Web Policy Agent Guide

/ Version 4.2

Latest update: 4.2.1.2

ForgeRock AS
201 Mission St., Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Guide to installing ForgeRock® Access Management web policy agents. ForgeRock Access Management provides authentication, authorization, entitlement, and federation software.

Table of Contents

Preface

This guide shows you how to install ForgeRock Access Management web server policy agents, as well as how to integrate with other access management software. Read the Web Agents Release Notes before you get started.

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

About ForgeRock Identity Platform™ Software

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

Chapter 1

Introducing Web Policy Agents

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.

1.1. About Web Policy Agents

A *policy agent* enforces policy for AM and protects all resources on the web server. The policy agent intercepts requests from users trying to access a protected web resource and denies access until the user has authorization from AM to access the resource.

Note

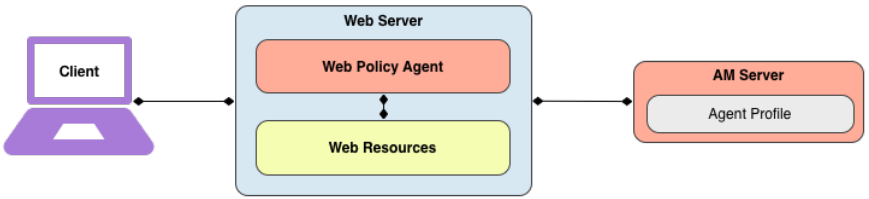
A single policy agent installation can hold multiple policy agent instances. Therefore, install only one policy agent per web server and configure as many agent instances as you require.

Installing more than one policy agent in a web server is not supported.

1.1.1. Web Policy Agent Components

The web policy agent provides fast installation and light touch integration to protect the resources on the supported web server. The web agent consists of a web server plugin matching the API requirements of the particular web server and a native module that interfaces with AM for its services.

Web Policy Agent



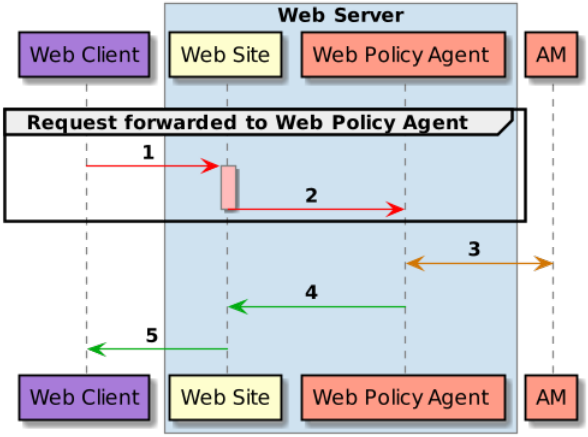
1.1.2. How the User, Web Policy Agent, and Access Management Interact

Imagine that a user attempts to access a protected resource before having authenticated by pointing the user's browser to a web page. Assume that you have configured AM to protect the web page. Then, the web policy agent intercepting the user's browser's request finds no session token in the request, and so redirects the user's browser to the AM login page for authentication. After the user has successfully authenticated, AM sets a session token in a browser cookie, and redirects the browser back to the page the user 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 AM. Given the valid session token, the policy agent gets a policy decision from AM concerning whether the user can access the page. If AM's Policy Service determines that the user is allowed to access the page, AM 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.

Web Policy Agent Interaction



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. Here is how it works:

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 AM policy. The policy agent acts to enforce policy, whereas the policy configuration and decisions are handled by AM.
3. The policy agent communicates with AM to get the policy decision to enforce.

4. For a resource to which AM approves access, the policy agent allows access.
5. The web server returns the requested access to the web client.

1.2. Web Policy Agent Features

The Web policy agent provides a number of additional features useful for your deployment, some of which are described below.

1.2.1. Multiple Sites and Virtual Host Support

Web policy agent instances can be configured to operate with multiple websites in IIS, and with multiple virtual hosts in Apache.

Each configuration instance is independent and has its own configuration file, debug logs, and audit logs. Each instance can connect to a different AM realm, or even different AM servers.

For more information, see "Installing Apache Web Policy Agents into a Virtual Host" and "Installing IIS Web Policy Agent".

1.2.2. Web Agent SSO Only Mode

The agent intercepts all inbound client requests to access a protected resource and processes the request based on a global configuration property, `com.sun.identity.agents.config.sso.only`. The configuration setting determines the mode of operation that should be carried out on the intercepted inbound request.

When `com.sun.identity.agents.config.sso.only` is `true`, the web policy agent only manages user authentication. The filter invokes the AM Authentication Service to verify the identity of the user. If the user's identity is verified, the user is issued a session token through AM's Session Service.

When `com.sun.identity.agents.config.sso.only` is `false`, which is the default, the web policy agents will also manage user authorization, by using the policy engine in AM.

For more information, see "Configuring Web Policy Agent SSO Properties".

1.2.3. Not-Enforced URL and Client IP Lists

The policy agent supports properties to bypass authentication and grant immediate access to resources not requiring protection, such as images, stylesheets, or static HTML pages.

You can configure a Not-Enforced URL List using the `com.sun.identity.agents.config.notenforced.url` property that grants the user access to resources whose URLs match those in the list.

For example, you can set URL patterns with wildcards in the AM console using the following patterns:

```
/logout.html
/images/*
/css/-*-
/*.jsp?locale=*
```

For more information on wildcard usage, see Specifying Resource Patterns with Wildcards.

The policy agent supports a Not-Enforced Client IP List, which specifies the client IP addresses that can be excluded from authentication and authorization. This property is useful to allow administrators access to the web site from a certain IP address or allow a search engine access to the web resources.

For finer control, you can configure a not-enforced policy that applies to requests to specified URLs, which also come from a list of specified IP addresses. See Not Enforced URL from IP Processing Properties (Not yet in the AM console)[2].

For more information on not-enforced lists, see "Configuring Web Policy Agent Application Properties".

1.2.4. Attribute Fetch Modes

Web policy agents provide the capability to fetch and inject user information into HTTP headers, request objects, and cookies and pass them on to the protected client applications. The client applications can then personalize content using these attributes in their web pages or responses.

Specifically, you can configure the type of attributes to be fetched and the associated mappings for the attributes names used in AM to those values used in the containers. The web policy agent securely fetches the user and session data from the authenticated user as well as policy response attributes.

For example, you can have a web page that addresses the user by name retrieved from the user profile, for example "Welcome Your Name!" AM populates part of the request (header, form data) with the CN from the user profile, and the web site consumes and displays it.

For more details, see Profile Attributes Processing Properties.

1.2.5. FQDN Checking

The web policy agent requires that clients accessing protected resources use valid URLs with fully qualified domain names (FQDNs). If invalid URLs are referenced, policy evaluation can fail as the FQDN will not match the requested URL, leading to blocked access to the resource. Misconfigured URLs can also result in incorrect policy evaluation for subsequent access requests.

There are cases where clients may specify resource URLs that differ from the FQDNs stored in AM policies, for example, in load balanced and virtual host environments. To handle these cases, the web policy agent supports FQDN Checking properties: `FQDN Default` and `FQDN Virtual Host Map` properties.

The `FQDN Default` property specifies the default URL with valid hostname. The property ensures that the policy agent can redirect to a URL with a valid hostname should it discover an invalid URL in the client request.

The `FQDN Virtual Host Map` property stores map keys and their corresponding values, allowing invalid URLs, load balanced URLs, and virtual host URLs to be correctly mapped to valid URLs. Each entry in the Map has precedence over the `FQDN Default` setting, so that if no valid URLs exist in the `FQDN Virtual Host Map` property, the agent redirects to the value specified in the `FQDN Default` property.

If you want the agent to redirect to a URL other than the one specified in the `FQDN Default` property, then it is good practice to include any anticipated invalid URLs in the `FQDN Virtual Host Map` property and map it to a valid URL.

For more details, see Fully Qualified Domain Name Checking Properties.

1.2.6. Cookie Reset Properties

AM provides cookie reset properties that the agent carries out prior to redirecting the client to a login page for authentication.

Cookie reset is typically used when multiple parallel authentication mechanisms are in play with the policy agent and another authentication system. The policy agent can reset the cookies set by the other mechanism before redirecting the client to a login page.

Note

To be able to set, and reset secure or HTTP Only cookies, in addition to the cookie reset properties, you must also set the relevant cookie option, as follows:

- To reset secure cookies, enable the `com.sun.identity.agents.config.cookie.secure` property.
- To reset HTTP only cookies, enable the `com.sun.identity.cookie.httponly` property.

For more information about these properties, see Cookie Properties.

The cookie reset properties include a name list specifying all of the cookies that will reset, a domain map specifying the domains set for each cookie, and a path map specifying the path from which the cookie will be reset.

If you have enabled attribute fetching using cookies to retrieve user data, it is good practice to use cookie reset, which will reset once you want to access an enforced URL without a valid session.

For more details, see Cookie Reset Properties.

1.2.7. Cross Domain Single Sign-On

Cross domain single sign-on (CDSSO) allows the web policy agent to transfer a validated stateful session ID between an AM domain and an application domain using a proprietary mechanism.

Normally, single sign-on cannot be implemented across domains as the session cookie from one domain (for example, website.com) is not accessible from another domain (for example, website.net).

AM's CDSSO solves this cross-domain problem and is best implemented in environments where all the domains are managed by the same organization, and where the AM server is configured to use stateful sessions. AM does not support CDSSO for deployments with stateless sessions.

The web policy agent works with an AM component called a `CDCServlet` that generates a self-submitting form containing the valid session token from one domain. The form gets auto-submitted to the policy agent endpoint via a POST operation. The policy agent processes the request and extracts the session ID, which is again validated by AM. If validation is successful, the policy agent sets the cookie in alternate domain. The client can then access a resource in that domain.

For more details, see *Configuring Cross Domain Single Sign-On*.

1.2.8. Supporting Load Balancers

The web policy agent provides a number of advanced properties for load balancer deployments fronting multiple policy agents. Properties are available to get the client IP and host name from the load balancer.

If the policy agent is running behind a load balancer, you can configure the policy agent to set a sticky cookie or a query parameter in the URL to ensure subsequent requests are routed to the same instance to preserve session data.

These mechanisms ensure that unauthenticated POST data can be preserved. Policy agents store POST data in the cache and do not share the data among the agents behind the load balancer.

For more details, see "Configuring Web Policy Agents Behind Load Balancers".

Also, web policy agents can communicate with an OpenAM site configured behind a load balancer. To improve OpenAM's server performance in this scenario, ensure that the value of the `amlbcookie` cookie is set up to the OpenAM's server ID. For more information, see *To Configure Site Load Balancing for Deployments With Stateful Sessions*.

Note

Web policy agents support more than one agent instance running on the same host by properly initializing the multi-process locks/semaphores during the bootstrap process.

Chapter 2

Implementing Web Policy Agents

You install policy agents in web servers and web application containers to enforce access policies AM applies to protected web sites and web applications. Policy agents depend on AM for all authentication and authorization decisions. The primary responsibility of policy agents is to enforce what AM decides in a way that is unobtrusive to the user.

Policy agent configuration is distinct from policy configuration. The only policy-like configurations that you apply to policy agents are:

- URLs to exclude from policy enforcement (*not enforced URLs*)
- Client IP addresses to exclude from policy enforcement (*not enforced IPs*)

2.1. Downloading and Unzipping the Agent

Navigate to the ForgeRock BackStage website and choose the agent to download based on your version, architecture, and operating system requirements. Remember to verify the checksum of the downloaded file against the checksum posted on the download page.

Unzip the file in the directory where you plan to store the policy agent's configuration and log files. The following directories are extracted:

`bin/`

Contains the installation and configuration program **agentadmin**.

`config/`

Contains configuration templates used by the **agentadmin** command during installation.

`instances/`

Contains configuration files, and audit and debug logs for individual instances of the web policy agents. The directory is empty when first extracted.

`legal/`

Contains licensing information including third-party licenses.

`lib/`

Contains shared libraries used by the policy agent.

`log/`

Contains log files written during installation. The directory is empty when first extracted.

2.2. Configuration Location

Policy agent configuration properties are either stored:

- Centrally, in the AM configuration store
- Locally, as a flat file

2.2.1. Centrally Stored Agent Configuration

By default, policy agent configuration settings are stored centrally in the AM configuration store. Storing the policy agent configuration centrally allows you to configure your policy agents by using the AM console, the **ssoadm** command line tool, or the REST API for easier management. Any property change made in AM is immediately communicated to the agent by using a notification. Many policy agent properties are hot-swap enabled, allowing the change to take effect immediately without restarting the policy agent.

You configure policy agents in realms. To access the centralized web policy agent configuration, select Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* in the AM console.

For more information on creating centrally-stored agent profiles, see "Creating Agent Profiles".

2.2.2. Locally Stored Agent Configuration

The policy agent installer can create a flat file with the agent configuration. The file is named `agent.conf` and is stored at the path `/web_agents/`*`agent_version`*`instances/Agent_`*`nnn`*`/config`.

If you choose to use a locally-stored agent configuration, you make all configuration changes by modifying property values in the `agent.conf` file. You cannot make changes using the AM console, command-line interface, or REST API.

When using a locally-stored agent configuration, provide valid values for configuration properties ending in the following strings:

- `.cookie.name`
- `.fqdn.default`
- `.agenturi.prefix`
- `.naming.url`

- `.login.url`
- `.instance.name`
- `.username`
- `.password`
- `.connection_timeout`
- `.policy_clock_skew`

The web policy agent installer populates properties required to connect to an AM instance. Additional properties are needed when settings are stored locally.

2.3. Creating Agent Profiles

A policy agent requires a profile to connect to and communicate with AM, regardless of whether it is stored centrally in AM or on the agent server.

To Create an Agent Profile in AM Using the Console

Create an agent profile using the AM console by performing the following steps:

1. In the AM console, navigate to Realms > *Realm Name* > Applications > Agents > Web, and then select the `New` button in the Agent table.
2. Complete the web form using the following hints:

   Name

   The name for the agent profile. This name is used during the agent installation.

   Password

   The password the agent uses to authenticate to AM. This password is used during the agent installation.

   Configuration

   The location where to the agent configuration is stored. Possible values are:

   - `Local`. The configuration is stored as a file in the agent installation. To manage the configuration, edit the file to add properties, remove properties, and change values.
   - `Centralized`. The configuration is stored in the AM configuration store. To manage the configuration, use the AM console.

Server URL

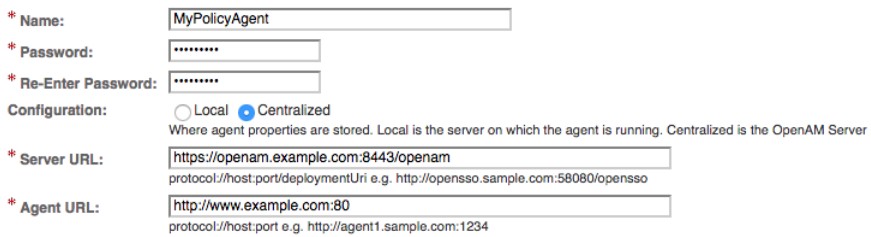
The full URL to an AM instance. If AM is deployed in a site configuration (behind a load balancer), enter the site URL.

In centralized configuration mode, Server URL is used to populate the agent profile for use with as login, logout, naming, and cross-domain SSO.

Agent URL

The URL the policy agent protects, such as `http://www.example.com:80`

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



To Create an Agent Profile Using the ssoadm Command

You can create a policy agent profile in AM using the **ssoadm** command-line tool. You do so by specifying the agent properties either as a list of attributes, or by using an agent properties file as shown below. Export an existing policy agent configuration before you start to see what properties you want to set when creating the agent profile.

Perform the following steps to create a policy agent profile using the **ssoadm** command:

1. Make sure the **ssoadm** command is installed. See the section Installing and Using the Tools in the *ForgeRock Access Management Install Guide*.

2. Determine the list of properties to set in the agent profile using the configuration exported previously and store them in a file, for example, `myPolicyAgent.properties`.

3. Create a password file, for example `$HOME/.pwd.txt`. The file should only contain the password string, on a single line.

   The password file must be read-only for the user who creates the policy agent profile, and must not be accessible to other users:

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

4. Create the agent profile, specifying `--agenttype WebAgent`:

```
$ ssoadm create-agent \
  --realm / \
  --agentname myPolicyAgent \
  --agenttype WebAgent \
  --adminid amadmin \
  --password-file $HOME/.pwd.txt \
  --datafile myPolicyAgent.properties

Agent configuration was created.
```

5. Review the new profile in the AM console under Realms > *Realm Name* > Applications > Agents > Web > *Agent Name*.

To Create an Agent Profile Group and Inherit Settings

Agent profile groups let you set up multiple agents to inherit settings from the group. To create a new agent profile group, perform the following steps:

1. In the AM console, navigate to Realms > *Realm Name* > Applications > Agents > Web.

2. Select New in the Group table, and provide a name for the group and the URL to the AM server in which to store the profile.

   After creating the group profile, you can select the link to the new group profile to fine-tune or export the configuration.

3. Inherit group settings by selecting your agent profile, and then selecting the group name in the Group drop-down list near the top of the profile page.

   You can then adjust inheritance by clicking Inheritance Settings on the OpenAM Services agent profile tab.

2.4. Configuring Web Policy Agents Behind Load Balancers

This section addresses the question of configuring policy agents on protected servers that operate behind network load balancers.

2.4.1. The Role of the Load Balancing Layer

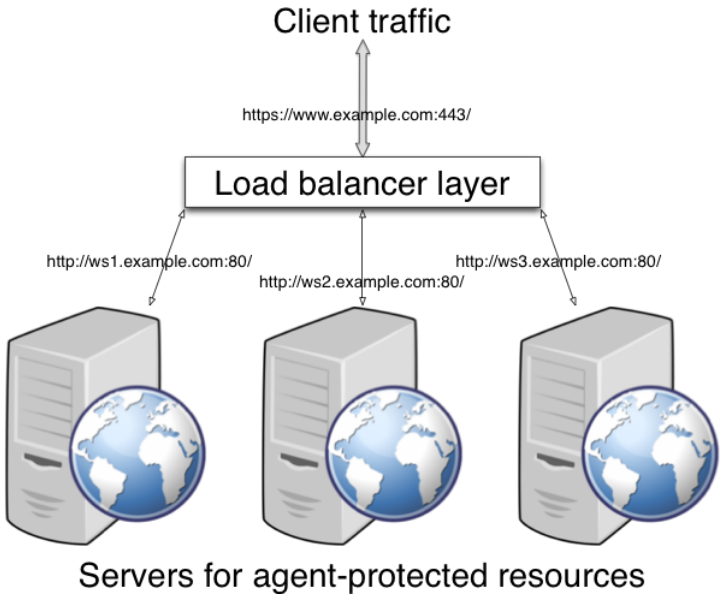
A load balancing layer that stands between clients and protected servers can distribute the client load, and fail client traffic over when a protected server goes offline. In the simplest case, the load balancing layer passes requests from the clients to servers and responses from servers to clients, managing the traffic so the client experience is as smooth as possible.

Load Balancing With the Same Protocol and Port

If your deployment has protocols and port numbers on the load balancer that match those of the protected servers, see "When Protocols and Port Number Match".

A load balancing layer can also offload processor-intensive public-key encryption algorithms involved in SSL transactions to a hardware accelerator, reducing the load on the protected servers. The client connects to the load balancer over HTTPS, but the load balancer connects to the servers over HTTP.

Load Balancing With SSL Offloading



If your deployment uses SSL offloading, see "When Protocols and Port Number Differ".

2.4.2. When Protocols and Port Number Match

When the protocol on the load balancer, such as HTTP or HTTPS, matches the protocol on the protected web server, and the port number the load balancer listens on, such as 80 or 443, matches the port number the protected web server listens on, then the main difference between URLs is in the host names. Map the agent host name to the host name for the load balancer.

To Map the Agent Host Name to the Load Balancer Host Name

When protocols and port numbers match, configure fully qualified domain name (FQDN) mapping.

This procedure explains how to do so for a centralized web policy agent profile configured in the AM console. The steps also mention the properties for web agent profiles that rely on local, file-based configurations:

1. Log in to the AM console as an administrative user with rights to modify the policy agent profile.

2. Navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* to open the web agent profile for editing.

3. In the Global tab page section Fully Qualified Domain Name Checking, make sure FQDN checking is selected (the default).

   The equivalent property setting is `com.sun.identity.agents.config.fqdn.check.enable=true`.

4. Set FQDN Default to the fully qualified domain name of the load balancer, such as `lb.example.com`, rather than the protected server FQDN where the policy agent is installed.

   The equivalent property setting is `com.sun.identity.agents.config.fqdn.default=lb.example.com`.

5. Set FQDN Virtual Host Map to map the protected server FQDN to the load balancer FQDN, for example, where the key `agent.example.com` (protected server) has value `lb.example.com` (load balancer).

   The equivalent property setting is `com.sun.identity.agents.config.fqdn.mapping[agent.example.com]=lb.example.com`.

6. Save your work, and then restart the protected server.

2.4.3. When Protocols and Port Number Differ

When the load balancer protocol and port, such as HTTPS and 443, differ from the protocol on the protected web server, such as HTTP and 80, then you must override these in the policy agent configuration.

To Override Protocol, Host, and Port

Use the Agent Deployment URI Prefix setting to override the agent protocol, host, and port with that of the load balancer.

Important

The web policy agent configuration for SSL offloading has the side effect of preventing FQDN checking and mapping. As a result, URL rewriting and redirection does not work correctly when the policy agent is accessed directly and not through the load balancer. This should not be a problem for client traffic, but potentially could be an issue for applications accessing the protected server directly, from behind the load balancer.

This procedure explains how to do so for a centralized web policy agent profile configured in the AM console. The steps also mention the properties for web agent profiles that rely on local, file-based configurations:

1. Log in to the AM console as an administrative user with rights to modify the policy agent profile.

2. Navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* to open the web agent profile for editing.

3. In the Global tab page Profile section, set the Agent Deployment URI Prefix to that of the load balancer.

The value you set here is used when overriding protocol, host, and port on the protected server with the web policy agent.

The property to set is `com.sun.identity.agents.config.agenturi.prefix`.

4. In the Advanced tab page Load Balancer section, enable Load Balancer Setup.

   The equivalent property setting is `com.sun.identity.agents.config.load.balancer.enable=true`.

5. Enable Override Request URL Protocol.

   The equivalent property setting is `com.sun.identity.agents.config.override.protocol=true`.

6. Enable Override Request URL Host.

   The equivalent property setting is `com.sun.identity.agents.config.override.host=true`.

7. Enable Override Request URL Port.

   The equivalent property setting is `com.sun.identity.agents.config.override.port=true`.

8. Enable Notification URL when the web policy agent gets notifications about configuration changes.

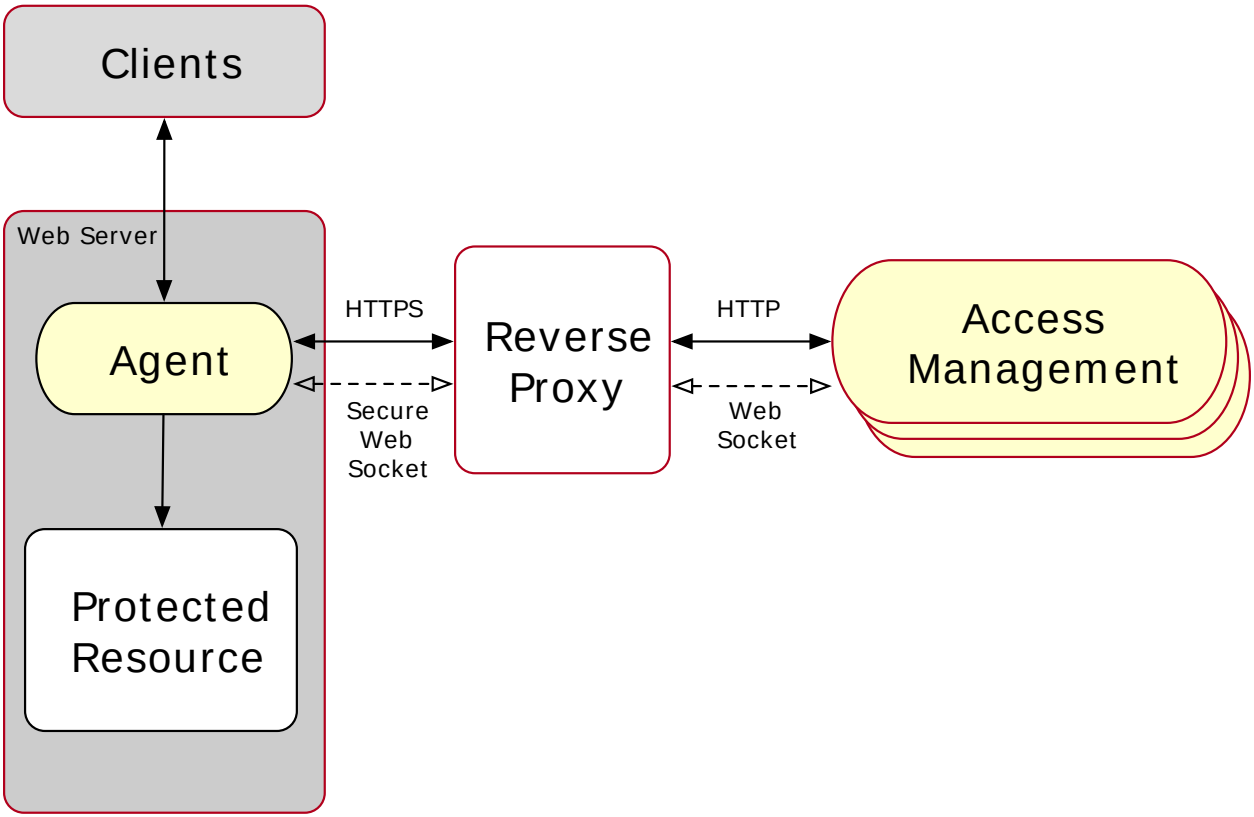
   The equivalent property setting is `com.sun.identity.agents.config.override.notification.url=true`.

9. Save your work, and then restart the protected server.

2.5. Preparing the Environment for Reverse Proxies

One of the most common deployment scenarios of AM and agents is to configure a reverse proxy between them to secure and load-balance traffic, as shown in the following diagram:

Reverse Proxy Configuration Between the Agent and AM



This section demonstrates how to configure Apache HTTP Server as a reverse proxy between AM and the agent, but you can use any reverse proxy that supports the WebSocket protocol. Refer to this section as an example of what you need to configure in your environment.

Before installing the agents in your environment, ensure you have configured the proxy as described in the following procedure:

To Configure Apache as a Reverse Proxy

This procedure demonstrates how to configure Apache HTTP Server as a reverse proxy between an agent and a single AM instance. Refer to the Apache documentation to configure Apache for load balancing and any other requirement for your environment.

1. Locate the `httpd.conf` file in your deployed reverse proxy instance.
2. Add the modules required for a proxy configuration as follows:

```
# Modules required for proxy
    LoadModule proxy_module modules/mod_proxy.so
    LoadModule proxy_http_module modules/mod_proxy_http.so
    LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
```

The `mod_proxy_wstunnel.so` module is required to support the WebSocket protocol used for notification between AM and the agents.

3. Add the proxy configuration. Consider the following directives:

```
[../../resources/reverse-proxy-example.conf]
```

Key:

❶ **RequestHeader**. Set this directive to `https` or `http` depending on the proxy configuration. If the proxy is configured for https, as in the example depicted in the diagram above, set the directive to `https`. Otherwise, set it to `http`.

In the next procedure you configure AM to recognize the forwarded header and use it in the `goto` parameter for redirecting back to the agent after authentication.

❷ **ProxyPass**. Set this directive to allow WebSocket traffic between AM and the agent.

If you have HTTPS configured between the proxy and AM, set the directive to use the `wss` protocol instead of `ws`.

❸ **ProxyPass**. Set this directive to allow HTTP traffic between AM and the agent.

❹ **ProxyPassReverseCookieDomain**. Set this directive to rewrite the domain string in `Set-Cookie` headers in the format *internal domain* (AM's domain) *public domain* (proxy's domain).

❺ **ProxyPassReverse**. Set this directive to the same value configured for the `ProxyPass` directive.

For more information about configuring Apache as a reverse proxy, refer to the Apache documentation.

4. Restart the reverse proxy instance.

5. Configure AM to obtain the header forwarded by the `RequestHeader` directive. See "To Configure Access Management to Use Forwarded Headers".

To Configure Access Management to Use Forwarded Headers

AM determines the base URL of the agent using the incoming HTTP request. When the agent is configured behind a proxy, AM cannot determine the base URL as expected, and functionality such as redirection using the `goto` parameter would not work.

AM supports configuring the Base URL Source service on the realm to configure options for obtaining the base URL.

1. Log in to the AM console with an administrative user, such as `amAdmin`.

2. Navigate to Realms > *Realm Name* > Services.

3. Select Add a Service, select Base URL Source, and then select Create, leaving the fields empty.
4. Configure the service with the following properties:
   - **Base URL Source**: X-Forwarded-* headers

     This property allows AM to retrieve the base URL from the `Forwarded` header field in the HTTP request. The Forwarded HTTP header field is standardized and specified in *RFC 7239*.
   - **Context path**: *AM's deployment uri*. For example, `/openam`.

   Leave the rest of the fields empty.

   Tip

   For more information about the Base URL Source service, see Base URL Source in the *ForgeRock Access Management Reference*.
5. Save your changes.
6. Install the agent. For more information, see "*Implementing Web Policy Agents*".

Chapter 3

Installing Web Policy Agents in Apache HTTP Server

This chapter covers prerequisites and installation procedures for Web Policy Agents 4.2 into Apache HTTP Servers 2.2.x, 2.4.x, and IBM HTTP Server.

3.1. Before You Install

1. Download the policy agent from BackStage. For more information, see "Downloading and Unzipping the Agent".
2. Consider the following points before installing web policy agents on Apache HTTP Server:
   - Avoid installing the web server and the web policy agent as root. Instead, create a web server user and install as that user.
   - The web policy agent replaces authentication functionality provided by Apache, for example, the `mod_auth_*` modules. Integration with built-in Apache httpd authentication directives, such as `AuthName`, `FilesMatch`, and `Require` is not supported.
   - SELinux can prevent the web server from accessing agent libraries and the agent from being able to write to audit and debug logs. See "*Troubleshooting*".
   - Ensure AM is installed and running, so that you can contact AM from the system running the policy agent.
   - Ensure the OpenSSL libraries are either located or referenced as shown in the following table:

OpenSSL Libraries Location by Operating System

Operating System	OpenSSL Library	Location or Variable
Windows 32-bit	`libeay32.dll` `ssleay32.dll`	`\windows\syswow64`
Windows 64-bit	`libeay32.dll` `ssleay32.dll`	`\windows\system32`
Linux	`libcrypto.so` `libssl.so`	`$LD_LIBRARY_PATH` or `$LD_LIBRARY_PATH_64`
Oracle Solaris X86/SPARC	`libcrypto.so`	`$LD_LIBRARY_PATH`

Operating System	OpenSSL Library	Location or Variable
	`libssl.so`	
AIX	`libcrypto.a` `libssl.a`	`$LIBPATH`

For information about supported OpenSSL libraries, see "Supported OpenSSL Versions" in the *Web Agents Release Notes*.

Note

On Windows operating systems the web policy agents use the native Windows SSL libraries by default if the AM server you will be connecting to uses SSL. You can choose to use OpenSSL instead.

3.1.1. Tuning Apache Multi-Processing Modules

Apache 2.0 and later comes with Multi-Processing Modules (MPMs) that extend the basic functionality of a web server to support the wide variety of operating systems and customizations for a particular site.

The key area of performance tuning for Apache is to run in worker mode ensuring that there are enough processes and threads available to service the expected number of client requests. Apache performance is configured in the `conf/extra/http-mpm.conf` file.

The key properties in this file are `ThreadsPerChild` and `MaxClients`. Together the properties control the maximum number of concurrent requests that can be processed by Apache. The default configuration allows for 150 concurrent clients spread across 6 processes of 25 threads each.

```
<IfModule mpm_worker_module>
   StartServers          2
   MaxClients          150
   MinSpareThreads      25
   MaxSpareThreads      75
   ThreadsPerChild      25
   MaxRequestsPerChild   0
</IfModule>
```

Important

For the policy agent notification feature, the `MaxSpareThreads`, `ThreadLimit` and `ThreadsPerChild` default values must *not* be altered; otherwise the notification queue listener thread cannot be registered.

Any other values apart from these three in the worker MPM can be customized. For example, it is possible to use a combination of `MaxClients` and `ServerLimit` to achieve a high level of concurrent clients.

3.2. Installing Apache Web Policy Agents

Complete the following procedures to install Web Policy Agents 4.2 into Apache HTTP Servers:

To Complete Pre-Installation Tasks

Perform the following steps to create the configuration required by the policy agent before installing it:

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Implementing Authorization Using the Access Management Console*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources. This allows you to test your policy agent after installation.

2. Create an agent profile in AM, required by the policy agent to connect and communicate with AM. For more information, see "Creating Agent Profiles".

3. Configure your AM instance to support single sign-on (SSO) or cross-domain SSO. Choose one of the following options depending on your environment:

   - To configure an AM instance and a policy agent on two different cookie domains, such as `example.org` and `example.net`, set up cross-domain SSO.

     For more information, see *Implementing Cross-Domain Single Sign-On*.

   - To configure an AM instance and a policy agent on the same cookie domain, such as `example.net`, set up SSO.

     For more information, see *Implementing Single Sign-On Within One Domain*.

4. Create a text file containing only the password specified when creating the agent profile, and protect it:

   Windows example:

   ```
   C:\> echo password > pwd.txt
   ```

   In Windows Explorer, right-click the password file, for example `pwd.txt`, select Read-Only, and then click OK.

   UNIX example:

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

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

To Install the Web Policy Agent into Apache

This procedure shows how to install into Apache 2.4. If installing into Apache 2.2, alter the path names accordingly.

1. Check the information in "Before You Install" before proceeding.
2. Shut down the Apache server where you plan to install the agent.
3. Make sure AM is running.
4. Run the **agentadmin --i** command to install the agent. You will be prompted to read and accept the software license agreement for the agent installation.

   - UNIX example:

     ```
     $ cd /web_agents/apache24_agent/bin/
     $ ./agentadmin --i
     ```

   - Windows example:

     ```
     C:\> cd web_agents\apache24_agent\bin
     C:\path\to\web_agents\apache24_agent\bin> agentadmin.exe --i
     ```

5. When prompted for information, enter the inputs appropriate for your deployment.

   Tip

   You can cancel web policy agent installation at anytime by pressing **CTRL+C**

   a. Enter the full path to the Apache HTTP Server configuration file. The installer modifies this file to include the web policy agent configuration and module.

      ```
      Enter the complete path to the httpd.conf file which is used by Apache HTTPD
      Server to store its configuration.
      [ q or 'ctrl+c' to exit ]
      Configuration file [/opt/apache/conf/httpd.conf]: /etc/httpd/conf/httpd.conf
      ```

   b. When installing the policy agent as the `root` user, the **agentadmin** command can change the directory ownership to the same user and group specified in the Apache configuration. Determine which user or group is running the Apache HTTP server by viewing the `Group` and `User` directives in the Apache HTTP server configuration file. Enter `yes` to alter directory ownership, press **Enter** to accept the default: `no`.

      ```
      Change ownership of created directories using
      User and Group settings in httpd.conf
      [ q or 'ctrl+c' to exit ]
      (yes/no): [no]: yes
      ```

Failure to set permissions causes issues, such as the Apache HTTP server not starting up, getting a blank page when accessing a protected resource, or the policy agent generating errors during log file rotation.

c. The installer can import settings from an existing web policy agent into the new installation and skips prompts for any values present in the existing configuration file. You will be required to re-enter the agent profile password.

Enter the full path to an existing agent configuration file to import the settings, or press **Enter** to skip the import.

```
To set properties from an existing configuration enter path to file
[ q or 'ctrl+c' to exit, return to ignore ]
Existing agent.conf file:
```

d. Enter the full URL of the AM instance the web policy agents will be using. Ensure that the deployment URI is specified.

```
Enter the URL where the AM server is running. Please include the
deployment URI also as shown below:
(http://openam.sample.com:58080/openam)
[ q or 'ctrl+c' to exit ]
OpenAM server URL: http://openam.example.com:8080/openam
```

e. Enter the full URL of the server the agent is running on.

```
Enter the Agent URL as shown below:
(http://agent.sample.com:1234)
[ q or 'ctrl+c' to exit ]
Agent URL: http://www.example.com:80
```

f. Enter the name given to the agent profile created in AM.

```
Enter the Agent profile name
[ q or 'ctrl+c' to exit ]
Agent Profile name: webagent4
```

g. Enter the AM realm containing the agent profile.

```
Enter the Agent realm/organization
[ q or 'ctrl+c' to exit ]
Agent realm/organization name: [/]: /
```

h. Enter the full path to the file containing the agent profile password created earlier.

```
Enter the path to a file that contains the password to be used
for identifying the Agent
[ q or 'ctrl+c' to exit ]
The path to the password file: /tmp/pwd.txt
```

i. The installer displays a summary of the configuration settings you specified.

- If a setting is incorrect, type `no`, or press **Enter**. The installer loops through the configuration prompts again, using your provided settings as the default. Press **Enter** to accept each one, or enter a replacement setting.
- If the settings are correct, type `yes` to proceed with installation.

```
Installation parameters:

   OpenAM URL: http://openam.example.com:8080/openam
   Agent URL: http://www.example.com:80
   Agent Profile name: webagent4
   Agent realm/organization name: /
   Agent Profile password source: /tmp/pwd.txt

Confirm configuration (yes/no): [no]: yes
Validating...
Validating... Success.
Cleaning up validation data...
Creating configuration...
Installation complete.
```

Upon successful completion, the installer adds the agent as a module to the Apache HTTP Server configuration file. You can find a backup configuration file in the Apache HTTP Server configuration directory, called `http.conf_amagent_date_and_time_of_installation`.

The installer also sets up configuration and log directories for the agent instance. 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 located under the directory `web_agents/apache24_agent/instances/agent_1/`.

The configuration files and log locations are as follows:

`config/agent.conf`

Contains the bootstrap properties the web policy agent requires to connect to AM and download its configuration. Also contains properties that are only used if you configure the web policy agent to use local configuration.

`logs/audit/`

Audit log directory, used if the `local` or `all` audit locations are enabled.

`logs/debug/`

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

6. Ensure the user or group running the Apache HTTP server has the appropriate permissions on the following directories:

Read Permission

- `/web_agents/apache_24_agent/lib`

Read and Write Permission

- `/web_agents/apache_24_agent/instances/agent_nnn`
- `/web_agents/apache_24_agent/log`

To determine which user or group is running the Apache HTTP server, check the `Group` and `User` directives in the Apache HTTP server configuration file.

Failure to set permissions causes issues, such as the Apache HTTP server not starting up, getting a blank page when accessing a protected resource, or the policy agent generating errors during log file rotation.

> **Note**
>
> You may see the same issues if SELinux is enabled in `enforcing` mode and it is not configured to allow access to agent directories. For more information, see "*Troubleshooting*".

7. Start the Apache HTTP server.

To Check the Policy Agent Installation

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

```
[Tue Sep 08 15:51:27.667625 2016] AH00163:
 Apache/2.4.6 (CentOS) OpenAM Web Agent/4.2 configured
 -- resuming normal operations
```

2. Make an HTTP request to a resource protected by the agent, then check the `/web_agents/apache24_agent/instances/Agent_1/logs/debug/debug.log` file to verify that no errors occurred on startup. Expected output should resemble the following:

```
2016-11-18 11:59:22.255 +0000    INFO [4900:6260]

    ########################################################
    OpenAM Web Agent
    Version: 4.2
    Revision: 5bf61d2
    Build date: Nov  8 2016 11:29:54
    ########################################################
```

3. (Optional) If you have a policy configured, you can test that your policy agent is processing requests. For example, when you make an HTTP request to a resource protected by the agent you should be redirected to AM to authenticate. As an example, authenticate as user `demo`, password `changeit`. After you authenticate, AM redirects you back to the resource you tried to access.

3.3. Installing Apache Web Policy Agents into a Virtual Host

Complete the following procedures to install Web Policy Agents 4.2 into Apache HTTP Server virtual hosts.

Installing into an Apache virtual host is a manual process, which involves copying an instance directory created by the **agentadmin** installer and adding to the Apache configuration file of the virtual host.

You will also need to have installed a web policy agent into the default root Apache configuration file before installing into a virtual host. See "Installing Apache Web Policy Agents".

To Prepare for Policy Agent Installation on an Apache Virtual Host

Perform the following steps to create the configuration required to install a policy agent on an Apache virtual host:

1. Install a web policy agent in the default root configuration of the Apache HTTP Server installation. For more information, see "Installing Apache Web Policy Agents"
2. Create an agent profile in AM for the policy agent. For more information, see "Creating Agent Profiles".
3. Create at least one policy in AM to protect resources on the virtual host, as described in the procedure *Implementing Authorization Using the Access Management Console*.

To Install the Web Policy Agent into Apache Virtual Hosts

This procedure assumes you have installed a web policy agent into the default root configuration of your Apache HTTP Server installation, with configuration in `/web_agents/apache24_agent/instances/agent_1`. To install into a virtual host, copy this configuration folder, modify required settings, and enable the web policy agent in the virtual host configuration file.

1. Check the information in "Before You Install" before proceeding.
2. Shut down the Apache server where you plan to install the agent.
3. Locate the web policy agent configuration instance to duplicate, and make a copy, for example `agent_2`:
   - UNIX example:
     ```
     $ cd /web_agents/apache24_agent/instances
     $ cp -r agent_1 agent_2
     ```
   - Windows example:
     ```
     c:\> cd c:\web_agents\apache24_agent\instances
     c:\path\to\web_agents\apache24_agent\instances> xcopy /E /I agent_1 agent_2
     ```

4. Give the user that runs the virtual host modify privileges to the new instance folder. The following examples demonstrate giving privileges to the `agent_2` configuration instance to a user named *apache*:

   - UNIX example:

     ```
     $ cd /web_agents/apache24_agent/instances
     $ chown -hR apache agent_2
     ```

   - Windows example:

     ```
     c:\> cd c:\web_agents\apache24_agent\instances
     c:\path\to\web_agents\apache24_agent\instances> icacls "agent_2" /grant apache:M
     ```

5. In the new instance folder, edit the `/config/agent.conf` configuration file as follows:

   a. Alter the value of `com.sun.identity.agents.config.username` to be the name of the agent profile you created in AM for the virtual host.

   b. Configure the virtual host's policy agent encryption key and password. Consider the following scenarios and choose the one that suits your environment best:

      - **Scenario 1**. The password of the virtual host's agent profile is the same as the password of the Apache root's agent profile[1].

        The encryption key and encryption password of the Apache root's agent and the virtual host's agent must match. Because you copied the configuration file, you do not need to perform any additional action.

      - **Scenario 2**. The password of the virtual host's agent profile is different from the password of the Apache root's agent profile[1].

        You need to generate a new encryption key and encrypt the new password before configuring them in the virtual host's agent profile. Perform the following steps:

        1. Generate a new encryption key by running the **agentadmin** command with the `--k` option. For example:

           ```
           $ agentadmin --k
           Encryption key value: YWM0OThlMTQtMzMxOS05Nw==
           ```

        2. Unix users only: Store the agent profile password in a file, for example, `newpassword.file`.

        3. Encrypt the agent's profile password with the encryption key by running the **agentadmin** command with the `--p` option.

           Unix example:

[1] The Apache root's profile refers to the web policy agent installation you performed as part of the prerequisites to install web policy agents on virtual hosts.

```
$ ./agentadmin --p "YWM0OThlMTQtMzMxOS05Nw==" "`cat newpassword.file`"
Encrypted password value: 07bJOSeM/G8ydO4=
```

Windows example:

```
C:\path\to\web_agents\agent\bin>agentadmin.exe --p "YWM0OThlMTQtMzMxOS05Nw==" "newpassword"
Encrypted password value: 07bJOSeM/G8ydO4=
```

4. In the virtual host's `agent.conf` file, set the following properties:

   - `com.sun.identity.agents.config.key`. Its value is the generated encryption key. For example:

     ```
     com.sun.identity.agents.config.key = YWM0OThlMTQtMzMxOS05Nw==
     ```

   - `com.sun.identity.agents.config.password`. Its value is the encrypted password. For example:

     ```
     com.sun.identity.agents.config.password = 07bJOSeM/G8ydO4=
     ```

c. Replace any references to the original instance directory with the new instance directory. For example, replace the string `agent_1` with `agent_2` wherever it occurs in the configuration file.

   Configuration options that are likely to require alterations include:

   - `com.sun.identity.agents.config.local.logfile`
   - `com.sun.identity.agents.config.local.audit.logfile`

d. Replace any references to the original website being protected with the new website being protected. For example, replace `http://www.example.com:80/amagent` with `http://customers.example.com:80/amagent`.

   Configuration options that are likely to require alterations include:

   - `com.sun.identity.client.notification.url`
   - `com.sun.identity.agents.config.agenturi.prefix`
   - `com.sun.identity.agents.config.fqdn.default`

e. Save and close the configuration file.

6. Edit the Apache HTTP Server configuration file. This is the same file specified when installing the web policy agent into the default Apache website. For example, `/etc/httpd/conf/httpd.conf`.

   a. At the end of the file the installer will have added three new lines of settings, for example:

   ```
   LoadModule amagent_module /web_agents/apache24_agent/lib/mod_openam.so
   AmAgent On
   AmAgentConf /web_agents/apache24_agent/bin/../instances/agent_1/config/agent.conf
   ```

Leave the first line, `LoadModule ...`, and move the other two lines into the virtual host configuration element of the default site, for example:

```
<VirtualHost *:80>
# This first-listed virtual host is also the default for *:80
ServerName www.example.com
ServerAlias example.com
DocumentRoot "/var/www/html"
AmAgent On
AmAgentConf /web_agents/apache24_agent/instances/agent_1/config/agent.conf
</VirtualHost>
```

b. Copy the same two lines into the new virtual host, and replace `agent_1` with the new agent configuration instance folder, for example `agent_2`:

```
<VirtualHost *:80>
ServerName customers.example.com
DocumentRoot "/var/www/customers"
AmAgent On
AmAgentConf /web_agents/apache24_agent/instances/agent_2/config/agent.conf
</VirtualHost>
```

Tip

If the new virtual host configuration is in a separate file, copy the two configuration lines into the `VirtualHost` element within that file.

7. Save and close the Apache HTTP Server configuration file.

8. Ensure the user or group running the Apache HTTP server has the appropriate permissions on the following directories:

Read Permission

- `/web_agents/apache_24_agent/lib`

Read and Write Permission

- `/web_agents/apache_24_agent/instances/agent_nnn`
- `/web_agents/apache_24_agent/log`

To determine which user or group is running the Apache HTTP server, check the `Group` and `User` directives in the Apache HTTP server configuration file.

Failure to set permissions causes issues, such as the Apache HTTP server not starting up, getting a blank page when accessing a protected resource, or the policy agent generating errors during log file rotation.

Note

You may see the same issues if SELinux is enabled in `enforcing` mode and it is not configured to allow access to agent directories. For more information, see "*Troubleshooting*".

9. Start the Apache HTTP server.

To Check the Policy Agent Installation

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

```
[Tue Sep 08 15:51:27.667625 2016] AH00163:
 Apache/2.4.6 (CentOS) OpenAM Web Agent/4.2 configured
 -- resuming normal operations
```

2. Make an HTTP request to a resource protected by the agent, then check the `/web_agents/apache24_agent/instances/Agent_1/logs/debug/debug.log` file to verify that no errors occurred on startup. Expected output should resemble the following:

```
2016-11-18 11:59:22.255 +0000    INFO [4900:6260]

    ######################################################
    OpenAM Web Agent
    Version: 4.2
    Revision: 5bf61d2
    Build date: Nov  8 2016 11:29:54
    ######################################################
```

3. (Optional) If you have a policy configured, you can test that your policy agent is processing requests. For example, when you make an HTTP request to a resource protected by the agent you should be redirected to AM to authenticate. As an example, authenticate as user `demo`, password `changeit`. After you authenticate, AM redirects you back to the resource you tried to access.

3.4. Installing Apache Web Policy Agents Silently

You can run a silent, non-interactive installation by running **agentadmin --s**, along with arguments used to configure the instance.

The required arguments, and the order in which to specify them are:

Web server configuration file

Enter the full path to the Apache HTTP server configuration file. The installer modifies this file to include the web policy agent configuration and module.

OpenAM URL

Enter the full URL of the AM instance the web policy agents will be using. Ensure the deployment URI is specified.

Agent URL

Enter the full URL of the server the agent is running on.

Realm

Enter the AM realm containing the agent profile.

Agent profile name

Enter the name given to the agent profile created in AM.

Agent profile password

Enter the full path to the file containing the agent profile password.

`--changeOwner`

To have the installer change the ownership of created directories to be the same User and Group as specified in the Apache configuration, specify the optional `--changeOwner` switch.

`--acceptLicence`

You can suppress the license agreement prompt during a silent, non-interactive install by including the `--acceptLicence` parameter. The inclusion of the option indicates that you have read and accepted the terms stated in the license. To view the license agreement, open `/path/to/web_agents/`*`agent_type`*`/legal/Forgerock_License.txt`.

`--forceInstall`

Optionally have the installer proceed with a silent installation even if it cannot connect to the specified AM server during installation, rather than exiting.

Complete the following procedures to install a web policy agent silently into Apache HTTP Server:

To Complete Pre-Installation Tasks

Perform the following steps to create the configuration required by the policy agent before installing it:

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Implementing Authorization Using the Access Management Console*. Consider creating

a simple policy, such as a policy that allows only authenticated users to access your resources. This allows you to test your policy agent after installation.

2. Create an agent profile in AM, required by the policy agent to connect and communicate with AM. For more information, see "Creating Agent Profiles".

3. Configure your AM instance to support single sign-on (SSO) or cross-domain SSO. Choose one of the following options depending on your environment:

   - To configure an AM instance and a policy agent on two different cookie domains, such as `example.org` and `example.net`, set up cross-domain SSO.

     For more information, see *Implementing Cross-Domain Single Sign-On*.

   - To configure an AM instance and a policy agent on the same cookie domain, such as `example.net`, set up SSO.

     For more information, see *Implementing Single Sign-On Within One Domain*.

4. Create a text file containing only the password specified when creating the agent profile, and protect it:

   Windows example:

   ```
   C:\> echo password > pwd.txt
   ```

   In Windows Explorer, right-click the password file, for example `pwd.txt`, select Read-Only, and then click OK.

   UNIX example:

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

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

To install Web Policy Agents in Apache Silently

1. Check the information in "Before You Install" before proceeding.

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

3. Make sure AM is running.

4. Run the **agentadmin --s** command with the required arguments. For example:

```
$ agentadmin --s \
  "/etc/httpd/conf/httpd.conf" \
  "http://openam.example.com:8080/openam" \
  "http://www.example.com:80" \
  "/" \
  "webagent4" \
  "/tmp/pwd.txt" \
  --changeOwner \
  --acceptLicence

OpenAM Web Agent for Apache Server installation.

Validating...
Validating... Success.
Cleaning up validation data...
Creating configuration...
Installation complete.
```

5. Ensure the user or group running the Apache HTTP server has the appropriate permissions on the following directories:

   Read Permission

   - `/web_agents/apache_24_agent/lib`

   Read and Write Permission

   - `/web_agents/apache_24_agent/instances/agent_nnn`
   - `/web_agents/apache_24_agent/log`

   To determine which user or group is running the Apache HTTP server, check the `Group` and `User` directives in the Apache HTTP server configuration file.

   Failure to set permissions causes issues, such as the Apache HTTP server not starting up, getting a blank page when accessing a protected resource, or the policy agent generating errors during log file rotation.

   Note

   You may see the same issues if SELinux is enabled in `enforcing` mode and it is not configured to allow access to agent directories. For more information, see "*Troubleshooting*".

6. Start the Apache HTTP server.

3.5. Removing Apache Web Policy Agents

Complete the following steps to remove an Apache HTTP Server policy agent:

To remove Web Policy Agents from Apache HTTP Server

1. Shut down the Apache server where the agent is installed.
2. Run **agentadmin --l** to output a list of the installed web policy agent configuration instances.

   Make a note of the ID value of the configuration instance you want to remove.
3. Run **agentadmin --r**, and specify the ID of the web policy agent configuration instance to remove. A warning is displayed. Type `yes` to proceed with removing the configuration instance.

```
$ ./agentadmin --r agent_3

Warning! This procedure will remove all OpenAM Web Agent references from
a Web server configuration. In case you are running OpenAM Web Agent in a
multi-virtualhost mode, an uninstallation must be carried out manually.

Continue (yes/no): [no]: yes

Removing agent_3 configuration...
Removing agent_3 configuration... Done.
```

4. Start the Apache HTTP server.

Chapter 4

Installing Web Policy Agents in Microsoft IIS

This section covers prerequisites and installation procedures for Web Agents 4.2 on Microsoft Internet Information Services (IIS).

4.1. Before You Install

1. Download the web policy agent from BackStage. For more information, see "Downloading and Unzipping the Agent".

2. Consider the following points before installing web policy agents on IIS servers:

   - Ensure AM is installed and running, so that you can contact AM from the system running the web policy agent.

   - Web policy agents use the native Windows SSL libraries by default if the AM server uses SSL/TLS. You can choose to use OpenSSL instead.

     If you choose to use OpenSSL, ensure the OpenSSL libraries are available in the correct locations, as shown in the table below:

OpenSSL DLL Locations on 32-bit and 64-bit Windows

OpenSSL DLL	Location
`libeay32.dll` `ssleay32.dll`	`\windows\syswow64`
`libeay32.dll` `ssleay32.dll`	`\windows\system32`

Note

Windows 64-bit servers require both 32-bit and 64-bit OpenSSL libraries.

For information about supported OpenSSL libraries, see "Supported OpenSSL Versions" in the *Web Agents Release Notes*.

To enable OpenSSL, perform the following steps:

- Configure the `org.forgerock.agents.config.secure.channel.disable` property by performing the following steps:

  a. Edit the `/web_agents/iis_agent/instances/Agent_nnn/config/agent.conf` file.

  b. Add the `org.forgerock.agents.config.secure.channel.disable=true` property under the Bootstrap Properties section.

  c. Restart the IIS service.

  You must perform this step whether the web agent is configured in centralized mode or not.

- A web agent configured for a site or a parent application protects any application configured within, *provided that the application inherits the parent's IIS configuration*. The same is true for protected applications containing applications within.

  Web agents configured in a site or parent application do not protect children applications that do not inherit the parent's IIS configuration.

- If an application requires a specific web agent configuration, follow the procedures in this section to create a new web agent instance for it. Configuring a web agent on an application overrides the application's parent web agent configuration, if any.

  Important

  You must install the web agent on the child application before installing a web agent in the parent. Trying to install a web agent on a child that is already protected will result in error.

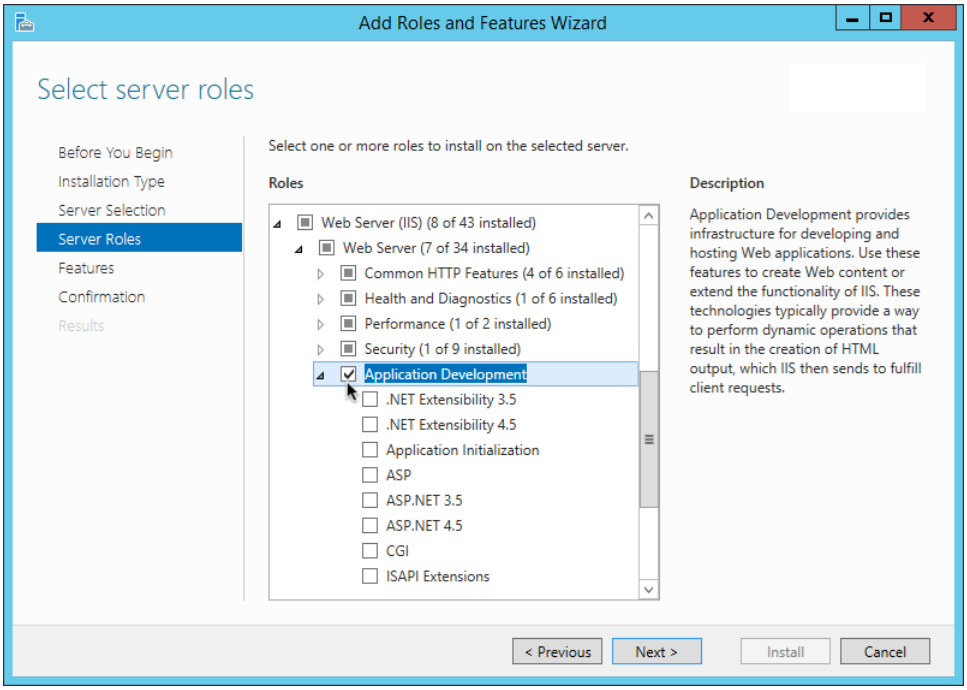
- You can disable the web agent protection at any level of the IIS hierarchy, with the following constraints:

  - Disabling the web agent in a parent application disables the protection on all children applications that do not have a specific web agent instance installed on them.

  - Disabling the web agent in a child application does not disable protection on its parent application.

- Web policy agents require that the *Application Development* component is installed alongside the core IIS services. Application Development is an optional component of the IIS web server. The component provides required infrastructure for hosting web applications.

Adding the Application Development Component to IIS



4.2. Installing IIS Web Policy Agent

Complete the following procedures to install Web Policy Agents 4.2 into Microsoft IIS servers.

To Complete Pre-Installation Tasks

Perform the following steps to create the configuration required by the policy agent before installing it:

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Implementing Authorization Using the Access Management Console*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources. This allows you to test your policy agent after installation.

2. Create an agent profile in AM, required by the policy agent to connect and communicate with AM. For more information, see "Creating Agent Profiles".

3. Configure your AM instance to support single sign-on (SSO) or cross-domain SSO. Choose one of the following options depending on your environment:

   - To configure an AM instance and a policy agent on two different cookie domains, such as `example.org` and `example.net`, set up cross-domain SSO.

     For more information, see *Implementing Cross-Domain Single Sign-On*.

   - To configure an AM instance and a policy agent on the same cookie domain, such as `example.net`, set up SSO.

     For more information, see *Implementing Single Sign-On Within One Domain*.

4. Create a text file containing only the password specified when creating the agent profile, and protect it:

   Windows example:

   ```
   C:\> echo password > pwd.txt
   ```

   In Windows Explorer, right-click the password file, for example `pwd.txt`, select Read-Only, and then click OK.

   UNIX example:

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

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

To Install the Policy Agent into IIS

1. Check the information in "Before You Install" before proceeding.
2. Log on to Windows as a user with administrator privileges.
3. Make sure AM is running.
4. Run **agentadmin.exe** with the `--i` switch to install the agent. You will be prompted to read and accept the software license agreement for the agent installation.

   ```
   c:\> cd web_agents\iis_agent\bin
   c:\web_agents\iis_agent\bin> agentadmin.exe --i
   ```

5. When prompted for information, enter the input appropriate for your deployment.

   Tip

   You can cancel the web policy agent installation at anytime by pressing **CTRL+C**

   a. Choose the site and application in which to install the web policy agent.

The **agentadmin** command reads the IIS server configuration and converts the IIS hierarchy into an ID composed of three values separated by the **.** character:

- The first value specifies an IIS site. The number `1` specifies the first site in the server.
- The second value specifies an application configured in an IIS site. The number `1` specifies the first application in the site.
- The third value specifies an internal value for the web policy agent.

The following is an example IIS server configuration read by the **agentadmin** command:

```
IIS Server Site configuration:
==================================
id       details
==================================

Default Web Site
application path:/, pool DefaultAppPool
1.1.1    virtualDirectory path:/, configuration: C:\inetpub\wwwroot\web.config

         MySite
         application path:/, pool: MySite
2.1.1    virtualDirectory path:/, configuration C:\inetpub\MySite\web.config
         application path:/MyApp1, pool: MySite
2.2.1    virtualDirectory path:/  configuration C:\inetpub\MySite\MyApp1\web.config
         application path:/MyApp1/MyApp2, pool: MySite
2.3.1    virtualDirectory path:/  configuration C:\inetpub\MySite\MyApp1\MyApp2\web.config

Enter IIS Server Site identification number.
[ q or 'ctrl+c' to exit ]
Site id: <userinput>2.1.1</userinput>
```

- The ID `2.1.1` corresponds to the first application, `/` configured in a second IIS site, `MySite`. You would choose this ID to install the web agent at the root of the site.
- The ID `2.2.1` corresponds to a second application, `MyApp1`, configured in a second IIS site, `MySite`. You would choose this ID to install the web agent in the `MyApp1` application.
- The ID `2.3.1` corresponds to a child application, `MyApp1/MyApp2`, configured in the second application, `MyApp1`, configured in a second IIS site, `MySite`. You would choose this ID to install the web agent in the sub-application, `MyApp1/MyApp2`.

b. The installer can import settings from an existing web agent on the new installation and skips prompts for any values present in the existing configuration file. You will be required to re-enter the agent profile password.

Enter the full path to an existing agent configuration file to import the settings, or press **Enter** to skip the import.

```
To set properties from an existing configuration enter path to file
[ q or 'ctrl+c' to exit, return to ignore ]
Existing agent.conf file:
```

c. Enter the full URL of the AM instance the web agents will be using. Ensure the deployment URI is specified.

> **Note**
>
> If your environment has a reverse proxy configured between AM and the agent, set the AM URL to the proxy URL instead. For example, `https://proxy.example.com:443/openam`. For more information about setting up the environment for reverse proxies, see "Preparing the Environment for Reverse Proxies".

```
Enter the URL where the AM server is running. Please include the
deployment URI also as shown below:
(http://openam.sample.com:58080/openam)
[ q or 'ctrl+c' to exit ]
OpenAM server URL: <userinput>https://openam.example.com:8443/openam</userinput>
```

d. Enter the full URL of the site the agent will be running in.

```
Enter the Agent URL as shown below:
(http://agent.sample.com:1234)
[ q or 'ctrl+c' to exit ]
Agent URL: <userinput>http://customers.example.com:8080</userinput>
```

e. Enter the name given to the agent profile created in AM.

```
Enter the Agent profile name
[ q or 'ctrl+c' to exit ]
Agent Profile name: <userinput>iisagent</userinput>
```

f. Enter the AM realm containing the agent profile.

```
Enter the Agent realm/organization
[ q or 'ctrl+c' to exit ]
Agent realm/organization name: [/]: <userinput>/</userinput>
```

g. Enter the full path to the file containing the agent profile password created earlier.

```
Enter the path to a file that contains the password to be used
for identifying the Agent
[ q or 'ctrl+c' to exit ]
The path to the password file: <userinput>c:\pwd.txt</userinput>
```

h. The installer displays a summary of the configuration settings you specified.

- If a setting is incorrect, type `no`, or press **Enter**. The installer loops through the configuration prompts using your provided settings as the default. Press **Enter** to accept each one, or enter a replacement setting.
- If the settings are correct, type `yes` to proceed with installation.

```
Installation parameters:

   OpenAM URL: https://openam.example.com:8443/openam
   Agent URL: http://customers.example.com:80
   Agent Profile name: iisagent
   Agent realm/organization name: /
   Agent Profile password source: c:\pwd.txt

Confirm configuration (yes/no): [no]: <userinput>yes</userinput>
Validating...
Validating... Success.
Cleaning up validation data...
Creating configuration...
Installation complete.
```

Upon successful completion, the installer adds the agent as a module to the IIS site configuration.

The installer also sets up configuration and log directories for the agent instance. 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 located under the directory `web_agents\iis_agent\instances\agent_1\`.

Note

The installer grants full access permissions on the created instance folder to the user that the selected IIS site is running under, so that log files can be written correctly.

The configuration files and log locations are as follows:

`config/agent.conf`

Contains the bootstrap properties the web policy agent requires to connect to AM and download its configuration. Also contains properties that are only used if you configure the web policy agent to use local configuration.

`logs/audit/`

Audit log directory, used if the `local` or `all` audit locations are enabled.

`logs/debug/`

Debug directory where the `debug.log` debug file resides. Useful in troubleshooting web policy agent issues.

6. (Optional) If you installed the web agent in an application, configure the web agent's CDSSO Redirect URI property, `com.sun.identity.agents.config.cdsso.redirect.uri`, to the application path by performing the following steps:

   a. Navigate to Realms > *Realm Name* > Agents > Web > *Agent Name* > SSO > Cross Domain SSO.

b. Add the application path to the default value of the CDSSO Redirect URI property. For example, if you installed the web agent in an application such as `MyApp1/MyApp2`, set the property to `MyApp1/MyApp2/agent/cdsso-oauth2`.

c. Save your changes.

4.3. Installing IIS Web Policy Agents Silently

You can run a silent, non-interactive installation by running **agentadmin.exe --s**, along with arguments used to configure the instance.

The required arguments, and the order in which to specify them are:

Web server configuration file

Enter the ID number of the IIS site in which to install the web policy agent. For a description of the supported IDs, see "To Install the Policy Agent into IIS".

> **Tip**
>
> To list the sites in an IIS server, run **agentadmin.exe --n**:

OpenAM URL

Enter the full URL of the AM instance the web policy agents will be using. Ensure the deployment URI is specified.

To balance agent connections to an AM site, enter the URL of the load balancer in front of the AM site.

> **Note**
>
> If your environment has a reverse proxy configured between AM and the agent, set the AM URL to the proxy URL instead. For example, `https://proxy.example.com:443/openam`. For more information about setting up the environment for reverse proxies, see "Preparing the Environment for Reverse Proxies".

Agent URL

Enter the full URL of the IIS site the agent will be running on.

Realm

Enter the AM realm containing the agent profile.

Agent profile name

Enter the name given to the agent profile created in AM.

Agent profile password

Enter the full path to the file containing the agent profile password.

`--changeOwner`

Optionally have the installer change the ownership of created directories to be the same user that is running the selected IIS site.

`--acceptLicence`

You can suppress the license agreement prompt during a silent, non-interactive install by including the `--acceptLicence` parameter. The inclusion of the option indicates that you have read and accepted the terms stated in the license. To view the license agreement, open `/path/to/web_agents/agent_type/legal/Forgerock_License.txt`.

`--forceInstall`

Add this optional switch to have the installer proceed with a silent installation even if it cannot connect to the specified AM server during installation, rather than exiting.

Complete the following procedures to install a web policy agent silently into an IIS server:

To Complete Pre-Installation Tasks

Perform the following steps to create the configuration required by the policy agent before installing it:

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Implementing Authorization Using the Access Management Console*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources. This allows you to test your policy agent after installation.
2. Create an agent profile in AM, required by the policy agent to connect and communicate with AM. For more information, see "Creating Agent Profiles".
3. Configure your AM instance to support single sign-on (SSO) or cross-domain SSO. Choose one of the following options depending on your environment:
   - To configure an AM instance and a policy agent on two different cookie domains, such as `example.org` and `example.net`, set up cross-domain SSO.

     For more information, see *Implementing Cross-Domain Single Sign-On*.
   - To configure an AM instance and a policy agent on the same cookie domain, such as `example.net`, set up SSO.

     For more information, see *Implementing Single Sign-On Within One Domain*.
4. Create a text file containing only the password specified when creating the agent profile, and protect it:

Windows example:

```
C:\> echo password > pwd.txt
```

In Windows Explorer, right-click the password file, for example `pwd.txt`, select Read-Only, and then click OK.

UNIX example:

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

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

To install Web Policy Agents in IIS Silently

1. Check the information in "Before You Install" before proceeding.
2. Make sure AM is running.
3. Run the **agentadmin --s** command with the required arguments. For example:

```
c:\web_agents\iis_agent\bin> <userinput>agentadmin.exe --s ^
  "2.1.1" ^
  "https://openam.example.com:8443/openam" ^
  "http://iis.example.com:80" ^
  "/" ^
  "iisagent" ^
  "c:\pwd.txt" ^
  --changeOwner ^
  --acceptLicence</userinput>

OpenAM Web Agent for IIS Server installation.

Validating...
Validating... Success.
Cleaning up validation data...
Creating configuration...
Installation complete.
```

4. (Optional) If you installed the web agent in a parent application, enable the web agent for its child applications by following the steps in

4.4. Enabling and Disabling IIS Web Policy Agents

The following table contains a list of procedures containing information about enabling and disabling IIS web agent protection:

Task	Section
Enable or disable web agent protection on an IIS site or application	Procedure 4.5
Enable or disable web agent protection on a child application	Procedure 4.6

To Enable and Disable a Web Policy Agent in an IIS site

Follow the steps on this procedure to enable and disable web agents installed in an application. Note that the **agentadmin** command only shows instances of the web agent; if you need to enable or disable the protection of children applications, see "To Disable And Enable Web Agent Protection for Children Applications".

1. Log on to Windows as a user with administrator privileges.
2. Run **agentadmin.exe --l** to output a list of the installed web policy agent configuration instances.

```
:\web_agents\iis_agent\bin> <userinput>agentadmin.exe --l</userinput>
OpenAM Web Agent configuration instances:

   id:            agent_1
   configuration: c:\web_agents\iis_agent\bin\..\instances\agent_1
   server/site:   2.2.1
```

   Make a note of the ID value of the configuration instance you want to disable or enable.
3. Perform one of the following steps:
   - To disable the web policy agent in a site, run **agentadmin.exe --d**, and specify the ID of the web policy agent configuration instance to disable.

```
c:\web_agents\iis_agent\bin> <userinput>agentadmin.exe --e agent_1</userinput>

Enabling agent_1 configuration...
Enabling agent_1 configuration... Done.
```

   - To enable the web policy agent in a site, run **agentadmin.exe --e**, and specify the ID of the web policy agent configuration instance to enable.

To Disable And Enable Web Agent Protection for Children Applications

Perform the steps in this procedure to enable and disable web agent protection for children applications:

1. Edit the child application's `web.config` configuration.
2. Decide whether to enable or disable web agent protection:
   - **Disabling web agent protection**. Add the following lines to the child application's `web.config` file:

```
<OpenAmModule enabled="false" configFile="C:\web_agents\iis_agent\instances\agent_1\config
\agent.conf" />
<modules>
   <add name="OpenAmModule64" preCondition="bitness64" />
</modules>
```

     Note that the path specified in `configFile` may be different for your environment.

- **Enabling web agent protection**. Web policy agents configured in a site or parent application also protect any applications that are inheriting the IIS configuration from that site or parent.

  If you have disabled the web policy agent's protection for a child application following the steps in this procedure, remove the lines added to the `web.config` file to enable protection again.

4.5. Enable IIS Basic Authentication and Password Replay Support

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

Given the proper configuration and with Active Directory as a user data store for AM, the IIS web policy agent can provide access to the IIS server variables. The instructions for configuring the capability follow in this section, though you should read the section in full, also paying attention to the required workarounds for Microsoft issues.

When configured as described, the policy agent requests IIS server variable values from AM, which gets them from Active Directory. The policy agent then sets the values in HTTP headers so that they can be accessed by your application.

The following IIS server variables all take the same value when set: `REMOTE_USER`, `AUTH_USER`, and `LOGON_USER`. The policy agent either sets all three, or does not set any of them.

When you enable Logon and Impersonation in the console (`com.sun.identity.agents.config.iis.logonuser=true` in the policy agent configuration), the policy agent performs Windows logon and sets the user impersonation token in the IIS session context.

When you enable Show Password in HTTP Header in the console (`com.sun.identity.agents.config.iis.password.header=true` in the policy agent configuration), the policy agent adds it in the `USER_PASSWORD` header.

The policy agent does not modify any other IIS server variables related to the authenticated user's session.

The policy agent works best with IIS running in Integrated, not Classic mode. In Classic mode, you cannot share sessions between the policy agent and another .NET application, so Logon and Impersonation are not operative. Furthermore IIS in Classic mode treats all modules as ISAPI extensions, and request processing is affected. It is therefore strongly recommended that you run IIS in Integrated mode:

- 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.

To Configure IIS Basic Authentication and Password Replay Support

Follow these steps:

1. Generate and store an encryption key:

   a. Generate the key using `com.sun.identity.common.DESGenKey` and the `.jar` files where you deployed AM, as in the following example. The Java command below is broken out into multiple lines for display purposes only:

   ```
   $ cd /tomcat/webapps/openam/WEB-INF/lib
   $ java -cp forgerock-util-3.0.0.jar:openam-core-14.0.0.jar:\
      openam-shared-14.0.0.jar com.sun.identity.common.DESGenKey
   Key ==> sxVoaDRAN0o=
   ```

   Windows users should use semi-colons (";"), instead of colons (":") in the commands. The Java command below is broken out into multiple lines for display purposes only:

   ```
   c:\> cd \tomcat\webapps\openam\WEB-INF\lib
   c:\> java -cp forgerock-util-3.0.0.jar;openam-core-14.0.0.jar; ^
       openam-shared-14.0.0.jar com.sun.identity.common.DESGenKey
   Key ==> sxVoaDRAN0o=
   ```

   b. In the AM console navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Microsoft IIS Server > Replay Password Key (property name: `com.sun.identity.agents.config.replaypasswd.key`), enter the generated key, and then click Save.

   c. In the AM console, navigate to Deployment > Servers > *Server Name* > Advanced > then add a property `com.sun.am.replaypasswd.key` with the key you generated as the value, and then click Save.

2. In the AM console, navigate to Realms > *Realm Name* > Authentication > Settings > Post Authentication Processing > Authentication Post Processing Classes, then add the class `com.sun.identity.authentication.spi.ReplayPasswd`, and then click Save.

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 data store, and you must configure the IIS web policy agent profile User ID Parameter and User ID Parameter Type so that the policy agent requests AM 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 the AM data store is configured to use Active Directory as the user data store.

In the AM console under Realms > *Realm Name* > Applications > 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 AM 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, navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Microsoft IIS Server, then select Show Password in HTTP Header, and then click Save.

5. To have the agent perform Windows logon (for user token impersonation), navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Microsoft IIS Server, then select Logon and Impersonation, and then click Save.

6. In the AM console, navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Microsoft IIS Server, then set Authentication Type to `basic`, and then click Save.

7. (Optional) If you access Microsoft Office from SharePoint pages, configure AM to persist the authentication cookie. For details, see "Persistent Cookie Module" in the *ForgeRock Access Management Authentication and Single Sign-On Guide*.

Chapter 5

Installing Web Policy Agents in NGINX Plus

This chapter covers prerequisites and installation procedures for Web Policy Agents 4.2 into NGINX Plus servers.

5.1. Before You Install

1. Download the policy agent from BackStage. For more information, see "Downloading and Unzipping the Agent".

2. Consider the following points before installing web policy agents on NGINX Plus servers:

   - Ensure AM is installed and running, so that you can contact AM from the system running the policy agent.

   - SELinux can prevent the web server from accessing agent libraries and the agent from being able to write to audit and debug logs. See "*Troubleshooting*".

5.2. Installing NGINX Plus Web Policy Agents

Complete the following procedures to install a web policy agent in an NGINX Plus server.

To Complete Pre-Installation Tasks

Perform the following steps to create the configuration required by the policy agent before installing it:

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Implementing Authorization Using the Access Management Console*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources. This allows you to test your policy agent after installation.

2. Create an agent profile in AM, required by the policy agent to connect and communicate with AM. For more information, see "Creating Agent Profiles".

3. Configure your AM instance to support single sign-on (SSO) or cross-domain SSO. Choose one of the following options depending on your environment:

- To configure an AM instance and a policy agent on two different cookie domains, such as `example.org` and `example.net`, set up cross-domain SSO.

  For more information, see *Implementing Cross-Domain Single Sign-On*.

- To configure an AM instance and a policy agent on the same cookie domain, such as `example.net`, set up SSO.

  For more information, see *Implementing Single Sign-On Within One Domain*.

4. Create a text file containing only the password specified when creating the agent profile, and protect it:

   Windows example:

   ```
   C:\> echo password > pwd.txt
   ```

   In Windows Explorer, right-click the password file, for example `pwd.txt`, select Read-Only, and then click OK.

   UNIX example:

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

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

To Install the Web Policy Agent into NGINX Plus

1. Check the information in "Before You Install" before proceeding.
2. Shut down the NGINX Plus server where you plan to install the agent.
3. Make sure AM is running.
4. Run the **agentadmin --i** command to install the agent. You will be prompted to read and accept the software license agreement for the agent installation:

   ```
   $ cd /web_agents/nginx12_agent/bin/
   $ ./agentadmin --i
   ```

5. When prompted for information, enter the inputs appropriate for your deployment.

   Tip

   You can cancel the policy agent installation at anytime by pressing **CTRL+C**

   a. Enter the full path to the NGINX Plus server configuration file, `nginx.conf`:

   ```
   Enter the complete path to your NGINX server configuration file
    [ q or 'ctrl+c' to exit ]
    [nginx.conf]: /etc/nginx/nginx.conf
   ```

b. The installer can import settings from an existing web policy agent into the new installation and skips prompts for any values present in the existing configuration file. You will be required to re-enter the agent profile password.

   Enter the full path to an existing agent configuration file to import the settings, or press **Enter** to skip the import:

```
To set properties from an existing configuration enter path to file
 [ q or 'ctrl+c' to exit, return to ignore ]
 Existing OpenSSOAgentBootstrap.properties file:
```

c. Enter the full URL of the AM instance this agent should connect to:

```
Enter the URL where the AM server is running. Please include the
 deployment URI also as shown below:
 (http://openam.sample.com:58080/openam)
 [ q or 'ctrl+c' to exit ]
 OpenAM server URL: https://openam.example.com:8443/openam
```

d. Enter the full URL of the server the agent is running on.

```
Enter the Agent URL as shown below:
 (http://agent.sample.com:1234)
 [ q or 'ctrl+c' to exit ]
 Agent URL: http://www.example.com:80
```

e. Enter the name given to the agent profile created in AM:

```
Enter the Agent profile name
 [ q or 'ctrl+c' to exit ]
 Agent Profile name: nginx_agent
```

f. Enter the AM realm containing the agent profile:

```
Enter the Agent realm/organization
 [ q or 'ctrl+c' to exit ]
 Agent realm/organization name: [/]: /
```

g. Enter the full path to the file containing the agent profile password created in the prerequisites:

```
Enter the path to a file that contains the password to be used
 for identifying the Agent
 [ q or 'ctrl+c' to exit ]
 The path to the password file: /tmp/pwd.txt
```

h. The installer displays a summary of the configuration settings you specified.

   - If a setting is incorrect, type `no`, or press **Enter**. The installer loops through the configuration prompts again, using your provided settings as the default. Press **Enter** to accept each one, or enter a replacement setting.

   - If the settings are correct, type `yes` to proceed with installation:

```
Installation parameters:

 OpenAM URL: https://openam.example.com:8443/openam
 Agent URL: http://www.example.com:80
 Agent Profile name: nginx_agent
 Agent realm/organization name: /
 Agent Profile password source: /tmp/pwd.txt

 Confirm configuration (yes/no): [no]:yes
 Validating...
 Validating... Success.

 Cleaning up validation data...

 Creating configuration...

 In order to complete the installation of the agent, update the configuration file /etc/nginx
/nginx.conf

 if this is the first agent in the installation, please insert the following directives into
 the top section of the NGINX configuration
 load_module /web_agents/nginx12_agent/lib/openam_ngx_auth_module.so;

 then insert the following directives into the server or location NGINX configuration sections
 that you wish this agent to protect:
 openam_agent on;
 openam_agent_configuration /web_agents/nginx12_agent/instances/agent_1/config/agent.conf;

 Please ensure that the agent installation files have read/write permissions for the NGINX
 server's user

 Please press any key to continue.

 Installation complete.
```

The installer sets up configuration and log directories for the agent instance. Each agent instance has its own numbered configuration and logs directories. The first agent is located under the directory `/web_agents/nginx12_agent/instances/agent_1/`.

The configuration files and log locations are as follows:

`config/agent.conf`

Contains the bootstrap properties the web policy agent requires to connect to AM and download its configuration. Also contains properties that are only used if the policy agent is configured to use local configuration.

`logs/audit/`

Audit log directory, used if the `local` or `all` audit locations are enabled.

`logs/debug/`

Debug directory that contains the `debug.log` file. Useful in troubleshooting policy agent issues.

6. Edit the NGINX Plus server configuration file `nginx.conf` to load the policy agent module `openam_ngx_auth_module.so`, if it is not already configured:

```
$ vi nginx.conf
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log notice;
pid        /var/run/nginx.pid;
load_module /web_agents/nginx12_agent/lib/openam_ngx_auth_module.so;
 ...
```

7. Edit the NGINX Plus server configuration file containing the context you want to protect and add policy agent directives to it. The following directives are supported:

   `openam_agent` ***`[on | off]`***

   Controls if an agent instance is `on` or `off` for a particular `http`, `server`, or `location` context.

   - Set the `openam_agent` directive to `on` for a context to protect it and its contents.

     If a context already protected requires a specific policy agent configuration, follow the procedures in this section again to create a new policy agent instance for it. The installer will configure the nex available policy agent instance, for example, `agent_2`.

   - Set the `openam_agent` directive to `off` for a context to disable the policy agent protection for that context and its contents. If the context has a parent, disabling the directive does not affect the protection for the parent.

   Consider the following examples:

   Example 1

```
server {
  listen       80 default_server;
  server_name  localhost;
  openam_agent on;
  openam_agent_configuration /web_agents/nginx12_agent/instances/agent_1/config/agent.conf;
  #charset koi8-r;
  #access_log  /var/log/nginx/log/host.access.log  main;

  location / {
    root   /www/;
    index  index.html index.htm;
  }

  location /customers {
    openam_agent on;
    openam_agent_configuration /web_agents/nginx12_agent/instances/agent_2/config/agent.conf;
    root   /www/customers
    index  index.html
  }

  location /market {
    root   /www/marketplace
    index  index.html
  }
}
```

The policy agent instance `agent_1` configured at the `server` context is protecting the `/` and `/market` `location` contexts. The `location` context `/customers` is protected by a second policy agent instance, `agent_2`.

Example 2

```
server {
  listen       80 default_server;
  server_name  localhost;
  openam_agent on;
  openam_agent_configuration /web_agents/nginx12_agent/instances/agent_1/config/agent.conf;
  #charset koi8-r;
  #access_log  /var/log/nginx/log/host.access.log  main;

   location / {
     root   /www/;
     index  index.html index.htm;
   }

   location /customers {
     openam_agent  off
     root   /www/customers
     index  index.html
   }

   location /market {
     root   /www/marketplace
     index  index.html
  }
 }
```

The policy agent instance `agent_1` is protecting the `server` context and the `/` and `/market` `location` contexts. Protection is disabled for the `/customers` `location` context.

`openam_agent_configuration` ***`/path/to/agent/config`***

Specifies the path to the configuration of the agent instance that protects a context. This directive is mandatory if the `openam_agent` directive is set to `on` for a context.

See the `openam_agent` directive for examples of use.

`openam_agent_instance` ***`numeric_value`***

Identifies the policy agent installation with a unique numeric value. Set up this directive in the `http` context only if there are multiple NGINX Plus servers installed in the same machine that also have a policy agent installed.

Consider the following example:

```
http {
 include       /etc/nginx/mime.types;
 default_type  application/octet-stream;

 openam_agent_instance 1
 ...
```

The `openam_agent_instance 1` directive is applied to the `http` context of the first NGINX Plus server. The next NGINX Plus server should configure the next available numeric value, for example, `openam_agent_instance 2`.

The numeric value specified byt the directive is not related to the number of policy agent instances configured at any given time.

8. Ensure the user or group running the NGINX Plus server has the appropriate permissions over the following directories:

Read Permission

- `/web_agents/nginx12_agent/lib`

Read and Write Permission

- `/web_agents/nginx12_agent/instances/agent_nnn`
- `/web_agents/nginx12_agent/log`

To determine which user or group is running the NGINX Plus server, check the `User` directive in the NGINX Plus server configuration file.

Failure to set permissions causes issues, such as the NGINX Plus server not starting up, getting a blank page when accessing a protected resource, or the policy agent generating errors during log file rotation.

Note

You may see the same issues if SELinux is enabled in `enforcing` mode and it is not configured to allow access to agent directories. For more information, see "*Troubleshooting*".

9. Start the NGINX Plus server.

Tip

The NGINX Plus server only sets the `REMOTE_USER` variable if the request contains an HTTP Authorization header, but the NGINX agent does not set an an HTTP Authorization header after the user has authenticated. Therefore, if you need to set the variable so CGI scripts can use it, configure the agent to create a custom header with the required attribute (see Profile Attributes Processing Properties) and then configure the NGINX Plus server to capture that header and convert it into the `REMOTE_USER` variable.

To Check the Policy Agent Installation

1. Check the NGINX Plus logs after starting the server to ensure the startup completed successfully.

   By default, the NGINX Plus server logs the startup messages in the operating system's syslog file, for example, `/var/log/messages`. Expected output should resemble the following:

```
Apr 25 02:17:38 tran systemd: Starting NGINX Plus - high performance web server...
Apr 25 02:17:38 FR-server nginx: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Apr 25 02:17:38 FR-server nginx: nginx: configuration file /etc/nginx/nginx.conf test is successful
Apr 25 02:17:38 FR-server systemd: Started NGINX Plus - high performance web server.
```

2. Make an HTTP request to a resource protected by the agent, then check the `/web_agents/nginx12_agent/instances/agent_1/logs/debug/debug.log` file to verify that no errors occurred on startup. Expected output should resemble the following:

```
2017-04-25 02:18:05.550 +0100    INFO [0x7f9675063700:9429]

 ###################################################
  OpenAM Web Agent
  Version: 4.2
  Revision: 3b116a1
  Container: NGINX Plus 12 Linux 64bit
  Build date: Apr 20 2017 18:03:17
 ###################################################
```

3. (Optional) If you have a policy configured, you can test that your policy agent is processing requests. For example, when you make an HTTP request to a resource protected by the agent, you should be redirected to AM to authenticate. As an example, authenticate as user `demo`,

password `changeit`. After you authenticate, AM redirects you back to the resource you tried to access.

5.3. Installing NGINX Plus Web Policy Agents Silently

You can run a silent, non-interactive installation by running **agentadmin --s**, along with arguments used to configure the instance, but you must finish the configuration by running manually steps 6, 7, and 8 of "To Install the Web Policy Agent into NGINX Plus".

The required arguments, and the order in which to specify them are:

Web server configuration file

Enter the full path to the NGINX Plus server configuration file. The installer modifies this file to include the policy agent configuration and module.

OpenAM URL

Enter the full URL of the AM instance the policy agents should connect to. Ensure the deployment URI is specified.

Agent URL

Enter the full URL of the server the agent is running on.

Realm

Enter the AM realm containing the agent profile.

Agent profile name

Enter the name of the agent profile created in AM.

Agent profile password

Enter the full path to the file containing the agent profile password.

`--acceptLicence`

You can suppress the license agreement prompt during a silent, non-interactive install by including the `--acceptLicence` parameter. The inclusion of the option indicates that you have read and accepted the terms stated in the license. To view the license agreement, open `/path/to/web_agents/agent_type/legal/Forgerock_License.txt`.

`--forceInstall`

Optionally have the installer proceed with a silent installation even if it cannot connect to the specified AM server during installation, rather than exiting.

Complete the following procedures to install a web policy agent silently into a NGINX Plus server:

To Complete Pre-Installation Tasks

Perform the following steps to create the configuration required by the policy agent before installing it:

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Implementing Authorization Using the Access Management Console*. Consider creating a simple policy, such as a policy that allows only authenticated users to access your resources. This allows you to test your policy agent after installation.
2. Create an agent profile in AM, required by the policy agent to connect and communicate with AM. For more information, see "Creating Agent Profiles".
3. Configure your AM instance to support single sign-on (SSO) or cross-domain SSO. Choose one of the following options depending on your environment:
   - To configure an AM instance and a policy agent on two different cookie domains, such as `example.org` and `example.net`, set up cross-domain SSO.

     For more information, see *Implementing Cross-Domain Single Sign-On*.
   - To configure an AM instance and a policy agent on the same cookie domain, such as `example.net`, set up SSO.

     For more information, see *Implementing Single Sign-On Within One Domain*.
4. Create a text file containing only the password specified when creating the agent profile, and protect it:

   Windows example:

   ```
   C:\> echo password > pwd.txt
   ```

   In Windows Explorer, right-click the password file, for example `pwd.txt`, select Read-Only, and then click OK.

   UNIX example:

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

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

To install Web Policy Agents in NGINX Plus Silently

1. Check the information in "Before You Install" before proceeding.
2. Shut down the NGINX Plus server where you plan to install the agent.
3. Make sure AM is running.

4. Run the **agentadmin --s** command with the required arguments. For example:

```
$ agentadmin --s \
 "/etc/nginx/nginx.conf" \
 "https://openam.example.com:8443/openam" \
 "http://www.example.com:80" \
 "/" \
 "nginx_agent" \
 "/tmp/pwd.txt" \
 --acceptLicence

OpenAM Web Agent for NGINX Server installation.

Validating...

Validating... Success.

Cleaning up validation data...

Creating configuration...

In order to complete the installation of the agent, update the configuration file /etc/nginx/nginx
.conf

if this is the first agent in the installation, please insert the following directives into the top
 section of the NGINX configuration
load_module /web_agents/nginx12_agent/lib/openam_ngx_auth_module.so;

then insert the following directives into the server or location NGINX configuration sections that you
 wish this agent to protect:
openam_agent on;
openam_agent_configuration /web_agents/nginx12_agent/instances/agent_3/config/agent.conf;

Please ensure that the agent installation files have read/write permissions for the NGINX server's
 user

Please press any key to continue.
```

5. Finish the configuration by performing steps 6, 7, and 8 of "To Install the Web Policy Agent into NGINX Plus".

5.4. Removing NGINX Plus Web Policy Agents

Complete the following steps to remove an NGINX Plus policy agent:

To remove Web Policy Agents from NGINX Plus Server

1. Shut down the NGINX Plus server where the agent is installed.
2. Run the **agentadmin --l** command to output a list of installed policy agent instances. For example:

```
$ ./agentadmin --l

 OpenAM Web Agent configuration instances:

 id:            agent_1
 configuration: /web_agents/nginx12_agent/instances/agent_1
 server/site:   /etc/nginx/nginx.conf

 id:            agent_2
 configuration: /web_agents/nginx12_agent/instances/agent_2
 server/site:   /etc/nginx/nginx.conf

 id:            agent_3
 configuration: /web_agents/nginx12_agent/instances/agent_3
 server/site:   /etc/nginx/nginx.conf
```

Make a note of the ID value of the configuration instance you want to remove.

3. Run the **agentadmin --r** command and specify the ID of the policy agent instance to remove. A warning is displayed. Type `yes` to remove the instance.

```
$ ./agentadmin --r agent_3

  Warning! This procedure will remove the OpenAM Web Agent configuration for agent_3
   but not references to it your NGINX server configuration file: /etc/nginx/nginx.conf.

   Continue (yes/no): [no]: yes

   In order to complete the removal of the agent from your NGINX installation,
   remove the openam_agent_ directives for this agent
   from your NGINX configuration file: /etc/nginx/nginx.conf
   and, if this is the only agent in the installation,
   remove the load_module directive for the openam_agent_auth_module
   in the NGINX configuration file.

   Please press any key to continue.


   Removing agent_3 configuration... Done.
```

4. Edit the NGINX Plus configuration file that contains the context protected by the removed policy agent instance.
5. Delete the `openam_agent_` directives from the context.

   If this is the last agent in the NGINX Plus server, remove the directive that loads the `openam_ngx_auth_module.so` library.
6. Restart the NGINX Plus server.

Chapter 6
Upgrading Web Policy Agents

The process of upgrading a policy agent consists of uninstalling the old agent and installing a new one. There is no requirement to create a new policy agent profile.

To upgrade Web Policy agents, perform the following procedure:

To Upgrade Web Policy Agents

1. Refer to the Web Agents Release Notes for information about changes in support and functionality.
2. Back up the policy agent installation and the web server configuration directories. For example:

   ```
   $  cp -r /path/to/web_agents/apache24_agent /path/to/backup
   $  cp -r /path/to/apache/httpd/conf /path/to/backup
   ```

   If the configuration is stored centrally in AM, back it up as described in the *ForgeRock Access Management Maintenance Guide*.
3. Redirect client traffic away from the protected web site.
4. Stop the web server where the policy agent is installed.
5. Remove the old policy agent.

   For example, to remove an old web policy agent installed in Apache HTTP server, see "Removing Apache Web Policy Agents". If the uninstall process has changed, refer to the version of the *Web Policy Agent Guide* that corresponds to your web policy agent.
6. Install the new policy agent.

   For example, to install the new policy agent in Apache HTTP server, see "*Installing Web Policy Agents in Apache HTTP Server*".

   Provide the `OpenSSOAgentBootstrap.properties` or `agent.conf` files to the installer if you want to reuse bootstrap properties, such as the AM URL, the agent profile name, and others.

   The installer creates a new `agent.conf` file containing adequate properties for the particular agent version.
7. Review the agent configuration:

- If the agent configuration is stored in the AM configuration store, review the Web Agents Release Notes and the ForgeRock Access Management Release Notes to check what is new and possible changes to AM and the agent. Then, adjust the agent configuration if required using the AM console.
- If the agent configuration is stored locally, review the Web Agents Release Notes, and the ForgeRock Access Management Release Notes to check what is new and possible changes to AM and the agent. Then, update the `agent.conf` file manually to contain the properties required for your environment. Use the backed-up copy of the configuration file for guidance.

8. If you provided the `OpenSSOAgentBootstrap.properties` or `agent.conf` files to the installer and you are upgrading from a web agent version earlier than 4.1.0 hotfix 23, re-encrypt the password specified in the `com.sun.identity.agents.config.password` property:

   a. Obtain the encryption key from the value of the `com.sun.identity.agents.config.key` property in the new `agent.conf` file.

   b. Unix users only: Store the agent profile password in a file, for example, `newpassword.file`.

   c. Encrypt the agent's profile password with the encryption key by running the **agentadmin** command with the `--p` option.

      Unix example:

      ```
      $ ./agentadmin --p "YWM0OThlMTQtMzMxOS05Nw==" "`cat newpassword.file`"
      Encrypted password value: 07bJOSeM/G8ydO4=
      ```

      Windows example:

      ```
      C:\path\to\web_agents\agent\bin>agentadmin.exe --p "YWM0OThlMTQtMzMxOS05Nw==" "newpassword"
      Encrypted password value: 07bJOSeM/G8ydO4=
      ```

   d. Set the encrypted password as the value of the `com.sun.identity.agents.config.password` property in the new `agent.conf` file.

9. Start the web server where the policy agent is installed.

10. Validate that the policy agent is performing as expected.

    For example, navigate to a protected page on the web site and confirm whether you can access it according to your configuration.

11. Allow client traffic to flow to the protected web site.

Chapter 7
Troubleshooting

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

Solutions to Common Issues

This section offers solutions to common problems when installing AM policy agents:

Q: I am running Agents 4.2 with AM 6.x with SSL and seeing 403 errors returned from the agent.

A: New agent profiles on AM 6.x are not compatible with Agents 4.x by default. You must set the `com.sun.identity.agents.config.login.url` and `com.sun.identity.agents.config.logout.url` properties in the agent profile for proper operation.

Q: When configuring the server so that my web policy agent in CDSSO mode listens for different virtual hosts, I get the following errors:

```
redirect_uri_mismatch. The redirection URI provided does not match a pre-registered value.
com.iplanet.sso.SSOException: Invalid Agent Root URL
com.iplanet.sso.SSOException: Goto URL not valid for the agent Provider ID
```

What should I do?

A: Web agents 4.2 only accept requests sent to the URL specified by the Agent Root URL for CDSSO property. For example, `http://agent.example.com:8080/`.

As a security measure, web agents prevent you from accessing the agent on URLs not defined in the Agent Root URL for CDSSO property. Add entries to this property when:

- Accessing the agent through different protocols. For example, `http://agent.example.com/` and `https://agent.example.com/`.
- Accessing the agent through different virtual host names. For example, `http://agent.example.com/` and `http://internal.example.com/`.
- Accessing the agent through different ports. For example, `http://agent.example.com/` and `http://agent.example.com:8080/`.

Q: I am trying to install a policy agent on Windows, which will connect to an AM server running over HTTPS, but the installer reports the following:

```
init_ssl(): ssleay32.dll is not available (error: 87)
init_ssl(): libeay32.dll is not available (error: 87)
```

A: If OpenSSL is correctly installed, on Windows 7 or Windows Server 2008 R2 systems, apply the update provided in Microsoft knowledge base article KB2533623. See Microsoft Security Advisory: Insecure library loading could allow remote code execution.

Q: I am trying to install the policy agent on a server with SELinux enabled in `enforcing` mode and I am getting error messages after installation, or the web server does not start up. What happened?

A: When installing policy agents on Linux or Unix servers, you must ensure that the user that runs the web server process has read and write permissions for the agent installation directory and files.

If SELinux is enabled in `enforcing` mode, you must also ensure that SELinux is configured to allow the web server process to perform read and write operations to the agent installation directory and files. By default, SELinux only allows the web server process to read files in well-known authorized locations, such as the `/var/www/html` directory.

For environments where security can be more relaxed, consider setting SELinux or the `httpd_t` context in `permissive` mode for troubleshooting purposes.

Refer to the Linux documentation for more information about configuring SELinux.

Q: My Apache HTTP 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 the Apache HTTP Server 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 the configuration. For example, if you have Apache HTTP Server configured to listen on port 8080, then set `ServerName` appropriately as in the following excerpt:

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

Q: My web server and web policy agent are installed as root, and the agent cannot rotate logs. I am seeing this error:

```
Could not rotate log file ... (error: 13)
```

What should I do?

A: First, avoid installing the web server (and therefore also the web policy agent) as root, but instead create a web server user and install as that user.

If however you cannot avoid installing the web server and policy agent as root, the you must give all users read and write permissions to the `logs/` directory under the agent instance directory (`/web_agents/`*`agent_version`*`/instances/agent_`*`nnn`*`/logs/`). Otherwise, the web policy agent fails to rotate log files with the error you observed.

Q: How do I increase security against possible phishing attacks through open redirect?

A: You can specify a list of valid URL resources against which AM validates the `goto` and `gotoOnFail` URL using the Valid `goto` URL Resource service.

AM only redirects a user if the `goto` and `gotoOnFail` URL matches any of the resources specified in this setting. If no setting is present, it is assumed that the `goto` and `gotoOnFail` URL is valid.

To set the Valid `goto` URL Resources, use the AM console, and navigate to Realms > *Realm Name* > Services. Click Add, select Validation Service, and then add one or more valid `goto` URLs.

You can use the "*" wildcard to define resources, where "*" matches all characters except "?". For example, you can use the wildcards, such as `https://website.example.com/*` or `https://website.example.com/*?*`. For more specific patterns, use resource names with wildcards as described in the procedure, *Constraining Post-Login Redirects*.

Q: After upgrading, the default Apache welcome page appear instead of my custom error pages. What should I do?

A: Check your Apache `ErrorDocument` configuration. If the custom error pages are not in the document root of the Apache server, you should enclose the `ErrorDocument` directives in `Directory` elements. For example:

```
<Directory "/web/docs">
    ErrorDocument 403 myCustom403Page.html
</Directory>
```

Refer to the Apache documentation for more details on the `ErrorDocument` directive.

Q: After starting a web agent installation, I see a failure in the logs:

```
2016-11-09 19:51:52 send_login_request(): authenticate response status code: 0 (empty)
2016-11-09 19:51:52 am_agent_login(): closing connection after failure
2016-11-09 19:51:52 error validating OpenAM agent configuration
2016-11-09 19:51:52 installation error
2016-11-09 19:51:52 installation exit
```

A: During a web agent installation, the installation can fail if AM's validation of the agent configuration exceeds the default timeout of 4 seconds.

You can set the `AM_NET_TIMEOUT` environment variable to change the default timeout, and then rerun the installation. For more information, see Web Policy Agent Environment Properties.

Chapter 8
Reference

8.1. Configuring Web Policy Agent Properties

When you create a web policy agent profile and install the agent, you can choose to store the agent configuration centrally and configure the agent using the AM console. Alternatively, you can choose to store the agent configuration locally and configure the agent by changing values in the properties file. This section covers centralized configuration, indicating the corresponding properties for use in a local configuration file where applicable. [1]

Some properties do not yet appear in the AM console, so they need to be configured as custom properties, see "Configuring Web Policy Agent Custom Properties", or locally in the agent properties configuration file, `agent.conf`.

Tip

To show the agent properties in configuration file format that correspond to what you see in the console, click Export Configuration after editing agent properties.

This corresponds to the local Java properties configuration file that is set up when you install an agent, for example in `agent_1/config/agent.conf`.

After changing properties specified as "Hot swap: no", you must restart the agent's container for the changes to take effect.

8.1.1. Configuring Web Policy Agent Global Properties

This section covers global web agent properties. After creating the agent profile, you access these properties in the AM console under Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Global.

This section describes the following property groups:

- Profile Properties
- General Properties
- Audit Properties
- Fully Qualified Domain Name Checking Properties

[1] The configuration file syntax is that of a standard Java properties file. See java.util.Properties.load for a description of the format. The value of a property specified multiple times is not defined.

Profile Properties

Group
`agentgroup`

For assigning the agent to a previously configured web agent group in order to inherit selected properties from the group.

Property: `agentgroup`

Password

Agent password used when creating the password file and when installing the agent.

Property: `userpassword`

Status

Status of the agent configuration.

Property: `sunIdentityServerDeviceStatus`

Location of Agent Configuration Repository

Whether the agent's configuration is managed centrally through AM (`centralized`) or locally in the policy agent configuration file (`local`).

If you change this to a local configuration, you can no longer manage the policy agent configuration through the AM console.

Property: `com.sun.identity.agents.config.repository.location`

Agent Configuration Change Notification

Enable agent to receive notification messages from AM server for configuration changes.

Property: `com.sun.identity.agents.config.change.notification.enable`

Enable Notifications

If enabled, the agent receives policy updates from the AM notification mechanism to maintain its internal cache. If disabled, the agent must poll AM for changes.

Property: `com.sun.identity.agents.config.notification.enable`

Hot swap: no

Agent Notification URL

URL used by agent to register notification listeners.

Property: `com.sun.identity.client.notification.url`

Hot swap: no

Agent Deployment URI Prefix

The default value is *`agent-root-URL`*`/amagent`.

Property: `com.sun.identity.agents.config.agenturi.prefix`

Hot swap: yes

Configuration Reload Interval

Interval in minutes to fetch agent configuration from AM. Used if notifications are disabled. Default: 60.

Property: `com.sun.identity.agents.config.polling.interval`

Hot swap: no

Configuration Cleanup Interval

Interval in minutes to cleanup old agent configuration entries unless they are referenced by current requests. Default: 30.

Property: `com.sun.identity.agents.config.cleanup.interval`

Hot swap: no

Agent Root URL for CDSSO

The agent root URLs for CDSSO. The valid value is in the format *`protocol://hostname:port/`* where *protocol* represents the protocol used, such as `http` or `https`, *hostname* represents the host name of the system where the agent resides, and *port* represents the port number on which the agent is installed. The slash following the port number is required.

If your agent system also has virtual host names, add URLs with the virtual host names to this list as well. AM checks that the `goto` URLs match one of the agent root URLs for CDSSO.

Property: `sunIdentityServerDeviceKeyValue[`*`n`*`]=agentRootURL=`*`protocol://hostname:port/`*

General Properties

SSO Only Mode

When enabled, the agent enforces authentication, so that upon verification of the user's identity, the user receives a session token.

When `true` , the web policy agent only manages user authentication. The filter invokes the AM Authentication Service to verify the identity of the user. If the user's identity is verified, the user is issued a session token through AM's Session Service.

When `false`, which is the default, the web policy agents will also manage user authorization, by using the policy engine in AM.

Property: `com.sun.identity.agents.config.sso.only`

Resources 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). The URL can be absolute or relative.

The following are not permitted in the URL:

- Wildcards
- The `.` directory specifier
- The `..` directory specifier

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

Agent Debug Level

Default is `Error`. Note that `All` is the same as `Message` now and provides the same output to better match AM's message levels.

Valid values for the property are:

- Error
- Warning
- Info
- Message
- All

Property: `com.sun.identity.agents.config.debug.level`

Agent Debug File Rotation

When enabled, rotate the debug file when specified file size is reached.

Property: `com.sun.identity.agents.config.debug.file.rotate`

Agent Debug File Size

Debug file size in bytes beyond which the log file is rotated. The minimum is 5242880 bytes (5 MB), and lower values are reset to 5 MB. AM sets a default of 10000000 bytes (approximately 10 MB).

Tip

If `com.sun.identity.agents.config.debug.file.rotate` is enabled, setting `com.sun.identity.agents.config.debug.file.size` to `-1` in the `agent.conf` file will rotate debug log files once every 24 hours rather than at a specified size limit.

Property: `com.sun.identity.agents.config.debug.file.size`

Default: 10000000

`com.sun.identity.agents.config.local.logfile` (Not yet in the AM console)[2]

Name of file stored locally on the agent that contains agent debug messages.

Default:

```
/web_agents/agent_version/instances/agent_nnn/logs/debug/debug.log
```

Audit Properties

Audit Access Types

Specifies the type of audit events to log. Valid values include:

- `LOG_NONE`. Disable audit logging.
- `LOG_ALLOW`. Log access allowed events.
- `LOG_DENY`. Log access denied events.
- `LOG_BOTH`. Log acccess allowed and access denied events

Default: `LOG_DENY`

Property: `com.sun.identity.agents.config.audit.accesstype`

Hot swap: yes

Audit Log Location

Specifies the location where the web agent logs audit messages. Valid values include:

- `REMOTE`. Log audit event messages to the file specified by the Remote Log Filename (`com.sun.identity.agents.config.remote.logfile`) property.
- `LOCAL`. Log audit event messages locally to the file specified by the `com.sun.identity.agents.config.local.audit.logfile` property.
- `ALL`. Log audit event messages to the file specified by the Remote Log Filename (`com.sun.identity.agents.config.remote.logfile`) property and locally to the file specified by the file specified by the `com.sun.identity.agents.config.local.audit.logfile` property.

Remote Log Filename

Name of the file stored on AM that contains agent audit messages when Audit Log Location is set to `REMOTE` or `ALL`.

Property: `com.sun.identity.agents.config.remote.logfile`

Hot swap: no

Remote Audit Log Interval

Periodic interval in minutes in which audit log messages are sent to the remote log file.

Default: `5`

Property: `com.sun.identity.agents.config.remote.log.interval`

Hot swap: no

Rotate Local Audit Log

When enabled, rotate local audit log files that have reached the size specified by the Local Audit Log Rotation Size property.

Default: `Disabled`

Property: `com.sun.identity.agents.config.local.log.rotate`

Hot swap: yes

Local Audit Log Rotation Size

Beyond this size limit in bytes, the agent rotates the local audit log file if rotation is enabled. The minimum is 5242880 bytes (5 MB), and lower values are reset to 5 MB. AM sets a default of 52428800 bytes (50 MB).

Default: `52428800`

Property: `com.sun.identity.agents.config.local.log.size`

Hot swap: yes

`com.sun.identity.agents.config.local.audit.logfile` (Not yet in the AM console)[2]

Name of file stored locally on the agent that contains agent audit messages if log location is `LOCAL` or `ALL`.

Default: `web_agents/`*`agent_type`*`/instances/agent_`*`nnn`*`/logs/audit/audit.log`

Fully Qualified Domain Name Checking Properties

FQDN Check

Enables checking of FQDN default value and FQDN map values.

Property: `com.sun.identity.agents.config.fqdn.check.enable`

FQDN Default

FQDN 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.

Property: `com.sun.identity.agents.config.fqdn.default`

FQDN Virtual Host Map

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 a virtual server`virtual.example.com` to `real.mydomain.example`, enter the keyword `valid`*`n`*, where *n* is an incrementing integer starting at `1`, in the Map Key field. Enter `virtual.example.com` in the Corresponding Map Value field.

In the configuration file, this corresponds to `com.sun.identity.agents.config.fqdn.mapping[valid1]=virtual.example.com`.

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.

Property: `com.sun.identity.agents.config.fqdn.mapping[`*`Source hostname / IP address`*`]=`*`Target FQDN`*

8.1.2. Configuring Web Policy Agent Application Properties

This section covers application web agent properties. After creating the agent profile, you access these properties in the AM console under Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Application.

This section describes the following property groups:

- Not Enforced URL Processing Properties
- Not Enforced IP Processing Properties
- Not Enforced URL from IP Processing Properties (Not yet in the AM console)[2]
- Profile Attributes Processing Properties
- Response Attributes Processing Properties

- Session Attributes Processing Properties
- Common Attributes Fetching Processing Properties

Not Enforced URL Processing Properties

Ignore Path Info for Not Enforced URLs

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 more information about Ignore Path Info, see Ignore Path Info Properties.

> **Note**
>
> The NGINX web policy agent does not support this setting.

Property: `com.sun.identity.agents.config.ignore.path.info.for.not.enforced.list`

Enable Regular Expressions for Not Enforced URLs (Not yet in the AM console)[2]

Enable use of Perl-compatible regular expressions in Not Enforced URL settings by using the following property under Advanced > Custom Properties in the agent profile.

Property: `com.forgerock.agents.notenforced.url.regex.enable`

Not Enforced URLs

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. The `-*-` wildcard can only be used in the path sections of a URL, not within the host, port, or protocol sections.

AM 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.

Property: `com.sun.identity.agents.config.notenforced.url[n]=Not enforced URL pattern`

If you enabled use of Perl-compatible regular expressions to match Not Enforced URLs, then all your settings must be done using regular expressions. (Do not mix settings; use either the mechanism described above or Perl-compatible regular expressions, but not both.)

The following example shows settings where no authentication is required for URLs whose path ends `/PublicServletA` or `/PublicServletB` (with or without query string parameters), and no authentication is required to access .png, .jpg, .gif, .js, or .css files under URLs that do not contain `/protectedA/` or `/protectedB/`.

```
com.sun.identity.agents.config.notenforced.url[0]=.*/(PublicServletA|PublicServletB)(\?.*|$)
com.sun.identity.agents.config.notenforced.url[1]=^(?!.*(/protectedA/|/protectedB/)).*\.(png|jpg|gif|
js|css)(\?.*|$)
```

Invert Not Enforced URLs

When set to `true`, enforce policy for the URLS and patterns specified in the Not Enforced URLs property instead of allowing access to them without authentication. Consider the following points when configuring this property:

- An empty Not Enforced URL property results in all URLs being enforced
- At least one URL must be enforced. To allow access to any URL without authentication, consider disabling the policy agent

Default: `false`

Property: `com.sun.identity.agents.config.notenforced.url.invert`

Fetch Attributes for Not Enforced URLs

When enabled, the agent fetches profile, response, and session attributes that are mapped by doing policy evaluation, and forwards these attributes to not enforced URLs.

Property: `com.sun.identity.agents.config.notenforced.url.attributes.enable`

Not Enforced IP Processing Properties

Not Enforced Client IP List

Specifies IP addresses or network CIDR notation for which no authentication is required. Supported values are:

- IPV4 and IPV6 addresses.
- IPV4 and IPV6 addresses specified in CIDR notation.
- IPV4 and IPV6 ranges of addresses delimited by the **-** character.
- Network ranges specified in CIDR notation.

For example:

```
com.sun.identity.agents.config.notenforced.ip[0]= 192.18.145.128
com.sun.identity.agents.config.notenforced.ip[2]= 192.168.145.128/24
com.sun.identity.agents.config.notenforced.ip[1]= 2001:5c0:9168:0:0:0:0:2/128
com.sun.identity.agents.config.notenforced.ip[3]= 192.168.1.0/24
com.sun.identity.agents.config.notenforced.ip[4]= 192.18.145.128-192.168.145.133
com.sun.identity.agents.config.notenforced.ip[5]= 2001:5c0:9168:0:0:0:0:1-2001:5c0:9168:0:0:0:0:2
```

Web agents stop evaluating not-enforced properties after reaching an invalid netmask in the list.

Property: `com.sun.identity.agents.config.notenforced.ip[n]`

Note

Loopback addresses are not considered valid IPs on the Not Enforced IP list. If specified, the policy agent ignores the loopback address.

Client IP Validation

When enabled, validate that the subsequent browser requests come from the same IP address that the SSO token is initially issued against.

Property: `com.sun.identity.agents.config.client.ip.validation.enable`

Not Enforced URL from IP Processing Properties (Not yet in the AM console)[2]

`org.forgerock.agents.config.notenforced.ipurl[n]`

Specifies a list of client IP addresses that do not require authentication when requesting the indicated URLs.

The supported format requires a list of IP addresses separated by spaces, the horizontal bar (|) character, and a list of URLs separated by spaces. For example:

```
org.forgerock.agents.config.notenforced.ipurl[n]=10.1.2.1 192.168.0.2|/public/*
```

In the preceding example, the IP addresses `10.1.2.1` and `192.168.0.2` can access any resource inside `/public` without authenticating.

The list of IP addresses supports IPv4 and IPv6 addresses specified by either CIDR or IP range notation:

- **IP range notation**

  Supported values are IPv4 and IPv6 ranges of addresses. For example:

  ```
  org.forgerock.agents.config.notenforced.ipurl[n]=192.168.1.1-192.168.1.10|/public/*
  org.forgerock.agents.config.notenforced.ipurl[n]=2001:5c0:9168:0:0:0:0:1-2001:5c0:9168:0:0:0:0:2|/
  public/*
  ```

  In the preceding IPv4 example, clients with IP addresses in the range `192.168.1.1-192.168.1.10` need not to authenticate to access the list of URLs included in `/public/*`.

- **CIDR notation**

  Supported values are specified in CIDR notation. For example:

  ```
  org.forgerock.agents.config.notenforced.ipurl[n]=192.168.1.0/24 192.168.100.0/24|/public/*
  org.forgerock.agents.config.notenforced.ipurl[n]=2001:5c0:9168:0:0:0:0:2/128|/public/*
  ```

  In the preceding IPv4 example, the IP addresses defined on the network `192.168.1` with netmask `255.255.255.0` and the network `192.168.100` with netmask `255.255.255.0` need not to authenticate to access the list of URLs included in `/public/*`.

The list of URLs can be specified by using either wildcards (*) or regular expressions:

- **Wildcards**

  The wildcard * matches all characters, except the question mark **?** character, 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`. For example:

  ```
  org.forgerock.agents.config.notenforced.ipurl[n]=192.6.8.0/24|/public/* /free_access/login*
  ```

  In the preceding example, the IP addresses specified in *`192.6.8.0/24`* do not need authenticating to access any resource inside the `/public` URI, or any resource (files or directories) that starts with `login` inside the `/free_access` URI.

- **Regular Expressions**

  To use regular expressions in the URL list, set the `org.forgerock.agents.config.notenforced.ext.regex.enable` property to `true` and use Perl-compatible regular expressions. For example:

  ```
  org.forgerock.agents.config.notenforced.ipurl[n]=192.6.8.0/24|.*\/private\/.*(png|jpg|gif)
  ```

  In the preceding example, the IP addresses specified in *`192.6.8.0/24`* do not need to authenticate to access any `png`, `jpg`, or `gif` images that are inside the `/private` URI.

`org.forgerock.agents.config.notenforced.ext.regex.enable`

Enable use of Perl-compatible regular expressions in Not Enforced URL from IP settings.

Profile Attributes Processing Properties

Profile Attribute Fetch Mode

When set to `HTTP_COOKIE` or `HTTP_HEADER`, profile attributes are introduced into the cookie or the headers, respectively.

Property: `com.sun.identity.agents.config.profile.attribute.fetch.mode`

Profile Attribute Map

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`.

Property: `com.sun.identity.agents.config.profile.attribute.mapping`

Response Attributes Processing Properties

Response Attribute Fetch Mode

When set to `HTTP_COOKIE` or `HTTP_HEADER`, response attributes are introduced into the cookie or the headers, respectively.

Property: `com.sun.identity.agents.config.response.attribute.fetch.mode`

Response Attribute Map

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`.

Property: `com.sun.identity.agents.config.response.attribute.mapping[`*`Response attribute`*`]=`*`HTTP header`*

Session Attributes Processing Properties

Session Attribute Fetch Mode

When set to `HTTP_COOKIE` or `HTTP_HEADER`, session attributes are introduced into the cookie or the headers, respectively.

Property: `com.sun.identity.agents.config.session.attribute.fetch.mode`

Session Attribute Map

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`.

Property: `com.sun.identity.agents.config.session.attribute.mapping[Session attribute]=HTTP header`

Common Attributes Fetching Processing Properties

Attribute Multi-Value Separator

Specifies separator for multiple values. Applies to all types of attributes, such as profile, session, and response attributes. Default: `|`.

Property: `com.sun.identity.agents.config.attribute.multi.value.separator`

8.1.3. Configuring Web Policy Agent SSO Properties

This section covers SSO web agent properties. After creating the agent profile, you access these properties in the AM console under Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > SSO.

This section describes the following property groups:

- Cookie Properties
- Cross Domain SSO Properties
- Cookie Reset Properties

Cookie Properties

Cookie Name

Name of the SSO token cookie used between the AM server and the agent. Default: `iPlanetDirectoryPro`.

Property: `com.sun.identity.agents.config.cookie.name`

Hot swap: no

Cookie Security

When enabled, the agent marks cookies secure, sending them only if the communication channel is secure.

Property: `com.sun.identity.agents.config.cookie.secure`

Hot swap: no

HTTPOnly Cookies (Not yet in the AM console)[2]

Agents with this property set to `true` mark cookies as HTTPOnly to mitigate against cross-site scripting. HTTPOnly cookies are not accessible through:

- The Document.cookie JavaScript property.
- The XMLHttpRequest API.
- The Request API.

Property: `com.sun.identity.cookie.httponly`

Cross Domain SSO Properties

Cross Domain SSO

Enables cross-domain single sign-oOn (CDSSO) for AM deployments that use stateful sessions. CDSSO is not supported for AM deployments that use stateless sessions.

Property: `com.sun.identity.agents.config.cdsso.enable`

CDSSO Servlet 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/cdcservlet`.

Property: `com.sun.identity.agents.config.cdsso.cdcservlet.url[`*`n`*`]=`*`Servlet URL`*

Cookies Domain List

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
```

Property: `com.sun.identity.agents.config.cdsso.cookie.domain[`*`n`*`]=`*`Cookie domain`*

Cookie Reset Properties

Cookie Reset

When enabled, the web agent resets (blanks) cookies in the response before redirecting to authentication by issuing a Set-Cookie header to the client. The following is an example of the header:

```
Set-Cookie myCookie= ; Max-Age=0; Expires=Thu, 01-Jan-1970 00:00:00 GMT; Domain=.my.default.fqdn
```

Property: `com.sun.identity.agents.config.cookie.reset.enable`

Cookie Reset Name List

Specifies a list of cookies to reset in the format `cookie_name[=value][;Domain=value][;Path=value]`. See the following table for an explanation of each of the attributes:

Cookie Reset Name List Format

Attribute	Description	Required?
`cookie_name`	Specifies the name of the cookie to reset. For example, `myCookie`	Yes
`value`	Specifies the values stored in the cookie. For example, `myDomainTrackingCookie`	No
`Domain`	Specifies the domain set for the cookie. For example, `www.example.com`. If not set, defaults to the value of the FQDN Default property (`com.sun.identity.agents.config.fqdn.default`)	No
`Path (Web Policy Agent 4.2 only)`	Specifies the cookie path, for example, `/myapp` If not set, it defaults to `/`	No

Examples:

```
com.sun.identity.agents.config.cookie.reset[0]=myCookie;Domain=www.example.com;Path=/myapp
com.sun.identity.agents.config.cookie.reset[1]=myTrackingCookie
com.sun.identity.agents.config.cookie.reset[2]=otherCookie=myCookieValue;Domain=www.forgerock.com
```

Property:`com.sun.identity.agents.config.cookie.reset[n]`

8.1.4. Configuring Access Management Services Properties

This section covers AM services web agent properties. After creating the agent profile, you access these properties in the AM console under Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > OpenAM Services.

This section describes the following property groups:

- Login URL Properties
- Logout URL Properties
- Agent Logout URL Properties
- Policy Client Service Properties

Login URL Properties

OpenAM Login URL

AM login page URL, such as `http://openam.example.com:8080/openam/UI/Login`, to which the agent redirects incoming users without sufficient credentials so that they can authenticate.

Property: `com.sun.identity.agents.config.login.url[n]=AM login URL`

OpenAM Conditional Login URL (Not yet in the AM console)[2]

Conditionally redirect users based on the incoming request URL. If the incoming request URL matches a specified domain name, the policy agent determines the appropriate redirection URL from a comma-delimited list of URLs.

The supported format is `FQDN|URL,URL[...]` with no spaces, where `FQDN` is the domain the incoming request must match, followed by a vertical bar (**|**), and `URL,URL[...]` is a comma-delimited list of URLs to which redirect incoming users.

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]=signin.example.com|http://openam3.example.com/openam/UI/
Login, http://openam4.example.com/openam/UI/Login
```

When configuring this property, consider the following points:

- If the FQDN Check property (`com.sun.identity.agents.config.fqdn.check.enable`) is enabled, the policy agent iterates through the list of URLs until it finds an appropriate redirect URL that matches the FQDN check values. Otherwise, the policy agent redirects the user to the first URL in the list.
- If CDSSO is enabled, configure the list of URLs so it takes CDSSO servlet URLs as defined in the `com.sun.identity.agents.config.cdsso.cdcservlet.url` property, rather than AM login URLs.

  Examples:

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

Property: `com.forgerock.agents.conditional.login.url[n]`

Regular Expression Conditional Login URL (Not yet in the AM console)[2]

Conditionally redirect users based on the incoming request URL, which is specified by a regular expression. If the incoming URL matches the regular expression, the policy agent determines the appropriate redirection URL from a comma-delimited list of URLs.

Examples:

```
org.forgerock.agents.config.conditional.login.pattern[0] = .*shop
org.forgerock.agents.config.conditional.login.url[0] = http://openam3.example.com/openam/UI/Login
,http://openam3.example.com/openam/UI/Login
```

When configuring this module, consider the following points:

- If the FQDN Check property (`com.sun.identity.agents.config.fqdn.check.enable`) is enabled, the policy agent iterates through the list of URLs until it finds an appropriate redirect URL that matches the FQDN check values. Otherwise, the policy agent redirects the user to the first URL in the list.
- If CDSSO is enabled, configure the list of URLs so it takes CDSSO servlet URLs as defined in the `com.sun.identity.agents.config.cdsso.cdcservlet.url` property, rather than AM login URLs.

  Examples:

```
org.forgerock.agents.config.conditional.login.pattern[0] = .*shop
org.forgerock.agents.config.conditional.login.url[0] = http://openam3.example.com/openam/cdcservlet
,http://openam4.example.com/openam/cdcservlet
```

Properties:

`org.forgerock.agents.config.conditional.login.pattern[n]`
`org.forgerock.agents.config.conditional.login.url[n]`

> **Note**
>
> These properties are supported for the NGINX Plus web policy agent only.

Agent Connection Timeout

Timeout period in seconds for an agent connection with AM auth server.

Property: `com.sun.identity.agents.config.auth.connection.timeout`

Default: 2

Polling Period for Primary Server

Interval in minutes, agent polls to check the primary server is up and running. Default: 5.

Property: `com.sun.identity.agents.config.poll.primary.server`

Hot swap: no

Logout URL Properties

OpenAM Logout URL

AM logout page URLs, such as `http://openam.example.com:8080/openam/UI/Logout`.

Property: `com.sun.identity.agents.config.logout.url[n]=AM logout URL`

Enable Logout URL Redirect (Not yet in the AM console)[2]

Logout URL redirect is enabled by default.

When this is disabled, instead of redirecting the user-agent, the policy agent performs session logout in the background and then continues processing access to the current URL. Disable this using Advanced > Custom Properties in the agent profile.

Property: `com.forgerock.agents.config.logout.redirect.disable`

Agent Logout URL Properties

Logout URL List

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

Property: `com.sun.identity.agents.config.agent.logout.url[n]=Agent logout URL`

Agent Logout URL Regular Expression (Not yet in the AM console)[2]

Perl-compatible regular expression that matches logout URLs. Set this using Advanced > Custom Properties in the agent profile.

For example, to match URLs with `protectedA` or `protectedB` in the path and `op=logout` in the query string, use the following setting:

```
com.forgerock.agents.agent.logout.url.regex= \
*(/protectedA\?|/protectedB\?/).*(\&op=logout\&)(.*|$)
```

When you use this property, the agent ignores the settings for Logout URL List.

Logout Cookies List for Reset

Cookies to be reset upon logout in the same format as the cookie reset list.

List of cookies to be reset upon logout in the format: `name[=value][;Domain=value]`.

For example `Cookie2=value;Domain=subdomain.domain.com`, which equates to: `com.sun.identity.agents.config.logout.cookie.reset[0]=Cookie2=value;Domain=subdomain.domain.com`

Property: `com.sun.identity.agents.config.logout.cookie.reset[n]=List of cookies`

Logout Redirect URL

User gets redirected to this URL after logout. Specify this property alongside a Logout URL List.

Property: `com.sun.identity.agents.config.logout.redirect.url`

Hot-swap: yes

Invalidate Logout Session

When set to `false`, the agent will not invalidate the session in AM before redirecting to the logout URL, specified by `com.sun.identity.agents.config.logout.redirect.url` for SAML applications.

Default: true

Property: `org.forgerock.agents.config.logout.session.invalidate`

Hot-swap: yes

Policy Client Service Properties

Policy Cache Polling Period

Polling interval in minutes during which an entry remains valid after being added to the agent's cache.

Property: `com.sun.identity.agents.config.policy.cache.polling.interval`

Hot swap: no

SSO Cache Polling Period

Polling interval in minutes during which an SSO entry remains valid after being added to the agent's cache.

Property: `com.sun.identity.agents.config.sso.cache.polling.interval`

Hot swap: no

User ID Parameter

Agent sets this value for User Id passed in the session from AM to the `REMOTE_USER` server variable. Default: `UserToken`.

Property: `com.sun.identity.agents.config.userid.param`

User ID Parameter Type

User ID can be fetched from either SESSION or LDAP attributes. Default: `SESSION`.

Property: `com.sun.identity.agents.config.userid.param.type`

Fetch Policies From The 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.

Property: `com.sun.identity.agents.config.fetch.from.root.resource`

Default: false

Hot swap: no

Retrieve Client Hostname

When enabled, get the client hostname through DNS reverse lookup for use in policy evaluation. This setting can impact performance.

Property: `com.sun.identity.agents.config.get.client.host.name`

Policy Clock Skew

Time in seconds used adjust time difference between agent system and AM. Clock skew in seconds = AgentTime - AMServerTime.

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 AM server time, the agent can make policy calls to the AM server before the policy subject cache has expired, or you can see infinite redirection occur.

Property: `com.sun.identity.agents.config.policy.clock.skew`

Hot swap: no

Realm

Realm where AM starts policy evaluation for this policy agent.

Default: / (top-level realm)

Edit this property when AM should start policy evaluation in a realm other than the top-level realm, `/`, when handling policy decision requests from this policy agent.

This property is recognized by AM, not the policy agent, and does not support realm aliases.

Property: `org.forgerock.openam.agents.config.policy.evaluation.realm`

Hot swap: yes

Application

Application where AM looks for policies to evaluate for this policy agent.

Default: `iPlanetAMWebAgentService`

Edit this property when AM should look for policies that belong to an application other than `iPlanetAMWebAgentService` when handling policy decision requests from this policy agent.

This property is recognized by AM, not the policy agent.

Property: `org.forgerock.openam.agents.config.policy.evaluation.application`

Hot swap: yes

8.1.5. Configuring Web Policy Agent Miscellaneous Properties

This section covers miscellaneous web agent properties. After creating the agent profile, you access these properties in the AM console under Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Miscellaneous.

This section describes the following property groups:

Advice Handling Properties

Composite Advice Handling (Not yet in the AM console)[2]

As of version 3.0.4, when set to `true`, the agent sends composite advice in the query (GET request) instead of sending it through a POST request.

Property: `com.sun.am.use_redirect_for_advice`

Locale Properties

Agent Locale

The default locale for the agent.

Property: `com.sun.identity.agents.config.locale`

Hot swap: no

Anonymous user Properties

Anonymous User

Enable or disable REMOTE_USER processing for anonymous users.

Property: `com.sun.identity.agents.config.anonymous.user.enable`

Cookie Processing Properties

Encode special characters in Cookies

When enabled, use URL encoding for special characters in cookies. This is useful when profile, session, and response attributes contain special characters, and the attributes fetch mode is set to `HTTP_COOKIE`.

> **Note**
>
> Set this property to `false` when configuring the environment variable `AM_AGENT_KEY`.

Property: `com.sun.identity.agents.config.encode.cookie.special.chars.enable`

Profile Attributes Cookie Prefix

Sets cookie prefix in the attributes headers. Default: `HTTP_`.

Property: `com.sun.identity.agents.config.profile.attribute.cookie.prefix`

Profile Attributes Cookie Maxage

Maximum age in seconds of custom cookie headers. Default: 300.

Property: `com.sun.identity.agents.config.profile.attribute.cookie.maxage`

URL Handling Properties

URL Comparison Case Sensitivity Check

When enabled, enforces case insensitivity in both policy and not enforced URL evaluation.

Property: `com.sun.identity.agents.config.url.comparison.case.ignore`

Encode URL's Special Characters

When enabled, encodes the URL which has special characters before doing policy evaluation.

Property: `com.sun.identity.agents.config.encode.url.special.chars.enable`

Ignore Naming URL Properties

Ignore Preferred Naming URL in Naming Request

When enabled, do not send a preferred naming URL in the naming request.

Property: `com.sun.identity.agents.config.ignore.preferred.naming.url`

Invalid URL properties

Invalid URL Regular Expression (Not yet in the AM console)[2]

Specifies a Perl-compatible regular expression to parse valid request URLs. The web agent rejects requests to invalid URLs with HTTP 403 Forbidden status without further processing.

For example, to filter out URLs containing a list of characters and words such as `/ /. . %00-%1f, %7f-%ff, %25, %2B, %2C, %7E, .info`, configure the following regular expression:

```
com.forgerock.agents.agent.invalid.url.regex=^(\/|\/.|.|.info|%2B|%00-%1f|%7f-%ff|%25|%2C|%7E).*$
```

Default: not set

Ignore Server Check Properties

Ignore Server Check

When enabled, do not check whether AM is up before doing a 302 redirect.

Property: `com.sun.identity.agents.config.ignore.server.check`

Ignore Path Info Properties

Ignore Path Info in Request URL

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 the 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.

Note

The NGINX web policy agent does not support this setting.

Property: `com.sun.identity.agents.config.ignore.path.info`

Multi-Byte Enable Properties

Native Encoding of Profile Attributes

When enabled, the agent encodes the LDAP header values in the default encoding of operating system locale. When disabled, the agent uses UTF-8.

Property: `com.sun.identity.agents.config.convert.mbyte.enable`

Goto Parameter Name Properties

Goto Parameter Name

Property used only when CDSSO is enabled. Only change the default `goto` value 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`.

Property: `com.sun.identity.agents.config.redirect.param`

Deprecated Agent Properties

Anonymous User Default Value

User ID of unauthenticated users. Default: `anonymous`.

Property: `com.sun.identity.agents.config.anonymous.user.id`

8.1.6. Configuring Web Policy Agent Advanced Properties

This section covers advanced web agent properties. After creating the agent profile, you access these properties in the AM console under Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced.

This section describes the following property groups:

Client Identification Properties

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 or 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`.

Client IP Address Header

HTTP header name that holds the IP address of the client.

Property: `com.sun.identity.agents.config.client.ip.header`

Client Hostname Header

HTTP header name that holds the hostname of the client.

Property: `com.sun.identity.agents.config.client.hostname.header`

Load Balancer Properties

Load Balancer Setup

Enable the `com.sun.identity.agents.config.load.balancer.enable` property to `true` in the `agent.conf` file if a load balancer is used for AM services.

This property must only exist once in the `agent.conf` file.

Property: `com.sun.identity.agents.config.load.balancer.enable`

Override Request URL 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`).

Property: `com.sun.identity.agents.config.override.protocol`

Override Request URL 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`).

Property: `com.sun.identity.agents.config.override.host`

Override Request URL 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`).

Property: `com.sun.identity.agents.config.override.port`

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`).

Property: `com.sun.identity.agents.config.override.notification.url`

`com.sun.identity.agents.config.postdata.preserve.stickysession.mode` (Not yet in the AM console)[2]

Specifies whether to create a cookie, or to append a query string to the URL to assist with sticky load balancing. Possible values are:

- `COOKIE`. The web agent creates a cookie with the value specified in the `com.sun.identity.agents.config.postdata.preserve.stickysession.value` property.
- `URL`. The web agent appends the value specified in the `com.sun.identity.agents.config.postdata.preserve.stickysession.value` to the URL query string.

`com.sun.identity.agents.config.postdata.preserve.stickysession.value` (Not yet in the AM console)[2]

Specifies a key-value pair separated by the **=** character that the web agent creates when evaluating the `com.sun.identity.agents.config.postdata.preserve.stickysession.mode` property.

For example, a setting of `lb=myserver` either sets an `lb` cookie with `myserver` value, or adds `lb=myserver` to the URL query string.

When configuring POST data preservation with cookies, set the cookie name in the cookie pair as the value configured in the `com.sun.identity.agents.config.postdata.preserve.lbcookie` property.

[2]Set this property as a custom property in AM, by navigating to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

Post Data Preservation Properties

POST Data Preservation

Enables HTTP POST data preservation.

Property: `com.sun.identity.agents.config.postdata.preserve.enable`

POST Data Entries Cache Period

POST cache entry lifetime in minutes.

Default: `10`

Property: `com.sun.identity.agents.config.postcache.entry.lifetime`

`com.sun.identity.agents.config.postdata.preserve.lbcookie` (Not yet in the AM console)[2]

Specifies the name of a cookie to use for enabling sticky load balancing under the following conditions:

- POST data preservation is enabled.
- The following load balancer web agent properties are set to `true`:
  - `com.sun.identity.agents.config.override.protocol`
  - `com.sun.identity.agents.config.override.host`
  - `com.sun.identity.agents.config.override.port`
- The web agent is behind a load balancer.
- The `com.sun.identity.agents.config.postdata.preserve.stickysession.mode` property is set to `COOKIE`.

`org.forgerock.agents.config.postdata.preserve.dir` (Not yet in the AM console)[2]

The directory local to the agent installation where the agent writes preserved POST data while requesting authorization to AM.

Default: `/web_agents/agent_type/log`

Post Data Preservation URI Prefix (Not yet in the AM console)[2]

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 AM. 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.

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

`org.forgerock.agents.pdp.javascript.repost` (Not yet in the AM console)[2]

When set to `true`, preserved post data will be resubmitted to the destination server after authentication by using JavaScript.

Sun Java System Proxy Server Properties

Override Proxy Server's Host and Port

When enabled ignore the host and port settings.

Property: `com.sun.identity.agents.config.proxy.override.host.port`

Hot swap: no

Microsoft IIS Server Properties

Authentication Type

The agent should normally perform authentication, so this is not required. If necessary, set to `none`.

Property: `com.sun.identity.agents.config.iis.auth.type`

Hot swap: no

Replay Password Key

DES key for decrypting the basic authentication password in the session.

Property: `com.sun.identity.agents.config.replaypasswd.key`

Filter Priority

The loading priority of filter, DEFAULT, HIGH, LOW, or MEDIUM.

Property: `com.sun.identity.agents.config.iis.filter.priority`

Filter configured with OWA

This property does not apply to Web Agents 4.2, although it may appear in the AM console

Property: `com.sun.identity.agents.config.iis.owa.enable`

Change URL Protocol to HTTPS

This property does not apply to Web Agents 4.2, although it may appear in the AM console

Property: `com.sun.identity.agents.config.iis.owa.enable.change.protocol`

Idle Session Timeout Page URL

This property does not apply to Web Agents 4.2, although it may appear in the AM console

Property: `com.sun.identity.agents.config.iis.owa.enable.session.timeout.url`

Show Password in HTTP Header

Set to `true` if encrypted password should be set in HTTP header `AUTH_PASSWORD`.

Property: `com.sun.identity.agents.config.iis.password.header`

Logon and Impersonation

Set to `true` if agent should do Windows Logon and User Impersonation.

Property: `com.sun.identity.agents.config.iis.logonuser`

IBM Lotus Domino Server Properties

Check User in Domino Database

When enabled, the agent checks whether the user exists in the Domino name database.

Property: `com.sun.identity.agents.config.domino.check.name.database`

Use LTPA token

Enable if the agent needs to use LTPA Token.

Property: `com.sun.identity.agents.config.domino.ltpa.enable`

LTPA Token Cookie Name

The name of the cookie that contains the LTPA token.

Property: `com.sun.identity.agents.config.domino.ltpa.cookie.name`

LTPA Token Configuration Name

The configuration name that the agent uses in order to employ the LTPA token mechanism.

Property: `com.sun.identity.agents.config.domino.ltpa.config.name`

LTPA Token Organization Name

The organization name to which the LTPA token belongs.

Property: `com.sun.identity.agents.config.domino.ltpa.org.name`

Custom Properties

Custom Properties

Additional properties to augment the set of properties supported by agentd. Such properties take the following forms.

- `customproperty=custom-value1`
- `customlist[0]=customlist-value-0`
- `customlist[1]=customlist-value-1`
- `custommap[key1]=custommap-value-1`
- `custommap[key2]=custommap-value-2`

Property: `com.sun.identity.agents.config.freeformproperties`

8.1.7. Configuring Web Policy Agent Custom Properties

This section covers custom web agent properties.

Note

These settings do not appear as configurable options in the AM console, so must be added as custom properties, or set in the local configuration file.

If using a centralized configuration, you create these properties in the AM console under Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

This section describes the following property groups:

- Bootstrap Properties
- Encryption Properties
- Naming URL and Failover Properties
- JSON-Formatted Response Properties
- Miscellaneous Custom Properties

Bootstrap Properties

These properties are only used within the local configuration file. They are not available in the AM console. The agent uses these bootstrap properties to connect to AM.

`com.sun.identity.agents.config.organization.name`

The AM realm where the agent profile is located.

Default: `/`

`com.sun.identity.agents.config.username`

The name of the agent profile in AM.

`com.sun.identity.agents.config.password`

The password required by the agent profile, encrypted with the key specified in `com.sun.identity.agents.config.key`.

`com.sun.identity.agents.config.key`

The encryption key used to encrypt the agent profile password, which should be provided in `com.sun.identity.agents.config.password`.

`org.forgerock.agents.config.tls`

Specifies a space-separated list of security protocols preceded by a dash **-** that will not be used when connecting to AM.

The supported protocols are the following:

- `SSLv3`
- `TLSv1`
- `TLSv1.1`
- `TLSv1.2` (Default)

This property is relevant to all policy agents using OpenSSL libraries.

Default: `-SSLv3 -TLSv1 -TLSv1.1`

You can also change the default value by setting an environment variable, `AM_SSL_OPTIONS`. For more information, see "Configuring Web Policy Agent Environment Variables".

Note

SSLv2 is disabled always regardless of setting.

`org.forgerock.agents.init.retry.max`

This is the maximum number of consecutive agent initialization retries.

The default value is 0 (not set).

`org.forgerock.agents.init.retry.wait`

This is the wait time in seconds between retries.

The default value is 0 (not set).

`com.sun.identity.agents.config.connect.timeout`

Set this property to the number of seconds to keep the socket connection open before timing out. Applies to both TCP *connect* and *receive* operations.

Default: 4 (seconds)

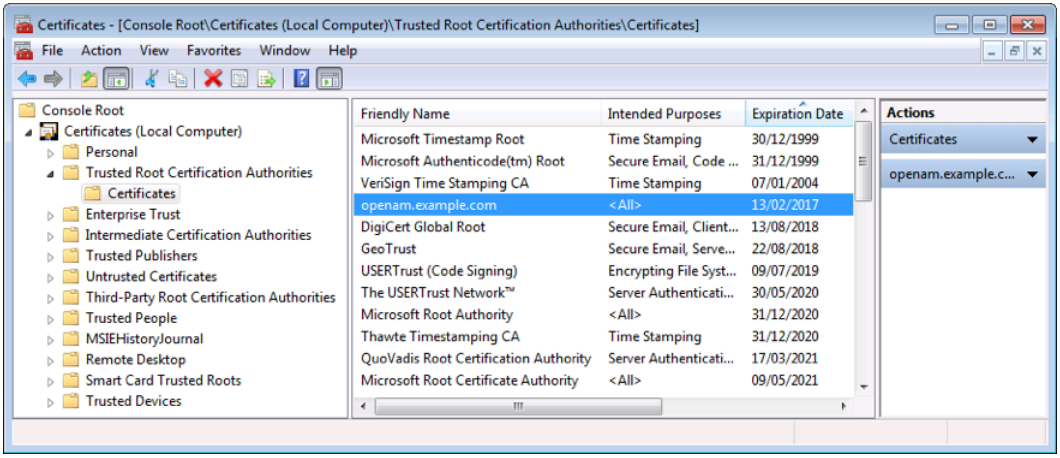
Encryption Properties

`com.forgerock.agents.config.cert.ca.file`

Set this property to the file name that contains one or more CA certificates. The file should be *Privacy Enhanced Mail* (PEM) encoded. AM requires PEM files to be base64-encoded ASCII data.

When using the Windows built-in Secure Channel API, set this property to the friendly name of the CA certificate file as it appears in the certificates snap-in. For example, the friendly name of the imported CA certificate in the image below is `openam.example.com`.

CA Friendly Name in the Windows Certificates Snap-in



You must set this property if `com.sun.identity.agents.config.trust.server.certs` is set to `false`.

`com.forgerock.agents.config.cert.file`

When AM is configured to perform client authentication, set this property to the name of the file that contains the public PEM-encoded client certificate that corresponds with the private key specified in `com.forgerock.agents.config.cert.key`.

When using the Windows built-in Secure Channel API, you can set this property to either the friendly name of the certificate file as it appears in the certificates snap-in, or the name of the file containing the client certificate in PKCS#12/PFX format.

To use a client certificate file in PKCS#12/PFX format:

1. Obtain your client certificate, ensuring the signing chain is intact, and that the key and CA certificate are included.
2. Run the **agentadmin** tool to generate an encrypted password for the certificate file:

```
C:\> cd web_agents\iis_agent\bin
C:\path\to\web_agents\iis_agent\bin> agentadmin.exe --p "Encryption Key" "Certificate File
 Password"

Encrypted password value: zck+6RKqjtc=
```

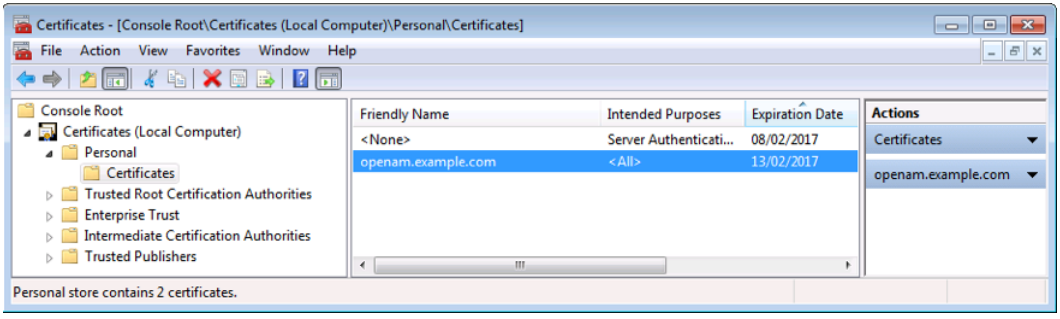
   The value used for *Encryption Key* comes from the `com.sun.identity.agents.config.key` property. The value of *Certificate File Password* should be the password required to access the client certificate file.
3. Use the encrypted password value in the agent configuration file as follows:

```
com.forgerock.agents.config.cert.file=C:\Certificates\myClientCertificate.pfx
com.forgerock.agents.config.cert.key=
com.forgerock.agents.config.cert.key.password=zck+6RKqjtc=
com.sun.identity.agents.config.trust.server.certs=false
```

4. Restart the agent.

If you do not want to use a file directly, enter the friendly name of the certificate as the value of the `com.forgerock.agents.config.cert.file` property instead. As an example, the friendly name of the imported certificate in the image below is `openam.example.com`.

Friendly Name in the Windows Certificates Snap-in



`com.forgerock.agents.config.cert.key`

Set this property to the name of the file that contains the private key. On UNIX systems, that key should be encoded in PEM format.

On Windows systems, that entry depends. If SSL mutual authentication is required with AM, that entry should contain the name of the private key or certificate imported in the Windows Certificate Manager, part of the Microsoft Management Console. For a web server, that should point to the Local Machine or Service certificate store, depending on the account associated with the Web server.

`com.forgerock.agents.config.cert.key.password`

Set this property to the obfuscated private key password. Obfuscate the password by using **agentadmin --p**, as demonstrated in the following example to generate the value:

Unix example:

```
$ cd /web_agents/agent-type/bin
$ ./agentadmin --p "key" "`cat newpassword.file`"
```

Here, *agent-type* corresponds to the file system directory for the particular agent type, such as `apache24_agent`, *newpassword.file* is a file containing the private key password, and *key* is the obfuscation key as specified by `com.sun.identity.agents.config.key`.

Windows example:

```
C:\cd C:\web_agents\agent-type\bin
$ agentadmin.exe --p "key" "newpassword"
```

Here, *agent-type* corresponds to the file system directory for the particular agent type, such as `apache24_agent`, *newpassword* is the new private key password, and *key* is the obfuscation key as specified by `com.sun.identity.agents.config.key`.

> **Tip**
>
> You can generate a new obfuscation key by using **agentadmin --k**.

This property is not used on Microsoft Windows systems.

`com.forgerock.agents.config.ciphers`

Set this property to a list of ciphers to support. The list consists of one or more cipher strings separated by colons, as defined in the man page for `ciphers` available at http://www.openssl.org/docs/apps/ciphers.html.

Default: `HIGH:MEDIUM`.

`com.sun.identity.agents.config.trust.server.certs`

When SSL is configured, set to `false` to trust the AM SSL certificate only if the certificate is found to be correct and valid. Default is `true` to make it easy to try SSL during evaluation.

> **Important**
>
> The default setting, `true`, means that the web policy agent trusts all server certificates. Change this to `false`, and test that your web policy agent can trust server certificates before deploying the policy agent in production.

org.forgerock.agents.config.secure.channel.disable

On Windows operating systems, web policy agents use the built-in Secure Channel API for SSL/TLS communications.

Set this property to `true` to disable the built-in Secure Channel API and use OpenSSL instead.

To use OpenSSL, you also need to set the `AM_SSL_SCHANNEL` environment variable to `false`. See "Configuring Web Policy Agent Environment Variables".

Note

Web Policy Agent 4.1.1 does not support the `org.forgerock.agents.config.secure.channel.disable` property. To enable OpenSSL, set the `AM_SLL_SCHANNEL` environment variable to `false`.

`org.forgerock.agents.config.cert.verify.depth` (Web Policy Agent 4.2 only)

Specifies the certificate verification depth for OpenSSL. The supported format is a number between `0` and `9`, as specified in the OpenSSL man pages.

To turn off server certification validation, set the `com.sun.identity.agents.config.trust.server.certs` property to `true`.

Default: `9`

Naming URL and Failover Properties

`com.forgerock.agents.ext.url.validation.default.url.set`

This property takes a comma-separated list of indexes for URL values indicating the order in which to fail over, where the indexes are taken from the values set for `com.sun.identity.agents.config.naming.url`, `com.sun.identity.agents.config.login.url`, `com.sun.identity.agents.config.cdsso.cdcservlet.url`, and `com.sun.identity.agents.config.logout.url`.

When using this failover capability make sure you synchronize URL settings in `com.sun.identity.agents.config.naming.url`, `com.sun.identity.agents.config.login.url`, `com.sun.identity.agents.config.cdsso.cdcservlet.url`, and `com.sun.identity.agents.config.logout.url`, such that each service shares the same index across all properties. This ensures the web agent fails over and fails back for all services.

Consider the following configuration:

```
com.sun.identity.agents.config.naming.url=https://openam1.example.com:8443/openam/ https://
openam2.example.com:8443/openam/
com.forgerock.agents.ext.url.validation.default.url.set=0,1
com.forgerock.agents.ext.url.validation.level=1
```

In the example, the web agent connects to `https://openam1.example.com:8443/openam/`, and will failover if necessary to `https://openam2.example.com:8443/openam/` since validation is enabled.

If you configure only one value on this property, the web agent will retry the connection until the server is available. Consider the following configuration:

```
com.sun.identity.agents.config.naming.url=https://openam1.example.com:8443/openam/ https://
openam2.example.com:8443/openam/
com.forgerock.agents.ext.url.validation.default.url.set=0
com.forgerock.agents.ext.url.validation.level=1
```

In the example, the web agent connects to `https://openam1.example.com:8443/openam/`. If the server becomes unavailable, the web agent retries the connection until the server is available again. Any other URL defined in the `com.sun.identity.agents.config.naming.url` property is ignored. The web agent behaves in this way also when URL validation is disabled.

This property has no default setting.

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

Specifies whether the web agent should check that the URLs specified in the `com.sun.identity.agents.config.naming.url` property are reachable during the bootstrap phase when the web agent reads its configuration, and then thereafter if the web agent is configured fail over when a URL becomes invalid.

If you leave URL validation disabled, then 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 URLs can cause the agent to crash.

When URL validation is enabled you should also ensure the `com.sun.identity.agents.config.connect.timeout` property is set to a low value, such as 4 seconds, which is the default.

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

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

This property can take the following values.

0

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

1

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

2 (Default)

Disable all URL validation.

`com.forgerock.agents.ext.url.validation.ping.interval`

Set this property to the seconds between validation requests against the current URL.

The value of the `com.sun.identity.agents.config.connect.timeout` property must not exceed this value.

Default: 60 (seconds)

`com.forgerock.agents.ext.url.validation.ping.miss.count`

If validation requests against the current URL fail this number of times in a row, the web policy agent fails over to the next service in `com.forgerock.agents.ext.url.validation.default.url.set`.

Default: 3

`com.forgerock.agents.ext.url.validation.ping.ok.count`

After failover, if validation requests against the default URL succeed this number of times in a row, the web policy agent fails back to that service, the first URL in the `com.forgerock.agents.ext.url.validation.default.url.set` list.

Default: 3

`com.sun.identity.agents.config.naming.url`

Specifies a space-separated list of AM URLs to which the web agent connects.

Forward Proxy Custom Properties

`com.sun.identity.agents.config.forward.proxy.host`

When AM and the agent communicate through a web proxy server configured in forward proxy mode, set this property to the proxy server host name.

`com.sun.identity.agents.config.forward.proxy.password`

When AM 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 property to the agent's password.

`com.sun.identity.agents.config.forward.proxy.port`

When AM and the agent communicate through a web proxy server configured in forward proxy mode, set this property to the proxy server port number.

`com.sun.identity.agents.config.forward.proxy.user`

When AM 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 property to the agent's user name.

JSON-Formatted Response Properties

`org.forgerock.agents.config.json.url[n]`

Use regular expressions to specify a list of resource URLs that should trigger JSON-formatted errors to be returned.

Example: `org.forgerock.agents.config.json.url[0]=https://www.example.com:8443/jsonClient/*`

`org.forgerock.agents.config.json.header[Header]=Value`

Specify HTTP headers and associated values that trigger JSON-formatted errors to be returned.

Example: `org.forgerock.agents.config.json.header[enableJsonResponse]=true`

`org.forgerock.agents.config.json.url.invert`

Set to `true` to invert the meaning of both the `org.forgerock.agents.config.json.url` and `org.forgerock.agents.config.json.header` properties. When inverted the specified values in those two properties will not trigger JSON-formatted responses. Any non-specified value will trigger JSON-formatted responses, instead.

Default: `false` (not set)

`org.forgerock.agents.config.json.response.code`

Specify an HTTP response code to return when a JSON-formatted error is triggered.

Example: `org.forgerock.agents.config.json.response.code=401`

Miscellaneous Custom Properties

`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`.

`org.forgerock.agents.config.keepalive.disable`

Specifies how the policy agent connects to AM during the session validation process.

Session validation is a process composed of several requests going to and coming from AM. When this property is set to `false`, the agent opens a single connection to AM which is reused to satisfy every request required for a session, then closes it. When set to `true`, the agent opens and closes a connection for every request required when validating a session.

Setting this property to `false` reduces the overhead of opening and closing connections to AM. However, if you use load balancers or reverse proxy servers that do not allow applications to keep connections open, you must set this property to `true`.

> **Note**
>
> If notifications are disabled, the web agent always behaves as if this property is set to `false` (Web Policy Agent 4.2 only).

Default: `true`

`org.forgerock.agents.config.cdsso.deny.cleanup.disable`

By default, web policy agents configured for CDSSO will delete SSO cookies when a page returns an HTTP 403 forbidden status code. Visiting any other protected page will require re-authentication.

To retain SSO cookies when an HTTP 403 forbidden status code is returned, set this property to `true`.

Default: `false`

`org.forgerock.agents.config.cdsso.original.url.redirect.param` (Web Agents 4.2 Only)

Specifies the name of the query parameter that contains the incoming request URL when the agent is working in CDSSO mode. The policy agent will add the query parameter to the URL when redirecting to AM for authentication.

For example, when configuring the property to the `originalURL` value, the policy agent creates a redirection URL similar to the following:

```
https://openam.example.com:8443/cdcservlet?...&goto=http://agent.example.com:80/
dummypost&originalURL=http://website.example.com
```

If no value is specified, the incoming request URL is not added to the URL when redirecting to AM for authentication.

Default: Not set

`org.forgerock.agents.config.cdsso.advice.cleanup.disable`

When set to `true`. the policy agent will not reset the session cookie on an authentication redirect if there is a policy advice present.

By default, the policy agent resets the session cookie in all configured domains on every authentication redirect when a policy advice is present.

Default: `false`

`org.forgerock.agents.config.skip.post.url[n]`

Specify a list of URLs that will not be processed by the web policy agent POST data inspector. This allows other modules on the same server to access the POST data directly.

The following example uses wildcards to add a file named `postreader.jsp` in the root of any protected website to the list of URLs that will not have their POST data inspected: `org.forgerock.agents.config.skip.post.url[0]=http*://*:*/postreader.jsp`

> **Note**
>
> Any URLs added to this property should also be added to the Not Enforced URLs (`com.sun.identity.agents.config.notenforced.url`) property. See Not Enforced URL Processing Properties.

Pattern-Match Policy Delegation (Web Agents 4.2 Only)

In environments where protected URLs are dynamic, the web agent's policy decision cache may not receive hits on subsequent policy validations. For example, a request to the resource at `http://www.example.com/myApp?param1=true` would not match a request for `http://www.example.com/myApp?param1=true&param2=true` even though the base URL is the same.

In these cases, the web agent may need to contact AM frequently for policy evaluation, which may cause a performance impact on both the agent and AM.

Although this is the expected behavior, pattern-match policy delegation allows the web agent to match inbound requests from an authenticated user against a regular expression stored alongside a policy decision in the policy cache. If there is a match, the web agent replays the policy decision without contacting AM.

Consider an example given the following regular expression and AM policies:

Regular expression:

```
org.forgerock.agents.config.policy.rule=^https:\/\/www.example.com\:443\/myapp.*
```

AM policies:

```
Policy 1 =  https://www.example.com:443/myapp/*
Policy 2 =  https://www.example.com:443/admin/*
```

- An unauthenticated user or client attempts to access a protected resource. For example, `https://www.example.com:443/myapp/index.html?param1=true`.
- The web agent redirects the request for authentication to AM and a policy decision. AM returns with the session token and policy decision.
- The web agent reads the policy decision and matches the protected URL against the regular expression. There is a match; therefore, the web agent stores the session token, the protected URL, and the regular expression in the policy cache.
- The same user or client, now authenticated, attempts to access another protected resource. For example, `https://www.example.com:443/myapp/index.html?param1=true&param2=true`.
- The web agent checks the session token in the policy cache and finds a match. Then, it checks for the policy decisions stored for that session and any regular expression.
- The web agent evaluates the newly requested URL against the regular expression stored in the cache for that session. There is a match; therefore, the web agent replays the policy decision in the cache without contacting AM.
- The authenticated user or client attemts to access another protected resource. For example, `https://www.example.com:443/admin/index.html?param1=false`.
- The web agent checks the session token in the session cache and finds a match. Then, it checks for the policy decisions stored for that session and any regular expression.
- The web agent evaluates the newly requested URL against the regular expression stored in the cache for that session. It does not match; therefore, the web agent contacts AM for a policy decision.

Do not set the `org.forgerock.agents.config.policy.rule` advanced property if:

- The policies configured in AM have policy response attributes.

- Every resource must be checked for policy decision due to security reasons.

Property: `org.forgerock.agents.config.policy.rule`

8.1.8. Configuring Web Policy Agent Environment Variables

This section covers web agent properties that are configured by using environment variables. You must restart the container in which web policy agents are running to apply changes to these settings.

Web Policy Agent Environment Properties

`AM_DEFAULT_LOG_LEVEL`

Configures the log level of garbage collector statistics for all policy agents instances in the container. The logs are written into the `/web_agents/type/log/agent.log` file every 3 seconds.

The default value of the `AM_DEFAULT_LOG_LEVEL` variable is `Error`. Increase it to `Message` or `All` for fine-grained detail.

Valid values for the variable are:

- Error
- Warning
- Info
- Message
- All

`AM_MAX_SHARED_POOL_SIZE`

Configure the maximum amount of shared memory, in bytes (32 MB) or hex (0x2000000), that the web policy agents use for caching. The maximum size the cache is allocated on startup to approximately 2 gigabytes (exactly 0x7FFFF000 bytes).

You can reduce the maximum size by setting `AM_MAX_SHARED_POOL_SIZE`, specified in bytes. You should not reduce the cache size to less than 10 megabytes. You cannot increase the default maximum cache size.

> **Warning**
>
> Reducing the size of the cache may affect web policy agent performance under heavy workloads, such as handling thousands of concurrent sessions, or fewer sessions with many non-repeating policy calls.

`AM_SSL_SCHANNEL`

Web policy agents installed on Windows operating systems use the built-in Secure Channel API by default for SSL/TLS communications.

To use OpenSSL on a Windows system, set this environment property to `false`.

`AM_AGENT_KEY` (Web Agents 4.2 only)

Specifies a shared secret to encrypt CDSSO login data when configuring web agents in CDSSO environments without sticky load balancing. Each web agent configured behind the load balancer must share the same secret.

When configured, web agents store CDSSO data encrypted and zipped into the `X-AMAGENT-TX` cookie. Since all web agents in the environment share the secret, any of them can decrypt the cookie and satisfy the CDSSO request.

> **Note**
>
> Ensure the `com.sun.identity.agents.config.encode.cookie.special.chars.enable` property is set to `false` when configuring the `AM_AGENT_KEY` environment variable.

`AM_AGENT_REST_LOGIN` (Web Agents 4.2 only)

Allows the agent to authenticate to AM servers configured behind a load balancer that does not support session stickiness. Set the environment variable to one of the following numbers, which indicates the REST API version used and then export the environment variable:

- `5`. Set the property to `5` for AM servers 5 and later.
- `13`. Set the property to `13` for AM servers 13.x.

`AM_LOG_ONE`(For single agent instance installations)

Enables fast direct logging for the agent instance.

In production environments, there can be a performance impact on the agent when debug logger mode is enabled. You can set an agent environment variable, `AM_LOG_ONE=1`, that enables direct log file input/output with little impact on the overall agent performance, even if the debug logger mode is turned on.

`AM_NET_TIMEOUT`

Specifies the timeout in seconds for the agent installer to contact AM during agent configuration validation. If not set, the default 4 seconds will be used.

For example, if the installer contacts AM to validate an agent configuration that lasts longer than 4 seconds, the installation will fail due to the default timeout of 4 seconds. You can extend this timeout by setting this environment variable.

`AM_SSL_OPTIONS`

Overrides the default SSL/TLS protocols for the agent, set in the `org.forgerock.agents.config.tls` bootstrap property (for more information, see Bootstrap Properties).

Specifies a space-separated list of security protocols preceded by a dash **-** that will *not* be used when connecting to AM.

The supported protocols are the following:

- `SSLv3`
- `TLSv1`
- `TLSv1.1` (Default)
- `TLSv1.2` (Default)
- `TLSv1.3`[3] (Default)

For example, to configure `TLSv1.1` or later, set the environment variable to `AM_SSL_OPTIONS = -SSLv3 -TLSv1`.

8.2. Configuring Agent Authenticators

An *agent authenticator* has read-only access to multiple agent profiles defined in the same realm, typically allowing an agent to read web service agent profiles.

After creating the agent profile, you access agent properties in the AM console under Realms > *Realm Name* > Applications > Agents > Agent Authenticator > *Agent Name*.

Password

Specifies the password the agent uses to connect to AM.

Status

Specifies whether the agent profile is active, and so can be used.

Agent Profiles allowed to Read

Specifies which agent profiles in the realm the agent authenticator can read.

Agent Root URL for CDSSO

Specifies the list of agent root URLs for CDSSO. The valid value is in the format `protocol://hostname:port/` where *protocol* represents the protocol used, such as `http` or `https`, *hostname* represents the host name of the system where the agent resides, and *port* represents the port number on which the agent is installed. The slash following the port number is required.

If your agent system also has virtual host names, add URLs with the virtual host names to this list as well. AM checks that `goto` URLs match one of the agent root URLs for CDSSO.

[3] TLSv1.3 is supported only when using OpenSSL 1.1.1 or later.

Command-Line Tool Reference

Table of Contents

Name

agentadmin — manage web policy agent installation

Synopsis

`agentadmin` {options}

Description

This command manages web policy agent installations.

Options

The following options are supported:

`--i`

Perform an interactive install of a new agent instance.

Usage: **agentadmin --i**

For more information, see:

- "Installing Apache Web Policy Agents"
- "Installing IIS Web Policy Agent"

`--s`

Perform a silent, non-interactive install of a new agent instance.

Usage: **agentadmin --s *web-server-config-file openam-url agent-url realm agent-profile-name agent-profile-password* [--changeOwner] [--acceptLicense] [--forceInstall]**

web-server-config-file

When installing in Apache HTTP Server, enter the full path to the Apache HTTP server configuration file. The installer modifies this file to include the web policy agent configuration and module.

When installing in Microsoft IIS, enter the ID number of the IIS site in which to install the web policy agent. To list the available sites in an IIS server and the relevant ID numbers, run **agentadmin.exe --n**.

openam-url

Enter the full URL of the AM instance that the web policy agents will use. Ensure the deployment URI is specified.

Example:

`https://openam.example.com:8443/openam`

agent-url

Enter the full URL of the server on which the agent is running.

Example:

`http://www.example.com:80`

realm

Enter the AM realm containing the agent profile.

agent-profile-name

Enter the name of the agent profile in AM.

agent-profile-password

Enter the full path to the agent profile password file.

`--changeOwner`

Use this option to change the ownership of the created directories to be the same user and group as specified in the Apache HTTP Server configuration, or the user that is running the selected IIS site.

`--acceptLicense`

When you run certain commands, you will be prompted to read and accept the software license agreement. You can suppress the license agreement prompt by including the optional `--acceptLicence` parameter. Specifying this options indicates that you have read and accepted the terms stated in the license.

To view the license agreement, open `/path/to/web_agents/agent_type/legal/Forgerock_License.txt`.

`--forceInstall`

Add this option to proceed with a silent installation even if it cannot connect to the specified AM server during installation, rather than exiting.

For more information, see:

- "Installing Apache Web Policy Agents Silently"
- "Installing IIS Web Policy Agents Silently"

`--n`

List the sites available in an IIS server.

Example:

```
c:\web_agents\iis_agent\bin> agentadmin.exe --n

 IIS Server Site configuration:

 Number of Sites: 2

 id: 1   name: "DEFAULT WEB SITE"
 id: 2   name: "CUSTOMERPORTAL"
```

--l

List existing configured agent instances.

Usage: **agentadmin --l**

Example:

```
$ ./agentadmin --l
OpenAM Web Agent configuration instances:

 id:            agent_1
 configuration: /opt/web_agents/apache24_agent/bin/../instances/agent_1
 server/site:   /etc/httpd/conf/httpd.conf

 id:            agent_2
 configuration: /opt/web_agents/apache24_agent/bin/../instances/agent_2
 server/site:   /etc/httpd/conf/httpd.conf

 id:            agent_3
 configuration: /opt/web_agents/apache24_agent/bin/../instances/agent_3
 server/site:   /etc/httpd/conf/httpd.conf
```

--r

Remove an existing agent instance.

Usage: **agentadmin --r *agent-instance***

agent-instance

The ID of the web policy agent configuration instance to remove.

Respond `yes` when prompted to confirm removal.

For more information, see:

- "Removing Apache Web Policy Agents"
- "Enabling and Disabling IIS Web Policy Agents"

--k

Generate a new signing key.

Usage: **agentadmin --k**

Examples:

- UNIX:

```
$ cd /web_agents/apache24_agent/bin/
$ ./agentadmin --k
Encryption key value: YWM0OThlMTQtMzMxOS05Nw==
```

- Windows:

```
C:\> cd web_agents\apache24_agent\bin
C:\web_agents\apache24_agent\bin> agentadmin --k
Encryption key value: YWM0OThlMTQtMzMxOS05Nw==
```

For more information, see Encryption Properties.

--p

Use a generated encryption key to encrypt a new password.

Usage: **agentadmin --p *encryption-key password***

encryption-key

An encryption key, generated by the **agentadmin --k** command.

password

The password to encrypt.

Examples:

- UNIX:

```
$ ./agentadmin --p "YWM0OThlMTQtMzMxOS05Nw==" "`cat newpassword.file`"
Encrypted password value: 07bJOSeM/G8ydO4=
```

- Windows:

```
C:\web_agents\apache24_agent\bin>agentadmin.exe --p "YWM0OThlMTQtMzMxOS05Nw==" "newpassword"
Encrypted password value: 07bJOSeM/G8ydO4=
```

For more information, see Encryption Properties.

--v

Display information about **agentadmin** build and version numbers, and available system resources.

For example:

```
OpenAM Web Agent for IIS Server 7.5, 8.x
 Version: 4.2
 Revision: 5ba11d2
 Build machine: WIN-6R2CH15R77
 Build date: Nov  8 2016 11:30:18

System Resources:
 total memory size: 7.7GB
 pre-allocated session/policy cache size: 1.0GB
 log buffer size: 128.5MB
 min audit log buffer size: 2MB, max 2.0GB
 total disk size: 162.4GB
 free disk space size: 89.6GB

System contains sufficient resources (with remote audit log feature enabled).
```

Return Codes

The **agentadmin** command returns `EXIT_SUCCESS` when an operation has completed successfully, and `EXIT_FAILURE` if the operation failed to complete.

The numerical return value will depend on the operating system in use, but is generally `0` for `EXIT_SUCCESS` and greater than zero for `EXIT_FAILURE`.

Appendix A. Getting Support

For more information or resources about AM and ForgeRock Support, see the following sections:

A.1. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.

  While many articles are visible to community members, ForgeRock customers have access to much more, including advanced information for customers using ForgeRock software in a mission-critical capacity.

- ForgeRock product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

A.2. Using the ForgeRock.org Site

The ForgeRock.org site has links to source code for ForgeRock open source software, as well as links to the ForgeRock forums and technical blogs.

If you are a *ForgeRock customer*, raise a support ticket instead of using the forums. ForgeRock support professionals will get in touch to help you.

A.3. Getting Support and Contacting ForgeRock

ForgeRock provides support services, professional services, training through ForgeRock University, and partner services to assist you in setting up and maintaining your deployments. For a general overview of these services, see https://www.forgerock.com.

ForgeRock has staff members around the globe who support our international customers and partners. For details, visit https://www.forgerock.com, or send an email to ForgeRock at info@forgerock.com.

Glossary

Access control	Control to grant or to deny access to a resource.
Account lockout	The act of making an account temporarily or permanently inactive after successive authentication failures.
Actions	Defined as part of policies, these verbs indicate what authorized subjects can do to resources.
Advice	In the context of a policy decision denying access, a hint to the policy enforcement point about remedial action to take that could result in a decision allowing access.
Agent administrator	User having privileges only to read and write policy agent profile configuration information, typically created to delegate policy agent profile creation to the user installing a policy agent.
Agent authenticator	Entity with read-only access to multiple agent profiles defined in the same realm; allows an agent to read web service profiles.
Application	In general terms, a service exposing protected resources. In the context of AM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.
Application type	Application types act as templates for creating policy applications. Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.

	Application types also define the internal normalization, indexing logic, and comparator logic for applications.
Attribute-based access control (ABAC)	Access control that is based on attributes of a user, such as how old a user is or whether the user is a paying customer.
Authentication	The act of confirming the identity of a principal.
Authentication chaining	A series of authentication modules configured together which a principal must negotiate as configured in order to authenticate successfully.
Authentication level	Positive integer associated with an authentication module, usually used to require success with more stringent authentication measures when requesting resources requiring special protection.
Authentication module	AM authentication unit that handles one way of obtaining and verifying credentials.
Authorization	The act of determining whether to grant or to deny a principal access to a resource.
Authorization Server	In OAuth 2.0, issues access tokens to the client after authenticating a resource owner and confirming that the owner authorizes the client to access the protected resource. AM can play this role in the OAuth 2.0 authorization framework.
Auto-federation	Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.
Bulk federation	Batch job permanently federating user profiles between a service provider and an identity provider based on a list of matched user identifiers that exist on both providers.
Circle of trust	Group of providers, including at least one identity provider, who have agreed to trust each other to participate in a SAML v2.0 provider federation.
Client	In OAuth 2.0, requests protected web resources on behalf of the resource owner given the owner's authorization. AM can play this role in the OAuth 2.0 authorization framework.
Conditions	Defined as part of policies, these determine the circumstances under which which a policy applies. Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved.

	Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.
Configuration datastore	LDAP directory service holding AM configuration data.
Cross-domain single sign-on (CDSSO)	AM capability allowing single sign-on across different DNS domains.
Delegation	Granting users administrative privileges with AM.
Entitlement	Decision that defines which resource names can and cannot be accessed for a given subject in the context of a particular application, which actions are allowed and which are denied, and any related advice and attributes.
Extended metadata	Federation configuration information specific to AM.
Extensible Access Control Markup Language (XACML)	Standard, XML-based access control policy language, including a processing model for making authorization decisions based on policies.
Federation	Standardized means for aggregating identities, sharing authentication and authorization data information between trusted providers, and allowing principals to access services across different providers without authenticating repeatedly.
Fedlet	Service provider application capable of participating in a circle of trust and allowing federation without installing all of AM on the service provider side; AM lets you create Java Fedlets.
Hot swappable	Refers to configuration properties for which changes can take effect without restarting the container where AM runs.
Identity	Set of data that uniquely describes a person or a thing such as a device or an application.
Identity federation	Linking of a principal's identity across multiple providers.
Identity provider (IdP)	Entity that produces assertions about a principal (such as how and when a principal authenticated, or that the principal's profile has a specified attribute value).
Identity repository	Data store holding user profiles and group information; different identity repositories can be defined for different realms.
Java EE policy agent	Java web application installed in a web container that acts as a policy agent, filtering requests to other applications in the container with policies based on application resource URLs.

Metadata	Federation configuration information for a provider.
Policy	Set of rules that define who is granted access to a protected resource when, how, and under what conditions.
Policy Agent	Agent that intercepts requests for resources, directs principals to AM for authentication, and enforces policy decisions from AM.
Policy Administration Point (PAP)	Entity that manages and stores policy definitions.
Policy Decision Point (PDP)	Entity that evaluates access rights and then issues authorization decisions.
Policy Enforcement Point (PEP)	Entity that intercepts a request for a resource and then enforces policy decisions from a PDP.
Policy Information Point (PIP)	Entity that provides extra information, such as user profile attributes that a PDP needs in order to make a decision.
Principal	Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities. When a Subject successfully authenticates, AM associates the Subject with the Principal.
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified subjects in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	AM unit for organizing configuration and identity information. Realms can be used for example when different parts of an organization have different applications and user data stores, and when different organizations use the same AM deployment. Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.
Resource	Something a user can access over the network such as a web page. Defined as part of policies, these can include wildcards in order to match multiple actual resources.
Resource owner	In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.

Resource server	In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.
Response attributes	Defined as part of policies, these allow AM to return additional information in the form of "attributes" with the response to a policy decision.
Role based access control (RBAC)	Access control that is based on whether a user has been granted a set of permissions (a role).
Security Assertion Markup Language (SAML)	Standard, XML-based language for exchanging authentication and authorization data between identity providers and service providers.
Service provider (SP)	Entity that consumes assertions about a principal (and provides a service that the principal is trying to access).
Session	The interval that starts with the user authenticating through AM and ends when the user logs out, or when their session is terminated. For browser-based clients, AM manages user sessions across one or more applications by setting a session cookie. See also Stateful session and Stateless session.
Session high availability	Capability that lets any AM server in a clustered deployment access shared, persistent information about users' sessions from the CTS token store. The user does not need to log in again unless the entire deployment goes down.
Session token	Unique identifier issued by AM after successful authentication. For a Stateful session, the session token is used to track a principal's session.
Single log out (SLO)	Capability allowing a principal to end a session once, thereby ending her session across multiple applications.
Single sign-on (SSO)	Capability allowing a principal to authenticate once and gain access to multiple applications without authenticating again.
Site	Group of AM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability. Use sticky load balancing based on `amlbcookie` values to improve site performance. The load balancer can also be used to protect AM services.
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateful session	An AM session that resides in the Core Token Service's token store. Stateful sessions might also be cached in memory on one or more

	AM servers. AM tracks stateful sessions in order to handle events like logout and timeout, to permit session constraints, and to notify applications involved in SSO when a session ends.
Stateless session	An AM session for which state information is encoded in AM and stored on the client. The information from the session is not retained in the CTS token store. For browser-based clients, AM sets a cookie in the browser that contains the session information.
Subject	Entity that requests access to a resource When a subject successfully authenticates, AM associates the subject with the Principal that distinguishes it from other subjects. A subject can be associated with multiple principals.
User data store	Data storage service holding principals' profiles; underlying storage can be an LDAP directory service, a relational database, or a custom `IdRepo` implementation.
Web policy agent	Native library installed in a web server that acts as a policy agent with policies based on web page URLs.