfig.vbs**; the agent administration and installation script, **IIS6Admin.vbs**; the certificate management tool **certutil.exe**; the password hashing tool **cryptit.exe**; additional .dll and support files.

`config`
: Configuration templates used by the scripts during configuration and installation

5.2. Installing IIS 6 Web Policy Agent

Complete the following procedures to install the policy agent.

Procedure 5.1. To Create the IIS 6 Web Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *Realm Name*> Agents > Web, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

   Password
   : Password the agent uses to authenticate to OpenAM

   Configuration
   : Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

   Server URL
   : The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

     In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

   Agent URL
   : The web server URL that the agent protects

     In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 5.2. To Create the Password File

1. Protect the password file you will create as appropriate.

2. Create a text file containing only the password.

```
C:\>notepad C:\Windows\Temp\pwd.txt
```

Procedure 5.3. To Configure Policy Agent Installation

1. Log on as a user with Administrator privileges.

2. Change to the directory where you unpacked the agent download.

```
C:\>cd web_agents\iis6_agent\bin
```

3. Create a configuration file using the **IIS6CreateConfig.vbs** script.

   Note

   The Web Site Identifier is the value of `id`, not the site name.

```
C:\web_agents\iis6_agent\bin>cscript IIS6CreateConfig.vbs config.txt
...
Enter the Agent Resource File Name [IIS6Resource.en] :

Enter the Agent URL (Example: http://agent.example.com:80) :
http://windows2003.example.com:80

Displaying the list of Web Sites and its corresponding Identifiers
Site Name (Site Id)
Default Web Site (1)

Web Site Identifier :
1
...
Enter the URL where the OpenAM server is running...:
http://openam.example.com:8080/openam

Please enter the Agent Profile name :
IIS 6 Web Agent

Enter the Agent profile password file :
C:\Windows\Temp\pwd.txt

-----------------------------------------------------
Agent Configuration file created : config.txt
-----------------------------------------------------
```

Procedure 5.4. To Install the Policy Agent into IIS 6

1. Log on as a user with Administrator privileges.

2. Make sure OpenAM is running.

3. Run **IIS6Admin.vbs** to install the agent.

```
C:\web_agents\iis6_agent\bin>cscript IIS6Admin.vbs -config config.txt
...
Enter the Agent Resource File Name [IIS6Resource.en] :

Creating the Agent Config Directory
Creating the OpenSSOAgentBootstrap.properties and
 OpenSSOAgentConfiguration.properties File
Updating the Windows Product Registry
Loading the IIS 6.0 Agent
Completed Configuring the IIS 6.0 Agent
```

4. Restart IIS 6.

```
C:\web_agents\iis6_agent\bin>iisreset

Attempting stop...
Internet services successfully stopped
Attempting start...
Internet services successfully restarted
```

Note

If the agent is in a different domain than the server, refer to *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

5. Take note of the configuration files and log locations.

   Each agent instance that you install on the system has its own configuration and logs directory. The agent protecting the Default Web Site (1) shown in the examples above has configuration and logs located under the directory `web_agents\iis6_agent\Identifier_1`. The number in the path to the agent configuration reflects the IIS site ID, unlike the other agents for which the number in the path is a counter. The number in the path therefore remains the same when you uninstall and then reinstall an agent to protect the same site.

   `config\OpenSSOAgentBootstrap.properties`
   Used to bootstrap the web policy agent, allowing the agent to connect to OpenAM and download its configuration

   `config\OpenSSOAgentConfiguration.properties`
   Only used if you configured the web policy agent to use local configuration

   `audit\`
   Operational audit log directory, only used if remote logging to OpenAM is disabled

   `debug\`
   Debug directory where the `amAgent` debug file resides. Useful in troubleshooting policy agent issues.

6. If your policy agent configuration is not in the top-level realm (/), then you must edit config\OpenSSOAgentBootstrap.properties to identify the sub-realm that has your policy agent configuration. Find com.sun.identity.agents.config.organization.name and change the / to the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

7. If the web policy agent performs naming URL validation, which you can configure by setting the `com.forgerock.agents.ext.url.validation.disable` property in `config\OpenSSOAgentBootstrap.properties`, then make sure the `IUSR_`*`MachineName`* user has read-write access to `C:\Windows\Temp\` before you start IIS.

8. If you have a policy configured, you can test your policy agent. For example, try to browse to a resource that your policy agent protects. You should be redirected to OpenAM to authenticate, for example as user `demo`, password `changeit`. After you authenticate, OpenAM then redirects you back to the resource you tried to access.

5.3. Custom IIS 6 Web Policy Agent Installation

When protecting multiple IIS 6 websites on the same host, use different configuration files for each site.

When preparing a scripted, silent installation, notice that the configuration file generated using **IIS6CreateConfig.vbs** is a text file containing all of the configuration information in clear text plus the encrypted password retrieved originally from the password file. Encrypt passwords using **cryptit.exe**.

```
C:\web_agents\iis6_agent\bin>cryptit.exe pwd-file encryption-key
```

5.4. Remove IIS 6 Web Policy Agent Software

To remove the web policy agent, log on as a user with Administrator privileges, run **cscript IIS6Admin.vbs -unconfig config.txt**, and then run **iisreset**.

Chapter 6. Installing the Microsoft IIS 7 Policy Agent

This chapter covers installation of the policy agent for Microsoft Internet Information Services 7.

6.1. Before You Install

Make sure OpenAM is installed, running, that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install Microsoft IIS 7 before you install the policy agent, and make sure that IIS 7 allows anonymous authentication. Make sure that IIS 7 listens on the URL used during the web policy agent installation, such as `http://windows7.example.com:80/`. Furthermore, you must reset IIS 7 after installing the policy agent.

Download the IIS 7 policy agent for 32 or 64-bit Windows from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Unpack the file in the directory where you plan to install the web policy agent. The agent you install stores its configuration and logs under this directory.

When you unpack the policy agent you download, you find the following directories under the `web_agents\iis7_agent\` directory.

`bin`
: Contains the configuration creation script, **IIS7CreateConfig.vbs**; the agent administration and installation script, **IIS7Admin.vbs**; the certificate management tool **certutil.exe**; the password hashing tool **cryptit.exe**; additional .dll and support files.

`config`
: Configuration templates used by the scripts during configuration and installation

6.2. Installing IIS 7 Web Policy Agent

Complete the following procedures to install the policy agent.

Procedure 6.1. To Create the IIS 7 Web Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *Realm Name*> Agents > Web, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

   Password
   : Password the agent uses to authenticate to OpenAM

   Configuration
   : Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

   Server URL
   : The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

     In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

   Agent URL
   : The web server URL that the agent protects

     In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 6.2. To Create the Password File

1. Protect the password file you will create as appropriate.

2. Create a text file containing only the password.

```
C:\>notepad C:\Windows\Temp\pwd.txt
```

Procedure 6.3. To Configure Policy Agent Installation

1. Log on as a user with Administrator privileges.

2. Change to the directory where you unpacked the agent download.

```
C:\>cd web_agents\iis7_agent\bin
```

3. Create a configuration file using the **IIS7CreateConfig.vbs** script.

   ## Note

   The Web Site Identifier is the value of id, not the site name.

```
C:\web_agents\iis7_agent\bin>cscript IIS7CreateConfig.vbs config.txt
...
Enter the Agent Resource File Name [IIS7Resource.en] :

Enter the Agent URL (Example: http://agent.example.com:80) :
http://windows7.example.com:80

Displaying the list of Web Sites and its corresponding Identifiers (id)

SITE "Default Web Site" (id:1,bindings:http/*:80:,state:Started)

Web Site Identifier :
1
...
Enter the URL where the OpenAM server is running...:
http://openam.example.com:8080/openam

Please enter the Agent Profile name :
IIS 7 Web Agent

Enter the Agent profile password file :
C:\Windows\Temp\pwd.txt

-----------------------------------------------------
Agent Configuration file created : config.txt
-----------------------------------------------------
```

Procedure 6.4. To Install the Policy Agent into IIS 7

1. Log on as a user with Administrator privileges.

2. Make sure OpenAM is running.

3. Run **IIS7Admin.vbs** to install the agent.

```
C:\web_agents\iis7_agent\bin>cscript IIS7Admin.vbs -config config.txt
...
Enter the Agent Resource File Name [IIS7Resource.en] :

Creating the Agent Config Directory
Creating the OpenSSOAgentBootstrap.properties and
 OpenSSOAgentConfiguration.properties File
Updating the Windows Product Registry
Installing policy web agent module in IIS (status: 0)
Adding policy web agent module to "Default Web Site" (status: 0)
```

```
Completed Configuring the IIS 7.0 Agent
```

4. Make sure the authentication method for IIS 7 is set to anonymous.

5. Restart IIS 7.

```
C:\web_agents\iis7_agent\bin>iisreset

Attempting stop...
Internet services successfully stopped
Attempting start...
Internet services successfully restarted
```

   Note

   If the agent is in a different domain than the server, refer to *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

6. Take note of the configuration files and log locations.

   Each agent instance that you install on the system has its own configuration and logs directory. The agent protecting the Default Web Site (id: 1) shown in the examples above has configuration and logs located under the directory `web_agents\iis7_agent\Identifier_1`. The number in the path to the agent configuration reflects the IIS site ID, unlike the other agents for which the number in the path is a counter. The number in the path therefore remains the same when you uninstall and then reinstall an agent to protect the same site.

   `config\OpenSSOAgentBootstrap.properties`
   : Used to bootstrap the web policy agent, allowing the agent to connect to OpenAM and download its configuration

   `config\OpenSSOAgentConfiguration.properties`
   : Only used if you configured the web policy agent to use local configuration

   `audit\`
   : Operational audit log directory, only used if remote logging to OpenAM is disabled

   `debug\`
   : Debug directory where the `amAgent` debug file resides. Useful in troubleshooting policy agent issues.

7. If your policy agent configuration is not in the top-level realm (/), then you must edit config\OpenSSOAgentBootstrap.properties to identify the sub-realm that has your policy agent configuration. Find

com.sun.identity.agents.config.organization.name and change the / to the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

8. If you have a policy configured, you can test your policy agent. For example, try to browse to a resource that your policy agent protects. You should be redirected to OpenAM to authenticate, for example as user `demo`, password `changeit`. After you authenticate, OpenAM then redirects you back to the resource you tried to access.

6.3. Custom IIS 7 Web Policy Agent Installation

When protecting multiple IIS 7 websites on the same host, use different configuration files for each site.

When preparing a scripted, silent installation, notice that the configuration file generated using **IIS7CreateConfig.vbs** is a text file containing all of the configuration information in clear text plus the encrypted password retrieved originally from the password file. Encrypt passwords using **cryptit.exe**.

```
C:\web_agents\iis7_agent\bin>cryptit.exe pwd-file encryption-key
```

6.4. Enable IIS 7 Basic Authentication & Password Replay Support

The IIS 7 web policy agent now supports IIS 7 basic authentication and password replay. You must use the appropriate software versions.

- For Microsoft Office integration, you must use Microsoft Office 2007 SP2 or later.
- For Microsoft SharePoint integration, you must use Microsoft SharePoint Server 2007 SP2 or later.

You must also apply workarounds as described for the following Microsoft issues.

Microsoft Support Issue: 841215
    Link: http://support.microsoft.com/kb/841215

    Description: Error message when you try to connect to a Windows SharePoint document library: "System error 5 has occurred"

    Summary: Enable Basic Authentication on the client computer.

Microsoft Support Issue: 870853
    Link: http://support.microsoft.com/kb/870853

Description: Office 2003 and 2007 Office documents open read-only in Internet Explorer

Summary: Add registry keys as described in Microsoft's support document.

Microsoft Support Issue: 928692
Link: http://support.microsoft.com/kb/928692

Description: Error message when you open a Web site by using Basic authentication in Expression Web on a computer that is running Windows Vista: "The folder name is not valid"

Summary: Edit the registry as described in Microsoft's support document.

Microsoft Support Issue: 932118
Link: http://support.microsoft.com/kb/932118

Description: Persistent cookies are not shared between Internet Explorer and Office applications

Summary: Add the web site the list of trusted sites.

Microsoft Support Issue: 943280
Link: http://support.microsoft.com/kb/943280

Description: Prompt for Credentials When Accessing FQDN Sites From a Windows Vista or Windows 7 Computer

Summary: Edit the registry as described in Microsoft's support document.

Microsoft Support Issue: 968851
Link: http://support.microsoft.com/kb/968851

Description: SharePoint Server 2007 Cumulative Update Server Hotfix Package (MOSS server-package): April 30, 2009

Summary: Apply the fix from Microsoft if you use SharePoint.

Microsoft Support Issue: 2123563
Link: http://support.microsoft.com/kb/2123563

Description: You cannot open Office file types directly from a server that supports only Basic authentication over a non-SSL connection

Summary: Enable SSL encryption on the web server.

Procedure 6.5. To Configure IIS 7 Basic Authentication & Password Replay Support

Follow these steps.

1. Generate and store an encryption key.

   a. Generate the key using `com.sun.identity.common.DESGenKey` using the .jars where you deployed OpenAM, as in the following example.

   ```
   $ cd /path/to/tomcat/webapps/openam/WEB-INF/lib
   $ java -cp openam-core-10.1.0.jar:openam-shared-10.1.0.jar
    com.sun.identity.common.DESGenKey
   Key ==> sxVoaDRAN0o=
   ```

   b. Store the key in the agent configuration on the property in the OpenAM console under Access Control > *realm-name* > Agents > Web > *agent-name* > Advanced > Microsoft IIS Server > Replay Password Key (property name: `com.sun.identity.agents.config.replaypasswd.key`), and then Save your work.

   c. Store the key in the server configuration in the OpenAM console under Configuration > Servers and Sites > *server-name* > Advanced > Add... to add the property `com.sun.am.replaypasswd.key` with the key you generated as the value, and then Save your work.

2. In the OpenAM console under Access Control > *realm-name* > Authentication > All Core Settings... > Authentication Post Processing Classes, add the class `com.sun.identity.authentication.spi.ReplayPasswd`, and then Save your work.

3. If you require Windows logon, or you need to use basic authentication with SharePoint or OWA, then you must configure Active Directory as a user date store, and you must configure the IIS 7 policy agent profile User ID Parameter and User ID Parameter Type so that the policy agent requests OpenAM to provide the appropriate account information from Active Directory in its policy response.

   Skip this step if you do not use SharePoint or OWA and no Windows logon is required.

   Make sure OpenAM data store is configured to use Active Directory as the user data store.

   In the OpenAM console under Access Control > *realm-name* > Agents > Web > *agent-name* > OpenAM Services > Policy Client Service, set User ID Parameter and User ID Parameter Type, and then Save your work.

For example if the real username for Windows domain logon in Active Directory is stored on the `samaccountname` attribute, then set the User ID Parameter to `samaccountname`, and the User ID Parameter Type to `LDAP`.

Setting the User ID Parameter Type to `LDAP` causes the policy agent to request that OpenAM get the value of the User ID Parameter attribute from the data store, in this case Active Directory. Given that information, the policy agent can set the HTTP headers `remote_user`, `auth_user`, or `logon_user` and `user_password` with Active Directory attribute values suitable for Windows logon, setting the remote user, and so forth.

4. To set the encrypted password in the AUTH_PASSWORD header, in the OpenAM console under Access Control > *realm-name* > Agents > Web > *agent-name* > Advanced > Microsoft IIS Server, select Show Password in HTTP Header, and then Save your work.

5. To have the agent perform Windows logon (for user token impersonation), in the OpenAM console under Access Control > *realm-name* > Agents > Web > *agent-name* > Advanced > Microsoft IIS Server, select Logon and Impersonation, and then Save your work.

6. In the OpenAM console under Access Control > *realm-name* > Agents > Web > *agent-name* > Advanced > Microsoft IIS Server, set Authentication Type to basic, and then Save your work.

7. To use the agent with SharePoint or Microsoft Office, configure OpenAM to support the `iPlanetDirectoryPro` as a persistent cookie.

   In the OpenAM console under Access Control > *realm-name* > Authentication > All Core Settings... > Persistent Cookie Mode, select Enabled, and then Save your work.

6.5. Remove IIS 7 Web Policy Agent Software

To remove the web policy agent, log on as a user with Administrator privileges, run **cscript IIS7Admin.vbs -unconfig config.txt**, and then run **iisreset**.

Chapter 7. Installing the Sun Web Server Policy Agent

This chapter covers installation of the policy agent for Sun Web Server.

7.1. Before You Install

Make sure OpenAM is installed, running, that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install Apache HTTP Server before you install the policy agent, and you must stop the server during installation.

You must install a Java 6 runtime environment, and set the `JAVA_HOME` environment variable. The policy agent installer requires Java.

```
$ echo $JAVA_HOME
/path/to/java1.6
$ which java
/usr/bin/java
```

Download the Sun Web Server policy agent for your platform from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Unzip the file in the directory where you plan to install the web policy agent. The agent you install stores its configuration and logs under this directory.

When you unzip the policy agent .zip download, you find the following directories under the `web_agents/sjsws_agent` directory.

`bin`
: Contains the installation and configuration program, **agentadmin**; the certificate management tool **certutil** and the password hashing tool **crypt_util**.

`config`
: Configuration templates used by the **agentadmin** command during installation

`data`
: Not used

`etc`
: Not used

`installer-logs`
: Location for log files written during installation

`lib`
: Shared libraries used by the web policy agent

`locale`
: Property files used by the installation program

7.2. Installing Sun Web Server Web Policy Agent

Complete the following procedures to install the policy agent.

Procedure 7.1. To Create the Sun Web Server Web Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *`Realm Name`*> Agents > Web, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

   Password
   : Password the agent uses to authenticate to OpenAM

   Configuration
   : Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

   Server URL
   : The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

     In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

   Agent URL
   : The web server URL that the agent protects

In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 7.2. To Create the Password File

1. Create a text file containing only the password.

   ```
   $ echo password > /tmp/pwd.txt
   ```

2. Protect the password file you create as appropriate for your operating system.

   ```
   $ chmod 400 /tmp/pwd.txt
   ```

Procedure 7.3. To Install the Policy Agent into Sun Web Server

1. Shut down Sun Web Server instance where you plan to install the agent.
2. Make sure OpenAM is running.
3. Run **agentadmin --install** to install the agent.

   ```
   $ /path/to/web_agents/sjsws_agent/bin/agentadmin --install
   ...
   -----------------------------------------------
   SUMMARY OF YOUR RESPONSES
   -----------------------------------------------
   Sun Java System Web Server Config Directory :
   /path/to/webserver7/https-www.example.com/config/
   OpenAM server URL : http://openam.example.com:8080/openam
   Agent URL : http://www.example.com:8080
   Agent Profile name : Sun Web Server Agent
   Agent Profile Password file name : /tmp/pwd.txt

   ...
   SUMMARY OF AGENT INSTALLATION
   -----------------------------
   Agent instance name: Agent_001
   Agent Bootstrap file location:
   /path/to/web_agents/sjsws_agent/Agent_001/config/
    OpenSSOAgentBootstrap.properties
   Agent Configuration Tag file location
   /path/to/web_agents/sjsws_agent/Agent_001/config/
    OpenSSOAgentConfiguration.properties
   Agent Audit directory location:
   /path/to/web_agents/sjsws_agent/Agent_001/logs/audit
   Agent Debug directory location:
   /path/to/web_agents/sjsws_agent/Agent_001/logs/debug


   Install log file location:
   /path/to/web_agents/sjsws_agent/installer-logs/audit/install.log
   ...
   ```

Upon successful completion, the installer has backed up and updated the Sun Web Server instance configuration, and has also set up configuration and log directories for the agent.

> **Note**
>
> If the agent is in a different domain than the server, refer to *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

4. Take note of the configuration files and log locations.

   Each agent instance that you install on the system has its own numbered configuration and logs directory. The first agent's configuration and logs are thus located under the directory `web_agents/sjsws_agent/Agent_001/`.

   `config/OpenSSOAgentBootstrap.properties`
   : Used to bootstrap the web policy agent, allowing the agent to connect to OpenAM and download its configuration

   `config/OpenSSOAgentConfiguration.properties`
   : Only used if you configured the web policy agent to use local configuration

   `logs/audit/`
   : Operational audit log directory, only used if remote logging to OpenAM is disabled

   `logs/debug/`
   : Debug log directory. Useful in troubleshooting policy agent issues.

5. If your policy agent configuration is not in the top-level realm (/), then you must edit config/OpenSSOAgentBootstrap.properties to identify the sub-realm that has your policy agent configuration. Find com.sun.identity.agents.config.organization.name and change the / to the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

6. Restart the Sun Web Server instance where you installed the agent.

7. Check that the agent protects the web site.

   If you have not yet configured any policies to allow access, then you should receive an HTTP 403 Forbidden error. In the above example, when accessing `http://www.example.com:8080/`, the content of the page returned appears in the browser as follows.

   ```
   Forbidden
   ```

```
Your client is not allowed to access the requested object.
```

7.3. Custom Sun Web Policy Agent Installation

For alternative installations, use **agentadmin --custom-install**.

When performing a scripted, silent installation, use **agentadmin --install --saveResponse *`response-file`*** to create a response file for scripted installation. Then install silently using **agentadmin --install --useResponse *`response-file`***.

With **./agentadmin --custom-install**, you can opt to create the policy agent profile during installation. The OpenAM administrator must first create an agent administrator user, as described in *Delegating Agent Profile Creation*, and provide you with the agent administrator user name and password. Before running the **./agentadmin --custom-install** command, put the password alone in a read-only file only the user installing can access, as for the agent password. When the **agentadmin** command prompts you to create the profile during installation, enter `true`, and then respond to the **agentadmin** prompts for the agent administrator user name and password file.

7.4. Remove Sun Web Policy Agent Software

Shut down the Sun Web Server before you uninstall the policy agent.

To remove the web policy agent, use **agentadmin --uninstall**.

Chapter 8. About OpenAM Java EE Policy Agents

OpenAM Java EE policy agents provide medium touch integration for web applications running in supported web application containers. Java EE policy agents require some configuration and code changes to deployed web applications. This chapter covers what Java EE policy agents do and how they work.

A *policy agent* enforces policy for OpenAM. A *J2EE policy agent* installed in a web application container intercepts requests from users trying to access resources in protected web applications. The agent denies access until the user has authorization from OpenAM to access a particular resource.

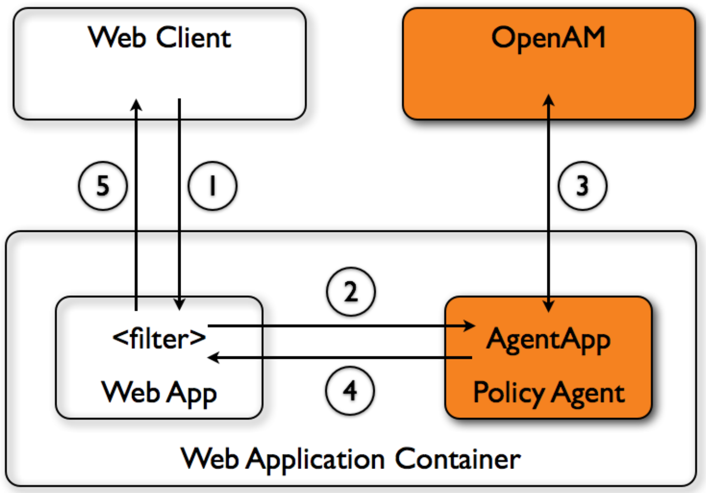
8.1. How the User, Application, Policy Agent, & OpenAM Interact

Imagine that a user attempts to access a protected resource before having authenticated by pointing her browser to a page in a protected application. Assume that you have configured OpenAM to protect the web application. You have therefore installed the J2EE agent in the web container, and also configured the protected web application to use the agent filter, thus sending requests through the agent. Then the J2EE policy agent intercepting her filtered browser's request finds no session token in the request, and so redirects the user's browser to the OpenAM login page for authentication. After the user has successfully authenticated, OpenAM sets a session token in a browser cookie, and redirects her browser back to the page she tried to access initially.

When the user's browser reiterates the request, the policy agent again checks that the request has a session token, finds a session token this time, and validates the session token with OpenAM. Given the valid session token, the policy agent gets a policy decision from OpenAM concerning whether the user can access the page. If OpenAM's Policy Service determines that the user is allowed to access the page, OpenAM responds to the policy agent that access should be granted. The J2EE policy agent then permits the page to be returned to the user's browser.

You can also configure J2EE agent filters to work in tandem with the J2EE security policies defined alongside the policies for OpenAM. In this case the filter ensures the J2EE security policy grants access to the resource before the agent gets a decision from OpenAM.

The following diagram shows how the pieces fit together when a Java EE client accesses a resource protected by a policy agent. This diagram is simplified to show only the essential principals rather than to describe every possible case.



A Java EE policy agent is a web application installed in the web application container. Other applications have filters configured to call the policy agent when a client requests access to a protected resource.

1. The web client requests access to a protected resource.
2. The web application filter settings put the request through the policy agent that protects the resource according to OpenAM policy. The policy agent acts to enforce policy, whereas the policy configuration and decisions are handled by OpenAM.
3. The policy agent communicates with OpenAM to get the policy decision to enforce.
4. For a resource to which OpenAM approves access, the policy agent allows access.
5. The web application returns the requested access to the web client.

8.2. How J2EE Policy Agents are Configured

You install J2EE policy agents in the web application containers serving web applications that you want to protect. J2EE policy agents are themselves web applications running in the container whose applications you configure OpenAM to protect. By default, the J2EE policy agent has only enough configuration at installation time to connect to OpenAM in order to get the rest of its configuration from the OpenAM configuration store. With nearly all configuration stored centrally, you can manage policy agents centrally from the OpenAM console.[1]

For each web application that you protect, you also configure at least the deployment descriptor to filter requests through the policy agent. ForgeRock delivers the J2EE policy agents with a sample application under `j2ee_agents/`*`container`*`_agent/sampleapp/` demonstrating the configuration to use to protect your web application.

You configure J2EE policy agents per OpenAM realm. Thus to access centralized configuration, you select Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`*. J2EE policy agent configuration is distinct from policy configuration. The only policy-like configuration that you apply to J2EE policy agents is indicating which URLs in the web server can be ignored (*not enforced URLs*) and which client IP address are exempt from policy enforcement (*not enforced IPs*).

For each aspect of J2EE policy agent configuration, you can configure the policy agent through the OpenAM console during testing, and then export the resulting configuration in order to script configuration in your production environment.

[1]You can opt to store the agent configuration locally if necessary.

Chapter 9. Installing the Apache Tomcat Policy Agent

This chapter covers installation of the policy agent for Apache Tomcat.

9.1. Before You Install

Make sure OpenAM is installed and running, and that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install Apache Tomcat before you install the policy agent, and you must stop the server during installation.

All of the Tomcat scripts must be present in `$CATALINA_HOME/bin`. The Tomcat Windows executable installer does not include the scripts, for example. If the scripts are not present in your installation, copy the contents of the `bin` directory from a .zip download of Tomcat of the same version as the one you installed.

You must install a Java 6 runtime environment, and set the `JAVA_HOME` environment variable.

```
$ echo $JAVA_HOME
/path/to/java1.6
$ which java
/usr/bin/java
```

Download the Tomcat policy agent from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Unzip the file in the directory where you plan to install the J2EE policy agent. The agent you install stores its configuration and logs under this directory.

When you unzip the policy agent, you find the following directories under the `j2ee_agents/tomcat_v6_agent` directory.

`bin`
: The installation and configuration program, **agentadmin**.

`config`
: Configuration templates used by the **agentadmin** command during installation

`data`
: Not used

`etc`
: Configuration templates used during installation

`installer-logs`
: Location for log files written during installation

`lib`
: Shared libraries used by the J2EE policy agent

`locale`
: Property files used by the installation program

`sampleapp`
: Sample application that demonstrates key features of the policy agent. Wait until you have installed the agent to deploy this.

9.2. Installing the Tomcat Policy Agent

Complete the following procedures to install the policy agent.

Procedure 9.1. To Create the Tomcat Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *`Realm Name`* > Agents > J2EE, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

   Password
   : Password the agent uses to authenticate to OpenAM

   Configuration
   : Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

   Server URL
   : The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

Agent URL
: The URL to the J2EE application that the agent protects

  In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 9.2. To Create the Password File

1. Create a text file containing only the password.

   ```
   $ echo password > /tmp/pwd.txt
   ```

2. Protect the password file you create as appropriate for your operating system.

   ```
   $ chmod 400 /tmp/pwd.txt
   ```

Procedure 9.3. To Install the Policy Agent into Tomcat

1. Shut down the Tomcat server where you plan to install the agent.

   ```
   $ /path/to/tomcat/bin/shutdown.sh
   ```

2. Make sure OpenAM is running.

3. Run **agentadmin --install** to install the agent.

   ```
   $ /path/to/j2ee_agents/tomcat_v6_agent/bin/agentadmin --install
   ...
   -----------------------------------------------
   SUMMARY OF YOUR RESPONSES
   -----------------------------------------------
   Tomcat Server Config Directory : /path/to/tomcat/conf
   OpenAM server URL : http://openam.example.com:8080/openam
   $CATALINA_HOME environment variable : /path/to/tomcat
   Tomcat global web.xml filter install : true
   Agent URL : http://www.example.com:8080/agentapp
   Agent Profile name : Tomcat Agent
   Agent Profile Password file name : /tmp/pwd.txt

   ...
   SUMMARY OF AGENT INSTALLATION
   -----------------------------
   Agent instance name: Agent_001
   Agent Bootstrap file location:
   /path/to/j2ee_agents/tomcat_v6_agent/Agent_001/config/
    OpenSSOAgentBootstrap.properties
   Agent Configuration file location
   /path/to/j2ee_agents/tomcat_v6_agent/Agent_001/config/
   ```

```
 OpenSSOAgentConfiguration.properties
Agent Audit directory location:
/path/to/j2ee_agents/tomcat_v6_agent/Agent_001/logs/audit
Agent Debug directory location:
/path/to/j2ee_agents/tomcat_v6_agent/Agent_001/logs/debug


Install log file location:
/path/to/j2ee_agents/tomcat_v6_agent/installer-logs/audit/install.log
...
```

Upon successful completion, the installer has added the agent configuration to Tomcat's configuration, and also set up configuration and log directories for the agent.

> **Note**
>
> If the agent is in a different domain than the server, refer to *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

4. Take note of the configuration files and log locations.

   Each agent instance that you install on the system has its own numbered configuration and logs directory. The first agent's configuration and logs are thus located under the directory `j2ee_agents/tomcat_v6_agent/Agent_001/`.

   `config/OpenSSOAgentBootstrap.properties`
   : Used to bootstrap the J2EE policy agent, allowing the agent to connect to OpenAM and download its configuration

   `config/OpenSSOAgentConfiguration.properties`
   : Only used if you configured the J2EE policy agent to use local configuration

   `logs/audit/`
   : Operational audit log directory, only used if remote logging to OpenAM is disabled

   `logs/debug/`
   : Debug directory where the `debug.out` debug file resides. Useful in troubleshooting policy agent issues.

5. If your policy agent configuration is not in the top-level realm (/), then you must edit config/OpenSSOAgentBootstrap.properties to identify the sub-realm that has your policy agent configuration. Find com.sun.identity.agents.config.organization.name and change the / to the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

6. If you choose not to let the installer install a global filter in Tomcat's `web.xml`, then you must add the filter manually for each protected application's `web.xml` configuration, following the opening <web-app> tag. The file for the sample application delivered with the agent is `/path/to/j2ee_agents/tomcat_v6_agent/sampleapp/etc/web.xml`.

```
<filter>
 <filter-name>Agent</filter-name>
 <display-name>Agent</display-name>
 <description>OpenAM Policy Agent Filter</description>
<filter-class>com.sun.identity.agents.filter.AmAgentFilter</filter-class>
</filter>
<filter-mapping>
 <filter-name>Agent</filter-name>
 <url-pattern>/*</url-pattern>
 <dispatcher>REQUEST</dispatcher>
 <dispatcher>INCLUDE</dispatcher>
 <dispatcher>FORWARD</dispatcher>
 <dispatcher>ERROR</dispatcher>
</filter-mapping>
```

7. Add the agent application web archive to Tomcat's webapps.

   $ cp /path/to/j2ee_agents/tomcat_v6_agent/etc/agentapp.war /path/to/tomcat/webapps/

8. Start the Tomcat server where you installed the agent.

```
$ /path/to/tomcat/bin/startup.sh
```

Procedure 9.4. To Check the Policy Agent Installation

1. Check the Tomcat server log after you start the server to make sure startup completed successfully.

```
$ tail -n 1 /path/to/tomcat/logs/catalina.out
INFO: Server startup in 810 ms
```

2. Check the `debug.out` debug log to verify that the agent did start up.

```
$ tail -n 7 /path/to/j2ee_agents/tomcat_v6_agent/Agent_001/logs/debug/debug.out
=======================================
Version: ...
Revision: 3111
Build Date: 20120915
Build Machine: builds.forgerock.org
=======================================
```

3. If you have a policy configured, you can test your policy agent. For example, try to browse to a resource that your policy agent protects. You should be redirected to OpenAM to authenticate, for example as user

`demo`, password `changeit`. After you authenticate, OpenAM then redirects you back to the resource you tried to access.

9.3. Silent Tomcat Policy Agent Installation

When performing a scripted, silent installation, use **agentadmin --install --saveResponse** ***`response-file`*** to create a response file for scripted installation. Then install silently using **agentadmin --install --useResponse** ***`response-file`***.

9.4. Remove Tomcat Policy Agent Software

Shut down the Tomcat server before you uninstall the policy agent.

```
$ /path/to/tomcat/bin/shutdown.sh
```

To remove the J2EE policy agent, use **agentadmin --uninstall**. You must provide the Tomcat server configuration directory location.

Uninstall does not remove the agent instance directory, but you can do so manually after removing the agent configuration from Tomcat.

Chapter 10. Installing the GlassFish Policy Agent

This chapter covers installation of the policy agent for GlassFish.

10.1. Before You Install

Make sure OpenAM is installed and running, and that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install GlassFish before you install the policy agent, and you must stop the domain with applications to protect during installation.

You must install a Java 6 runtime environment, and set the `JAVA_HOME` environment variable.

```
$ echo $JAVA_HOME
/path/to/java1.6
$ which java
/usr/bin/java
```

Download the GlassFish policy agent from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Unzip the file in the directory where you plan to install the J2EE policy agent. The agent you install stores its configuration and logs under this directory.

When you unzip the policy agent, you find the following directories under the `j2ee_agents/appserver_v10_agent` directory.

`bin`
: The installation and configuration program, **agentadmin**.

`config`
: Configuration templates used by the **agentadmin** command during installation

`data`
: Not used

`etc`
: Agent web application used during installation

`installer-logs`
: Location for log files written during installation

`lib`
: Shared libraries used by the J2EE policy agent

`locale`
: Property files used by the installation program

`sampleapp`
: Sample application that demonstrates key features of the policy agent. Wait until you have installed the agent to deploy this.

10.2. Installing the GlassFish Policy Agent

Complete the following procedures to install the policy agent.

Procedure 10.1. To Create the GlassFish Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *`Realm Name`* > Agents > J2EE, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

   Password
   : Password the agent uses to authenticate to OpenAM

   Configuration
   : Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

   Server URL
   : The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

     In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

Agent URL
: The URL to the J2EE agent application, such as `http://www.example.com:8080/agentapp`

  In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 10.2. To Create the Password File

1. Create a text file containing only the password.

   ```
   $ echo password > /tmp/pwd.txt
   ```

2. Protect the password file you create as appropriate for your operating system.

   ```
   $ chmod 400 /tmp/pwd.txt
   ```

Procedure 10.3. To Install the Policy Agent into GlassFish

1. Shut down the GlassFish domain where you plan to install the agent.

   ```
   $ /path/to/glassfish/bin/asadmin stop-domain domain1
   Waiting for the domain to stop ....
   Command stop-domain executed successfully.
   ```

2. Make sure OpenAM is running.

3. Run **agentadmin --install** to install the agent.

   ```
   $ /path/to/j2ee_agents/appserver_v10_agent/bin/agentadmin --install
   ...
   -----------------------------------------------
   SUMMARY OF YOUR RESPONSES
   -----------------------------------------------
   Application Server Config Directory :
   /path/to/glassfish/glassfish/domains/domain1/config
   Application Server Instance name : server
   OpenAM server URL : http://openam.example.com:8080/openam
   Agent URL : http://www.example.com:8080/agentapp
   Agent Profile name : GlassFish Agent
   Agent Profile Password file name : /tmp/pwd.txt

   ...
   SUMMARY OF AGENT INSTALLATION
   -----------------------------
   Agent instance name: Agent_001
   Agent Bootstrap file location:
   /path/to/j2ee_agents/appserver_v10_agent/Agent_001/config/
    OpenSSOAgentBootstrap.properties
   Agent Configuration file location
   /path/to/j2ee_agents/appserver_v10_agent/Agent_001/config/
   ```

```
 OpenSSOAgentConfiguration.properties
Agent Audit directory location:
/path/to/j2ee_agents/appserver_v10_agent/Agent_001/logs/audit
Agent Debug directory location:
/path/to/j2ee_agents/appserver_v10_agent/Agent_001/logs/debug


Install log file location:
/path/to/j2ee_agents/appserver_v10_agent/installer-logs/audit/install.log
...
```

Upon successful completion, the installer has updated the GlassFish configuration, and also set up configuration and log directories for the agent.

> **Note**
>
> If the agent is in a different domain than the server, refer to *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

4. Take note of the configuration files and log locations.

   Each agent instance that you install on the system has its own numbered configuration and logs directory. The first agent's configuration and logs are thus located under the directory `j2ee_agents/appserver_v10_agent/Agent_001/`.

   `config/OpenSSOAgentBootstrap.properties`
   : Used to bootstrap the J2EE policy agent, allowing the agent to connect to OpenAM and download its configuration

   `config/OpenSSOAgentConfiguration.properties`
   : Only used if you configured the J2EE policy agent to use local configuration

   `logs/audit/`
   : Operational audit log directory, only used if remote logging to OpenAM is disabled

   `logs/debug/`
   : Debug directory where the debug file resides. Useful in troubleshooting policy agent issues.

5. If your policy agent configuration is not in the top-level realm (/), then you must edit config/OpenSSOAgentBootstrap.properties to identify the sub-realm that has your policy agent configuration. Find com.sun.identity.agents.config.organization.name and change the / to the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

6. To protect a web application, you must add the following filter to the application's `web.xml` configuration, following the opening <web-app> tag. The file for the sample application delivered with the agent is `/path/to/j2ee_agents/appserver_v10_agent/sampleapp/etc/web.xml`.

```
<filter>
 <filter-name>Agent</filter-name>
 <display-name>Agent</display-name>
 <description>OpenAM Policy Agent Filter</description>
<filter-class>com.sun.identity.agents.filter.AmAgentFilter</filter-class>
</filter>
<filter-mapping>
 <filter-name>Agent</filter-name>
 <url-pattern>/*</url-pattern>
 <dispatcher>REQUEST</dispatcher>
 <dispatcher>INCLUDE</dispatcher>
 <dispatcher>FORWARD</dispatcher>
 <dispatcher>ERROR</dispatcher>
</filter-mapping>
```

7. Start the GlassFish domain where you installed the agent.

```
$ /path/to/glassfish/bin/asadmin start-domain domain1
Waiting for domain1 to start .....
Successfully started the domain : domain1
domain  Location: /path/to/glassfish/glassfish/domains/domain1
Log File: /path/to/glassfish/glassfish/domains/domain1/logs/server.log
Admin Port: 4848
Command start-domain executed successfully.
```

8. Deploy the agent web application.

```
cd /path/to/glassfish/glassfish/bin/asadmin
    $ deploy --name agentapp --contextroot /agentapp
 /path/to/j2ee_agents/appserver_v10_agent/etc/agentapp.war
```

9. Check your work by quickly deploying the sample application, `/path/to/j2ee_agents/appserver_v10_agent/sampleapp/dist/agentsample.ear` through the GlassFish administration console, and verifying that the agent redirects to OpenAM for authentication and that access is denied after successful login to OpenAM. (Access is denied because when no policy exists for a protected resource the default decision for OpenAM is to deny all access.)

10.3. Silent GlassFish Policy Agent Installation

When performing a scripted, silent installation, use **agentadmin --install --saveResponse** ***`response-file`*** to create a response file for scripted installation. Then install silently using **agentadmin --install --useResponse** ***`response-file`***.

10.4. Removing GlassFish Policy Agent Software

Shut down the GlassFish domain before you uninstall the policy agent.

```
$ /path/to/glassfish/bin/asadmin stop-domain domain1
    Waiting for the domain to stop ....
    Command stop-domain executed successfully.
```

To remove the J2EE policy agent, use **agentadmin --uninstall**. You must provide the GlassFish configuration directory location, and the instance name.

Uninstall does not remove the agent instance directory, but you can do so manually after removing the agent configuration from GlassFish.

Chapter 11. Installing the JBoss Application Server Policy Agent

This chapter covers installation of the policy agent for JBoss Application Server.

11.1. Before You Install

Make sure OpenAM is installed and running, and that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install JBoss before you install the policy agent, and you must stop the server during installation.

You must install a Java 6 runtime environment, and set the `JAVA_HOME` environment variable.

```
$ echo $JAVA_HOME
/path/to/java1.6
$ which java
/usr/bin/java
```

Download the JBoss policy agent from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Unzip the file in the directory where you plan to install the J2EE policy agent. The agent you install stores its configuration and logs under this directory.

When you unzip the policy agent, you find the following directories under the `j2ee_agents/jboss_v42_agent` directory.

`bin`
: The installation and configuration program, **agentadmin**.

`config`
: Configuration templates used by the **agentadmin** command during installation

`data`
: Not used

`etc`
: Agent web application and configuration templates used during installation

`installer-logs`
: Location for log files written during installation

`lib`
: Shared libraries used by the J2EE policy agent

`locale`
: Property files used by the installation program

`sampleapp`
: Sample application that demonstrates key features of the policy agent. Wait until you have installed the agent to deploy this.

11.2. Installing the JBoss Policy Agent

Complete the following procedures to install the policy agent.

Procedure 11.1. To Create the JBoss Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *`Realm Name`* > Agents > J2EE, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

   Password
   : Password the agent uses to authenticate to OpenAM

   Configuration
   : Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

   Server URL
   : The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

Agent URL
: The URL to the J2EE agent application, such as `http://www.example.com:8080/agentapp`

  In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 11.2. To Create the Password File

1. Create a text file containing only the password.

```
$ echo password > /tmp/pwd.txt
```

2. Protect the password file you create as appropriate for your operating system.

```
$ chmod 400 /tmp/pwd.txt
```

Procedure 11.3. To Install the Policy Agent into JBoss

1. Shut down the JBoss server where you plan to install the agent.

2. Make sure OpenAM is running.

3. Run **agentadmin --install** to install the agent.

```
$ /path/to/j2ee_agents/jboss_v42_agent/bin/agentadmin --install
...
-----------------------------------------------
SUMMARY OF YOUR RESPONSES
-----------------------------------------------
JBoss Server Config Directory : /path/to/jboss/server/default/conf
JBoss Server Home Directory : /path/to/jboss
OpenAM server URL : http://openam.example.com:8080/openam
Agent URL : http://www.example.com:8080/agentapp
Agent Profile name : JBoss Agent
Agent Profile Password file name : /tmp/pwd.txt
Agent permissions gets added to java permissions policy file : false

...
SUMMARY OF AGENT INSTALLATION
-----------------------------
Agent instance name: Agent_001
Agent Bootstrap file location:
/path/to/j2ee_agents/jboss_v42_agent/Agent_001/config/
 OpenSSOAgentBootstrap.properties
Agent Configuration file location
/path/to/j2ee_agents/jboss_v42_agent/Agent_001/config/
```

```
 OpenSSOAgentConfiguration.properties
Agent Audit directory location:
/path/to/j2ee_agents/jboss_v42_agent/Agent_001/logs/audit
Agent Debug directory location:
/path/to/j2ee_agents/jboss_v42_agent/Agent_001/logs/debug


Install log file location:
/path/to/j2ee_agents/jboss_v42_agent/installer-logs/audit/install.log
...
```

Upon successful completion, the installer has updated the JBoss configuration, created a `JBOSS_HOME/bin/setAgentClasspathdefault.sh` script, added the agent web application under `JBOSS_HOME/server/default/deploy/`, and also set up configuration and log directories for the agent.

Note

If the agent is in a different domain than the server, refer to *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

4. Take note of the configuration files and log locations.

   Each agent instance that you install on the system has its own numbered configuration and logs directory. The first agent's configuration and logs are thus located under the directory `j2ee_agents/jboss_v42_agent/Agent_001/`.

   `config/OpenSSOAgentBootstrap.properties`
   : Used to bootstrap the J2EE policy agent, allowing the agent to connect to OpenAM and download its configuration

   `config/OpenSSOAgentConfiguration.properties`
   : Only used if you configured the J2EE policy agent to use local configuration

   `logs/audit/`
   : Operational audit log directory, only used if remote logging to OpenAM is disabled

   `logs/debug/`
   : Debug directory where the debug file resides. Useful in troubleshooting policy agent issues.

5. If your policy agent configuration is not in the top-level realm (/), then you must edit config/OpenSSOAgentBootstrap.properties to identify the sub-realm that has your policy agent configuration. Find com.sun.identity.agents.config.organization.name and change the `/` to

the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

6. To protect a web application, you must add the following filter to the application's `web.xml` configuration, following the opening <web-app> tag. The file for the sample application delivered with the agent is `/path/to/j2ee_agents/jboss_v42_agent/sampleapp/etc/web.xml`.

```
<filter>
 <filter-name>Agent</filter-name>
 <display-name>Agent</display-name>
 <description>OpenAM Policy Agent Filter</description>
<filter-class>com.sun.identity.agents.filter.AmAgentFilter</filter-class>
</filter>
<filter-mapping>
 <filter-name>Agent</filter-name>
 <url-pattern>/*</url-pattern>
 <dispatcher>REQUEST</dispatcher>
 <dispatcher>INCLUDE</dispatcher>
 <dispatcher>FORWARD</dispatcher>
 <dispatcher>ERROR</dispatcher>
</filter-mapping>
```

   You must also add the following security domain specification to the application's `jboss.xml` and `jboss-web.xml` configuration files.

```
<security-domain>java:/jaas/AMRealm</security-domain>
```

Procedure 11.4. To Run JBoss After Agent Installation

1. Render the script to set the agent classpath executable.

```
$ chmod +x $JBOSS_HOME/bin/setAgentClasspathdefault.sh
```

2. Open the `JBOSS_HOME/bin/run.sh` script for editing, and locate the following code block.

```
if [ "x$JBOSS_CLASSPATH" = "x" ]; then
    JBOSS_CLASSPATH="$JBOSS_BOOT_CLASSPATH"
else
    JBOSS_CLASSPATH="$JBOSS_CLASSPATH:$JBOSS_BOOT_CLASSPATH"
fi
if [ "x$JAVAC_JAR_FILE" != "x" ]; then
    JBOSS_CLASSPATH="$JBOSS_CLASSPATH:$JAVAC_JAR_FILE"
fi
```

3. Edit the `JBOSS_HOME/bin/run.sh` script to set the classpath needed for the agent, by adding these lines after the code block you located in the previous step.

```
if [ -r "setAgentClasspathdefault.sh" ]; then
    . $JBOSS_HOME/bin/setAgentClasspathdefault.sh
fi
```

4. Start the JBoss server where you installed the agent.

```
$ cd $JBOSS_HOME ; ./bin/run.sh -b 0.0.0.0
...
16:30:31,172 INFO  [ServerImpl] JBoss ... Started in 1m:44s:759ms
```

5. If you have a policy configured, you can test your policy agent. For example, try to browse to a resource that your policy agent protects. You should be redirected to OpenAM to authenticate, for example as user `demo`, password `changeit`. After you authenticate, OpenAM then redirects you back to the resource you tried to access.

11.3. Silent JBoss Policy Agent Installation

When performing a scripted, silent installation, use **agentadmin --install --saveResponse** ***`response-file`*** to create a response file for scripted installation. Then install silently using **agentadmin --install --useResponse** ***`response-file`***.

11.4. Removing JBoss Policy Agent Software

Shut down the JBoss server before you uninstall the policy agent.

To remove the J2EE policy agent, use **agentadmin --uninstall**. You must provide the JBoss configuration directory location.

Uninstall does not remove the agent instance directory, but you can do so manually after removing the agent configuration from JBoss.

Chapter 12. Installing the Jetty Server Policy Agent

This chapter covers installation of the policy agent for Jetty.

12.1. Before You Install

Make sure OpenAM is installed and running, and that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install Jetty before you install the policy agent, and you must stop the server during installation.

You must install a Java 6 runtime environment, and set the `JAVA_HOME` environment variable.

```
$ echo $JAVA_HOME
/path/to/java1.6
$ which java
/usr/bin/java
```

Download the Jetty policy agent from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Note

Command line examples in this chapter show Jetty accessed remotely. If you are following the examples and have issues accessing Jetty remotely, you might have to change the test filter settings in `/path/to/jetty/webapps/test/WEB-INF/web.xml`.

```
<filter>
 <filter-name>TestFilter</filter-name>
 <filter-class>com.acme.TestFilter</filter-class>
 <init-param>
  <param-name>remote</param-name>
  <param-value>true</param-value> <!-- default: false -->
 </init-param>
</filter>
```

Unzip the file in the directory where you plan to install the J2EE policy agent. The agent you install stores its configuration and logs under this directory.

When you unzip the policy agent, you find the following directories under the `j2ee_agents/jetty_v61_agent` directory.

`bin`
: The installation and configuration program, **agentadmin**.

`config`
: Configuration templates used by the **agentadmin** command during installation

`data`
: Not used

`etc`
: Agent web application used during installation

`installer-logs`
: Location for log files written during installation

`lib`
: Shared libraries used by the J2EE policy agent

`locale`
: Property files used by the installation program

`sampleapp`
: Sample application that demonstrates key features of the policy agent. Wait until you have installed the agent to deploy this.

12.2. Installing the Jetty Policy Agent

Complete the following procedures to install the policy agent.

Procedure 12.1. To Create the Jetty Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *`Realm Name`* > Agents > J2EE, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

Password
: Password the agent uses to authenticate to OpenAM

Configuration
: Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

Server URL
: The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

  In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

Agent URL
: The URL to the J2EE agent application, such as `http://www.example.com:8080/agentapp`

  In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 12.2. To Create the Password File

1. Create a text file containing only the password.

   ```
   $ echo password > /tmp/pwd.txt
   ```

2. Protect the password file you create as appropriate for your operating system.

   ```
   $ chmod 400 /tmp/pwd.txt
   ```

Procedure 12.3. To Install the Policy Agent into Jetty

1. Shut down the Jetty server where you plan to install the agent.
2. Make sure OpenAM is running.
3. Run **agentadmin --install** to install the agent.

   ```
   $ /path/to/j2ee_agents/jetty_v61_agent/bin/agentadmin --install
   ...
   -----------------------------------------------
   SUMMARY OF YOUR RESPONSES
   -----------------------------------------------
   Jetty Server Config Directory : /path/to/jetty/etc
   OpenAM server URL : http://openam.example.com:8080/openam
   ```

```
Jetty installation directory. : /path/to/jetty
Agent URL : http://www.example.com:8080/agentapp
Agent Profile name : Jetty Agent
Agent Profile Password file name : /tmp/pwd.txt

...
SUMMARY OF AGENT INSTALLATION
-----------------------------
Agent instance name: Agent_001
Agent Bootstrap file location:
/path/to/j2ee_agents/jetty_v61_agent/Agent_001/config/
 OpenSSOAgentBootstrap.properties
Agent Configuration file location
/path/to/j2ee_agents/jetty_v61_agent/Agent_001/config/
 OpenSSOAgentConfiguration.properties
Agent Audit directory location:
/path/to/j2ee_agents/jetty_v61_agent/Agent_001/logs/audit
Agent Debug directory location:
/path/to/j2ee_agents/jetty_v61_agent/Agent_001/logs/debug


Install log file location:
/path/to/j2ee_agents/jetty_v61_agent/installer-logs/audit/install.log
...
```

Upon successful completion, the installer has updated Jetty's `start.jar` to reference the agent, set up the agent web application, and also set up configuration and log directories for the agent.

Note

If the agent is in a different domain than the server, refer to *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

4. Take note of the configuration files and log locations.

   Each agent instance that you install on the system has its own numbered configuration and logs directory. The first agent's configuration and logs are thus located under the directory `j2ee_agents/jetty_v61_agent/Agent_001/`.

   `config/OpenSSOAgentBootstrap.properties`
   : Used to bootstrap the J2EE policy agent, allowing the agent to connect to OpenAM and download its configuration

   `config/OpenSSOAgentConfiguration.properties`
   : Only used if you configured the J2EE policy agent to use local configuration

   `logs/audit/`
   : Operational audit log directory, only used if remote logging to OpenAM is disabled

`logs/debug/`
: Debug directory where the `debug.out` debug file resides. Useful in troubleshooting policy agent issues.

5. If your policy agent configuration is not in the top-level realm (/), then you must edit config/OpenSSOAgentBootstrap.properties to identify the sub-realm that has your policy agent configuration. Find com.sun.identity.agents.config.organization.name and change the `/` to the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

6. To protect a web application, you must add the following filter to the application's `web.xml` configuration, following the opening <web-app> tag. The file for the sample application delivered with the agent is `/path/to/j2ee_agents/jetty_v61_agent/sampleapp/etc/web.xml`.

```
<filter>
 <filter-name>Agent</filter-name>
 <display-name>Agent</display-name>
 <description>OpenAM Policy Agent Filter</description>
<filter-class>com.sun.identity.agents.filter.AmAgentFilter</filter-class>
</filter>
<filter-mapping>
 <filter-name>Agent</filter-name>
 <url-pattern>/*</url-pattern>
 <dispatcher>REQUEST</dispatcher>
 <dispatcher>INCLUDE</dispatcher>
 <dispatcher>FORWARD</dispatcher>
 <dispatcher>ERROR</dispatcher>
</filter-mapping>
```

7. Start the Jetty server where you installed the agent.

```
$ cd /path/to/jetty ; java -jar start.jar
...
2011-09-15 12:49:55.469:INFO::Extract file:/path/to/jetty/webapps/agentapp.war
...
2011-09-15 12:50:14.163:INFO::Started SelectChannelConnector@0.0.0.0:8080
```

8. If you have a policy configured, you can test your policy agent. For example, try to browse to a resource that your policy agent protects. You should be redirected to OpenAM to authenticate, for example as user `demo`, password `changeit`. After you authenticate, OpenAM then redirects you back to the resource you tried to access.

12.3. Silent Jetty Policy Agent Installation

When performing a scripted, silent installation, use **agentadmin --install --saveResponse** ***`response-file`*** to create a response file for scripted installation. Then install silently using **agentadmin --install --useResponse** ***`response-file`***.

12.4. Removing Jetty Policy Agent Software

Shut down the Jetty server before you uninstall the policy agent.

To remove the J2EE policy agent, use **agentadmin --uninstall**. You must provide the Jetty configuration directory location.

Uninstall does not remove the agent instance directory, but you can do so manually after removing the agent configuration from Jetty.

Chapter 13. Installing the IBM WebSphere Policy Agent

This chapter covers installation of the policy agent for IBM WebSphere.

13.1. Before You Install

Make sure OpenAM is installed and running, and that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install WebSphere before you install the policy agent, and you must stop the server during installation.

You must install a Java 6 runtime environment, and set the `JAVA_HOME` environment variable.

```
$ echo $JAVA_HOME
/path/to/java1.6
$ which java
/usr/bin/java
```

If you are using IBM Java, see Procedure 13.1, “To Install With IBM Java”.

Download the WebSphere policy agent from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Unzip the file in the directory where you plan to install the J2EE policy agent. The agent you install stores its configuration and logs under this directory.

When you unzip the policy agent, you find the following directories under the `j2ee_agents/websphere_v61_agent` directory.

`bin`
: The installation and configuration program, **agentadmin**.

`config`
: Configuration templates used by the **agentadmin** command during installation

`data`
: Not used

`etc`
: Agent web application that handles notifications and Cross Domain SSO

`installer-logs`
: Location for log files written during installation

`lib`
: Shared libraries used by the J2EE policy agent

`locale`
: Property files used by the installation program

`sampleapp`
: Sample application that demonstrates key features of the policy agent. Wait until you have installed the agent to deploy this.

Procedure 13.1. To Install With IBM Java

The WebSphere policy agent runs with IBM Java. In order to install the policy agent using IBM Java on platforms other than AIX, you must first change the **agentadmin** script to use IBMJCE.

1. Open the file, `bin/agentadmin` (`bin/agentadmin.bat` on Windows), for editing.

2. Edit the line specifying `AGENT_OPTS` on platforms other than AIX.

   ```
   AGENT_OPTS="-DamKeyGenDescriptor.provider=IBMJCE \
    -DamCryptoDescriptor.provider=IBMJCE -DamRandomGenProvider=IBMJCE"
   ```

3. Edit the last line to include the IBMJCE settings before the classpath is set.

   ```
   $JAVA_VM \
    -DamCryptoDescriptor.provider=IBMJCE -DamKeyGenDescriptor.provider=IBMJCE \
    -classpath "$AGENT_CLASSPATH" $AGENT_OPTS \
    com.sun.identity.install.tools.launch.AdminToolLauncher $*
   ```

4. Save your work.

   You can now install the WebSphere policy agent with IBM Java as described in Section 13.2, “Installing the WebSphere Policy Agent”.

13.2. Installing the WebSphere Policy Agent

Complete the following procedures to install the policy agent.

Procedure 13.2. To Create the WebSphere Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *Realm Name* > Agents > J2EE, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

   Password
   : Password the agent uses to authenticate to OpenAM

   Configuration
   : Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

   Server URL
   : The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

     In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

   Agent URL
   : The URL to the J2EE agent application, such as `http://www.example.com:8080/agentapp`

     In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 13.3. To Create the Password File

1. Create a text file containing only the password.

   ```
   $ echo password > /tmp/pwd.txt
   ```

2. Protect the password file you create as appropriate for your operating system.

   ```
   $ chmod 400 /tmp/pwd.txt
   ```

Procedure 13.4. To Install the Policy Agent into WebSphere

1. Shut down the WebSphere server where you plan to install the agent.

2. Make sure OpenAM is running.

3. Run **agentadmin --install** to install the agent.

   ```
   $ /path/to/j2ee_agents/websphere_v61_agent/bin/agentadmin --install
   ...
   -----------------------------------------------
   SUMMARY OF YOUR RESPONSES
   -----------------------------------------------
   Instance Config Directory :
   /path/to/WebSphere/AppServer/profiles/AppSrv01/config/cells/wwwNode01Cell/
    nodes/wwwNode01/servers/server1

   Instance Server name : server1
   WebSphere Install Root Directory : /path/to/WebSphere/AppServer
   OpenAM server URL : http://openam.example.com:8080/openam
   Agent URL : http://www.example.com:9080/agentapp
   Agent Profile name : WebSphere Agent
   Agent Profile Password file name : /tmp/pwd.txt

   ...
   SUMMARY OF AGENT INSTALLATION
   -----------------------------
   Agent instance name: Agent_001
   Agent Bootstrap file location:
   /path/to/j2ee_agents/websphere_v61_agent/Agent_001/config/
    OpenSSOAgentBootstrap.properties
   Agent Configuration file location
   /path/to/j2ee_agents/websphere_v61_agent/Agent_001/config/
    OpenSSOAgentConfiguration.properties
   Agent Audit directory location:
   /path/to/j2ee_agents/websphere_v61_agent/Agent_001/logs/audit
   Agent Debug directory location:
   /path/to/j2ee_agents/websphere_v61_agent/Agent_001/logs/debug


   Install log file location:
   /path/to/j2ee_agents/websphere_v61_agent/installer-logs/audit/install.log
   ...
   ```

   Upon successful completion, the installer has updated the WebSphere configuration, copied the agent libraries to WebSphere's external library directory, and also set up configuration and log directories for the agent.

   ## Note

   If the agent is in a different domain than the server, refer to *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

4. Take note of the configuration files and log locations.

Each agent instance that you install on the system has its own numbered configuration and logs directory. The first agent's configuration and logs are thus located under the directory `j2ee_agents/websphere_v61_agent/Agent_001/`.

`config/OpenSSOAgentBootstrap.properties`
: Used to bootstrap the J2EE policy agent, allowing the agent to connect to OpenAM and download its configuration

`config/OpenSSOAgentConfiguration.properties`
: Only used if you configured the J2EE policy agent to use local configuration

`logs/audit/`
: Operational audit log directory, only used if remote logging to OpenAM is disabled

`logs/debug/`
: Debug directory where the debug file resides. Useful in troubleshooting policy agent issues.

5. If your policy agent configuration is not in the top-level realm (/), then you must edit config/OpenSSOAgentBootstrap.properties to identify the sub-realm that has your policy agent configuration. Find com.sun.identity.agents.config.organization.name and change the `/` to the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

6. Restart the WebSphere server.

Procedure 13.5. To Protect Applications After Agent Installation

1. Deploy the `/path/to/j2ee_agents/websphere_v61_agent/etc/agentapp.war` agent application in WebSphere.

2. For each web application to protect, add the following filter to the application's `web.xml` configuration, following the opening <web-app> tag. The file for the sample application delivered with the agent is `/path/to/j2ee_agents/websphere_v61_agent/sampleapp/etc/web.xml`.

```
<filter>
 <filter-name>Agent</filter-name>
 <display-name>Agent</display-name>
 <description>OpenAM Policy Agent Filter</description>
<filter-class>com.sun.identity.agents.filter.AmAgentFilter</filter-class>
</filter>
<filter-mapping>
 <filter-name>Agent</filter-name>
 <url-pattern>/*</url-pattern>
```

```
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>ERROR</dispatcher>
 </filter-mapping>
```

You might also have to update additional configuration files. See the sample application located under `/path/to/j2ee_agents/websphere_v61_agent/sampleapp` for examples.

3. If you have a policy configured, you can test your policy agent. For example, try to browse to a resource that your policy agent protects. You should be redirected to OpenAM to authenticate, for example as user `demo`, password `changeit`. After you authenticate, OpenAM then redirects you back to the resource you tried to access.

13.3. Silent WebSphere Policy Agent Installation

When performing a scripted, silent installation, use **agentadmin --install --saveResponse** ***`response-file`*** to create a response file for scripted installation. Then install silently using **agentadmin --install --useResponse** ***`response-file`***.

13.4. Removing WebSphere Policy Agent Software

Shut down the WebSphere server before you uninstall the policy agent.

To remove the J2EE policy agent, use **agentadmin --uninstall**. You must provide the WebSphere configuration directory location.

Uninstall does not remove the agent instance directory, but you can do so manually after removing the agent configuration from WebSphere.

13.5. Notes About WebSphere Network Deployment

When using WebSphere Application Server Network Deployment, you must install policy agents on the Deployment Manager, on each Node Agent, and on each Application Server. Installation requires that you stop and then restart the Deployment Manager, each Node Agent, and each Application Server in the Network Deployment.

Before installation, synchronize each server configuration with the profile saved by the Deployment Manager using the **syncNode** command. After agent installation, copy the server configuration for each node, stored in `server.xml`, to the corresponding Deployment Manager profile. After you have synchronized the configurations, you must restart the Deployment Manager for the Network Deployment.

Chapter 14. Installing the Oracle WebLogic Policy Agent

This chapter covers installation of the policy agent for Oracle WebLogic.

14.1. Before You Install

Make sure OpenAM is installed and running, and that you can contact OpenAM from the system running the policy agent. Next, create a profile for your policy agent as described in the *Administration Guide* section on *Creating Agent Profiles*. To protect resources with the agent also create at least one policy as described in the section on *Configuring Policies*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources, in order to test your policy agent after installation.

You must install WebLogic before you install the policy agent, and you must stop the server during installation.

You must install a Java 6 runtime environment, and set the `JAVA_HOME` environment variable.

```
$ echo $JAVA_HOME
/path/to/java1.6
$ which java
/usr/bin/java
```

Download the WebLogic policy agent from the download page. Also verify the checksum of the file you download against the checksum posted on the download page.

Unzip the file in the directory where you plan to install the J2EE policy agent. The agent you install stores its configuration and logs under this directory.

When you unzip the policy agent, you find the following directories under the `j2ee_agents/weblogic_v10_agent` directory.

`bin`
: The installation and configuration program, **agentadmin**.

`config`
: Configuration templates used by the **agentadmin** command during installation

`data`
: Not used

`etc`
: Agent web application and startup configuration

`installer-logs`
: Location for log files written during installation

`lib`
: Shared libraries used by the J2EE policy agent

`locale`
: Property files used by the installation program

`sampleapp`
: Sample application that demonstrates key features of the policy agent. Wait until you have installed the agent to deploy this.

14.2. Installing the WebLogic Policy Agent

Complete the following procedures to install the policy agent.

Procedure 14.1. To Create the WebLogic Agent Profile

Regardless of whether you store configurations centrally in OpenAM or locally with your agents, the agent requires a profile so that it can connect to and communicate with OpenAM.

1. In the OpenAM console, browse to Access Control > *`Realm Name`* > Agents > J2EE, and then click the New... button in the Agent table.

2. Complete the web form using the following hints.

   Name
   : The name for the agent profile used when you install the agent

   Password
   : Password the agent uses to authenticate to OpenAM

   Configuration
   : Centralized configurations are stored in the OpenAM configuration store. You can manage the centralized configuration through the OpenAM console. Local configurations are stored in a file alongside the agent.

   Server URL
   : The full URL to an OpenAM instance, or if OpenAM is deployed in a site configuration (behind a load balancer) then the site URL

     In centralized configuration mode, the Server URL is used to populate the agent profile for services such as Login, Logout, Naming, and Cross Domain SSO.

Agent URL
: The URL to the J2EE agent application, such as `http://www.example.com:8080/agentapp`

  In centralized configuration mode, the Agent URL is used to populate the Agent Profile for services such as notifications.

Procedure 14.2. To Create the Password File

1. Create a text file containing only the password.

   ```
   $ echo password > /tmp/pwd.txt
   ```

2. Protect the password file you create as appropriate for your operating system.

   ```
   $ chmod 400 /tmp/pwd.txt
   ```

Procedure 14.3. To Install the Policy Agent into WebLogic

1. Shut down the WebLogic server where you plan to install the agent.

2. Make sure OpenAM is running.

3. Run **agentadmin --install** to install the agent.

   ```
   $ /path/to/j2ee_agents/weblogic_v10_agent/bin/agentadmin --install
   ...
   -----------------------------------------------
   SUMMARY OF YOUR RESPONSES
   -----------------------------------------------
   Startup script location :
   /path/to/domain/mydomain/bin/startWebLogic.sh
   WebLogic Server instance name : AdminServer
   WebLogic home directory : /path/to/wlserver
   OpenAM server URL : http://openam.example.com:8080/openam
   Agent URL : http://www.example.com:7001/agentapp
   Agent Profile name : WebLogic Agent
   Agent Profile Password file name : /tmp/pwd.txt

   ...
   SUMMARY OF AGENT INSTALLATION
   -----------------------------
   Agent instance name: Agent_001
   Agent Bootstrap file location:
   /path/to/j2ee_agents/weblogic_v10_agent/Agent_001/config/
    OpenSSOAgentBootstrap.properties
   Agent Configuration file location
   /path/to/j2ee_agents/weblogic_v10_agent/Agent_001/config/
    OpenSSOAgentConfiguration.properties
   Agent Audit directory location:
   /path/to/j2ee_agents/weblogic_v10_agent/Agent_001/logs/audit
   ```

```
Agent Debug directory location:
/path/to/j2ee_agents/weblogic_v10_agent/Agent_001/logs/debug


Install log file location:
/path/to/j2ee_agents/weblogic_v10_agent/installer-logs/audit/install.log
...
```

Upon successful completion, the installer has updated the WebLogic configuration, copied the agent libraries to WebLogic's library directory, and also set up configuration and log directories for the agent.

> **Note**
>
> If the agent is in a different domain than the server, refer to *Administration Guide* procedure, *Configuring Cross-Domain Single Sign On*.

4. Take note of the configuration files and log locations.

   Each agent instance that you install on the system has its own numbered configuration and logs directory. The first agent's configuration and logs are thus located under the directory `j2ee_agents/weblogic_v10_agent/Agent_001/`.

   `config/OpenSSOAgentBootstrap.properties`
   : Used to bootstrap the J2EE policy agent, allowing the agent to connect to OpenAM and download its configuration

   `config/OpenSSOAgentConfiguration.properties`
   : Only used if you configured the J2EE policy agent to use local configuration

   `logs/audit/`
   : Operational audit log directory, only used if remote logging to OpenAM is disabled

   `logs/debug/`
   : Debug directory where the debug file resides. Useful in troubleshooting policy agent issues.

5. If your policy agent configuration is not in the top-level realm (/), then you must edit config/OpenSSOAgentBootstrap.properties to identify the sub-realm that has your policy agent configuration. Find com.sun.identity.agents.config.organization.name and change the / to the path to your policy agent profile. This allows the policy agent to properly identify itself to the OpenAM server.

6. Restart the WebLogic server.

First you must get the environment settings from the script installed with the policy agent, then run the startup script.

```
$ . ./domain/mydomain/bin/setAgentEnv_AdminServer.sh
$ ./domain/mydomain/bin/startWebLogic.sh
```

7. Configure shutdown classes for the environment.

    a. In WebLogic console, browse to Environment > Startup & Shutdown Classes.

    b. Click Lock & Edit.

    c. Click New.

    d. Select the Shutdown Class option, and then click Next.

    e. Provide the Name `Agent`, and the Class Name `org.forgerock.agents.weblogic.v10.lifecycle.ShutdownListener`.

    f. Select the appropriate targets to call the shutdown class once per Java Virtual Machine, and then click Finish.

    g. Click Activate Changes.

Procedure 14.4. To Protect Applications After Agent Installation

1. Deploy the `/path/to/j2ee_agents/weblogic_v10_agent/etc/agentapp.war` agent application in WebLogic.

2. For each web application to protect, add the following filter to the application's `web.xml` configuration, following the opening <web-app> tag. The file for the sample application delivered with the agent is `/path/to/j2ee_agents/weblogic_v10_agent/sampleapp/etc/web.xml`.

```
<filter>
 <filter-name>Agent</filter-name>
 <display-name>Agent</display-name>
 <description>OpenAM Policy Agent Filter</description>
<filter-class>com.sun.identity.agents.filter.AmAgentFilter</filter-class>
</filter>
<filter-mapping>
 <filter-name>Agent</filter-name>
 <url-pattern>/*</url-pattern>
 <dispatcher>REQUEST</dispatcher>
 <dispatcher>INCLUDE</dispatcher>
 <dispatcher>FORWARD</dispatcher>
 <dispatcher>ERROR</dispatcher>
</filter-mapping>
```

You might also have to update additional configuration files. See the sample application located under `/path/to/j2ee_agents/weblogic_v10_agent/sampleapp` for examples.

3. If you have a policy configured, you can test your policy agent. For example, try to browse to a resource that your policy agent protects. You should be redirected to OpenAM to authenticate, for example as user `demo`, password `changeit`. After you authenticate, OpenAM then redirects you back to the resource you tried to access.

14.3. Silent WebLogic Policy Agent Installation

When performing a scripted, silent installation, use **agentadmin --install --saveResponse** ***`response-file`*** to create a response file for scripted installation. Then install silently using **agentadmin --install --useResponse** ***`response-file`***.

14.4. Post Installation of WebLogic Policy Agent

After installing WebLogic, some configuration is required before the policy agent will work.

Procedure 14.5. To Configure the WebLogic Policy Agent

WebLogic is unique in that it requires additional configuration after the installation is complete.

1. Go to the WebLogic Server Administratotion Console and login.
2. Click `Security realms`.
3. Click the name of the realm to use for OpenAM.
4. Click `Providers` > `Authentication`.
5. Click `Lock & Edit` > `New`.
6. Enter the desired type in `Type as AgentAuthenticator`, provide a name, and click `OK`.
7. Click on the name of the agent authenticator you just created.
8. Use `OPTIONAL` for the control flag and save.
9. Click on `Providers` to display the Authentication Providers Table.
10. Click on `Default Authenticator`, use `OPTIONAL` for the control flag, and save.

11. Activate the changes once the default authenticator is done saving.

    You will need to restart the WebLogic Server to implement the changes.

14.5. Removing WebLogic Policy Agent Software

Shut down the WebLogic server before you uninstall the policy agent.

To remove the J2EE policy agent, use **agentadmin --uninstall**. You must provide the WebLogic configuration directory location.

Uninstall does not remove the agent instance directory, but you can do so manually after removing the agent configuration from WebLogic.

Chapter 15. Troubleshooting

This chapter offers solutions to issues during installation of OpenAM policy agents.

Solutions to Common Issues

This section offers solutions to common problems when installing OpenAM policy agents.

Q: I am trying to install the policy agent on SELinux and I am getting error messages after installation. What happened?

A: SELinux must be properly configured to connect the web policy agent and OpenAM nodes. Either re-configure SELinux or disable it, then reinstall the policy agent.

Q: My Apache HTTPD server is not using port 80. But when I install the web policy agent it defaults to port 80. How do I fix this?

A: You probably set `ServerName` in Apache HTTPD's configuration to the host name, but did not specify the port number.

Instead you must set both the host name and port number for `ServerName` in Apache HTTPD's configuration. For example, if you have Apache HTTPD configured to listen on port 8080, then set `ServerName` appropriately as in the following excerpt.

```
<VirtualHost *:8080>
ServerName www.localhost.example:8080
```

Q: I am trying to install the WebSphere policy agent on Linux. The system has IBM Java. When I run **agentadmin --install**, the script fails to encrypt the password from the password file, ending with this message:

```
ERROR: An unknown error has occurred (null). Please try again.
```

What should I do?

A: You must edit **agentadmin** to use IBMJCE, and then try again.

See *To Install With IBM Java*.

Appendix A. Web Agent Configuration Properties

Web Agents use the following configuration properties. Bootstrap properties are always configured locally, whereas other agent configuration properties are either configured centrally in OpenAM or locally using the agent properties file.

A.1. Bootstrap Configuration Properties

These properties are set in `config/OpenSSOAgentBootstrap.properties`.

`com.forgerock.agents.ext.url.validation.disable`

The bootstrap configuration property, `com.forgerock.agents.ext.url.validation.disable`, lets you configure naming URL validation during the initial bootstrap phase when the policy agent reads its configuration. When URL validation is fully disabled the policy agent does not need to connect to OpenAM during the bootstrap phase.

If you leave URL validation disabled, make sure that the URLs in the policy agent bootstrap configuration file are valid and correct. As the policy agent performs no further validation after the bootstrap phase, incorrect naming URLs can cause the agent to crash.

To enable full URL validation, set the property as shown:

```
com.forgerock.agents.ext.url.validation.disable = 0
```

This property can take the following values.

0

Fully validate naming URLs specified by using the `com.sun.identity.agents.config.naming.url` property. The web policy agent logs into and logs out of OpenAM to check that a naming URL is valid.

1

Check that naming URLs are valid by performing an HTTP GET, which should receive an HTTP 200 response.

2 (Default)

Disable all naming URL validation.

When naming URL validation is enabled, then set the following properties.

- `com.forgerock.agents.ext.url.validation.poll.interval`
- `com.forgerock.agents.ext.url.validation.scan.interval`
- `com.sun.identity.agents.config.connect.timeout`
- `com.sun.identity.agents.config.receive.timeout`

`com.forgerock.agents.ext.url.validation.poll.interval`
: Set this property to the number of seconds to wait before contacting OpenAM to check that naming URLs are valid. This property should be set to a larger value than `com.forgerock.agents.ext.url.validation.scan.interval`. For example, `com.forgerock.agents.ext.url.validation.poll.interval = 10`.

`com.forgerock.agents.ext.url.validation.scan.interval`
: Set this property to the number of seconds to wait before rescanning the list of valid naming URLs, and persisting the list to a temporary file. This property should be set to a smaller value than `com.forgerock.agents.ext.url.validation.poll.interval`. For example, `com.forgerock.agents.ext.url.validation.scan.interval = 3`.

`com.sun.identity.agents.config.certdb.password`
: When SSL is configured, set this to the password for the certificate database.

`com.sun.identity.agents.config.certdb.prefix`
: When SSL is configured, set this property if the certificate databases in the directory specified by `com.sun.identity.agents.config.sslcert.dir` have a prefix.

`com.sun.identity.agents.config.certificate.alias`
: When SSL is configured, set this to the alias of the certificate used to authenticate.

`com.sun.identity.agents.config.connect.timeout`
: Set this to the number of milliseconds to keep the socket connection open before timing out. If you have the web policy agent perform naming URL validation, then set this property to a reasonable value such as 2000 (2 seconds). The default value is 0 which implies no timeout.

`com.sun.identity.agents.config.debug.file`
: Set this to the full path of the agent's debug log file.

`com.sun.identity.agents.config.debug.level`
: Default is `Error`. Increase to `Message` or even `All` for fine-grained detail.

  Set the level in the configuration file by module using the format `*module*[:*level*][,*module*[:*level*]]*`, where *`module`* is one of

`AuthService`, `NamingService`, `PolicyService`, `SessionService`, `PolicyEngine`, `ServiceEngine`, `Notification`, `PolicyAgent`, `RemoteLog`, or `all`, and *`level`* is one of the following.

- `0`: Disable logging from specified module

  At this level the agent nevertheless logs messages having the level value `always`.

- `1`: Log error messages
- `2`: Log warning and error messages
- `3`: Log info, warning, and error messages
- `4`: Log debug, info, warning, and error messages
- `5`: Like level 4, but with even more debugging messages

When you omit *`level`*, the agent uses the default level, which is the level associated with the `all` module.

The following example used in the local configuration sets the log overall level to debug for all messages.

```
com.sun.identity.agents.config.debug.level=all:4
```

`com.sun.identity.agents.config.forward.proxy.host`
: When OpenAM and the agent communicate through a web proxy server configured in forward proxy mode, set this to the proxy server host name.

`com.sun.identity.agents.config.forward.proxy.password`
: When OpenAM and the agent communicate through a web proxy server configured in forward proxy mode and the proxy server has the agent authenticate using Basic Authentication, set this to the agent's password.

`com.sun.identity.agents.config.forward.proxy.port`
: When OpenAM and the agent communicate through a web proxy server configured in forward proxy mode, set this to the proxy server port number.

`com.sun.identity.agents.config.forward.proxy.user`
: When OpenAM and the agent communicate through a web proxy server configured in forward proxy mode and the proxy server has the agent authenticate using Basic Authentication, set this to the agent's user name.

`com.sun.identity.agents.config.key`
: Set this to the encryption key used to encrypt the agent profile password.

`com.sun.identity.agents.config.local.logfile`
: Set this to the full path for agent's audit log file.

`com.sun.identity.agents.config.naming.url`
: Set this to the naming service URL(s) used for naming lookups in OpenAM. Separate multiple URLs with single space characters.

`com.sun.identity.agents.config.organization.name`
: Set this to the realm name where the agent authenticates to OpenAM.

`com.sun.identity.agents.config.password`
: Set this to the encrypted version of the password for the agent authenticator. Use the command **./agentadmin --encrypt** ***`agentInstance`*** ***`passwordFile`*** to get the encrypted version.

`com.sun.identity.agents.config.profilename`
: Set this to the agent profile name.

`com.sun.identity.agents.config.receive.timeout`
: Set this to the number of milliseconds to wait for a response from OpenAM before timing out and dropping the connection. If you have the web policy agent perform naming URL validation, then set this property to a reasonable value such as 2000 (2 seconds). The default value is 0 which implies no timeout.

`com.sun.identity.agents.config.sslcert.dir`
: When SSL is configured, set this to the directory containing SSL certificate databases.

`com.sun.identity.agents.config.tcp.nodelay.enable`
: Set to `true` to enable the socket option `TCP_NODELAY`. Default is `false`.

`com.sun.identity.agents.config.trust.server.certs`
: When SSL is configured, set to `false` to trust the OpenAM SSL certificate only if the certificate is found to be correct and valid. Default is `true`.

`com.sun.identity.agents.config.username`
: Set this to the user name of the agent authenticator.

A.2. Agent Configuration Properties

These properties are set in `config/OpenSSOAgentConfiguration.properties` if your agent uses local configuration. If your agent uses centralized configuration, the properties are set in OpenAM.

`com.forgerock.agents.cache_control_header.enable`
: Set this property to `true` to enable use of Cache-Control headers that prevent proxies from caching resources accessed by unauthenticated users. Default: `false`.

`com.forgerock.agents.cdsso.disable.redirect.on_post`

Set this property to `true` to disable the HTTP 302 redirect after the LARES POST. By default, the policy agent does an HTTP redirect after processing the LARES POST message. Default: `false`.

This property applies only to Apache HTTPD 2.2 and 2.4, Lotus Domino, and Varnish Cache policy agents. Other policy agents do not redirect after processing the LARES POST message.

`com.forgerock.agents.conditional.login.url`

To conditionally redirect users based on the incoming request URL, set this property.

This takes the incoming request URL to match, a vertical bar (`|`), and then a comma-separated list of URLs to which to redirect incoming users.

If the string before the vertical bar matches an incoming request URL, then the policy agent uses the list of URLs to determine how to redirect the user-agent. If the global property FQDN Check (`com.sun.identity.agents.config.fqdn.check.enable`) is enabled for the policy agent, then the policy agent iterates through the list until it finds an appropriate redirect URL that matches the FQDN check. Otherwise, the policy agent redirects the user-agent to the first URL in the list.

Examples: `com.forgerock.agents.conditional.login.url[0]= login.example.com/|http://openam1.example.com/openam/UI/Login, http://openam2.example.com/openam/UI/Login, com.forgerock.agents.conditional.login.url[1]= login.example.com| http://openam3.example.com/openam/UI/Login, http:// openam4.example.com/openam/UI/Login`

`com.forgerock.agents.config.notenforced.ip.handler`

As of version 3.0.4, web policy agents with this property set to `cidr` can use IPv4 netmasks and IP ranges instead of wildcards as values for `com.sun.identity.agents.config.notenforced.ip` addresses. Version 3.0.5 adds support for IPv6, including the IPv6 loopback address, `::1`.

When the parameter is defined, wildcards are ignored in `com.sun.identity.agents.config.notenforced.ip` settings. Instead, you can use settings such as those shown in the following examples.

Netmask Example

To disable policy agent enforcement for addresses in 192.168.1.1 to 192.168.1.255, use the following setting.

```
com.sun.identity.agents.config.notenforced.ip = 192.168.1.1/24
```

The following example shows a configuration using IPv6.

```
com.sun.identity.agents.config.notenforced.ip = 2001:5c0:9168:0:0:0:0:2/128
```

Currently the policy agent stops evaluating properties after reaching an invalid netmask in the list.

IP Range Example

To disable policy agent enforcement for addresses between 192.168.1.1 to 192.168.4.3 inclusive, use the following setting.

```
com.sun.identity.agents.config.notenforced.ip = 192.168.1.1-192.168.4.3
```

The following example shows a configuration using IPv6.

```
com.sun.identity.agents.config.notenforced.ip = 2001:5c0:9168:0:0:0:0:1-2001:5c0:9168:0:0:0:0:2
```

`com.forgerock.agents.config.pdpuri.prefix`

If you run multiple web servers with policy agents behind a load balancer that directs traffic based on the request URI, and you need to preserve POST data, then set this property.

By default, policy agents use a dummy URL for POST data preservation, `http://`*`agent.host`*`:`*`port`*`/dummypost/sunpostpreserve`, to handle POST data across redirects to and from OpenAM. When you set this property, the policy agent prefixes the property value to the dummy URL path. In other words, when you set `com.forgerock.agents.config.pdpuri.prefix = app1`, the policy agent uses the dummy URL, `http://`*`agent.host`*`:`*`port`*`/app1/dummypost/sunpostpreserve`.

Next, use the prefix you set when you define load balancer URI rules. This ensures that clients end up being redirected to the policy agent that preserved the POST data.

`com.sun.identity.agents.config.access.denied.url`

The URL of the customized access denied page. If no value is specified (default), then the agent returns an HTTP status of 403 (Forbidden).

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > Resources Access Denied URL.

`com.sun.identity.agents.config.agent.logout.url`

List of application logout URLs, such as `http://www.example.com/logout.html`. The user is logged out of the OpenAM session when these URLs are accessed. When using this property, specify a value for the Logout Redirect URL property.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > OpenAM Services > Logout URL List.

`com.sun.identity.agents.config.agenturi.prefix`
The default value is *`agent-root-URL`*`/amagent`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > Agent Deployment URI Prefix.

`com.sun.identity.agents.config.anonymous.user.enable`
Enable or disable REMOTE_USER processing for anonymous users.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Miscellaneous > Anonymous User.

`com.sun.identity.agents.config.anonymous.user.id`
User ID of unauthenticated users. Default: `anonymous`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Miscellaneous > Anonymous User Default Value.

`com.sun.identity.agents.config.attribute.multi.value.separator`
Specifies separator for multiple values. Applies to all types of attributes such as profile, session and response attributes. Default: `|`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Application > Attribute Multi Value Separator.

`com.sun.identity.agents.config.audit.accesstype`
Types of messages to log based on user URL access attempts.

Valid values for the configuration file property include `LOG_NONE`, `LOG_ALLOW`, `LOG_DENY`, and `LOG_BOTH`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > Audit Access Types.

`com.sun.identity.agents.config.auth.connection.timeout`
Timeout period in seconds for an agent connection with OpenAM auth server. Default: 2

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > OpenAM Services > Agent Connection Timeout.

`com.sun.identity.agents.config.cdsso.cdcservlet.url`
List of URLs of the available CDSSO controllers that the agent can use for CDSSO processing. For example, `http://openam.example.com:8080/openam/cdcservelet`.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > SSO > CDSSO Servlet URL.

`com.sun.identity.agents.config.cdsso.cookie.domain`

List of domains, such as `.example.com`, in which cookies have to be set in CDSSO. If this property is left blank, then the fully qualified domain name of the cookie for the agent server is used to set the cookie domain, meaning that a host cookie rather than a domain cookie is set.

To set the list to `.example.com`, and `.example.net` using the configuration file property, include the following.

```
com.sun.identity.agents.config.cdsso.cookie.domain[0]=.example.com
com.sun.identity.agents.config.cdsso.cookie.domain[1]=.example.net
```

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > SSO > Cookies Domain List.

`com.sun.identity.agents.config.cdsso.enable`

Enables Cross Domain Single Sign On.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > SSO > Cross Domain SSO.

`com.sun.identity.agents.config.cleanup.interval`

Interval in minutes to cleanup old agent configuration entries unless they are referenced by current requests. Default: 30.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Global > Configuration Cleanup Interval.

`com.sun.identity.agents.config.client.hostname.header`

HTTP header name that holds the hostname of the client.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > Client Hostname Header.

`com.sun.identity.agents.config.client.ip.header`

HTTP header name that holds the IP address of the client.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > Client IP Address Header.

`com.sun.identity.agents.config.client.ip.validation.enable`

When enabled, validate that the subsequent browser requests come from the same IP address that the SSO token is initially issued against.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Application > Client IP Validation.

`com.sun.identity.agents.config.convert.mbyte.enable`

When enabled, the agent encodes the LDAP header values in the default encoding of operating system locale. When disabled, the agent uses UTF-8.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Miscellaneous > Native Encoding of Profile Attributes.

`com.sun.identity.agents.config.cookie.name`

Name of the SSO Token cookie used between the OpenAM server and the agent. Default: `iPlanetDirectoryPro`.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > SSO > Cookie Name.

`com.sun.identity.agents.config.cookie.reset.enable`

When enabled, agent resets cookies in the response before redirecting to authentication.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > SSO > Cookie Reset.

`com.sun.identity.agents.config.cookie.reset`

List of cookies in the format `*name*[=*value*][;Domain=*value*]`.

Concrete examples include the following with two list items configured.

- `LtpaToken`, corresponding to `com.sun.identity.agents.config.cookie.reset[0]=LtpaToken`. The default domain is taken from FQDN Default.
- `token=value;Domain=subdomain.domain.com`, corresponding to `com.sun.identity.agents.config.cookie.reset[1]= token=value;Domain=subdomain.domain.com`

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > SSO > Cookie Reset Name List.

`com.sun.identity.agents.config.cookie.secure`
: When enabled, the agent marks cookies secure, sending them only if the communication channel is secure.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > SSO > Cookie Security.

`com.sun.identity.agents.config.debug.file.rotate`
: When enabled, rotate the debug file when specified file size is reached.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > Agent Debug File Rotation.

`com.sun.identity.agents.config.debug.file.size`
: Debug file size in bytes beyond which the log file is rotated. The minimum is 3000 bytes, and lower values are reset to 3000 bytes. Default: 10 MB.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > Agent Debug File Size.

`com.sun.identity.agents.config.debug.level`
: Default is `Error`. Increase to `Message` or even `All` for fine-grained detail.

  You can set the level in the configuration file by module using the format *`module`*`[:`*`level`*`][,`*`module`*`[:`*`level`*`]]*`, where *`module`* is one of `AuthService`, `NamingService`, `PolicyService`, `SessionService`, `PolicyEngine`, `ServiceEngine`, `Notification`, `PolicyAgent`, `RemoteLog`, or `all`, and *`level`* is one of the following.

  - `0`: Disable logging from specified module

    At this level the agent nevertheless logs messages having the level value `always`.

  - `1`: Log error messages
  - `2`: Log warning and error messages
  - `3`: Log info, warning, and error messages
  - `4`: Log debug, info, warning, and error messages
  - `5`: Like level 4, but with even more debugging messages

  When you omit *`level`*, the agent uses the default level, which is the level associated with the `all` module.

The following example used in the local configuration sets the log overall level to debug for all messages.

```
com.sun.identity.agents.config.debug.level=all:4
```

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Global > Agent Debug Level.

`com.sun.identity.agents.config.domino.check.name.database`
: When enabled, the agent checks whether the user exists in the Domino name database.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > Check User in Domino Database.

`com.sun.identity.agents.config.domino.ltpa.config.name`
: The configuration name that the agent uses in order to employ the LTPA token mechanism.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > LTPA Token Configuration Name.

`com.sun.identity.agents.config.domino.ltpa.cookie.name`
: The name of the cookie that contains the LTPA token.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > LTPA Token Cookie Name.

`com.sun.identity.agents.config.domino.ltpa.enable`
: Enable if the agent needs to use LTPA Token.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > Use LTPA token.

`com.sun.identity.agents.config.domino.ltpa.org.name`
: The organization name to which the LTPA token belongs.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > LTPA Token Organization Name.

`com.sun.identity.agents.config.encode.cookie.special.chars.enable`
: When enabled, encode special chars in cookie by URL encoding. This is useful when profile, session, and response attributes contain special characters, and the attributes fetch mode is set to `HTTP_COOKIE`.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Miscellaneous > Encode special chars in Cookies.

`com.sun.identity.agents.config.encode.url.special.chars.enable`
When enabled, encodes the URL which has special characters before doing policy evaluation.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Miscellaneous > Encode URL's Special Characters.

`com.sun.identity.agents.config.fetch.from.root.resource`
When enabled, the agent caches the policy decision of the resource and all resources from the root of the resource down. For example, if the resource is `http://host/a/b/c`, then the root of the resource is `http://host/`. This setting can be useful when a client is expect to access multiple resources on the same path. Yet, caching can be expensive if very many policies are defined for the root resource.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > OpenAM Services > Fetch Policies from Root Resource.

`com.sun.identity.agents.config.fqdn.check.enable`
Enables checking of FQDN default value and FQDN map values.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Global > FQDN Check.

`com.sun.identity.agents.config.fqdn.default`
Fully qualified domain name that the users should use in order to access resources. Without this value, the web server can fail to start, thus you set the property on agent installation, and only change it when absolutely necessary.

This property ensures that when users access protected resources on the web server without specifying the FQDN, the agent can redirect the users to URLs containing the correct FQDN.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Global > FQDN Default.

`com.sun.identity.agents.config.fqdn.mapping`
Enables virtual hosts, partial hostname and IP address to access protected resources. Maps invalid or virtual name keys to valid FQDN

values so the agent can properly redirect users and the agents receive cookies belonging to the domain.

To map `myserver` to `myserver.mydomain.example`, enter `myserver` in the Map Key field, and enter `myserver.mydomain.example` in the Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.fqdn.mapping[myserver]= myserver.mydomain.example`.

Invalid FQDN values can cause the web server to become unusable or render resources inaccessible.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > FQDN Virtual Host Map.

`com.sun.identity.agents.config.get.client.host.name`
When enabled, get the client hostname through DNS reverse lookup for use in policy evaluation. This setting can impact performance.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > OpenAM Services > Retrieve Client Hostname.

`com.sun.identity.agents.config.ignore.path.info`
When enabled, strip path info from the request URL while doing the Not Enforced List check, and URL policy evaluation. This is designed to prevent a user from accessing a URI by appending the matching pattern in the policy or not enforced list.

For example, if the not enforced list includes `http://host/*.gif`, then stripping path info from the request URI prevents access to `http://host/index.html` by using `http://host/index.html?hack.gif`.

However, when a web server is configured as a reverse proxy for a J2EE application server, the path info is interpreted to map a resource on the proxy server rather than the application server. This prevents the not enforced list or the policy from being applied to the part of the URI below the application server path if a wildcard character is used.

For example, if the not enforced list includes `http://host/webapp/servcontext/*` and the request URL is `http://host/webapp/servcontext/example.jsp`, the path info is `/servcontext/example.jsp` and the resulting request URL with path info stripped is `http://host/webapp/`, which does not match the not enforced list. Thus when this property is enabled, path info is not stripped from teh request URL even if there is a wildcard in the not enforced list or policy.

Make sure therefore when this property is enabled that there is nothing following the wildcard in the not enforced list or policy.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Miscellaneous > Ignore Path Info in Request URL.

`com.sun.identity.agents.config.ignore.path.info.for.not.enforced.list`
When enabled, the path info and query are stripped from the request URL before being compared with the URLs of the not enforced list for those URLs containing a wildcard character. This prevents a user from accessing `http://host/index.html` by requesting `http://host/index.html/hack.gif` when the not enforced list includes `http://host/*.gif`.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Application > Ignore Path Info for Not Enforced URLs.

`com.sun.identity.agents.config.ignore.preferred.naming.url`
When enabled, do not send a preferred naming URL in the naming request.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Miscellaneous > Ignore Preferred Naming URL in Naming Request.

`com.sun.identity.agents.config.ignore.server.check`
When enabled, do not check whether OpenAM is up before doing a 302 redirect.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Miscellaneous > Ignore Server Check.

`com.sun.identity.agents.config.iis.auth.type`
The agent should normally perform authentication, so this is not required. If necessary, set to `none`.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > Authentication Type.

`com.sun.identity.agents.config.iis.filter.priority`
The loading priority of filter, DEFAULT, HIGH, LOW, or MEDIUM.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > Filter Priority.

`com.sun.identity.agents.config.iis.owa.enable`
Enable if the IIS agent filter is configured for OWA.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > Filter configured with OWA.

`com.sun.identity.agents.config.iis.owa.enable.change.protocol`
Enable to avoid IE6 security pop-ups.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > Change URL Protocol to https.

`com.sun.identity.agents.config.iis.owa.enable.session.timeout.url`
URL of the local idle session timeout page.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > Idle Session Timeout Page URL.

`com.sun.identity.agents.config.load.balancer.enable`
Enable if a load balancer is used for OpenAM services.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > Load Balancer Setup.

`com.sun.identity.agents.config.local.log.rotate`
When enabled, audit log files are rotated when reaching the specified size.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Global > Rotate Local Audit Log.

`com.sun.identity.agents.config.local.log.size`
Beyond this size limit in bytes the agent rotates the local audit log file if rotation is enabled. Default: 50 MB

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Global > Local Audit Log Rotation Size.

`com.sun.identity.agents.config.locale`
The default locale for the agent.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Miscellaneous > Agent Locale.

`com.sun.identity.agents.config.log.disposition`
Specifies where audit messages are logged. By default, audit messages are logged remotely.

Valid values for the configuration file property include `REMOTE`, `LOCAL`, and `ALL`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > Audit Log Location.

`com.sun.identity.agents.config.login.url`
OpenAM login page URL, such as `http://openam.example.com:8080/openam/UI/Login`, to which the agent redirects incoming users without sufficient credentials so then can authenticate.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > OpenAM Services > OpenAM Login URL.

`com.sun.identity.agents.config.logout.cookie.reset`
Cookies to be reset upon logout in the same format as the cookie reset list.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > OpenAM Services > Logout Cookies List for Reset.

`com.sun.identity.agents.config.logout.redirect.url`
User gets redirected to this URL after logout. Specify this property alongside a Logout URL List.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > OpenAM Services > Logout Redirect URL.

`com.sun.identity.agents.config.logout.url`
OpenAM logout page URL, such as `http://openam.example.com:8080/openam/UI/Logout`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > OpenAM Services > OpenAM Logout URL.

`com.sun.identity.agents.config.notenforced.ip`
No authentication and authorization are required for the requests coming from these client IP addresses.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Application > Not Enforced Client IP List.

`com.sun.identity.agents.config.notenforced.url.attributes.enable`
: When enabled, the agent fetches profile attributes for not enforced URLs by doing policy evaluation.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Application > Fetch Attributes for Not Enforced URLs.

`com.sun.identity.agents.config.notenforced.url.invert`
: Only enforce not enforced list of URLs. In other words, enforce policy only for those URLs and patterns specified in the list.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Application > Invert Not Enforced URLs.

`com.sun.identity.agents.config.notenforced.url`
: List of URLs for which no authentication is required. You can use wildcards to define a pattern for a URL.

  The `*` wildcard matches all characters except question mark (`?`), cannot be escaped, and spans multiple levels in a URL. Multiple forward slashes do not match a single forward slash, so `*` matches `mult/iple/dirs`, yet `mult/*/dirs` does not match `mult/dirs`.

  The `-*-` wildcard matches all characters except forward slash (`/`) or question mark (`?`), and cannot be escaped. As it does not match `/`, `-*-` does not span multiple levels in a URL.

  OpenAM does not let you mix `*` and `-*-` in the same URL.

  Examples include `http://www.example.com/logout.html`, `http://www.example.com/images/*`, `http://www.example.com/css/-*-`, and `http://www.example.com/*.jsp?locale=*`.

  Trailing forward slashes are not recognized as part of a resource name. Therefore `http://www.example.com/images//` and `http://www.example.com/images` are equivalent.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Application > Not Enforced URLs.

`com.sun.identity.agents.config.notification.enable`
: If enabled, the agent receives policy updates from the OpenAM notification mechanism to maintain its internal cache. If disabled, the agent must poll OpenAM for changes.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > Enable Notifications.

`com.sun.identity.agents.config.override.host`
Enable if the agent is sitting behind a SSL/TLS off-loader, load balancer, or proxy such that the host name users use is different from the host name the agent uses. When enabled, the host is overridden with the value from the Agent Deployment URI Prefix (property: `com.sun.identity.agents.config.agenturi.prefix`).

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Advanced > Override Request URL Host.

`com.sun.identity.agents.config.override.notification.url`
Enable if the agent is sitting behind a SSL/TLS off-loader, load balancer, or proxy such that the URL users use is different from the URL the agent uses. When enabled, the URL is overridden with the value from the Agent Deployment URI Prefix (property: `com.sun.identity.agents.config.agenturi.prefix`).

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Advanced > Override Notification URL.

`com.sun.identity.agents.config.override.port`
Enable if the agent is sitting behind a SSL/TLS off-loader, load balancer, or proxy such that the port users use is different from the port the agent uses. When enabled, the port is overridden with the value from the Agent Deployment URI Prefix (property: `com.sun.identity.agents.config.agenturi.prefix`).

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Advanced > Override Request URL Port.

`com.sun.identity.agents.config.override.protocol`
Enable if the agent is sitting behind a SSL/TLS off-loader, load balancer, or proxy such that the protocol users use is different from the protocol the agent uses. When enabled, the protocol is overridden with the value from the Agent Deployment URI Prefix (property: `com.sun.identity.agents.config.agenturi.prefix`).

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Advanced > Override Request URL Protocol.

`com.sun.identity.agents.config.policy.cache.polling.interval`
Polling interval in minutes during which an entry remains valid after being added to the agent's cache.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > OpenAM Services > Policy Cache Polling Period.

`com.sun.identity.agents.config.policy.clock.skew`
Time in seconds used adjust time difference between agent system and OpenAM. Clock skew in seconds = AgentTime - OpenAMServerTime.

Use this property to adjust for small time differences encountered despite use of a time synchronization service. When this property is not set and agent time is greater than OpenAM server time, the agent can make policy calls to the OpenAM server before the policy subject cache has expired, or you can see infinite redirection occur.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > OpenAM Services > Policy Clock Skew.

`com.sun.identity.agents.config.poll.primary.server`
Interval in minutes, agent polls to check the primary server is up and running. Default: 5.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > OpenAM Services > Polling Period for Primary Server.

`com.sun.identity.agents.config.polling.interval`
Interval in minutes to fetch agent configuration from OpenAM. Used if notifications are disabled. Default: 60.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Global > Configuration Reload Interval.

`com.sun.identity.agents.config.postcache.entry.lifetime`
POST cache entry lifetime in minutes. Default: 10.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced > POST Data Entries Cache Period.

`com.sun.identity.agents.config.postdata.preserve.enable`
Enables HTTP POST data preservation. This feature is available in the Apache 2.2, Microsoft IIS 6, Microsoft IIS 7, and Sun Java System Web Server web policy agents as of version 3.0.3.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Advanced > POST Data Preservation.

`com.sun.identity.agents.config.postdata.preserve.lbcookie`
When HTTP POST data preservation is enabled, override properties are set to true, and the agent is behind a load balancer, then this property sets the name and value of the sticky cookie to use.

`com.sun.identity.agents.config.profile.attribute.cookie.maxage`
Maximum age in seconds of custom cookie headers. Default: 300.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Miscellaneous > Profile Attributes Cookie Maxage.

`com.sun.identity.agents.config.profile.attribute.cookie.prefix`
Sets cookie prefix in the attributes headers. Default: `HTTP_`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Miscellaneous > Profile Attributes Cookie Maxage.

`com.sun.identity.agents.config.profile.attribute.fetch.mode`
When set to `HTTP_COOKIE` or `HTTP_HEADER`, profile attributes are introduced into the cookie or the headers, respectively.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Application > Profile Attribute Fetch Mode.

`com.sun.identity.agents.config.profile.attribute.mapping`
Maps the profile attributes to HTTP headers for the currently authenticated user. Map Keys are LDAP attribute names, and Map Values are HTTP header names.

To populate the value of profile attribute CN under `CUSTOM-Common-Name`: enter CN in the Map Key field, and enter `CUSTOM-Common-Name` in the Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.profile.attribute.mapping[cn]=CUSTOM-Common-Name`.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by `HTTP_`, lower case letters become upper case, and hyphens (`-`) become underscores (`_`). For example, `common-name` becomes `HTTP_COMMON_NAME`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Application > Profile Attribute Map.

`com.sun.identity.agents.config.proxy.override.host.port`

When enabled ignore the host and port settings for Sun Java System Proxy.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Advanced > Override Proxy Server's Host and Port.

`com.sun.identity.agents.config.redirect.param`

Property used only when CDSSO is enabled. Only change the default value, `goto` when the login URL has a landing page specified such as, `com.sun.identity.agents.config.cdsso.cdcservlet.url = http://openam.example.com:8080/openam/cdcservlet?goto= http://www.example.com/landing.jsp`. The agent uses this parameter to append the original request URL to this cdcservlet URL. The landing page consumes this parameter to redirect to the original URL.

As an example, if you set this value to `goto2`, then the complete URL sent for authentication is `http://openam.example.com:8080/openam/cdcservlet?goto= http://www.example.com/landing.jsp?goto2=http://www.example.com/original.jsp`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Miscellaneous > Goto Parameter Name.

`com.sun.identity.agents.config.remote.log.interval`

Periodic interval in minutes in which audit log messages are sent to the remote log file. Default: 5

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > Remote Audit Log Interval.

`com.sun.identity.agents.config.remote.logfile`

Name of file stored on OpenAM server that contains agent audit messages if log location is remote or all.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > Remote Log Filename.

`com.sun.identity.agents.config.replaypasswd.key`

DES key for decrypting the basic authentication password in the session for IIS.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Advanced > Replay Password Key.

`com.sun.identity.agents.config.response.attribute.fetch.mode`

When set to `HTTP_COOKIE` or `HTTP_HEADER`, response attributes are introduced into the cookie or the headers, respectively.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Application > Response Attribute Fetch Mode.

`com.sun.identity.agents.config.response.attribute.mapping`

Maps the policy response attributes to HTTP headers for the currently authenticated user. The response attribute is the attribute in the policy response to be fetched.

To populate the value of response attribute `uid` under `CUSTOM-User-Name`: enter `uid` in the Map Key field, and enter `CUSTOM-User-Name` in the Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.response.attribute.mapping[uid]=Custom-User-Name`.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by `HTTP_`, lower case letters become upper case, and hyphens (`-`) become underscores (`_`). For example, `response-attr-one` becomes `HTTP_RESPONSE_ATTR_ONE`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Application > Response Attribute Map.

`com.sun.identity.agents.config.session.attribute.fetch.mode`

When set to `HTTP_COOKIE` or `HTTP_HEADER`, session attributes are introduced into the cookie or the headers, respectively.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Application > Session Attribute Fetch Mode.

`com.sun.identity.agents.config.session.attribute.mapping`

Maps session attributes to HTTP headers for the currently authenticated user. The session attribute is the attribute in the session to be fetched.

To populate the value of session attribute `UserToken` under `CUSTOM-userid`: enter `UserToken` in the Map Key field, and enter `CUSTOM-userid` in the Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.session.attribute.mapping[UserToken]=CUSTOM-userid`.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by `HTTP_`, lower case letters become upper case, and hyphens (-) become underscores (_). For example, `success-url` becomes `HTTP_SUCCESS_URL`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Application > Session Attribute Map.

`com.sun.identity.agents.config.sso.cache.polling.interval`
Polling interval in minutes during which an SSO entry remains valid after being added to the agent's cache.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > OpenAM Services > SSO Cache Polling Period.

`com.sun.identity.agents.config.sso.only`
When enabled, agent only enforces authentication (SSO), but no policies for authorization.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > SSO Only Mode.

`com.sun.identity.agents.config.url.comparison.case.ignore`
When enabled, enforce case sensitivity in both policy and not enforced URL evaluation.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Miscellaneous > URL Comparison Case Sensitivity Check.

`com.sun.identity.agents.config.userid.param`
Agent sets this value for User Id passed in the session from OpenAM to the REMOTE_USER server variable. Default: UserToken.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > OpenAM Services > User ID Parameter.

`com.sun.identity.agents.config.userid.param.type`
User ID can be fetched from either SESSION and LDAP attributes. Default: `SESSION`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > OpenAM Services > User ID Parameter Type.

`com.sun.identity.client.notification.url`
URL used by agent to register notification listeners.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > Web > *`Agent Name`* > Global > Agent Notification URL.

`com.sun.identity.cookie.httponly`
Set this property to `true` to mark `iPlanetDirectoryPro` cookies as HTTPOnly, preventing scripts and third-party programs from accessing the cookies.

Appendix B. Java EE Agent Configuration Properties

Java EE Agents use the following configuration properties. Bootstrap properties are always configured locally, whereas other agent configuration properties are either configured centrally in OpenAM or locally using the agent properties file.

B.1. Bootstrap Configuration Properties

These properties are set in `config/OpenSSOAgentBootstrap.properties`.

`am.encryption.pwd`
: When using an encrypted password, set this to the encryption key used to encrypt the agent profile password.

`com.iplanet.am.naming.url`
: Set this to the naming service URL(s) used for naming lookups in OpenAM. Separate multiple URLs with single space characters.

`com.iplanet.am.service.secret`
: When using a plain text password, set this to the password for the agent profile, and leave `am.encryption.pwd` blank.

  When using an encrypted password, set this to the encrypted version of the password for the agent profile. Use the command **./agentadmin --encrypt *`agentInstance`* *`passwordFile`*** to get the encrypted version.

`com.iplanet.am.services.deploymentDescriptor`
: Set this to the URI under which OpenAM is deployed, such as `/openam`.

`com.iplanet.services.debug.directory`
: Set this to the full path of the agent's debug log directory where the agent writes debug log files.

`com.sun.identity.agents.app.username`
: Set this to the agent profile name.

`com.sun.identity.agents.config.local.logfile`
: Set this to the full path for agent's audit log file.

`com.sun.identity.agents.config.lock.enable`
: Set this to `true` to require an agent restart to allow agent configuration changes, even for hot-swappable parameters. Default is `false`.

`com.sun.identity.agents.config.organization.name`
: Set this to the realm name where the agent authenticates to OpenAM.

`com.sun.identity.agents.config.profilename`
: Set this to the profile name used to fetch agent configuration data. Unless multiple agents use the same credentials to authenticate, this is the same as `com.sun.identity.agents.app.username`.

`com.sun.identity.agents.config.service.resolver`
: Set this to the class name of the service resolver used by the agent.

`com.sun.services.debug.mergeall`
: When set to `on`, the default, the agent writes all debug messages to a single file under `com.iplanet.services.debug.directory`.

B.2. Agent Configuration Properties

These properties are set in `config/OpenSSOAgentConfiguration.properties` if your agent uses local configuration. If your agent uses centralized configuration, the properties are set in OpenAM.

`com.forgerock.agents.conditional.login.url`
: To conditionally redirect users based on the incoming request URL, set this property.

  This takes the incoming request URL to match, a vertical bar (`|`), and then a comma-separated list of URLs to which to redirect incoming users.

  If the string before the vertical bar matches an incoming request URL, then the policy agent uses the list of URLs to determine how to redirect the user-agent. If the global property FQDN Check (`com.sun.identity.agents.config.fqdn.check.enable`) is enabled for the policy agent, then the policy agent iterates through the list until it finds an appropriate redirect URL that matches the FQDN check. Otherwise, the policy agent redirects the user-agent to the first URL in the list.

  Examples: `com.forgerock.agents.conditional.login.url[0]= login.example.com/|http://openam1.example.com/openam/UI/Login, http://openam2.example.com/openam/UI/Login, com.forgerock.agents.conditional.login.url[1]= login.example.com| http://openam3.example.com/openam/UI/Login, http:// openam4.example.com/openam/UI/Login`

`com.iplanet.am.cookie.name`
: Name of the SSO Token cookie used between the OpenAM server and the agent. Default: `iPlanetDirectoryPro`.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > Cookie Name.

`com.iplanet.am.sdk.remote.pollingTime`
: If notifications are not enabled and set to a value other than zero, specifies the time in minutes after which the agent polls to update cached user management data. Default: 1

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > User Data Cache Polling Time.

`com.iplanet.am.server.host`
: Specifies the OpenAM authentication service host name.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > OpenAM Authentication Service Host Name.

`com.iplanet.am.server.port`
: Specifies the OpenAM authentication service port number.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > OpenAM Authentication Service Port.

`com.iplanet.am.server.protocol`
: Specifies the protocol used by the OpenAM authentication service.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > OpenAM Authentication Service Protocol.

`com.iplanet.am.session.client.polling.enable`
: When enabled, the session client polls to update the session cache rather than relying on notifications from OpenAM.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > Enable Client Polling.

`com.iplanet.am.session.client.polling.period`
: Specifies the time in seconds after which the session client requests an update from OpenAM for cached session information. Default: 180

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > Client Polling Period.

`com.iplanet.security.encryptor`
: Specifies the agent's encryption provider class.

Default: `com.iplanet.services.util.JCEEncryption`

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Miscellaneous > Encryption Provider.

`com.iplanet.services.debug.level`
: Default is `Error`. Increase to `Message` or even `All` for fine-grained detail.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > Agent Debug Level.

`com.sun.identity.agents.config.access.denied.uri`
: Specifies the URIs of custom pages to return when access is denied. The key is the web application name. The value is the custom URI.

To set a global custom access denied URI for applications without other custom access denied URIs defined, leave the key empty and set the value to the global custom access denied URI, `/sample/accessdenied.html`.

To set a custom access denied URI for a specific application, set the key to the name of the application, and the value to the application access denied URI, such as `/myApp/accessdenied.html`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Resource Access Denied URI.

`com.sun.identity.agents.config.agent.host`
: Specifies the host name of the agent protected server to show to client browsers, rather than the actual host name.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Advanced > Alternative Agent Host Name.

`com.sun.identity.agents.config.agent.port`
: Specifies the port number of the agent protected server to show to client browsers, rather than the actual port number.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Advanced > Alternative Agent Port Name.

`com.sun.identity.agents.config.agent.protocol`
: Specifies the protocol used to contact the agent from the browser client browsers, rather than the actual protocol used by the server. Either `http` or `https`.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Advanced > Alternative Agent Protocol.

`com.sun.identity.agents.config.amsso.cache.enable`
: When enabled, the agent exposes SSO Cache through the agent SDK APIs.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > SSO > SSO Cache Enable.

`com.sun.identity.agents.config.attribute.cookie.encode`
: When enabled, attribute values are URL encoded before being set as a cookie.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Attribute Cookie Encode.

`com.sun.identity.agents.config.attribute.cookie.separator`
: Specifies the separator for multiple values of the same attribute when it is set as a cookie. Default: `|`.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Cookie Separator Character.

`com.sun.identity.agents.config.attribute.date.format`
: Specifies the `java.text.SimpleDateFormat` of date attribute values used when an attribute is set in an HTTP header. Default: `EEE, d MMM yyyy hh:mm:ss z`.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Fetch Attribute Date Format.

`com.sun.identity.agents.config.audit.accesstype`
: Types of messages to log based on user URL access attempts.

  Valid values for the configuration file property include `LOG_NONE`, `LOG_ALLOW`, `LOG_DENY`, and `LOG_BOTH`.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Global > Audit Access Types.

`com.sun.identity.agents.config.auth.handler`
Specifies custom authentication handler classes for users authenticated with the application server. The key is the web application name and the value is the authentication handler class name.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Custom Authentication Handler.

`com.sun.identity.agents.config.bypass.principal`
Specifies a list of principals the agent bypasses for authentication and search purposes, such as `guest` or `testuser`.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Miscellaneous > Bypass Principal List.

`com.sun.identity.agents.config.cdsso.cdcservlet.url`
List of URLs of the available CDSSO controllers that the agent can use for CDSSO processing. For example, `http://openam.example.com:8080/openam/cdcservelet`.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > SSO > CDSSO Servlet URL.

`com.sun.identity.agents.config.cdsso.clock.skew`
When set to a value other than zero, specifies the clock skew in seconds that the agent accepts when determining the validity of the CDSSO authentication response assertion.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > SSO > CDSSO Clock Skew.

`com.sun.identity.agents.config.cdsso.enable`
Enables Cross Domain Single Sign On.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > SSO > Cross Domain SSO.

`com.sun.identity.agents.config.cdsso.redirect.uri`
Specifies a URI the agent uses to process CDSSO requests.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > SSO > CDSSO Redirect URI.

`com.sun.identity.agents.config.cdsso.trusted.id.provider`

Specifies the list of OpenAM servers or identity providers the agent trusts when evaluating CDC Liberty Responses.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > SSO > CDSSO Trusted ID Provider.

`com.sun.identity.agents.config.client.hostname.header`

If the agent is behind a proxy or load balancer, then the agent can get client IP and host name values from the proxy or load balancer. For proxies and load balancer that support providing the client IP and host name in HTTP headers, you can use the following properties.

When multiple proxies are load balancers sit in the request path, the header values can include a comma-separated list of values with the first value representing the client, as in `client,next-proxy,first-proxy`.

HTTP header name that holds the hostname of the client.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Advanced > Client Hostname Header.

`com.sun.identity.agents.config.client.ip.header`

Similar to `com.sun.identity.agents.config.client.hostname.header`, HTTP header name that holds the IP address of the client.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Advanced > Client IP Address Header.

`com.sun.identity.agents.config.cookie.reset.domain`

Specifies how names from `com.sun.identity.agents.config.cookie.reset.name` correspond to cookie domain values when the cookie is reset.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > SSO > Cookie Reset Domain Map.

`com.sun.identity.agents.config.cookie.reset.enable`

When enabled, agent resets cookies in the response before redirecting to authentication.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > SSO > Cookie Reset.

`com.sun.identity.agents.config.cookie.reset.name`
: List of cookies to reset if `com.sun.identity.agents.config.cookie.reset.enable` is enabled.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > SSO > Cookie Reset Name List.

`com.sun.identity.agents.config.cookie.reset.path`
: Specifies how names from the `com.sun.identity.agents.config.cookie.reset.name` correspond to cookie paths when the cookie is reset.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > SSO > Cookie Reset Path Map.

`com.sun.identity.agents.config.default.privileged.attribute`
: Specifies the list of privileged attributes granted to all users with a valid OpenAM session, such as `AUTHENTICATED_USERS`.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Default Privileged Attribute.

`com.sun.identity.agents.config.filter.mode`
: Specifies how the agent filters requests to protected web applications. The global value functions as a default, and applies for protected applications that do not have their own filter settings. Valid settings include the following.

  `ALL`
  : Enforce both the J2EE policy defined for the web container where the protected application runs, and also OpenAM policies.

    When setting the filter mode to `ALL`, set the Map Key, but do not set any Corresponding Map Value.

  `J2EE_POLICY`
  : Enforce only the J2EE policy defined for the web container where the protected application runs.

  `NONE`
  : Do not enforce policies to protect resources. In other words, turn off access management. Not for use in production.

  `SSO_ONLY`
  : Enforce only authentication, not policies.

`URL_POLICY`
: Enforce only OpenAM, URL resource based policies.

  When setting the filter mode to `URL_POLICY`, set the Map Key to the application name and the Corresponding Map Value to `URL_POLICY`.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > Agent Filter Mode.

`com.sun.identity.agents.config.fqdn.check.enable`
: Enables checking of FQDN default value and FQDN map values.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > FQDN Check.

`com.sun.identity.agents.config.fqdn.default`
: Fully qualified domain name that the users should use in order to access resources.

  This property ensures that when users access protected resources on the web server without specifying the FQDN, the agent can redirect the users to URLs containing the correct FQDN.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > FQDN Default.

`com.sun.identity.agents.config.fqdn.mapping`
: Enables virtual hosts, partial hostname and IP address to access protected resources. Maps invalid or virtual name keys to valid FQDN values so the agent can properly redirect users and the agents receive cookies belonging to the domain.

  To map `myserver` to `myserver.mydomain.example`, enter `myserver` in the Map Key field, and enter `myserver.mydomain.example` in the Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.fqdn.mapping[myserver]= myserver.mydomain.example`.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > FQDN Virtual Host Map.

`com.sun.identity.agents.config.httpsession.binding`
: When enabled the agent invalidates the HTTP session upon login failure, when the user has no SSO session, or when the principal user name does not match the SSO user name.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Global > HTTP Session Binding.

`com.sun.identity.agents.config.ignore.path.info`

When enabled, strip path info from the request URL while doing the Not Enforced List check, and URL policy evaluation. This is designed to prevent a user from accessing a URI by appending the matching pattern in the policy or not enforced list.

For example, if the not enforced list includes `/*.gif`, then stripping path info from the request URL prevents access to `http://host/index.html` by using `http://host/index.html?hack.gif`.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Miscellaneous > Ignore Path Info in Request URL.

`com.sun.identity.agents.config.legacy.redirect.uri`

Specifies a URI the agent uses to redirect legacy user agent requests.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Miscellaneous > Legacy User Agent Redirect URI.

`com.sun.identity.agents.config.legacy.support.enable`

When enabled, provide support for legacy browsers.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Miscellaneous > Legacy User Agent Support Enable.

`com.sun.identity.agents.config.legacy.user.agent`

List of header values that identify legacy browsers. Entries can use the wildcard character, `*`.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Miscellaneous > Legacy User Agent List.

`com.sun.identity.agents.config.load.interval`

Interval in seconds to fetch agent configuration from OpenAM. Used if notifications are disabled. Default: 0

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Global > Configuration Reload Interval.

`com.sun.identity.agents.config.local.log.rotate`
When enabled, audit log files are rotated when reaching the specified size.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > Rotate Local Audit Log.

`com.sun.identity.agents.config.local.log.size`
Beyond this size limit in bytes the agent rotates the local audit log file if rotation is enabled. Default: 50 MB

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > Local Audit Log Rotation Size.

`com.sun.identity.agents.config.locale.country`
The default country for the agent.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Miscellaneous > Locale Country.

`com.sun.identity.agents.config.locale.language`
The default language for the agent.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Miscellaneous > Locale Language.

`com.sun.identity.agents.config.log.disposition`
Specifies where audit messages are logged. By default, audit messages are logged remotely.

Valid values for the configuration file property include `REMOTE`, `LOCAL`, and `ALL`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > Audit Log Location.

`com.sun.identity.agents.config.login.attempt.limit`
When set to a value other than zero, this defines the maximum number of failed login attempts allowed during a single browser session, after which the agent blocks requests from the user.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > Login Attempt Limit.

`com.sun.identity.agents.config.login.content.file`
: Full path name to the file containing custom login content when Use Internal Login is enabled.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Login Content File Name.

`com.sun.identity.agents.config.login.error.uri`
: Specifies the list of absolute URIs corresponding to a protected application's `web.xml` `form-error-page` element, such as `/myApp/jsp/error.jsp`.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Login Error URI.

`com.sun.identity.agents.config.login.form`
: Specifies the list of absolute URIs corresponding to a protected application's `web.xml` `form-login-page` element, such as `/myApp/jsp/login.jsp`.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Login Form URI.

`com.sun.identity.agents.config.login.url.prioritized`
: When enabled, OpenAM uses the priority defined in the OpenAM Login URL list as the priority for Login and CDSSO URLs when handling failover.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > OpenAM Services > Login URL Prioritized.

`com.sun.identity.agents.config.login.url.probe.enabled`
: When enabled, OpenAM checks the availability of OpenAM Login URLs before redirecting to them.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > OpenAM Services > Login URL Probe.

`com.sun.identity.agents.config.login.url.probe.timeout`
: Timeout period in milliseconds for OpenAM to determine whether to failover between Login URLs when Login URL Probe is enabled. Default: 2000

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > OpenAM Services > Login URL Probe Timeout.

`com.sun.identity.agents.config.login.url`
OpenAM login page URL, such as `http://openam.example.com:8080/openam/UI/Login`, to which the agent redirects incoming users without sufficient credentials so then can authenticate.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > OpenAM Services > OpenAM Login URL.

`com.sun.identity.agents.config.login.use.internal`
When enabled, the agent uses the internal default content file for the login.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Use Internal Login.

`com.sun.identity.agents.config.logout.application.handler`
Specifies how logout handlers map to specific applications. The key is the web application name. The value is the logout handler class.

To set a global logout handler for applications without other logout handlers defined, leave the key empty and set the value to the global logout handler class name, `GlobalApplicationLogoutHandler`.

To set a logout handler for a specific application, set the key to the name of the application, and the value to the logout handler class name.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Application Logout Handler.

`com.sun.identity.agents.config.logout.entry.uri`
Specifies the URIs to return after successful logout and subsequent authentication. The key is the web application name. The value is the URI to return.

To set a global logout entry URI for applications without other logout entry URIs defined, leave the key empty and set the value to the global logout entry URI, `/welcome.html`.

To set a logout entry URI for a specific application, set the key to the name of the application, and the value to the application logout entry URI, such as `/myApp/welcome.html`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Logout Entry URI.

`com.sun.identity.agents.config.logout.handler`
Specifies custom logout handler classes to log users out of the application server. The key is the web application name and the value is the logout handler class name.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Custom Logout Handler.

`com.sun.identity.agents.config.logout.introspect.enabled`
When enabled, the agent checks the HTTP request body to locate the Logout Request Parameter you set.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Logout Introspect Enabled.

`com.sun.identity.agents.config.logout.request.param`
Specifies parameters in the HTTP request that indicate logout events. The key is the web application name. The value is the logout request parameter.

To set a global logout request parameter for applications without other logout request parameters defined, leave the key empty and set the value to the global logout request parameter, `logoutparam`.

To set a logout request parameter for a specific application, set the key to the name of the application, and the value to the application logout request parameter, such as `logoutparam`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Logout Request Parameter.

`com.sun.identity.agents.config.logout.uri`
Specifies request URIs that indicate logout events. The key is the web application name. The value is the application logout URI.

To set a global logout URI for applications without other logout URIs defined, leave the key empty and set the value to the global logout URI, `/logout.jsp`.

To set a logout URI for a specific application, set the key to the name of the application, and the value to the application logout page.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Application Logout URI.

`com.sun.identity.agents.config.logout.url.prioritized`
When enabled, OpenAM uses the priority defined in the OpenAM Logout URL list as the priority for Logout URLs when handling failover.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > Logout URL Prioritized.

`com.sun.identity.agents.config.logout.url.probe.enabled`
When enabled, OpenAM checks the availability of OpenAM Logout URLs before redirecting to them.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > Logout URL Probe.

`com.sun.identity.agents.config.logout.url.probe.timeout`
Timeout period in milliseconds for OpenAM to determine whether to failover between Logout URLs when Logout URL Probe is enabled. Default: 2000

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > Logout URL Probe Timeout.

`com.sun.identity.agents.config.logout.url`
OpenAM logout page URLs, such as `http://openam.example.com:8080/openam/UI/Logout`. The user is logged out of the OpenAM session when accessing these URLs.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > OpenAM Logout URL.

`com.sun.identity.agents.config.notenforced.ip.cache.enable`
When enabled, the agent caches evaluation of the not enforced IP list.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Not Enforced IP Cache Flag.

`com.sun.identity.agents.config.notenforced.ip.cache.size`
When caching is enabled, this limits the number of not enforced addresses cached. Default: 1000

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Not Enforced IP Cache Size.

`com.sun.identity.agents.config.notenforced.ip.invert`
Only enforce the not enforced list of IP addresses. In other words, enforce policy only for those client addresses and patterns specified in the list.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Not Enforced IP Invert List.

`com.sun.identity.agents.config.notenforced.ip`
No authentication and authorization are required for the requests coming from these client IP addresses.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Not Enforced Client IP List.

`com.sun.identity.agents.config.notenforced.refresh.session.idletime`
When enabled, the agent reset the session idle time when granting access to a not enforced URI, prolonging the time before the user must authenticate again.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Refresh Session Idle Time.

`com.sun.identity.agents.config.notenforced.uri.cache.enable`
When enabled, the agent caches evaluation of the not enforced URI list.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Not Enforced URIs Cache Enabled.

`com.sun.identity.agents.config.notenforced.uri.cache.size`
When caching is enabled, this limits the number of not enforced URIs cached.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Not Enforced URIs Cache Size.

`com.sun.identity.agents.config.notenforced.uri.invert`
Only enforce not enforced list of URIs. In other words, enforce policy only for those URIs and patterns specified in the list.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Invert Not Enforced URIs.

`com.sun.identity.agents.config.notenforced.uri`
List of URIs for which no authentication is required, and the agent does not protect access. You can use wildcards to define a pattern for a URI.

The `*` wildcard matches all characters except question mark (`?`), cannot be escaped, and spans multiple levels in a URI. Multiple forward slashes do not match a single forward slash, so `*` matches `mult/iple/dirs`, yet `mult/*/dirs` does not match `mult/dirs`.

The `-*-` wildcard matches all characters except forward slash (`/`) or question mark (`?`), and cannot be escaped. As it does not match `/`, `-*-` does not span multiple levels in a URI.

OpenAM does not let you mix `*` and `-*-` in the same URI.

Examples include `/logout.html`, `/images/*`, `/css/-*-`, and `/*.jsp?locale=*`.

Trailing forward slashes are not recognized as part of a resource name. Therefore `/images//` and `/images` are equivalent.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application > Not Enforced URIs.

`com.sun.identity.agents.config.policy.env.get.param`
Specifies the list of HTTP GET request parameters whose names and values the agents sets in the environment map for URL policy evaluation by the OpenAM server.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > URL Policy Env GET Parameters.

`com.sun.identity.agents.config.policy.env.jsession.param`
Specifies the list of HTTP session attributes whose names and values the agents sets in the environment map for URL policy evaluation by the OpenAM server.

For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > URL Policy Env jsession Parameters.

`com.sun.identity.agents.config.policy.env.post.param`

Specifies the list of HTTP POST request parameters whose names and values the agents sets in the environment map for URL policy evaluation by the OpenAM server.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > OpenAM Services > URL Policy Env POST Parameters.

`com.sun.identity.agents.config.port.check.enable`

When enabled, activate port checking, correcting requests on the wrong port.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Miscellaneous > Port Check Enable.

`com.sun.identity.agents.config.port.check.file`

Specifies the name of the file containing the content to handle requests on the wrong port when port checking is enabled.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Miscellaneous > Port Check File.

`com.sun.identity.agents.config.port.check.setting`

Specifies which ports correspond to which protocols. The agent uses the map when handling requests with invalid port numbers during port checking.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Miscellaneous > Port Check Setting.

`com.sun.identity.agents.config.privileged.attribute.mapping.enable`

When enabled, lets you use Privileged Attribute Mapping.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Enable Privileged Attribute Mapping.

`com.sun.identity.agents.config.privileged.attribute.mapping`

Maps OpenAM UUIDs to principal names specified in your web application's deployment descriptor, such as `com.sun.identity.agents.config.privileged.attribute.mapping [id\=manager,ou\=group,o\=openam] = am_manager_role` or `com.sun.identity.agents.config.privileged.attribute.mapping [id\=employee,ou\=group,o\=openam] = am_employee_role`

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > 0Privileged Attribute Mapping.

`com.sun.identity.agents.config.privileged.attribute.tolowercase`
Specifies how privileged attribute types should be converted to lower case.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Privileged Attributes To Lower Case.

`com.sun.identity.agents.config.privileged.attribute.type`
Specifies the list of privileged attribute types fetched for each user.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Privileged Attribute Type.

`com.sun.identity.agents.config.privileged.session.attribute`
Specifies the list of session property names, such as `UserToken` which hold privileged attributes for authenticated users.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Privileged Session Attribute.

`com.sun.identity.agents.config.profile.attribute.fetch.mode`
When set to `HTTP_COOKIE` or `HTTP_HEADER`, profile attributes are introduced into the cookie or the headers, respectively.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Profile Attribute Fetch Mode.

`com.sun.identity.agents.config.profile.attribute.mapping`
Maps the profile attributes to HTTP headers for the currently authenticated user. Map Keys are LDAP attribute names, and Map Values are HTTP header names.

To populate the value of profile attribute CN under `CUSTOM-Common-Name`: enter CN in the Map Key field, and enter `CUSTOM-Common-Name` in the Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.profile.attribute.mapping[cn]=CUSTOM-Common-Name`.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by `HTTP_`, lower case letters become upper case, and hyphens (`-`) become underscores (`_`). For example, `common-name` becomes `HTTP_COMMON_NAME`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Profile Attribute Mapping.

`com.sun.identity.agents.config.redirect.attempt.limit`
: When set to a value other than zero, this defines the maximum number of redirects allowed for a single browser session, after which the agent blocks the request.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > Redirect Attempt Limit.

`com.sun.identity.agents.config.redirect.param`
: Property used only when CDSSO is enabled. Only change the default value, `goto` when the login URL has a landing page specified such as, `com.sun.identity.agents.config.cdsso.cdcservlet.url = http://openam.example.com:8080/openam/cdcservlet?goto= http://www.example.com/landing.jsp`. The agent uses this parameter to append the original request URL to this cdcserlet URL. The landing page consumes this parameter to redirect to the original URL.

  As an example, if you set this value to `goto2`, then the complete URL sent for authentication is `http://openam.example.com:8080/openam/cdcservlet?goto= http://www.example.com/landing.jsp?goto2=http://www.example.com/original.jsp`.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Miscellaneous > Goto Parameter Name.

`com.sun.identity.agents.config.remote.logfile`
: Name of file stored on OpenAM server that contains agent audit messages if log location is remote or all.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > Remote Log Filename.

`com.sun.identity.agents.config.response.attribute.fetch.mode`
: When set to `HTTP_COOKIE` or `HTTP_HEADER`, response attributes are introduced into the cookie or the headers, respectively. When set to `REQUEST_ATTRIBUTE`, response attributes are part of the HTTP response.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Response Attribute Fetch Mode.

`com.sun.identity.agents.config.response.attribute.mapping`

Maps the policy response attributes to HTTP headers for the currently authenticated user. The response attribute is the attribute in the policy response to be fetched.

To populate the value of response attribute `uid` under `CUSTOM-User-Name`: enter `uid` in the Map Key field, and enter `CUSTOM-User-Name` in the Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.response.attribute.mapping[uid]=Custom-User-Name`.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by `HTTP_`, lower case letters become upper case, and hyphens (`-`) become underscores (`_`). For example, `response-attr-one` becomes `HTTP_RESPONSE_ATTR_ONE`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Response Attribute Map.

`com.sun.identity.agents.config.response.header`

Specifies the custom headers the agent sets for the client. The key is the header name. The value is the header value.

For example, `com.sun.identity.agents.config.response.header[Cache-Control]=no-cache`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > Custom Response Header.

`com.sun.identity.agents.config.session.attribute.fetch.mode`

When set to `HTTP_COOKIE` or `HTTP_HEADER`, session attributes are introduced into the cookie or the headers, respectively. When set to `REQUEST_ATTRIBUTE`, session attributes are part of the HTTP response.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Session Attribute Fetch Mode.

`com.sun.identity.agents.config.session.attribute.mapping`

Maps session attributes to HTTP headers for the currently authenticated user. The session attribute is the attribute in the session to be fetched.

To populate the value of session attribute `UserToken` under `CUSTOM-userid`: enter `UserToken` in the Map Key field, and enter `CUSTOM-userid` in the Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.session.attribute.mapping[UserToken]=CUSTOM-userid`.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by `HTTP_`, lower case letters become upper case, and hyphens (`-`) become underscores (`_`). For example, `success-url` becomes `HTTP_SUCCESS_URL`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Session Attribute Map.

`com.sun.identity.agents.config.user.attribute.name`
Specifies the data store attribute that contains the user ID.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > User Attribute Name.

`com.sun.identity.agents.config.user.mapping.mode`
Specifies the mechanism used to determine the user ID.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > User Mapping Mode.

`com.sun.identity.agents.config.user.principal`
When enabled, OpenAM uses both the principal user name and also the user ID for authentication.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > User Principal Flag.

`com.sun.identity.agents.config.user.token`
Specifies the session property name for the authenticated user's ID. Default: `UserToken`.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Global > User Token Name.

`com.sun.identity.agents.config.verification.handler`
Specifies custom verification classes to validate user credentials with the local user repository. The key is the web application name and the value is the validation handler class name.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Application > Custom Verification Handler.

`com.sun.identity.agents.config.webservice.authenticator`
: Specifies an implementation class of interface `com.sun.identity.agents.filter.IWebServiceAuthenticator` that can be used to authenticate web-service requests.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Advanced > Web Service Authenticator.

`com.sun.identity.agents.config.webservice.autherror.content`
: Specifies a file the agent uses to generate an authorization error fault for the client application.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Advanced > Web Service Authorization Error Content File.

`com.sun.identity.agents.config.webservice.enable`
: Enable web service processing.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Advanced > Web Service Enable.

`com.sun.identity.agents.config.webservice.endpoint`
: Specifies a list of web application end points that represent web services.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Advanced > Web Service End Points.

`com.sun.identity.agents.config.webservice.internalerror.content`
: Specifies a file the agent uses to generate an internal error fault for the client application.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Advanced > Web Service Internal Error Content File.

`com.sun.identity.agents.config.webservice.process.get.enable`
: When enabled, the agent processes HTTP GET requests for web service endpoints.

  For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > Advanced > Web Service Process GET Enable.

`com.sun.identity.agents.config.webservice.responseprocessor`
: Specifies a class implementing `com.sun.identity.agents.filter.IWebServiceResponseProcessor`, used to process web service reponses.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Advanced > Web Service Response Processor.

`com.sun.identity.agents.notification.enabled`
: When enabled, OpenAM sends notification about changes to policy.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > Enable Policy Notifications.

`com.sun.identity.agents.polling.interval`
: Specifies the time in minutes after which the policy cache is refreshed. Default: 3

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > Policy Client Polling Interval.

`com.sun.identity.client.notification.url`
: URL used by agent to register notification listeners.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Global > Agent Notification URL.

`com.sun.identity.idm.remote.notification.enabled`
: When enabled, receive notification from OpenAM to update user management data caches.

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > Enable Notification of User Data Caches.

`com.sun.identity.policy.client.booleanActionValues`
: Specifies the values, such as `allow` and `deny`, that are associated with boolean policy decisions.

  Default: `iPlanetAMWebAgentService|GET|allow|deny:iPlanetAMWebAgentService|POST|allow|deny`

  For centralized configurations this property is configured under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services > Policy Client Boolean Action Values.

`com.sun.identity.policy.client.cacheMode`

Set to cache mode subtree when only a small number of policy rules are defined. For large numbers of policy rules, set to self.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > OpenAM Services > Policy Client Cache Mode.

`com.sun.identity.policy.client.clockSkew`

Time in seconds used adjust time difference between agent system and OpenAM. Clock skew in seconds = AgentTime - OpenAMServerTime.

Default: 10.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > OpenAM Services > Policy Client Clock Skew.

`com.sun.identity.policy.client.resourceComparators`

Specifies the comparators used for service names in policy.

Default: `serviceType=iPlanetAMWebAgentService| class=com.sun.identity.policy.plugins.HttpURLResourceName| wildcard=*| delimiter=/|caseSensitive=false`

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > OpenAM Services > Policy Client Resource Comparators.

`com.sun.identity.sm.cacheTime`

If notifications are not enabled and set to a value other than zero, specifies the time in minutes after which the agent polls to update cached service configuration data. Default: 1

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > OpenAM Services > Service Data Cache Time.

`com.sun.identity.sm.notification.enabled`

When enabled, receive notification from OpenAM to update service configuration data caches.

For centralized configurations this property is configured under Access Control > *`Realm Name`* > Agents > J2EE > *`Agent Name`* > OpenAM Services > Enable Notification of Service Data Caches.

Index