



User Guide

/ Web Agents 5.8.2.1

Latest update: 5.8.2.1

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

Copyright © 2011-2021 ForgeRock AS.

Abstract

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



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

ForgeRock® and ForgeRock Identity Platform™ are trademarks of ForgeRock Inc. or its subsidiaries in the U.S. and in other countries. Trademarks are the property of their respective owners.

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

DejaVu Fonts

Bitstream Vera Fonts Copyright

Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the names of Gnome, the Gnome Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the Gnome Foundation or Bitstream Inc., respectively. For further information, contact: fonts at gnome dot org.

Arev Fonts Copyright

Copyright (c) 2006 by Tavmjong Bah. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the modifications to the Bitstream Vera Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Tavmjong Bah" or the word "Arev".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Tavmjong Bah Arev" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL TAVMJONG BAH BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the name of Tavmjong Bah shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from Tavmjong Bah. For further information, contact: tavmjong @ free . fr.

FontAwesome Copyright

Copyright (c) 2017 by Dave Gandy, <https://fontawesome.com/>.

This Font Software is licensed under the SIL Open Font License, Version 1.1. See <https://opensource.org/licenses/OFL-1.1>.

Table of Contents

Preface	iv
1. Introducing Web Agents	1
Web Agent Components	1
Configuration Location	2
Request Process Flow	4
Web Agent Features	7
2. Preparing for Installation	29
Downloading and Unzipping Web Agents	29
Configuring AM to Sign Authentication Information	30
Creating Agent Profiles	32
Preparing your Environment for Secure Communication Between the Agents and AM	34
Supporting Load Balancers and Reverse Proxies Between AM and the Agents	35
3. Configuring Environments With Load Balancers and Reverse Proxies	36
Regarding Communication Between AM and Agents	38
Regarding Communication Between Clients and Agents	40
4. Installing Web Agents	48
Installing the Apache Web Agent	48
Installing the IIS Web Agent	66
Installing the NGINX Plus Web Agent	82
5. Post-Installation Tasks	95
Configuring Audit Logging	95
Configuring Whether Unix Web Agents Should Share Runtime Resources and Shared Memory	97
Supporting Load Balancers and Reverse Proxies Between Clients and Agents ...	100
Configuring Web Agents to Secure Communication with AM	100
6. Upgrading Web Agents	106
7. Removing Web Agents	109
Removing the Apache Web Agent	109
Removing the IIS Web Agent	110
Removing the NGINX Plus Web Agent	111
8. Troubleshooting	113
9. Reference	121
Configuring Web Agent Properties	121
Configuring Web Agent Environment Variables	195
Configuring Agent Authenticators	201
Command-Line Tool Reference	201
Configuring Apache HTTP Server as a Reverse Proxy Example	210
Glossary	212

Preface

This guide shows you how to install ForgeRock Access Management web server agents, as well as how to integrate with ForgeRock Access Management. Read the [Release Notes](#).

This guide is written for anyone installing web 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 Agents

A *web agent* is an Access Management add-on component that operates as a policy enforcement point (PEP) for a website deployed on a web server.

Web agents intercept inbound requests to websites and interact with AM to:

- Ensure that clients provide appropriate authentication.
- Enforce AM resource-based policies ¹.

This chapter covers how web agents work and how they can protect your websites.

Web Agent Components

Web agents comprise two main components:

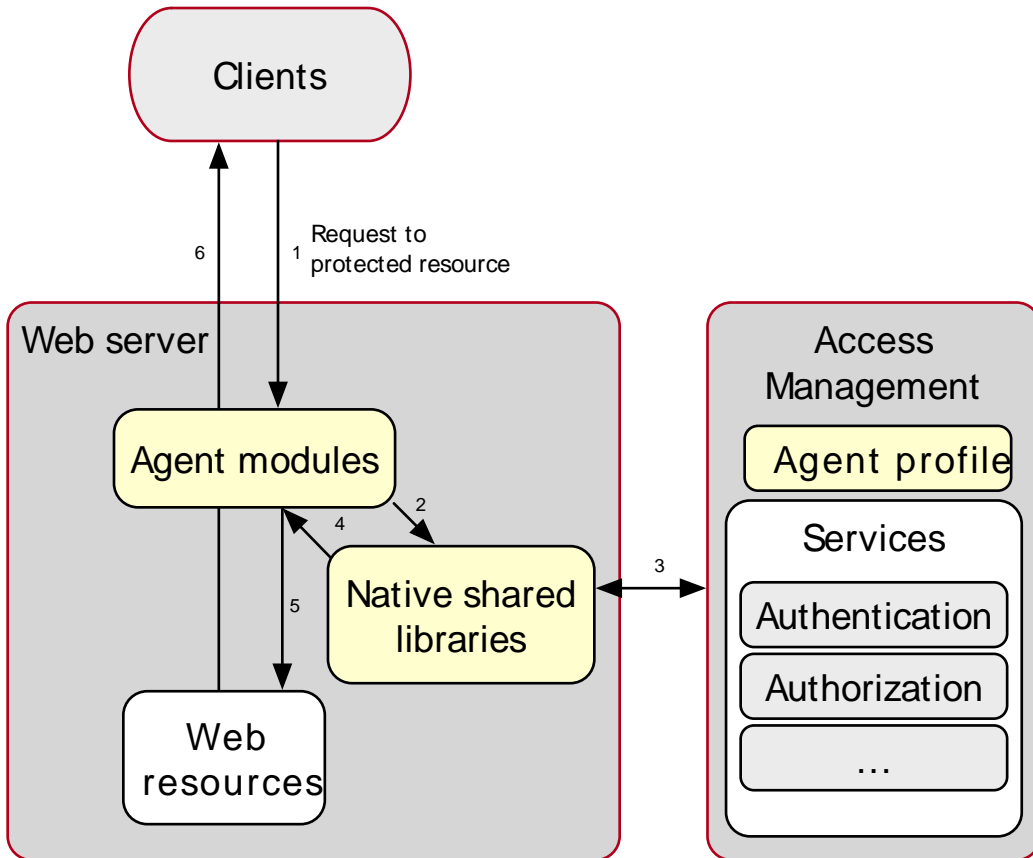
- **Agent Modules.** Intercept and process inbound requests to protected resources.
- **Native Shared Libraries.** Enable agents to interact with AM.

The *agent profile* is not strictly part of the web agent, but plays an important part in the agent's operation. It contains a set of configuration properties that define the web agent's behavior.

The following figure illustrates the web agent's components when the agent profile is stored in AM's configuration store:

¹ You can configure the web agent to only enforce user authentication. For more information, see "Web Agent Single Sign-on (SSO) Only Mode".

Web Agent



Configuration Location

Web agent configuration properties determine the behavior of the agent. AM stores configuration properties either centrally or locally:

- **Centralized configuration**

AM stores the web agent properties in the AM configuration store. Storing the agent configuration centrally allows you to configure your agents using the AM console, the **soadm** command, and the REST API.

To access the centralized web agent configuration, navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* in the AM console.

You can configure properties that are not present in the UI as *custom properties* in the Advanced tab. For a list of property names, see the "*Reference*".

+ *Where are the Custom Properties?*

In the AM console, go to Realms > *Realm Name* > Applications > Agents > Web > *Web Agent Name* > Advanced.

Custom Properties



When defined, properties and value pairs set as custom properties are the source of truth for that property. Therefore, make sure you do not configure a property as a custom property if it has a UI counterpart, since it may result in configuration mistakes.

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

- **Local configuration**

The web agent installer creates the file `/web_agents/agent_version/instances/Agent_nnn/config/agent.conf` to store the web agent configuration properties. The installer populates this file with enough information to make the web agent start. To manage the configuration, edit the file to add properties, remove properties, and change value. You cannot update this file using the AM console, the `ssoadm` command, or the REST API.

The `agent.conf` must contain at least the following properties:

```
### Bootstrap properties
com.sun.identity.agents.config.organization.name = /
com.sun.identity.agents.config.username = ApacheAgentProfile
com.sun.identity.agents.config.password = o70uvnaDnQ==
com.sun.identity.agents.config.key = OGM1MWewZWMtNmM4Zi00Yg=
com.sun.identity.agents.config.naming.url = https://openam.example.com:8443/openam

### Configuration properties
com.sun.identity.agents.config.repository.location = local
org.forgerock.openam.agents.config.jwt.name = am-auth-jwt
com.sun.identity.agents.config.cdsso.redirect.uri = agent/cdsso-oauth2
org.forgerock.openam.agents.config.policy.evaluation.application = iPlanetAMWebAgentService
org.forgerock.openam.agents.config.policy.evaluation.realm = /
com.sun.identity.agents.config.polling.interval = 60
com.sun.identity.agents.config.sso.cache.polling.interval = 3
com.sun.identity.agents.config.policy.cache.polling.interval = 3
com.sun.identity.agents.config.cookie.name = iPlanetDirectoryPro
com.sun.identity.agents.config.debug.file.size = 10000000
com.sun.identity.agents.config.local.logfile = /web_agents/agent_type/instances/agent_1/logs/debug/
debug.log
com.sun.identity.agents.config.local.audit.logfile = /web_agents/agent_type/instances/agent_1/logs/
audit/audit.log
com.sun.identity.agents.config.debug.level = Error
```

The properties previously discussed are provided with an example value. For information on each of these properties, see "Configuring Web Agent Properties".

Request Process Flow

Suppose you wanted to withdraw money from your bank account using an ATM. The ATM would not allow you to access your account unless you identified yourself to the bank with your card and PIN number. For a joint account, you may also require additional authorization to access the funds.

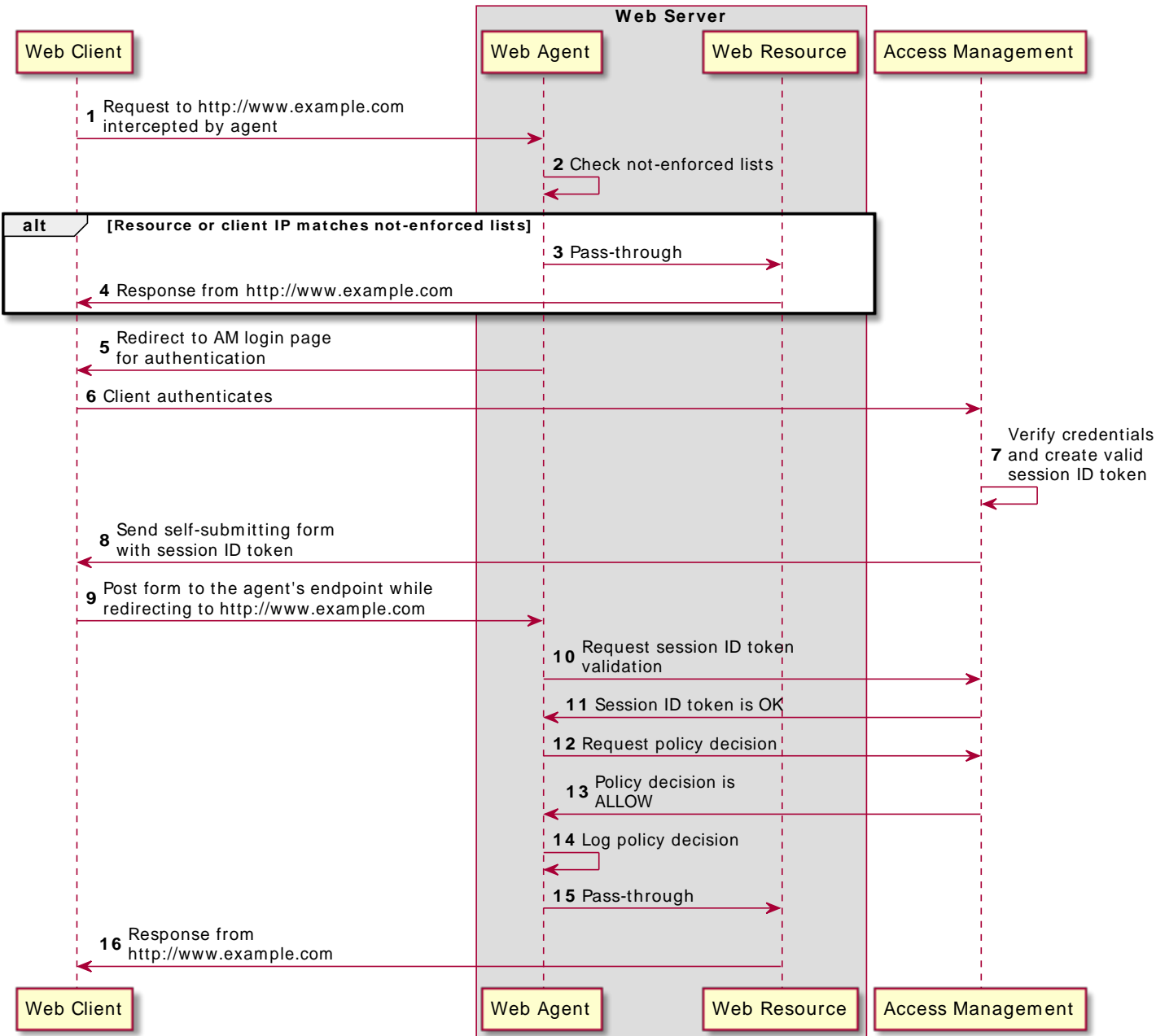
Web agents work on a similar premise. When a client requests access to a resource, the web agent intercepts the request. Then, AM validates the identity of the client as well as authorizes access the protected resource.

The following sequence diagram shows the simplified² flow that occurs when an unauthenticated client requests a resource protected by a web agent and AM:

²For a detailed diagram, see [Single Sign-On](#) in the *ForgeRock Access Management Authentication and Single Sign-On Guide*.

Web Agent Interaction

Web Agent Process Flow



1. An unauthenticated client attempts to access a resource at www.example.com. The agent intercepts the inbound request.
2. The agent evaluates whether the requested resource or the client IP address matches any rule contained in the *not-enforced lists*.
3. *Alternate Flow*. The requested resource or the client IP address matches a not-enforced rule. The agent allows access to the resource.
4. *Alternate Flow*. The client receives a response from www.example.com. The flow ends.
5. The requested resource or the client IP address does not match a not-enforced rule. The agent redirects the client to log in to AM.
6. The client authenticates to AM.

To protect against reply attacks, the agent issues *pre-authentication* cookies, named `agent-authn-tx`, to identify authentication requests to AM.

Depending on the value of `Multivalue` for `Pre-Authn Cookie`, the agent issues one pre-authentication cookie for *each* request, or one pre-authentication cookie for *all* requests.

The pre-authentication cookie expires after 5 minutes, or after the time specified in `Profile Attributes Cookie Maxage`.

If POST data preservation is enabled, the request expires after the time specified in `POST Data Entries Cache Period`, which is by default 10 minutes. In this case, consider increasing `Profile Attributes Cookie Maxage` to at least 10 minutes.

7. AM's Authentication Service verifies the client's credentials and creates a valid OpenID Connect (OIDC) ID token with session information.
8. AM sends the client a self-submitting form with the session ID token.
9. The client posts the self-submitting form to the agent's endpoint while redirecting to www.example.com again. The agent intercepts the requests and consumes the form.
10. The agent contacts AM to validate the session contained in the ID token.
11. AM validates the session.
12. The agent contacts AM's Policy Service, requesting a decision about whether the client is authorized to access the resource.
13. AM's Policy Service returns `ALLOW`.
14. The agent writes the policy decision to the audit log.
15. The agent enforces the policy decision. Since the Policy Service returned `ALLOW`, the agent performs a pass-through operation to return the resource to the client.
16. The client accesses the resource at www.example.com.

Web Agent Features

The Web Agent provides the following features to help you protect your applications:

- Multiple Sites and Virtual Host Support
- Web Agent Single Sign-on (SSO) Only Mode
- Not-Enforced URL and Client IP Lists
- Notification System
- Attribute Fetch Modes
- FQDN Checking
- Cookie Reset Properties
- Cross-Domain Single Sign-On
- Supporting Load Balancers
- Continuous Security
- Login Redirection and Login Conditional Redirection
- Logout Redirection
- POST Data Preservation
- Caching Capabilities
- Connection Pooling

Multiple Sites and Virtual Host Support

Web 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 Agents on a Virtual Host" and "Installing the IIS Web Agent".

Web Agent Single Sign-on (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 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 agents will also manage user authorization, by using the policy engine in AM.

For more information, see "SSO Properties".

Not-Enforced URL and Client IP Lists

The web 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](#).

To add not enforced URLs, navigate to Applications > Agents > Web > *Agent Name* > Application, and configure the Not Enforced URLs property.

You can specify the HTTP method that must be used to access the URL in order for it to be not enforced. For example, if you did not want to enforce `OPTIONS` HTTP requests to your scripts, you can specify a not-enforced URL rule as follows:

```
com.sun.identity.agents.config.notenforced.url[OPTIONS,1]=/scripts/*
```

Create separate rules to match multiple HTTP methods for a single URL, for example:

```
com.sun.identity.agents.config.notenforced.url[OPTIONS,1]=/scripts/*  
com.sun.identity.agents.config.notenforced.url[TRACE,2]=/scripts/*
```

Tip

Due to the different format for not enforced rules that apply an HTTP method filter, when using centralized configuration you must create these rules as Custom Properties. To do so, navigate to Applications > Agents > Web > *Agent Name* > Advanced, and add the not enforced rule into the Custom Properties field.

The web 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](#).

For more information on not-enforced lists, see ["Application Properties"](#).

Notification System

AM can notify web agents about configuration and session state changes through WebSockets. Web agents can subscribe to three notification feeds:

- **Configuration Notifications.** When the administrator makes a change to a hot-swappable web agent configuration property, AM sends a notification to the web agent to reread the agent profile from AM.

Configuration notifications are applicable when you store the web agent profile in AM's configuration data store.

- **Session Notifications.** When a client logs out or a CTS-based session expires, AM sends a notification to the web agent to remove the client's entry from the session cache.
- **Policy Notifications.** When an administrator changes a policy, AM sends a notification to the web agent to empty the session and policy cache.

Enabling notifications affects the validity of the web agent caches. For more information, see ["Caching Capabilities"](#). To enable notifications, configure the Agent Configuration Change Notification and Enable Notifications properties as described in the [Profile Global Properties](#) section.

The AM advanced server configuration property, `org.forgerock.openam.notifications.agents.enabled`, controls whether the AM server sends notifications to connected web agents. This property is enabled by default.

Note

Ensure that load balancers and reverse proxies configured in your environment support WebSockets.

Attribute Fetch Modes

Web 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 web 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](#).

FQDN Checking

The web 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 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 web 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](#).

Cookie Reset Properties

Web agents can reset cookies prior to redirecting the client to a login page for authentication by issuing a Set-Cookie header to the client to reset the cookie values.

Cookie reset is typically used when multiple parallel authentication mechanisms are in play with the web agent and another authentication system. The web 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](#).

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

For more information about cookie reset properties, see [Cookie Reset](#).

Cross-Domain Single Sign-On

Cross-domain single sign-on (CDSSO) is an AM capability that lets users access multiple independent services from a single login session, using the web agent to transfer a validated session ID on a single DNS domain or across domains.

Without AM's CDSSO, SSO cannot be implemented across domains; the session cookie from one domain would not be accessible from another domain. For example, in a configuration where the AM server (`openam.example.com`) is in a different DNS domain than the web agent (`myapp.website.com`), single sign-on would not be possible.

Web Agents work in CDSSO mode by default, regardless of the DNS domain of the AM servers and the DNS domain of the web agents.

For more information and implementation details, see [Single Sign-On](#) and [Implementing CDSSO in the ForgeRock Access Management Authentication and Single Sign-On Guide](#).

Supporting Load Balancers

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

If the web agent is running behind a load balancer, you can configure the web 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. Web 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 Environments With Load Balancers and Reverse Proxies](#)".

Also, web agents can communicate with an AM site configured behind a load balancer. To improve AM server performance in this scenario, ensure that the value of the `amlbcookie` cookie is set up to the AM's server ID. For more information, see [Configuring Site Sticky Load Balancing in the ForgeRock Access Management Setup Guide](#).

Continuous Security

Because web agents are the first point of contact between users and your business applications, they can collect inbound login requests' cookie and header information which an AM server-side authorization script can then process.

For example, you may decide that only incoming requests containing the `InternalNetwork` cookie can access intranet resources outside working hours.

For more information about configuring continuous security properties, see [Continuous Security Properties](#).

Login Redirection and Login Conditional Redirection

Web agents provide the capability to redirect users to a specific AM instance, an AM site, or a website other than AM. You can also redirect users based on the incoming request URL by configuring conditional redirection, which is available for login and logout requests.

For example, you can configure the web agent such that any login request made from the `france.example.com` domain is redirected to the `openam.france.example.com` AM site. You can also configure the web agent to redirect any user to a specific page after logout.

Web agents support the following login modes:

- "Default Login Redirection Mode"

The default login mode of the agent, which uses OpenID Connect ID tokens as session tokens and the AM UI end user pages to log in users.

Use the default redirection mode for all new implementations.

- "Custom Login Redirection Mode"

A special login mode meant to support environments with custom login pages that are upgrading from Web Agents 4.x. It lets the agent use AM-specific SSO tokens as session tokens.

Additionally, you can redirect users conditionally to AM instances or sites, custom pages, or realms. Use conditional redirection with any of the login modes.

Default Login Redirection Mode

The agent redirects unauthenticated users to the `/oauth2/authorize` endpoint. Therefore, unauthenticated users must be able to reach, at least, AM's `/oauth2/authorize` endpoint, as well as the AM user pages to which AM redirects for authentication.

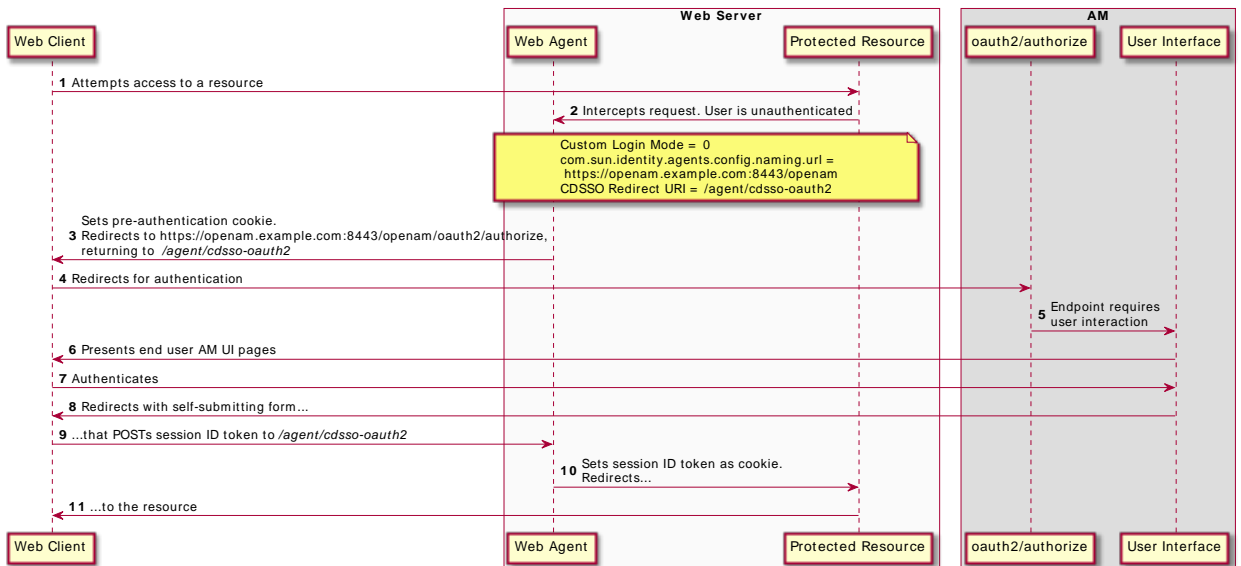
+ *If AM Is Behind a Proxy...*

Expose the following endpoints:

- `oauth2/authorize`
- `json/authenticate`
- `json/sessions`
- `json/serverinfo`
- `XUI/*`

After a successful authentication, AM returns the user's session as an ID token.

Default Login Redirection Flow



During the default flow, the agent uses the endpoint configured in the CDSSO Redirect URI property to process authentication requests. This endpoint is not the same as what the agent uses when processing custom login redirection mode requests.

Keep in mind:

- In this login mode, the agent redirects to the AM instance list defined by the `com.sun.identity.agents.config.naming.url` bootstrap property. This is the default.

To redirect to different AM instances or sites conditionally, or redirect to the realm to which users must authenticate, see "Conditional Redirection".

Reference Information

- Login URL Properties

Custom Login Redirection Mode

Enable the custom login redirection mode when your environment has custom login pages (as part of a migration from an earlier version of the agents). Custom login pages must be defined in the not-enforced URL or IP lists.

Important

The custom login redirection mode requires AM 6 or later.

Agents configured for the custom login redirection mode will use the default login redirection mode if the redirection URL contains the `/oauth2/authorize` endpoints. They will also use it as a fallback mechanism, in case they cannot see or validate the SSO token.

The custom login redirection mode supports two scenarios, depending on whether the custom login pages are in the same domain as the agent:

+ *Why Is the Domain Important?*

Cookies are only accessible to the domain they are set to. If the custom login pages set the SSO token cookie in the `example.com` domain, and the agent is in the `internal.com` domain, it will not be able to see the cookie.

Depending on your environment, the agent will still manage to contact AM to validate the cookie even if it cannot see it, but in other cases, you will need to configure an additional property.

- "Custom login pages are in the same domain as the agent"
- "Custom login pages are *not* in the same domain as the agent"

Custom login pages are in the same domain as the agent

In this scenario, the custom login pages set the SSO token in the domain where the agent is. Therefore, the agent can see the SSO token cookie and validate it against the AM endpoints.

You can configure the agent so that, at the end of the login flow, it does one of the following:

- Redirect the client to the protected resource they tried to access originally.

In this case, the agent tracks the user authentication using the pre-authentication cookie, and uses the special `/agent/custom-login-response` endpoint to process the authentication request. Unlike the endpoint for the default login flow, this one is not configurable.

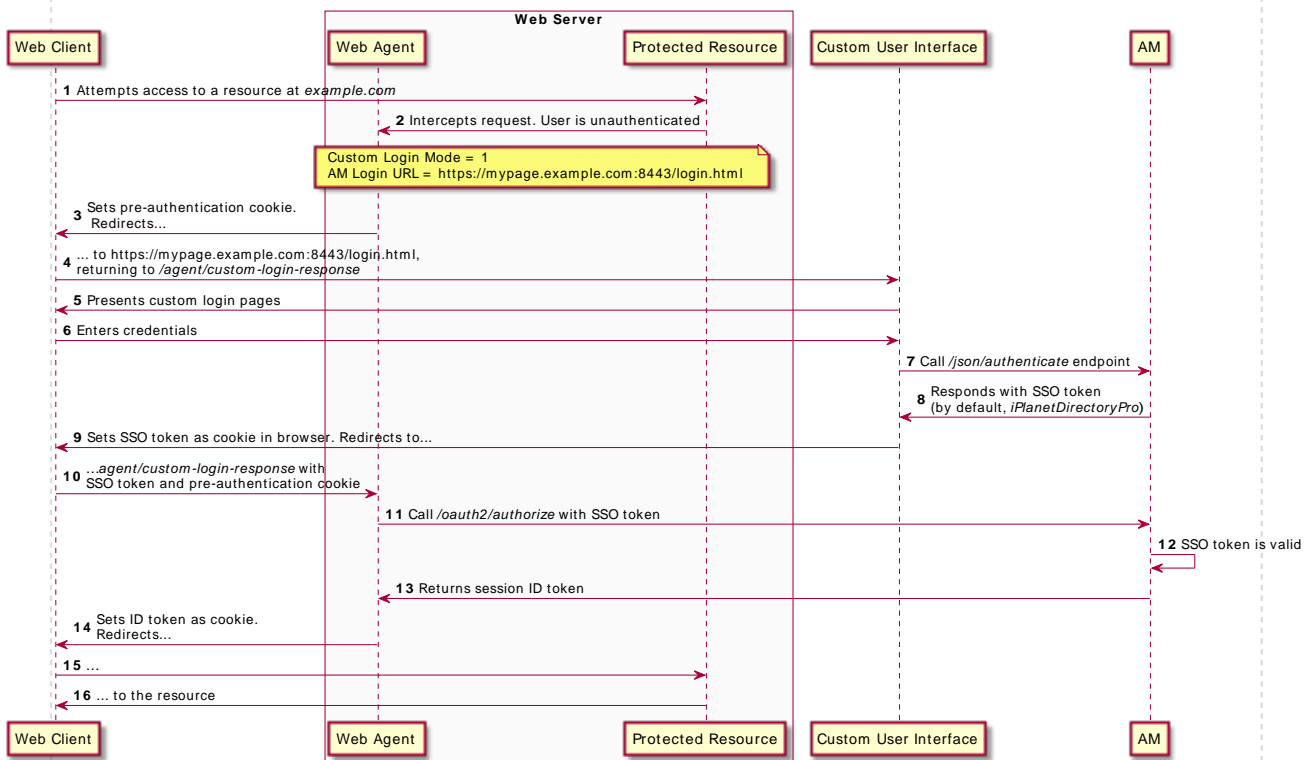
Then, the agent converts the SSO Token into an ID Token.

+ Example

+ Properties Involved

- Custom Login Mode (`org.forgerock.openam.agents.config.allow.custom.login`)
- AM Login URL (`com.sun.identity.agents.config.login.url`)

Custom Login Mode In the Same Domain that Redirects to the Protected Resource



- (Migration mode) Redirect the client with a `goto` query parameter to the originally requested resource

The custom login pages obtain the SSO token from AM, but the agent does not create the pre-authentication cookie, which is used (among other things) to protect against CSRF attacks.

Part of this flow happens outside the agent control, and therefore, the SSO token may expire or become invalid before the agent has a chance to validate it. In these cases, the user/client will need to authenticate again.

This mode only operates on HTTP GET requests. POST requests are not supported.

Caution

This is not a standard flow, and this feature is evolving. Use it only when migrating from earlier versions of the agents. Contact ForgeRock if you suspect your environment has a similar use case.

Custom login pages are not in the same domain as the agent

In this scenario, the login pages set the SSO token cookie in the login domain. Since the agent is in a different domain, it cannot see the cookie; therefore, it redirects to AM to follow the "Default Login Redirection Mode". If AM can validate the SSO token, it will return an ID token as part of the default redirection login flow.

Keep in mind:

- You must ensure that the login pages *do not* set the SSO token cookie with the `SameSite=Strict` attribute.
- If, for any reason, AM could not be able to validate the SSO token (for example, because it cannot recognize the domain set for the cookie), it will redirect the end user to authenticate again using the "Default Login Redirection Mode".

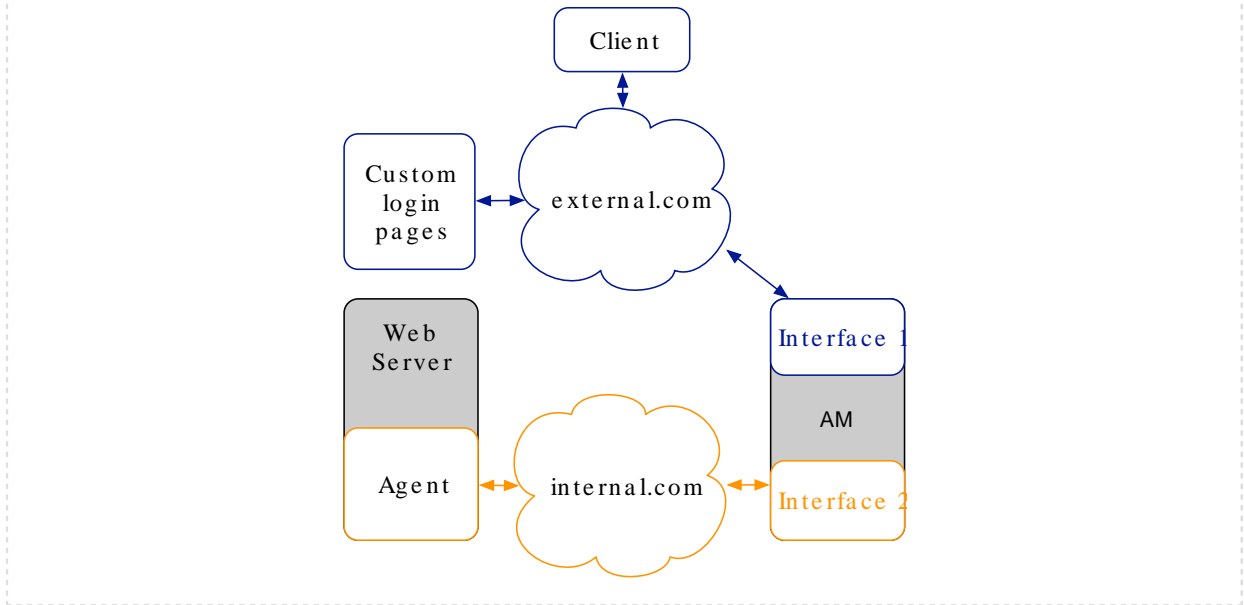
AM must be visible to the custom login pages, either because they both are in the same network/domain, or because you exposed the relevant AM endpoints using a proxy:

Shared Network

The server where AM is running has two interfaces: one connected to the internal network, where the agent is, and another connected to the external network, where the custom login pages are.

+ *Diagram*

The web server where the protected resources are may be connected to the external network in different ways; with two interfaces, or through a proxy. It is not important for the purposes of custom login, so it is not shown in the following diagram:



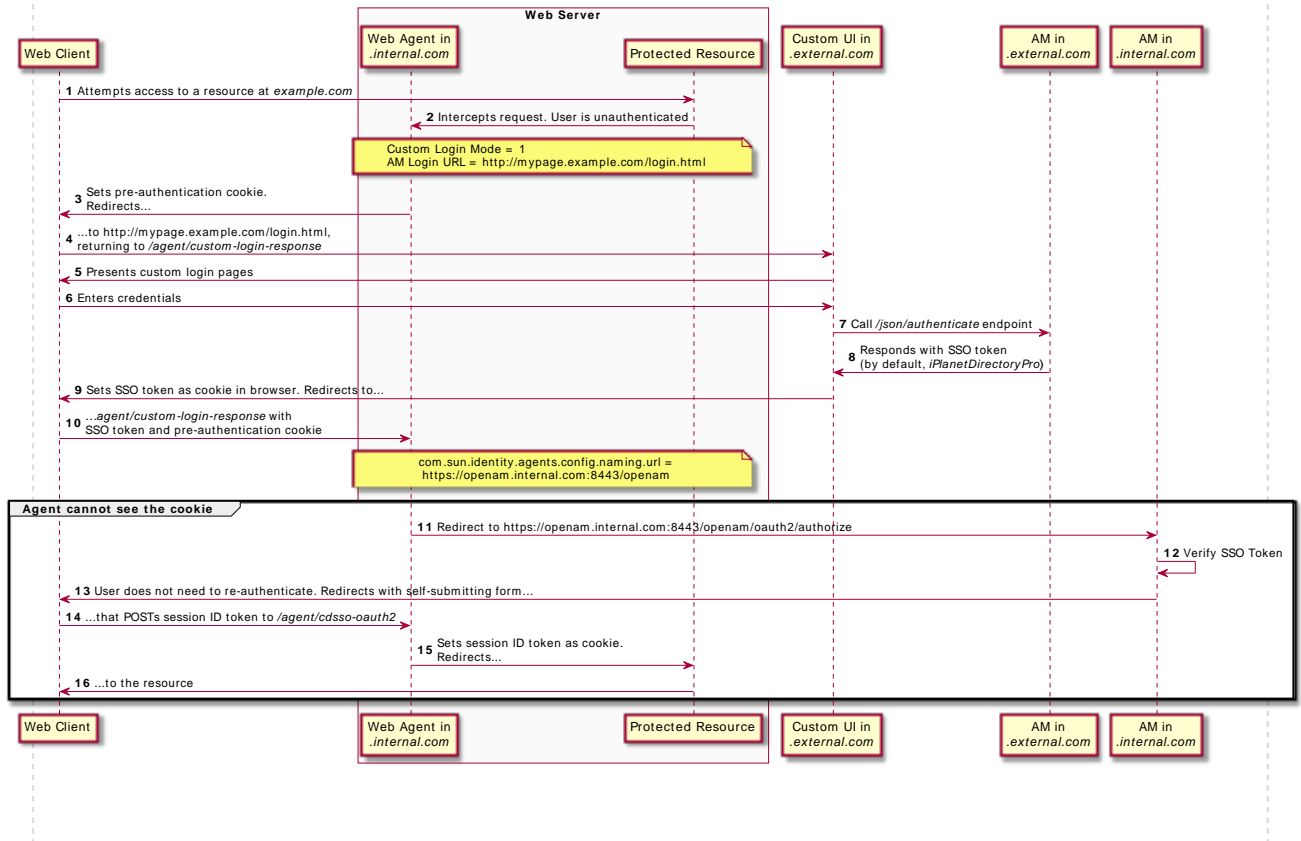
+ *Example*

+ *Properties Involved*

- Custom Login Mode (`org.forgerock.openam.agents.config.allow.custom.login`)

- AM Login URL (`com.sun.identity.agents.config.login.url`)

Custom Login Mode in Different Domains (Without Proxy, in the Same Network)

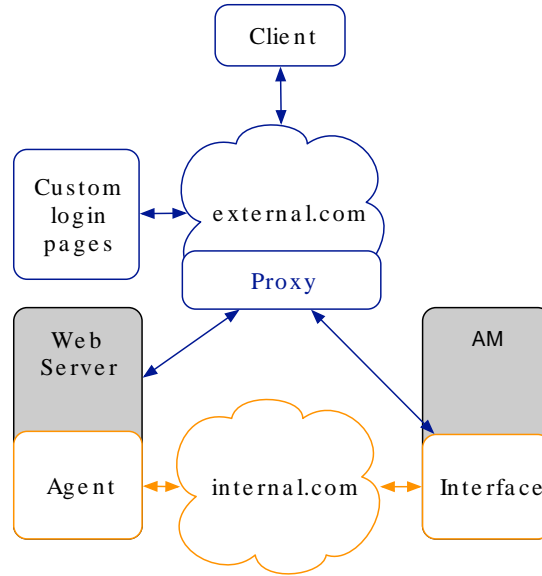


Using a Proxy

The server where AM is running has one interface to the internal network, where the agent is. A proxy hides AM from the external network, which forwards traffic to the `/oauth2/authorize` endpoint.

+ Diagram

The web server where the protected resources are may be connected to the external network in different ways; with two interfaces, or through a proxy. It is not important for the purposes of custom login, so it is not shown in the following diagram:



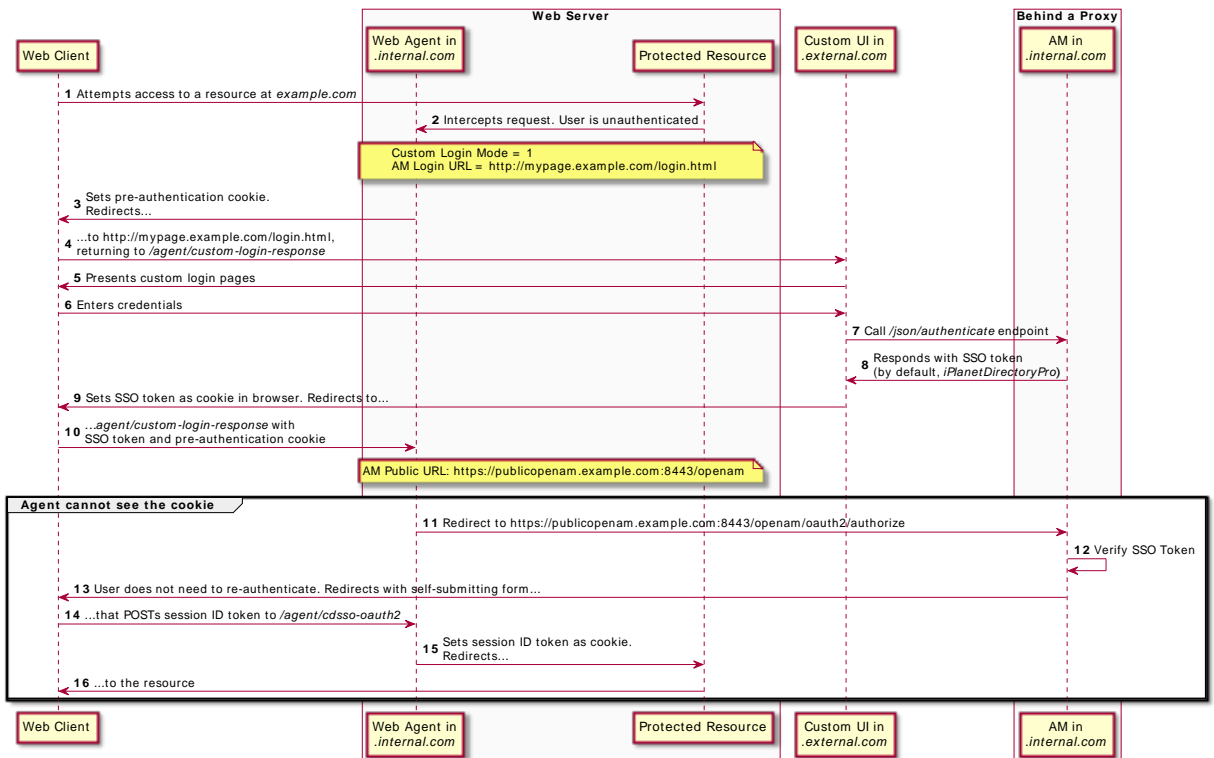
+ *Example*

+ *Properties Involved*

- Custom Login Mode (`org.forgerock.openam.agents.config.allow.custom.login`)
- AM Login URL (`com.sun.identity.agents.config.login.url`)

- Public AM URL (`com.forgerock.agents.public.am.url`)

Custom Login Mode in Different Domains (With proxy)



Reference information:

- Login URL Properties
- Profile Global Properties

Conditional Redirection

Conditional redirection allows the agent to redirect the end user to different AM instances or sites, or to different custom pages depending on the incoming request. Use conditional redirection, for example, to specify the realm to which users must authenticate.

Web agents supports the following types of conditional login redirection:

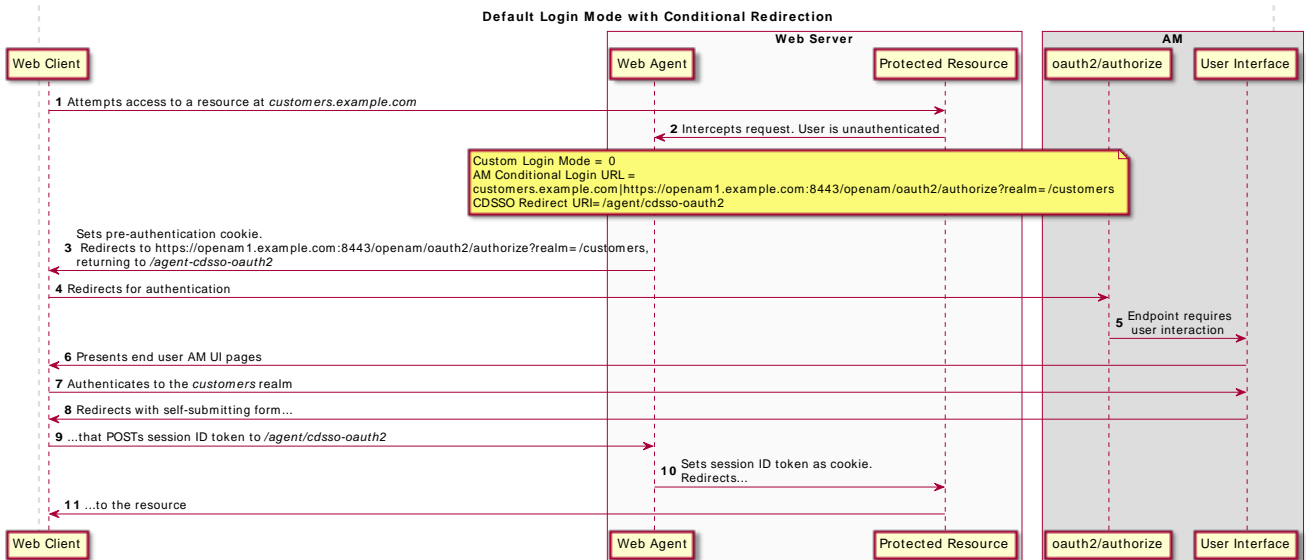
- Redirect to different AM realms, instances, or sites conditionally for authentication. Depending on the incoming request URL, the agent redirects to the URLs defined in the AM Conditional Login URL property.
- Redirect to different AM realms, instances, or sites for authentication, if the incoming URL matches a regular expression. In this case, configure the Regular Expression AM Conditional Login URL properties.

Examples:

+ *Default Login Mode with Conditional Redirection*

+ *Properties Involved*

- Custom Login Mode (`org.forgerock.openam.agents.config.allow.custom.login`)
- AM Conditional Login URL (`com.forgerock.agents.conditional.login.url`)
- CDSO Redirect URI (`com.sun.identity.agents.config.cdsso.redirect.uri`)



You can also use conditional redirection during the custom login redirection mode.

Related: "Default Login Redirection Mode".

Reference information

- Login URL Properties

Logout Redirection

Web agents can redirect users on logout to a specific AM page or to a custom logout page in your web server.

+ *Properties involved*

- Logout URL List (`com.sun.identity.agents.config.agent.logout.url`)
- Agent Logout URL Regular Expression (`com.forgerock.agents.agent.logout.url.regex`)
- OpenAM Logout URL (`com.sun.identity.agents.config.logout.url`)
- Logout Redirect URL (`com.sun.identity.agents.config.logout.redirect.url`)
- Invalidate Logout Session (`org.forgerock.agents.config.logout.session.invalidate`)
- Disabled Logout Redirection (`com.forgerock.agents.config.logout.redirect.disable`)
- Logout Cookies List for Reset (`com.sun.identity.agents.config.logout.cookie.reset`)

The logout flow is triggered when the incoming URL matches one of the values configured in the Logout URL List or the Agent Logout URL Regular Expression properties.

These pages must exist in your web server and should be the logout pages for your application.

If the incoming URL matches a logout URL, the agent creates a URL and redirects the web client to it. The URL contains:

- A logout page in your application or in AM. This page is configured in the OpenAM Logout URL property.

If the Invalidate Logout Session property is enabled, the *agent* invalidates the session in AM. Configure this if the Logout URL List property is set to a page in your application, and your application *does not handle* the session invalidation process.

If it is disabled, the logout page is responsible for invalidating the user session. Configure this if the Logout URL List property page is a SAML v2.0 logout page, the AM logout page, or a page in your application that can handle the session invalidation process.

- A `goto` parameter. Its value is the URL configured in the Logout Redirect URL property.

Configure this if you want the user to end on a specific page of your application after logout. For example, the landing page, or a login page. This page must exist in your web server.

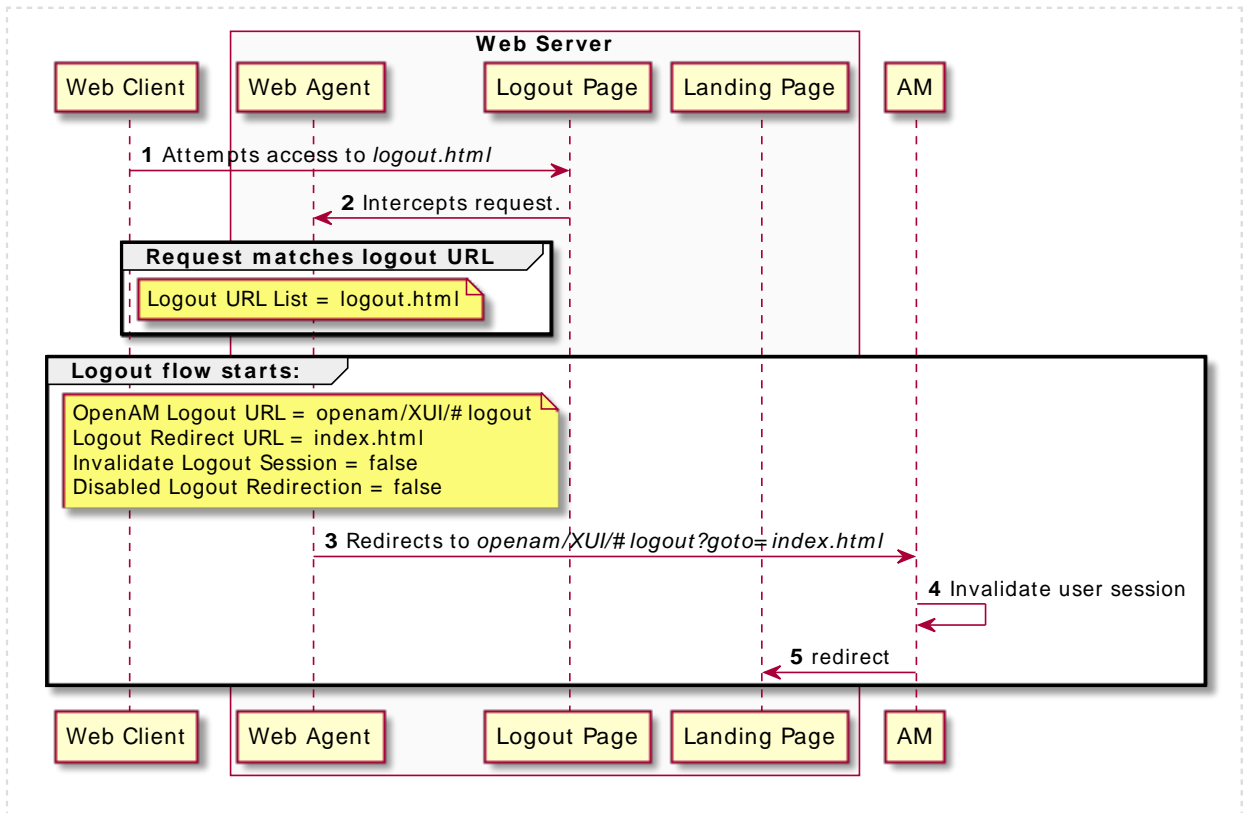
If the Disabled Logout Redirection property is true, the agent does not add the `goto` parameter, and the web client will remain in the logout page.

Tip

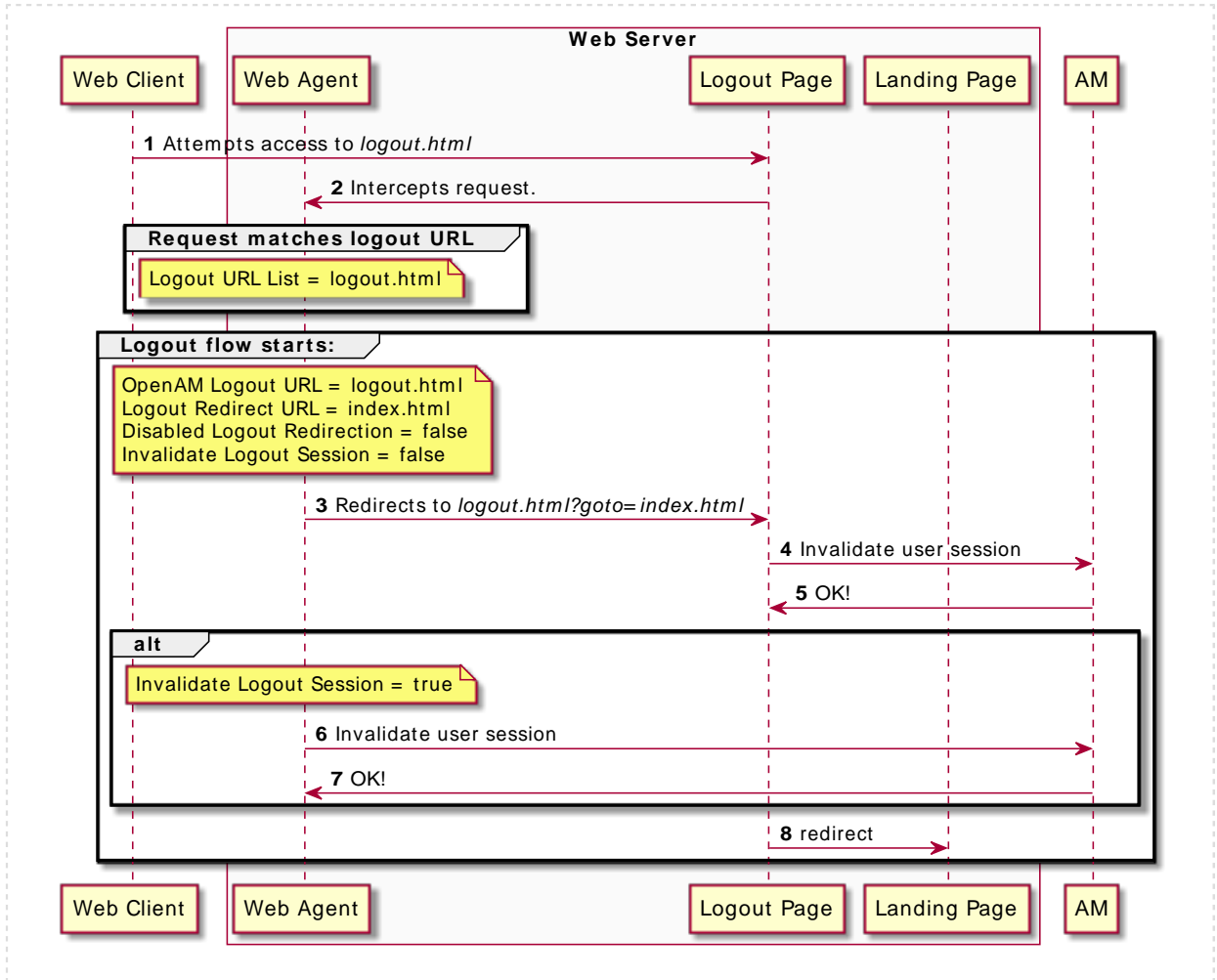
You can also configure the agent to reset specific cookies during logout by configuring the Logout Cookies List for Reset property.

Examples:

+ *Logout Flow with AM as the Logout Page*



+ *Logout Flow with the Application Serving the Logout Page*



Reference information:

- Logout URL Properties
- Agent Logout URL Properties

POST Data Preservation

Web agents can preserve HTML form data submitted as an HTTP POST by unauthenticated clients.

At a high level, when an unauthenticated client posts HTML POST data to a protected resource, the web agent stores the data in its cache and redirects the client to the login screen. Upon successful

authentication, the agent recovers the data stored in the cache and autosubmits it to the protected resource.

Consider enabling POST data preservation if users or clients in your environment submit large amounts of data, such as blog posts and wiki pages, and their sessions are short-lived.

Web agents guarantee the integrity of the data and the authenticity of the client as follows:

- Each unauthenticated form POST to a protected resource generates a random unique identifier. This identifier is then handled as follows:
 - The agent places it into a cookie and provides the cookie to the client.
 - The agent sends it to AM along with the authentication request for the client.
- After authentication, AM returns the session for the client alongside with the unique identifier. If the client cannot provide the identifier (because the cookie is missing) or the identifier differs from the one returned by AM, the web agent denies access to the stored POST data.

The unique identifier and the cookie protect the client against cross-site request forgery (CSRF) attacks by ensuring a request cannot be replayed after authentication unless it was originally sent in the same browser session within a finite time.

For more information about the POST data preservation cache and its properties, see "Caching Capabilities" and Post Data Preservation.

Caching Capabilities

Web Agents support the following caches to speed up agent operations:

Configuration Cache

The configuration cache stores web agent configuration properties.

When a web agent starts up, it either makes a call to AM to retrieve a copy of the agent profile (centralized configuration) or reads the agent profile from the local configuration file (local configuration). Then, the agent stores the configuration in its cache. The information stored in the cache is valid until one of the following events occur:

- AM notifies the agent of changes to hot-swappable web agent configuration properties. This only applies to deployments that use centralized configuration.
- The information in cache reaches the expiration time specified by the `com.sun.identity.agents.config.polling.interval` property.

When a configuration property in the cache is invalid, the web agent clears the cached property value and rereads it from the agent profile.

Session and Policy Decision Cache

Stored in the shared memory pool defined by the `AM_MAX_SESSION_CACHE_SIZE` environment variable, the session and policy decision cache stores session information, and the results of previous policy decisions.

The default size of the cache is 16 MB, but you may need to increase its size if you plan to hold many active sessions in the cache at any given time. For more information about the environment variable, see "Configuring Web Agent Environment Variables".

After authentication, AM presents the client with an ID token containing session information. The web agent stores part of that session information in the cache. When a client attempts to access a protected resource, the web agent checks whether there is a policy decision cached for the resource:

- If there is a cached policy decision, the agent reuses it without contacting AM.
- If there is no cached policy decision, the validity of the client's session determines the agent's behavior:
 - If the client's session is valid, the web agent requests a policy decision from AM, caches it, and then enforces it.
 - If the client's session is not valid, the agent redirects the client to AM for authentication regardless of why the session is invalid. The web agent does not specify the reason why the client needs to authenticate.

Once the client authenticates and the session is cached, the web agent requests a policy decision from AM, caches it, and then enforces it.

Session and policy decisions are valid in the cache until one of the following events occur:

Session and Policy Decision Validity in Cache

Event	What is invalidated?
Session contained in the ID token expires	Session and policy decisions related to the session
Client logs out from AM (and session notifications are enabled)	Session and policy decisions related to the session
Session reaches the expiration time specified by the <code>com.sun.identity.agents.config.sso.cache.polling.interval</code> property	Session
Policy decision reaches the expiration time specified by the <code>com.sun.identity.agents.config.policy.cache.polling.interval</code> property	Policy decision
Administrator makes a change to policy configuration (and policy notifications are enabled)	All sessions and all policy decisions

Important

A web agent that loses connectivity with AM cannot request policy decisions. Therefore, the web agent denies access to inbound requests that do not have a policy decision cached until the connection is restored(*).

For more information about properties related to the session and policy decision cache, see Policy Client Service Properties.

Policy Cache

The policy cache builds upon the policy decision cache. It downloads and stores details about policies from AM, and uses the downloaded policies to make authorization decisions, without contacting AM each time.

Web agents use the policy cache without contacting AM in the following situations:

- A requested resource matches the resource pattern of a policy that has been cached due to a previous evaluation.
- A requested resource **does not** match any cached policy patterns. In this case, the agent denies access immediately.
- A requested resource matches the resource pattern of a simple policy that applies to the **All Authenticated Users** virtual group.

If the resource matches the policy used for a previous policy decision, the agent does not request policy evaluation from AM. Therefore, policy conditions based on scripts, LDAP filter conditions, or session properties, which rely on attributes that can vary during a session, may not be enforced.

To reduce this risk, you should:

- Enable the session property change notification feature. See "Notification System".
- Reduce the amount of time that sessions can remain in the agent session cache. See Policy Client Service Properties.

Caveats

The following caveats apply when using the policy cache:

- If you have a large number of policies, for example more than one million in an UMA deployment, the time to download the policies and the memory consumption of the agent may affect performance.
- The agent downloads the policy rules, and uses them to evaluate policies locally. If a policy is customized in AM in a way that changes the way it is evaluated (for example, a wildcard or delimiter is changed), the policy decision made by the agent might not match the policy defined in AM.

- Even though delimiters and wildcards are configurable in AM (Configure > Global Services > Policy Configuration > Global Attributes > Resource Comparator), the policy cache only supports the default configuration.

Do not enable the agent's policy cache if your policies use custom delimiters and/or wildcards.

Enable the policy cache by creating an environment variable named `AM_POLICY_CACHE_MODE`.

Change the location of the policy cache by creating an environment variable named `AM_POLICY_CACHE_DIR`.

For more information about properties related to the policy cache, see "Configuring Web Agent Environment Variables".

POST Data Preservation Cache

Stored in files saved in the agent installation directory, the POST data preservation cache stores short-lived POST data.

When POST data preservation is enabled (`com.sun.identity.agents.config.postdata.preserve.enable`), the web agent caches HTML form data submitted as an HTTP POST by unauthenticated clients. By default, this data is stored in the directory specified by the `org.forgerock.agents.config.postdata.preserve.dir` property.

POST data information is cached for the amount of time specified by the POST Data Entries Cache Period (`com.sun.identity.agents.config.postcache.entry.lifetime`) property.

For more information about POST data preservation, see "POST Data Preservation" and Post Data Preservation.

Connection Pooling

By default, the agent uses connection pooling to improve performance when AM is available over low bandwidth connections, or to throttle the maximum number of connections made by the agent.

When AM is available over high bandwidth connections, connection pooling can reduce performance.

Enable and disable connection pooling with the bootstrap property `org.forgerock.agents.config.connection.pool.enable`.

Chapter 2

Preparing for Installation

This chapter covers tasks to perform before installing web agents in your environment. The following table contains a list of the tasks:

Task	Section
Download web agent binaries	Section
Secure communications between AM and the web agents	Section
Create agent profiles	Section
Ensure that the correct SSL libraries are available to the web agent	Section
Configure your environment when communication between AM and agents happens behind load balancers or reverse proxies	Section

Downloading and Unzipping Web Agents

Download the product software from the [ForgeRock BackStage download site](#). 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 web 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 agents. The directory is empty when first extracted.

Important

Agent configuration files are created in `instances/agent_n/config/agent.conf`.

Ensure this path, including the parent path, does not exceed 260 characters in length.

Legal/

Contains licensing information including third-party licenses.

lib/

Contains shared libraries used by the web agent.

log/

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

When the web agent is running, the directory may also contain the following files:

- POST data preservation files (configurable in the `org.forgerock.agents.config.postdata.preserve.dir` property).
- The `system_n.log` file, where the agent logs information related to agent tasks running in the background.

Web agents timestamp events in coordinated universal time (UTC).

- The backup of the site and application configuration files created after running the **agentadmin -g** command (IIS web agent only).
- Files related to the web agent caches (IIS web agent only).

Configuring AM to Sign Authentication Information

AM communicates all authentication and authorization information to web agents using OpenID Connect ID tokens. To secure the integrity of the JSON payload (outlined in the JSON Web Algorithm specification RFC 7518), AM and the web agent support signing the tokens for communication with the RS256 algorithm.

AM also uses an HMAC signing key to protect requested **ACR** claims values between sending the user to the authentication endpoint, and returning from successful authentication.

By default, AM uses a demo key and an autogenerated secret for these purposes. For production environments, perform the steps in one of the following procedures to create new key aliases and configure them in AM:

- "To Configure AM Secret IDs for the Agents' OAuth 2.0 Provider in AM 6.0 or earlier"

- "To Configure AM Secret IDs for the Agents' OAuth 2.0 Provider in AM 6.5 or later"

To Configure AM Secret IDs for the Agents' OAuth 2.0 Provider in AM 6.0 or earlier

By default, AM 6.0 or earlier signs the session ID tokens with the `test` key alias provided in AM's JCEKS keystore and sign the claims with a secret autogenerated at time.

Perform the following steps to create and set up a new key and a new secret in AM 6.0 or earlier:

1. Create the following aliases in one of the secret stores configured in AM, for example, the default JCEKS keystore:
 - a. Create an RSA key pair.

For more information about creating a key alias in the AM keystore, see the section *Creating Key Aliases* of the *ForgeRock Access Management Security Guide*.
 - b. Create an HMAC secret.
2. In the AM console, navigate to `Configure > Global Services > OAuth2 Provider`.
3. Perform the following actions:
 - a. Replace the `test` key alias in the `ID Token Signing Key Alias for Agent Clients` field with the new RSA key alias.
 - b. Replace the value in the `Authenticity Secret` field with the new HMAC secret.

Note that you may already have a secret configured for this secret ID, since it is also used for signing certain OpenID Connect ID tokens and remote consent requests.
 - c. Save your changes.

No further configuration is required in the agents.

To Configure AM Secret IDs for the Agents' OAuth 2.0 Provider in AM 6.5 or later

By default, AM 6.5 or later is configured to:

- Sign the session ID tokens with the secret mapped to the `am.global.services.oauth2.oidc.agent.idtoken.signing` secret ID. This secret ID defaults to the `rsajwt signingkey` key alias provided in AM's JCEKS keystore.
- Sign the claims with the secret mapped to the `am.services.oauth2.jwt.authenticity.signing` secret ID. This secret ID defaults to the `hmac signingtest` key alias available in AM's JCEKS keystore.

Perform the following steps to create and set up new keys on a keystore secret store:

1. Create the following aliases in one of the secret stores configured in AM, for example, the default JCEKS keystore:

- a. Create an RSA key pair.
 - b. Create an HMAC secret.
2. In the AM console, navigate to Configure > Secret Stores > *Keystore Secret Store Name* > Mappings.
 3. Configure the following secret IDs:
 - a. Configure the new RSA key alias in the `am.global.services.oauth2.oidc.agent.idtoken.signing` secret ID.
 - b. Configure the new HMAC secret in the `am.services.oauth2.jwt.authenticity.signing` secret ID.

Note that you may already have a secret configured for this secret ID, since it is also used for signing certain OpenID Connect ID tokens and remote consent requests. For more information, see [Secret ID Default Mappings](#) in the *ForgeRock Access Management Security Guide*.

- c. Save your changes.

For more information about secret stores, see the chapter [Configuring Secret Stores](#) of the *ForgeRock Access Management Security Guide*.

No further configuration is required in the agents.

Creating Agent Profiles

A web 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, go to Realms > *Realm Name* > Applications > Agents > Web, and add an agent.
2. Complete the web form using the following hints:

Agent ID

The ID of the agent profile, used during the agent installation.

Agent URL

The URL the 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.

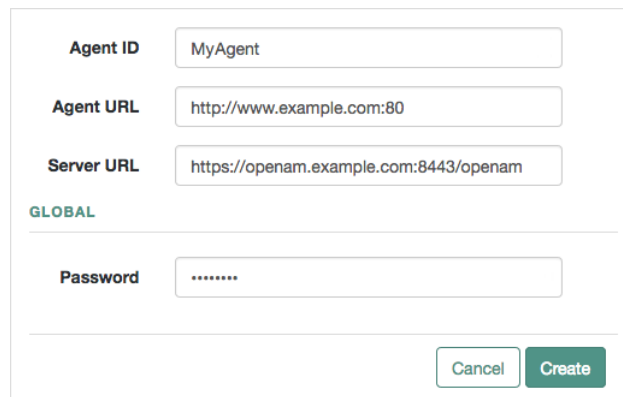
Server URL

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.

Password

The password the agent uses to authenticate to AM. Use this password when installing an agent.



The screenshot shows a configuration form with the following fields:

- Agent ID:** MyAgent
- Agent URL:** http://www.example.com:80
- Server URL:** https://openam.example.com:8443/openam

Below these fields is a section labeled **GLOBAL** containing:

- Password:** A masked password field (represented by seven dots).

At the bottom right of the form are two buttons: **Cancel** and **Create**.

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 Add Group in the Group tab, and provide an ID 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.

Tip

You can also create agent profiles by using the `/realm-config/agents/WebAgent/{id}` endpoint in the REST API.

For more information, navigate to the API Explorer in your AM instance.

Preparing your Environment for Secure Communication Between the Agents and AM

Web agents require either OpenSSL or the Windows built-in Secure Channel API to be available at install time. Unix agents only support OpenSSL, while Windows agents support both OpenSSL and the Windows Secure Channel API.

For information about supported OpenSSL versions, see "OpenSSL Requirements" in the *Release Notes*.

Before installing web agents, ensure that the OpenSSL libraries are located or referenced as shown in the following table:

OpenSSL Library Location by Operating System

Operating System	OpenSSL Library	Location or Variable
Windows 32-bit	libeay32.dll ssleay32.dll libcrypto-1_1.dll ^a libssl-1_1.dll ^a	\windows\syswow64
Windows 64-bit	libeay64.dll ssleay64.dll libcrypto-1_1-x64.dll ^a libssl-1_1.dll ^a	\windows\system32
Linux	libcrypto.so libssl.so	\$LD_LIBRARY_PATH or \$LD_LIBRARY_PATH_64
AIX	libcrypto.so libssl.so	\$LIBPATH

^aOpenSSL 1.1.0+ only

Note

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

Supporting Load Balancers and Reverse Proxies Between AM and the Agents

When your environment has reverse proxies or load balancers configured between the agents and AM, you must perform additional configuration in both AM and your environment before installing the agents.

Failure to do so may cause the agent installation to fail, or it may compromise the agent's functionality.

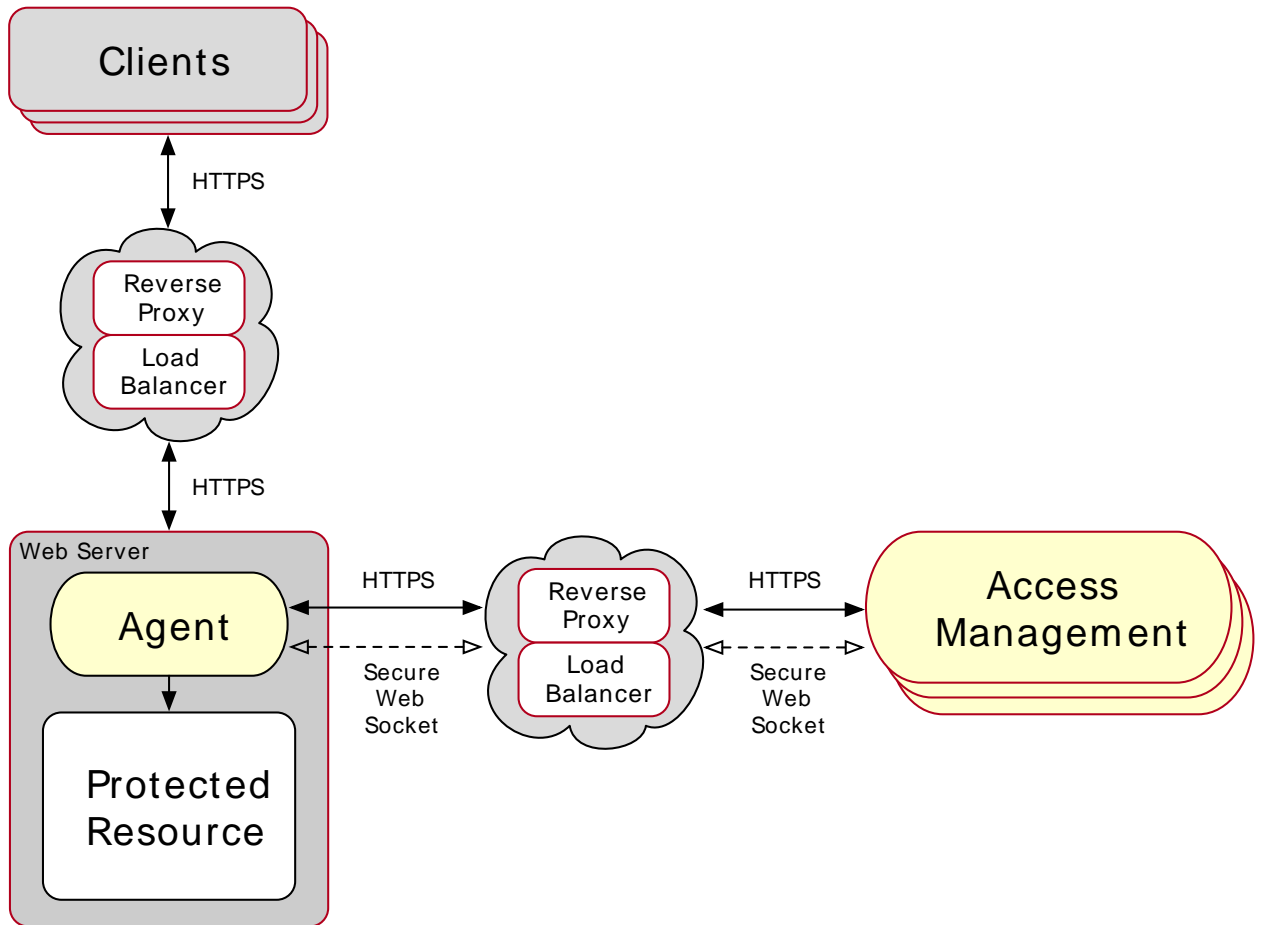
For more information, see "*Configuring Environments With Load Balancers and Reverse Proxies*".

Chapter 3

Configuring Environments With Load Balancers and Reverse Proxies

When working with AM and agents, the most common deployment scenario is to configure a load balancer and a reverse proxy between the clients and the agents, and another load balancer and reverse proxy between the agent and an AM site, as shown in the following diagram:

Web Agents in Environments with Load Balancers and Reverse Proxies



Usually, you want to anonymize client traffic as it gets into your network by using a reverse proxy, then balance the load among different web servers and agents.

AM sites are usually deployed behind a load balancer so the load can be spread among different instances. A reverse proxy may be deployed in front of the AM site to protect its APIs, too.

Note that the reverse proxy and the load balancer may be the same entity. In very complex environments, there may be more than the depicted load balancers and reverse proxies deployed in the network.

In any case, when installing web agents in an environment with load balancers or reverse proxies, you must consider the communication between the clients and the web agents, and between the agents and the AM servers.

Refer to the following sections for more information:

- "Regarding Communication Between AM and Agents".
- "Regarding Communication Between Clients and Agents".

Regarding Communication Between AM and Agents

Before attempting to install web agents in an environment where AM is behind a load balancer, reverse proxy, or both, consider the following points:

Agent's IP Address and/or FQDN

When a load balancer or a reverse proxy is configured between AM and the web agents, the agents' IP addresses and FQDNs are concealed by the load balancer/reverse proxy's own IP or FQDN. As a result, AM cannot determine the agents' base URL as expected.

This could cause trouble during the installation process and also hinder functionality such as redirection using the `goto` parameter.

Therefore, you must configure the following:

- The load balancer or reverse proxy, to forward the agents' IP address and/or FQDN in a header.
- The AM site, to recover the forwarded headers. For more information, see "Configuring AM to Use Forwarded Headers".

Note

A load balancer or reverse proxy conceals the AM instances' IP addresses and FQDNs. When installing web agents, use the load balancer or reverse proxy IP address or FQDN as the point of contact for the AM site.

AM Sessions and Session Stickiness

When web agents communicate with an AM site that is behind a load balancer, improve policy evaluation performance by setting AM's sticky cookie (by default, `amlbcookie`) to the AM's server ID. For more information, see *Configuring Site Sticky Load Balancing in the ForgeRock Access Management Setup Guide*.

Important

When configuring multiple agents behind a load balancer or reverse proxy, take into consideration whether you use one or multiple agent profiles, since it impacts sticky load balancer requirements:

- If the agents are configured with multiple agent profiles, you must configure sticky load balancing. This is because the agent profile name is contained in the session ID token the agent and AM use to communicate. Without session stickiness, there is no way to make sure that the appropriate session ID token ends in the appropriate web agent instance.

To disable validation of the `aud` claim in the session ID token, enable the `com.forgerock.agents.jwt.aud.disable` property or configure `com.forgerock.agents.jwt.aud.whitelist` property. This way, you can have multiple agent profiles without sticky load balancing.

We recommend that you use this approach sparingly and mostly for migrations, because, for security reasons, agents should validate all the claims in the session ID tokens.

- If multiple agents are configured with the same agent profile, you can decide whether to configure sticky load balancing or not depending on other requirements of your environment.

WebSockets

Your load balancers and reverse proxies must support the WebSocket protocol for communication between the web agents and the AM servers.

For more information, refer to the load balancer or proxy documentation.

Tip

For an example of how to configure Apache HTTP as a reverse proxy, see "Configuring Apache HTTP Server as a Reverse Proxy Example".

Configuring AM to Use Forwarded Headers

When web agents are behind a load balancer or reverse proxy, you must configure AM to recover the forwarded headers that expose the agents' real IP address or FQDN.

To Configure AM to Use Forwarded Headers

To configure how AM obtains the base URL of web agents, use the Base URL Source service:

1. Log in to the AM console as 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.

Regarding Communication Between Clients and Agents

When your environment has load balancers or reverse proxies between clients and agents, you must consider the following points:

Client's IP Address and/or FQDNs

When configuring web agents behind a load balancer or reverse proxy, the clients' IP addresses and FQDNs are hidden by the load balancer's IP or FQDN, which results in agents not being able to determine the clients' base URLs.

Therefore, you must configure the load balancer or reverse proxy to forward the client's IP address and/or the client's FQDN in a header. Failure to do so will prevent the agent from performing policy evaluation, and applying not-enforced and conditional login/logout rules.

For more information, see "Configuring Client Identification Properties".

POST Data Preservation

When using POST data preservation, you must use sticky load balancing to ensure that the client always hits the same agent and, therefore, their saved POST data.

Web agents provide properties to set either a sticky cookie or a URL query string for load balancers and reverse proxies.

For more information, see "Configuring POST Data Preservation for Load Balancers or Reverse Proxies".

Web Server FQDNs, Ports, and Protocols

When the protected web servers and their agents are behind a load balancer or reverse proxy, it is imperative that the agent is configured to match the load balancer FQDN, port, and protocol.

Failure to do so would make the agent to return HTTP 403 errors when clients request access to resources.

There are two use-cases:

- The load balancer or reverse proxy forwards requests and responses between clients and protected web servers only. In this case, ports and protocols configured in the web server match those on the load balancer or reverse proxy, but FQDNs do not.
- The load balancer or reverse proxy also performs SSL offloading, terminating the SSL traffic and converting the requests reaching the web server to HTTP. This reduces the load on the protected servers, since the processing of the public key is usually done by a hardware accelerator.

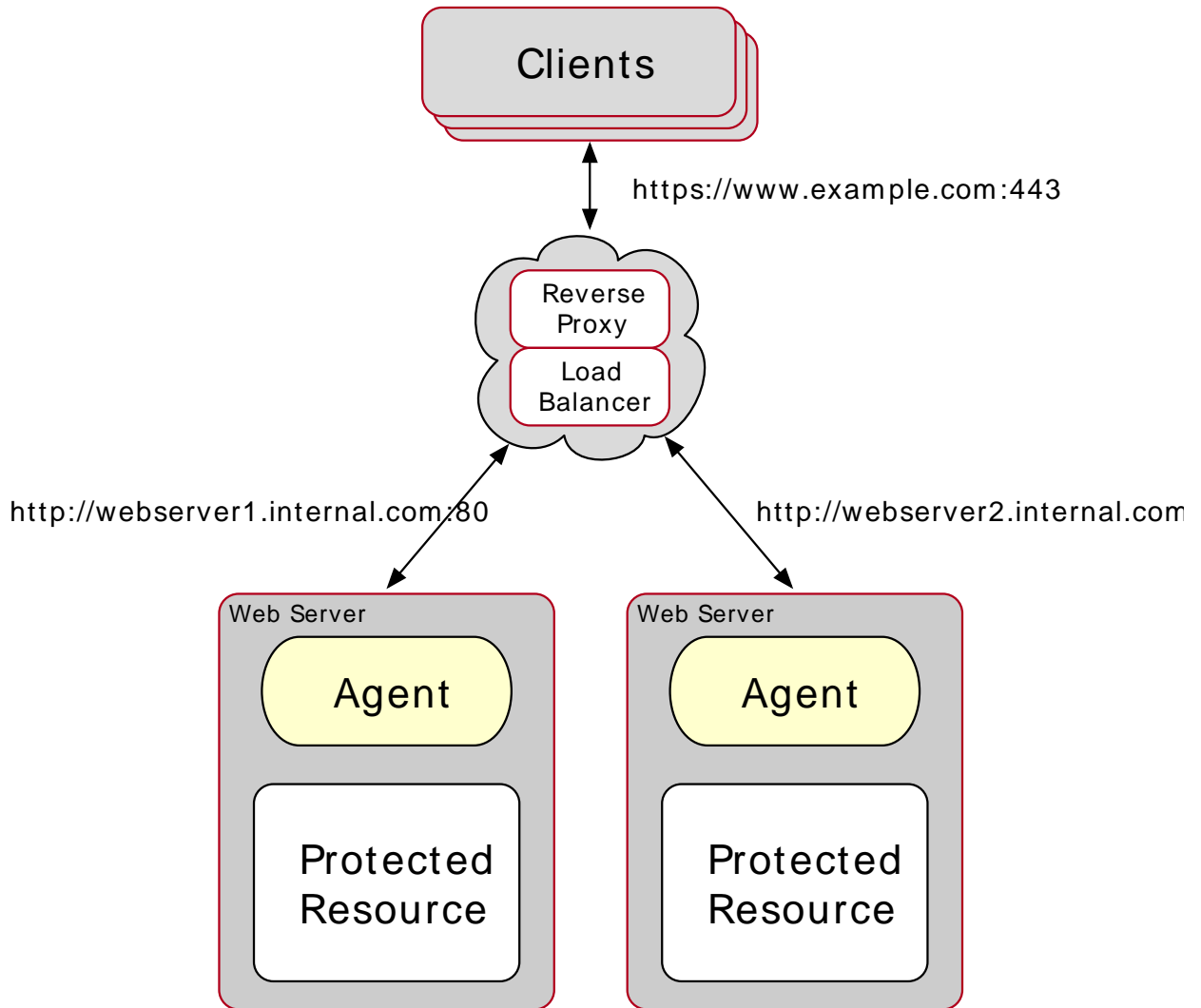
In this case, neither ports, protocols, or FQDNs match.

For more information about matching FQDNs, ports and protocols, see "Matching Protected Web Server Ports, Protocols, and FQDNs".

Matching Protected Web Server Ports, Protocols, and FQDNs

When the protocol and port configured on the load balancer or reverse proxy differ from those configured on the protected web server, you must override them in the web agent configuration. The following diagram illustrates this scenario:

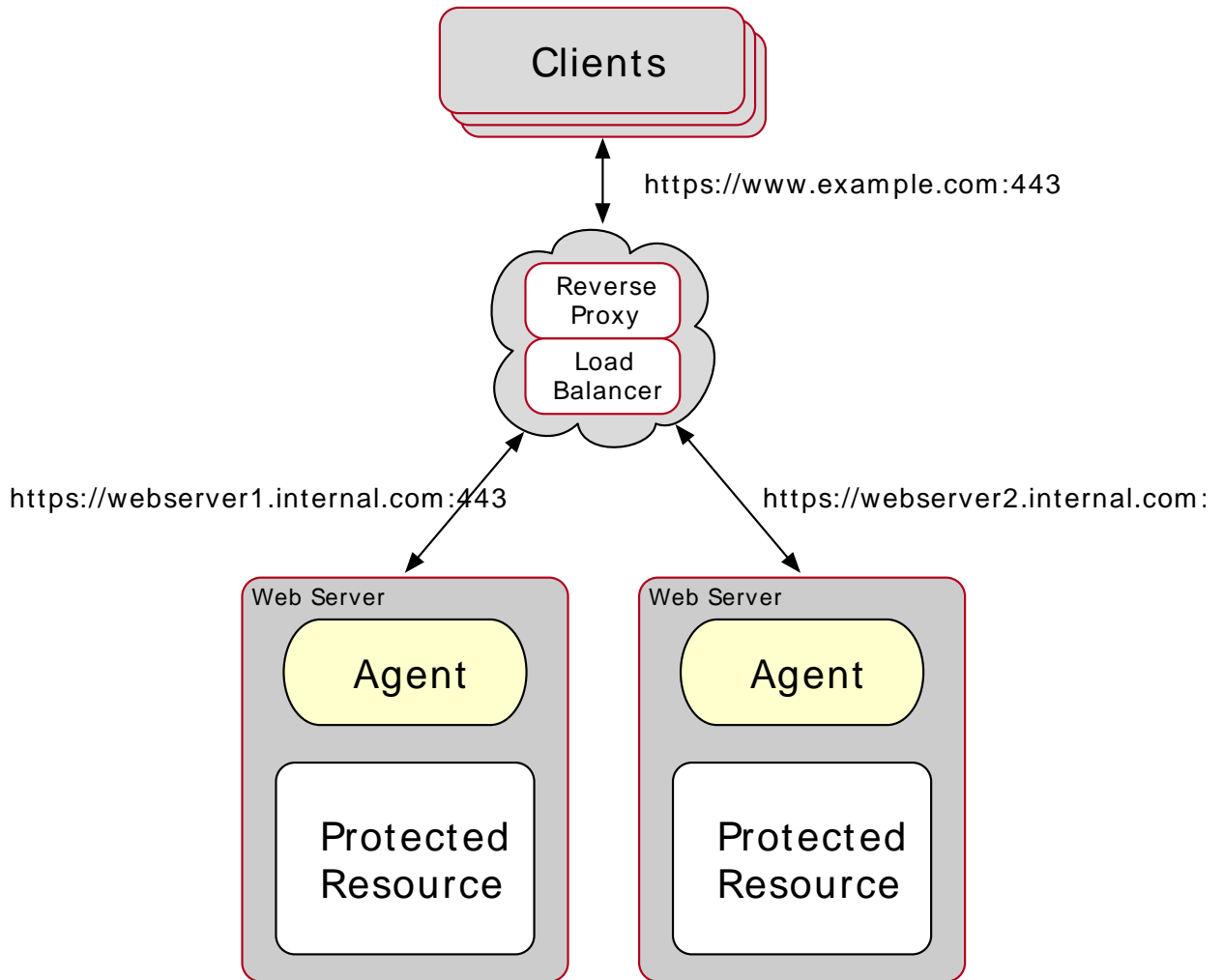
Different Protocol, Port, and FQDN



In this case, configure the web agents following the steps in "To Override Protocol, Host, and Port".

When the protocol and port configured on the load balancer or reverse proxy match those configured on the protected web server, you must map the agent host name to the load balancer or reverse proxy host name. The following diagram illustrates this scenario:

Same Protocol and Port, Different FQDN



In this case, configure the web agents following the steps in "To Map the Agent Host Name to the Load Balancer or Reverse Proxy Host Name".

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 or reverse proxy.

Note

The following headers, when defined on the proxy or load-balancer, override the value of the `com.sun.identity.agents.config.agenturi.prefix` property:

- `X-Forwarded-Proto`
- `X-Forwarded-Host`
- `X-Forwarded-Port`

Do not configure the agent to override its hostname, port, or protocol, if you are already using these headers.

The web 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 web agent is accessed directly and not through the load balancer or proxy. 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 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 web agent profile.
2. Navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name*.
3. In the Global tab, set the Agent Deployment URI Prefix (`com.sun.identity.agents.config.agenturi.prefix`) to that of the load balancer or proxy. For example, `https://external.example.com:443`.

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

4. In the Advanced tab, perform the following steps:

- a. Enable Override Request URL Protocol.

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

- b. Enable Override Request URL Host.

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

- c. Enable Override Request URL Port.

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

Tip

Depending on your configuration, you may only need to override a combination of protocol, hostname, or the port. In these cases, set the Agent Deployment URI Prefix property appropriately, and enable the required override properties only.

5. Save your work.
6. Restart the web server where the agent is installed.

To Map the Agent Host Name to the Load Balancer or Reverse Proxy 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 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 web agent profile.
2. Navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name*.
3. In the Global tab, enable FQDN check.

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

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

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

5. Append the FQDN of the load balancer or proxy to the Agent Root URL for CDSSO field.

The equivalent property setting is `sunIdentityServerDeviceKeyValue[n]=lb.example.com`.

6. Map the load balancer or proxy FQDN to the FQDN where the web agent is installed in the FQDN Virtual Host Map key-pair map. For example, set the key `agent.example.com` (protected server) and a value `lb.example.com` (load balancer or proxy).

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

7. Save your work.
8. Restart the web server where the agent is installed.

Configuring Client Identification Properties

After configuring your proxies or load balancers to forward the client's FQDN and/or IP address, configure the web agents to check the appropriate headers.

To Configure the Web Agent Client Identification Properties

This procedure explains how to configure the client identification properties for a centralized web 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 with a user that has permissions to modify the web agent profile.
2. Navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced.
3. (Optional) In the Client IP Address Header field, configure the name of the header containing the IP address of the client. For example, *X-Forwarded-For*.

The equivalent property setting is `com.sun.identity.agents.config.client.ip.header=X-Forwarded-For`.

Configure this property if your AM policies are IP address-based, you configured the agent for not-enforced IP rules, or if you configured the agent to take any decision based on the client's IP address.

4. (Optional) In the Client Hostname Header field, configure the name of the header containing the FQDN of the client. For example, *X-Forwarded-Host*.

The equivalent property setting is `com.sun.identity.agents.config.client.hostname.header=X-Forwarded-Host`.

Configure this property if your AM policies are URL-based, you configured the agent for not-enforced URL rules, or if you configured the agent to take any decision based on the client's URL.

5. Save your changes.

Configuring POST Data Preservation for Load Balancers or Reverse Proxies

When configuring POST data preservation behind a load balancer or a reverse proxy, you must configure both your load balancer/reverse proxy and the web agents for session stickiness.

To Configure POST Data Preservation Stickiness Properties

1. Log in to the AM console with a user that has permissions to modify the web agent profile.
2. Navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced.
3. Decide whether the web agent should create a cookie or append a string to the URL to assist with sticky load balancing.

In the Advanced field, configure the `com.sun.identity.agents.config.postdata.preserve.stickysession.mode` property with one of the following options:

- **COOKIE.** The web agent will create a cookie for POST data preservation session stickiness. The contents of the cookie is configured in the next step.
 - **URL.** The web agent will append to the URL a string specified in the next step.
4. In the Advanced field, configure a key-pair value separated by the = character as the value of the `com.sun.identity.agents.config.postdata.preserve.stickysession.value` property.

For example, specifying `lb=myserver` either sets a cookie called `lb` with `myserver` as a value, or appends `lb=myserver` to the URL query string.
 5. (Optional) When using cookies, in the Advanced field, configure the name of the sticky cookie as the value of the `com.sun.identity.agents.config.postdata.preserve.lbcookie` property. For example, `lb`.
 6. Save your changes.
 7. Configure your load balancer or reverse proxy to ensure session stickiness when the cookie or URL query parameter are present.

Chapter 4

Installing Web Agents

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

When installing web agents consider that a single web agent installation can hold multiple web agent instances. As installing more than one web agent in a web server is not supported, install only one web agent per web server and configure as many agent instances as you require.

The following table contains a list of sections containing information about installing web agents on supported platforms:

Task	Section
Install web agents on Apache HTTP Server or IBM HTTP Server	Section
Install web agents on Microsoft Internet Information Services (IIS)	Section
Install web agents on NGINX Plus	Section

Installing the Apache Web Agent

This section covers prerequisites and installation procedures for Web Agents 5.8.2.1 on Apache HTTP Servers and IBM HTTP Servers.

The examples on this chapter use Apache and the Apache HTTP Server agent path. For IBM HTTP Servers, replace the Apache HTTP Server agent path, `apache_24_agent`, with the IBM HTTP agent path, `httpserver7_agent`.

Before You Install

1. Ensure you have completed the tasks in "*Preparing for Installation*".
2. Consider the following points before installing web agents on Apache:
 - The web agent replaces authentication functionality provided by Apache, for example, the `mod_auth_*` modules. Integration with built-in Apache 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 web agent.

Tuning Apache Multi-Processing Modules

The Apache HTTP Server and the IBM HTTP Server include 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.

You must configure and tune the MPMs before installing the Apache Web Agent, as follows:

- Configure either the `mpm-event` or the `mpm-worker` modules for Unix-based servers, or the `mpm_winnt` module for Windows servers.

The `prefork-mpm` module may cause performance issues to both the agent and AM.

- Ensure that there are enough processes and threads available to service the expected number of client requests.

MPM-related performance is configured in the `conf/extra/http-mpm.conf` file. The key properties in this file are `ThreadsPerChild` and `MaxClients`. Together, these 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>
```

For the web 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.

Installing the Apache Web Agent

Complete the following procedures to install Web Agent 5.8.2.1 on Apache:

To Complete Pre-Installation Tasks

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

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Configuring Policies*.
2. Configure AM to protect the cross-domain single sign-on (CDSSO) cookie from hijacking. For more information, see *Enabling Restricted Tokens for CDSSO Session Cookies in the ForgeRock Access Management Security Guide*.
3. 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

Note

Windows 2008 R2 users must not create the password file using PowerShell.

Unix example:

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

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

4. (Optional) If AM is configured to perform client authentication, or if the container where the agent will be installed is configured to validate AM's server certificate, set up the required environment variables before installing the agent.

For more information, see *Web Agent Installer Environment Variables*.

To Install the Apache Web Agent

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 the web agent installation at anytime by pressing **CTRL+C**

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

```
Enter the complete path to the httpd.conf file which is used by Apache HTTP
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 web 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 server by viewing the **Group** and **User** directives in the Apache 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 server not starting up, getting a blank page when accessing a protected resource, or the web agent generating errors during log file rotation.

- c. 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:
```

- d. Enter the full URL of the AM instance the web agents will be using. Ensure that 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 "Configuring Apache HTTP Server as a Reverse Proxy Example".

```
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. Realms are case-sensitive.

```
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 configuration file. You can find a backup configuration file in the Apache configuration directory, called [http.conf_agent_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 agent requires to connect to AM and download its configuration. Also contains properties that are only used if you configure the web 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 agent issues.

6. (Unix only) Configure whether the Apache agent instance should share runtime resources and shared memory, or not.

For more information, see "Configuring Whether Unix Web Agents Should Share Runtime Resources and Shared Memory".

7. (Unix only) 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`

Apply execute permissions on the folders listed above, recursively, for the user that runs the Apache HTTP server.

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 web 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](#)".

8. Run the configuration validator for the new agent instance you just created.

The validator will ensure, among other things, that WebSocket communication between your web server and AM is possible.

Perform the following steps to find the agent instance and run the `agentadmin` command:

- a. Change directories to the location where your web agent instances are installed. For example, `/path/to/web_agents/agent_name/instances`.
- b. Find the agent instance you just created, for example, `agent_2`.
- c. Run the `agentadmin --Vi` command. On Unix systems, ensure that you run the command as the user running the web server processes.

Windows Example:

```
C:\web_agents\iis_agent\bin> agentadmin --Vi ^
agent_2 am_user C:\path\to\am_user_password_file /
```

Unix Example:

```
$ sudo -u daemon /path/to/web_agents/agent_name/bin/agentadmin --Vi \
agent_2 am_user /path/to/am_user_password_file /
```

```
Running configuration validation for agent_2:
```

```
Agent instance is configured with 1 naming.url value(s):
1. https://openam.example.com:8443/openam is valid
selected https://openam.example.com:8443/openam as naming.url value
validate_bootstrap_configuration: ok
validate_ssl_libraries: ok
validate_agent_login: ok
get_allocator_blockspace_sz(): trying for configured cache size 16777216 bytes
validate_system_resources: ok
validate_session_profile: ok
validate_websocket_connection: ok
validate_worker_init_shutdown: ok
```

```
Result: 7 out of 7 tests passed, 0 skipped.
```

Do not use the `--vi` option to check the instance configuration while the agent is actively protecting a website, as the agent instance may become unresponsive. Instead, use the `--v` option only. For more information about the `--vi` option, see "Command-Line Tool Reference".

If `validate_websocket_connection` is not ok, ensure your web server and the network infrastructure between the web server and the AM servers support WebSockets.

Web agents require WebSocket communication.

9. Start the Apache server.

To Check the Apache Web 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/5.8.2.1 configured
-- resuming normal operations
```

2. Make an HTTP request to a resource protected by the agent, then check the `/web_agents/apache24_agent/log/system_0.log` file to verify that no errors occurred on startup. Expected output should resemble the following:

```
22019-03-26 08:17:44 GMT INFO
[0x7fb89e7a6700:22]: OpenAM Web Agent Version: 5.8.2.1
Revision: ab12cde, Container: Apache 2.4 Linux 64bit (Centos6),
Build date: Mar 21 2019 13:43:12
```

3. (Optional) If you have a policy configured, you can test that your web 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.

Installing Apache Web Agents on a Virtual Host

Complete the following procedures to install Web Agent 5.8.2.1 on Apache virtual hosts.

Installing on 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.

To Prepare for Web Agent Installation on an Apache Virtual Host

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

1. Install a web agent in the default root configuration of the Apache installation. For more information, see "Installing the Apache Web Agent"
2. Create an agent profile in AM for the web 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 *Configuring Policies*.

To Install the Apache Web Agent on Apache Virtual Hosts

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

- 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.
- Configure the virtual host's web 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 profile¹.

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 profile¹.

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: YWM00ThlMTQtMzMxOS05Nw==
```

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:

```
$ ./agentadmin --p "YWM00ThlMTQtMzMxOS05Nw==" "`cat newpassword.file`"
Encrypted password value: 07bJ0SeM/G8yd04=
```

Windows example:

```
$ agentadmin.exe --p "YWM00ThlMTQtMzMxOS05Nw==" "newpassword"
Encrypted password value: 07bJ0SeM/G8yd04=
```

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

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

```
com.sun.identity.agents.config.key = YWM00ThLMTQtMzMx0S05Nw==
```

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

```
com.sun.identity.agents.config.password = 07bJ0SeM/G8yd04=
```

- 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`

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

- Save and close the configuration file.

- Edit the Apache configuration file. This is the same file specified when installing the web agent on the default Apache website. For example, `/etc/httpd/conf/httpd.conf`.

- 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 on 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 on 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 on the `VirtualHost` element within that file.

7. Save and close the Apache configuration file.
8. (Unix only) Configure whether the Apache agent instance should share runtime resources and shared memory, or not.

For more information, see "Configuring Whether Unix Web Agents Should Share Runtime Resources and Shared Memory".

9. (Unix only) 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`

Apply execute permissions on the folders listed above, recursively, for the user that runs the Apache HTTP server.

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 web 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](#)".

10. Run the configuration validator for the new agent instance you just created.

The validator will ensure, among other things, that WebSocket communication between your web server and AM is possible.

Perform the following steps to find the agent instance and run the **agentadmin** command:

- Change directories to the location where your web agent instances are installed. For example, `/path/to/web_agents/agent_name/instances`.
- Find the agent instance you just created, for example, `agent_2`.
- Run the **agentadmin --Vi** command. On Unix systems, ensure that you run the command as the user running the web server processes.

Windows Example:

```
C:\web_agents\iis_agent\bin> agentadmin --Vi ^
agent_2 am_user C:\path\to\am_user_password_file /
```

Unix Example:

```
$ sudo -u daemon /path/to/web_agents/agent_name/bin/agentadmin --Vi \
agent_2 am_user /path/to/am_user_password_file /
```

Running configuration validation for agent_2:

```
Agent instance is configured with 1 naming.url value(s):
1. https://openam.example.com:8443/openam is valid
selected https://openam.example.com:8443/openam as naming.url value
validate_bootstrap_configuration: ok
validate_ssl_libraries: ok
validate_agent_login: ok
get_allocator_blockspace_sz(): trying for configured cache size 16777216 bytes
validate_system_resources: ok
validate_session_profile: ok
validate_websocket_connection: ok
validate_worker_init_shutdown: ok
```

Result: 7 out of 7 tests passed, 0 skipped.

Do not use the **--Vi** option to check the instance configuration while the agent is actively protecting a website, as the agent instance may become unresponsive. Instead, use the **--V** option only. For more information about the **--Vi** option, see "[Command-Line Tool Reference](#)".

If `validate_websocket_connection` is **not ok**, ensure your web server and the network infrastructure between the web server and the AM servers support WebSockets.

Web agents require WebSocket communication.

11. Start the Apache server.

To Check the Apache Web 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/5.8.2.1 configured  
-- resuming normal operations
```

2. Make an HTTP request to a resource protected by the agent, then check the `/web_agents/apache24_agent/log/system_0.log` file to verify that no errors occurred on startup. Expected output should resemble the following:

```
22019-03-26 08:17:44 GMT INFO  
[0x7fb89e7a6700:22]: OpenAM Web Agent Version: 5.8.2.1  
Revision: ab12cde, Container: Apache 2.4 Linux 64bit (Centos6),  
Build date: Mar 21 2019 13:43:12
```

3. (Optional) If you have a policy configured, you can test that your web 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.

Installing the Apache Web Agent 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 configuration file. The installer modifies this file to include the web agent configuration and module.

OpenAM URL

Enter the full URL of the AM instance the web 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 "Configuring Apache HTTP Server as a Reverse Proxy Example".

Agent URL

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

Realm

Enter the AM realm containing the agent profile. Realms are case-sensitive.

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

On Unix systems, use this option to set the ownership of created directories to the user and group as specified in the Apache configuration file.

Note that this option will only change the ownership of the files and directories if you run the **agentadmin** command as the **root** user or using the **sudo** command.

If you cannot run the **agentadmin** as the **root** user or using the **sudo** command, you must change the ownership manually. Refer to the next steps for more information.

--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 agent silently on Apache:

To Complete Pre-Installation Tasks

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

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Configuring Policies*.
2. Configure AM to protect the cross-domain single sign-on (CDSSO) cookie from hijacking. For more information, see [Enabling Restricted Tokens for CDSSO Session Cookies in the ForgeRock Access Management Security Guide](#).
3. 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

Note

Windows 2008 R2 users must not create the password file using PowerShell.

Unix example:

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

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

4. (Optional) If AM is configured to perform client authentication, or if the container where the agent will be installed is configured to validate AM's server certificate, set up the required environment variables before installing the agent.

For more information, see [Web Agent Installer Environment Variables](#).

To install the Apache Web Agent 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:

```
$ sudo 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. (Unix only) Configure whether the Apache agent instance should share runtime resources and shared memory, or not.

For more information, see "Configuring Whether Unix Web Agents Should Share Runtime Resources and Shared Memory".

6. (Unix only) 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`

Apply execute permissions on the folders listed above, recursively, for the user that runs the Apache HTTP server.

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 web 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. Run the configuration validator for the new agent instance you just created.

The validator will ensure, among other things, that WebSocket communication between your web server and AM is possible.

Perform the following steps to find the agent instance and run the **agentadmin** command:

- a. Change directories to the location where your web agent instances are installed. For example, `/path/to/web_agents/agent_name/instances`.
- b. Find the agent instance you just created, for example, `agent_2`.
- c. Run the **agentadmin --Vi** command. On Unix systems, ensure that you run the command as the user running the web server processes.

Windows Example:

```
C:\web_agents\iis_agent\bin> agentadmin --Vi ^
agent_2 am_user C:\path\to\am_user_password_file /
```

Unix Example:

```
$ sudo -u daemon /path/to/web_agents/agent_name/bin/agentadmin --Vi \
agent_2 am_user /path/to/am_user_password_file /
```

Running configuration validation for agent_2:

```
Agent instance is configured with 1 naming.url value(s):
1. https://openam.example.com:8443/openam is valid
selected https://openam.example.com:8443/openam as naming.url value
validate_bootstrap_configuration: ok
validate_ssl_libraries: ok
validate_agent_login: ok
get_allocator_blockspace_sz(): trying for configured cache size 16777216 bytes
validate_system_resources: ok
validate_session_profile: ok
validate_websocket_connection: ok
validate_worker_init_shutdown: ok
```

Result: 7 out of 7 tests passed, 0 skipped.

Do not use the **--Vi** option to check the instance configuration while the agent is actively protecting a website, as the agent instance may become unresponsive. Instead, use the **--V** option only. For more information about the **--Vi** option, see "[Command-Line Tool Reference](#)".

If `validate_websocket_connection` is **not ok**, ensure your web server and the network infrastructure between the web server and the AM servers support WebSockets.

Web agents require WebSocket communication.

8. Start the Apache server.

To Check the Apache Web 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/5.8.2.1 configured  
-- resuming normal operations
```

2. Make an HTTP request to a resource protected by the agent, then check the `/web_agents/apache24_agent/log/system_0.log` file to verify that no errors occurred on startup. Expected output should resemble the following:

```
22019-03-26 08:17:44 GMT INFO  
[0x7fb89e7a6700:22]: OpenAM Web Agent Version: 5.8.2.1  
Revision: ab12cde, Container: Apache 2.4 Linux 64bit (Centos6),  
Build date: Mar 21 2019 13:43:12
```

3. (Optional) If you have a policy configured, you can test that your web 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.

Installing the IIS Web Agent

This section covers prerequisites and installation procedures for Web Agents 5.8.2.1 on IIS.

Before You Install

1. Ensure you have completed the tasks in "*Preparing for Installation*".
2. Consider the following points before installing web agents on IIS servers:
 - Ensure AM is installed and running, so that you can contact AM from the system running the web agent.
 - Web agents requires IIS to be run in Integrated mode.
 - A web agent configured for a site or a parent application protects any application configured within. The same is true for protected applications containing applications within.

The following restrictions apply to the previous statements:

- Web agents configured in a site or parent application do not protect children applications that do not inherit the parent's IIS configuration.
- Agents configured for a site or parent application running under a 64-bit pool *will not* protect child applications running under 32-bit pools due to architectural differences; 32-bit applications cannot load 64-bit web agent libraries and, therefore, will not be protected.

The same is true for the opposite scenario.

In this case, the child applications require their own web agent installation, as explained in the next item of this list. Both 32-bit and 64-bit agent libraries are supplied with the IIS Web Agent binaries.

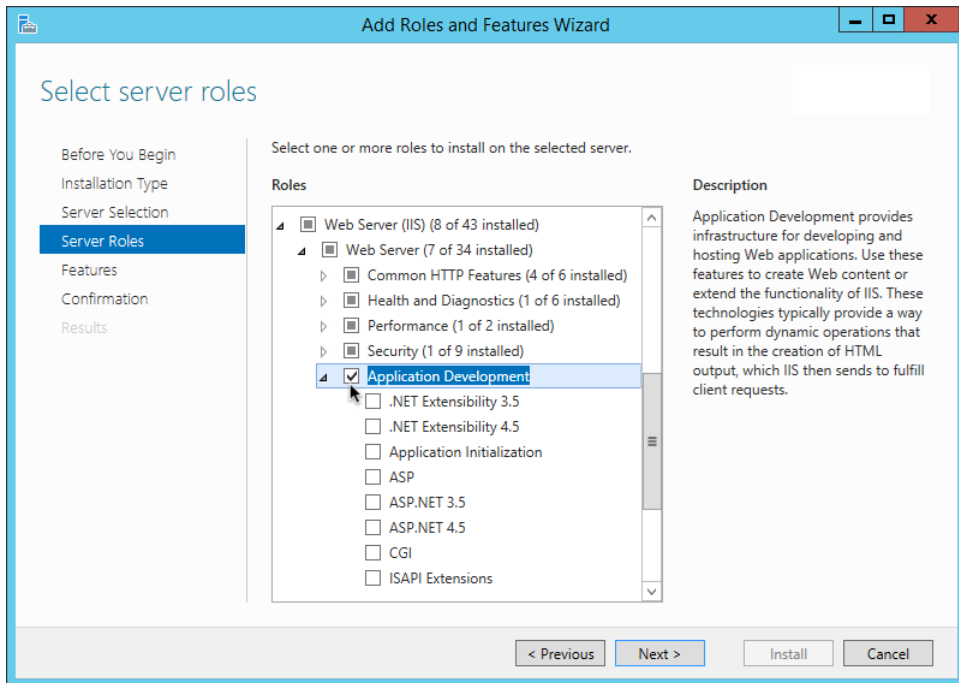
- If an application requires a specific web agent configuration or, for example, the application is a 32-bit application configured within a 64-bit site, follow the procedures in this section to create a new web agent instance for it. Configuring a web agent on an application overrides the application's 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 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



Installing the IIS Web Agent

Complete the following procedures to install Web Agent 5.8.2.1 on IIS servers.

To Complete Pre-Installation Tasks

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

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Configuring Policies*.
2. Configure AM to protect the cross-domain single sign-on (CDSSO) cookie from hijacking. For more information, see [Enabling Restricted Tokens for CDSSO Session Cookies in the ForgeRock Access Management Security Guide](#).
3. 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

Note

Windows 2008 R2 users must not create the password file using PowerShell.

Unix example:

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

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

4. (Optional) If AM is configured to perform client authentication, or if the container where the agent will be installed is configured to validate AM's server certificate, set up the required environment variables before installing the agent.

For more information, see [Web Agent Installer Environment Variables](#).

To Install the IIS Web Agent

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 agent installation at anytime by pressing **CTRL+C**

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

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

```
IIS Server Site configuration:
=====
id      details
=====
        Default Web Site
1.1.1   application path:/, pool DefaultAppPool
        virtualDirectory path:/, configuration: C:\inetpub\wwwroot\web.config

        MySite
2.1.1   application path:/, pool: MySite
        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: 2.1.1
```

- 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 "Configuring Apache HTTP Server as a Reverse Proxy Example".

```
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 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: http://customers.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: iisagent
```

- f. Enter the AM realm containing the agent profile. Realms are case-sensitive.

```
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 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: c:\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 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]: 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 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 agent requires to connect to AM and download its configuration. Also contains properties that are only used if you configure the web 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 agent issues.

6. Ensure the application pool identity related to the IIS site has the appropriate permissions on the following agent installation folders:

- `\web_agents\iis_agent\lib`
- `\web_agents\iis_agent\log`

- `\web_agents\iis_agent\instances\agent_nnn`

To change the ACLs for files and folders related to the agent instance, run the **agentadmin --o** command. For example:

```
C:\web_agents\iis_agent\bin>agentadmin.exe --o "ApplicationPoolIdentity1" "C:\web_agents\iis_agent\lib"
```

For more information about the **agentadmin --o**, see "Command-Line Tool Reference".

Failure to set permissions cause issues, such as getting a blank page when accessing a protected resource, or the web agent generating errors during log file rotation.

7. Run the configuration validator for the new agent instance you just created.

The validator will ensure, among other things, that WebSocket communication between your web server and AM is possible.

Perform the following steps to find the agent instance and run the **agentadmin** command:

- a. Change directories to the location where your web agent instances are installed. For example, `/path/to/web_agents/agent_name/instances`.
- b. Find the agent instance you just created, for example, `agent_2`.
- c. Run the **agentadmin --Vi** command. On Unix systems, ensure that you run the command as the user running the web server processes.

Windows Example:

```
C:\web_agents\iis_agent\bin> agentadmin --Vi ^  
agent_2 am_user C:\path\to\am_user_password_file /
```

Unix Example:

```
$ sudo -u daemon /path/to/web_agents/agent_name/bin/agentadmin --Vi \  
agent_2 am_user /path/to/am_user_password_file /
```

Running configuration validation for agent_2:

```
Agent instance is configured with 1 naming.url value(s):  
1. https://openam.example.com:8443/openam is valid  
selected https://openam.example.com:8443/openam as naming.url value  
validate_bootstrap_configuration: ok  
validate_ssl_libraries: ok  
validate_agent_login: ok  
get_allocator_blockspace_sz(): trying for configured cache size 16777216 bytes  
validate_system_resources: ok  
validate_session_profile: ok  
validate_websocket_connection: ok  
validate_worker_init_shutdown: ok
```

Result: 7 out of 7 tests passed, 0 skipped.

Do not use the `--vi` option to check the instance configuration while the agent is actively protecting a website, as the agent instance may become unresponsive. Instead, use the `--v` option only. For more information about the `--vi` option, see "Command-Line Tool Reference".

If `validate_websocket_connection` is `not ok`, ensure your web server and the network infrastructure between the web server and the AM servers support WebSockets.

Web agents require WebSocket communication.

8. (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.

Installing IIS Web 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 web agent. For a description of the supported IDs, see "To Install the IIS Web Agent".

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 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 "Configuring Apache HTTP Server as a Reverse Proxy Example".

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. Realms are case-sensitive.

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.

--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 agent silently on an IIS server:

To Complete Pre-Installation Tasks

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

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Configuring Policies*.
2. Configure AM to protect the cross-domain single sign-on (CDSSO) cookie from hijacking. For more information, see *Enabling Restricted Tokens for CDSSO Session Cookies in the ForgeRock Access Management Security Guide*.
3. 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

Note

Windows 2008 R2 users must not create the password file using PowerShell.

Unix example:

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

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

4. (Optional) If AM is configured to perform client authentication, or if the container where the agent will be installed is configured to validate AM's server certificate, set up the required environment variables before installing the agent.

For more information, see [Web Agent Installer Environment Variables](#).

To install Web 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> agentadmin.exe --s ^
"2.1.1" ^
"https://openam.example.com:8443/openam" ^
"http://iis.example.com:80" ^
"/" ^
"iisagent" ^
"c:\pwd.txt" ^
--acceptLicence
```

OpenAM Web Agent for IIS Server installation.

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

4. Ensure the application pool identity related to the IIS site has the appropriate permissions on the following agent installation folders:
 - `\web_agents\iis_agent\lib`
 - `\web_agents\iis_agent\log`

- `\web_agents\iis_agent\instances\agent_nnn`

To change the ACLs for files and folders related to the agent instance, run the **agentadmin --o** command. For example:

```
C:\web_agents\iis_agent\bin>agentadmin.exe --o "ApplicationPoolIdentity1" "C:\web_agents\iis_agent\lib"
```

For more information about the **agentadmin --o**, see "Command-Line Tool Reference".

Failure to set permissions cause issues, such as getting a blank page when accessing a protected resource, or the web agent generating errors during log file rotation.

5. Run the configuration validator for the new agent instance you just created.

The validator will ensure, among other things, that WebSocket communication between your web server and AM is possible.

Perform the following steps to find the agent instance and run the **agentadmin** command:

- a. Change directories to the location where your web agent instances are installed. For example, `/path/to/web_agents/agent_name/instances`.
- b. Find the agent instance you just created, for example, `agent_2`.
- c. Run the **agentadmin --Vi** command. On Unix systems, ensure that you run the command as the user running the web server processes.

Windows Example:

```
C:\web_agents\iis_agent\bin> agentadmin --Vi ^  
agent_2 am_user C:\path\to\am_user_password_file /
```

Unix Example:

```
$ sudo -u daemon /path/to/web_agents/agent_name/bin/agentadmin --Vi \  
agent_2 am_user /path/to/am_user_password_file /
```

Running configuration validation for agent_2:

```
Agent instance is configured with 1 naming.url value(s):  
1. https://openam.example.com:8443/openam is valid  
selected https://openam.example.com:8443/openam as naming.url value  
validate_bootstrap_configuration: ok  
validate_ssl_libraries: ok  
validate_agent_login: ok  
get_allocator_blockspace_sz(): trying for configured cache size 16777216 bytes  
validate_system_resources: ok  
validate_session_profile: ok  
validate_websocket_connection: ok  
validate_worker_init_shutdown: ok
```

Result: 7 out of 7 tests passed, 0 skipped.

Do not use the `--vi` option to check the instance configuration while the agent is actively protecting a website, as the agent instance may become unresponsive. Instead, use the `--V` option only. For more information about the `--vi` option, see "Command-Line Tool Reference".

If `validate_websocket_connection` is **not ok**, ensure your web server and the network infrastructure between the web server and the AM servers support WebSockets.

Web agents require WebSocket communication.

- (Optional) If you installed the web agent in a parent application, enable the web agent for its child applications by following the steps in "To Disable And Enable Web Agent Protection for Children Applications".

Enabling and Disabling IIS Web 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 4.14
Enable or disable web agent protection on a child application	Procedure 4.15

To Disable and Enable Web Agents

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".

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

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

- Perform one of the following steps:
 - To disable the web agent in a site, run **agentadmin.exe --d**, and specify the ID of the web agent configuration instance to disable.

```
c:\web_agents\iis_agent\bin> agentadmin.exe --d agent_1

Disabling agent_1 configuration...
Disabling agent_1 configuration... Done.
```

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

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

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

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 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 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.

Enable IIS Basic Authentication and Password Replay Support

The IIS web 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 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 web agent requests IIS server variable values from AM, which gets them from Active Directory. The web 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 web 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 web agent configuration), the web 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 web agent configuration), the web agent adds it in the `USER_PASSWORD` header.

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

The web agent requires that IIS runs in Integrated mode. Consider the following points for integration with additional Microsoft products:

- 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. Use the **openssl** tool to generate a suitable encryption key, as follows:

```
$ openssl rand -base64 32  
e63+9+KQZ0fYu3B2nRsy1tyU3rvqLjMuP9EXKTAxisw=
```

2. In the AM console, navigate to Deployment > Servers > *Server Name* > Advanced, and then add a property `com.sun.am.replaypasswd.key` with the encryption key you generated in a previous step as the value.
3. Navigate to Realms > *Realm Name* > Authentication > Settings > Post Authentication Processing, and in the Authentication Post Processing Classes property, add the class `com.sun.identity.authentication.spi.ReplayPasswd`.
4. Restart AM or the container where it runs.
5. In the AM console navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced

- a. In the Replay Password Key (`com.sun.identity.agents.config.replaypasswd.key`) property, enter the encryption key generated in a previous step.
 - b. (Optional) To have the agent perform Windows logon (for user token impersonation), enable the Logon and Impersonation (`com.sun.identity.agents.config.iis.logonuser`) property.
 - c. Save our changes.
6. (Optional) To set the encrypted password in the IIS `AUTH_PASSWORD` server variable, navigate to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced, and then enable the Show Password in HTTP Header property (property name: `com.sun.identity.agents.config.iis.password.header`).
 7. (Optional) 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 agent profile User ID Parameter and User ID Parameter Type so that the web 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 web 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 web 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.

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

Installing the NGINX Plus Web Agent

This section covers prerequisites and installation procedures for Web Agents 5.8.2.1 into NGINX Plus servers.

The examples in this chapter use the NGINX Plus R17 agent path. For other supported versions, replace the R17 agent path with the required version.

Before You Install

1. Ensure you have completed the tasks in "*Preparing for Installation*".
2. Consider the following points before installing web agents on NGINX Plus servers:
 - Ensure AM is installed and running, so that you can contact AM from the system running the web 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*".

Installing the NGINX Plus Web Agent

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

To Complete Pre-Installation Tasks

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

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Configuring Policies*.
2. Configure AM to protect the cross-domain single sign-on (CDSSO) cookie from hijacking. For more information, see [Enabling Restricted Tokens for CDSSO Session Cookies](#) in the *ForgeRock Access Management Security Guide*.
3. 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

Note

Windows 2008 R2 users must not create the password file using PowerShell.

Unix example:

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

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

4. (Optional) If AM is configured to perform client authentication, or if the container where the agent will be installed is configured to validate AM's server certificate, set up the required environment variables before installing the agent.

For more information, see [Web Agent Installer Environment Variables](#).

To Install the NGINX Plus Web Agent

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/nginx17_agent/bin/  
$ ./agentadmin --i
```

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

Tip

You can cancel the web 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 agent to 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:

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 "Configuring Apache HTTP Server as a Reverse Proxy Example".

```
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. Realms are case-sensitive:

```
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/nginx17_agent/lib/openamngx_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/nginx17_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/nginx17_agent/instances/agent_1/`.

The configuration files and log locations are as follows:

`config/agent.conf`

Contains the bootstrap properties the web agent requires to connect to AM and download its configuration. Also contains properties that are only used if the web 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 web agent issues.

6. Finish the NGINX Plus web agent installation by performing the steps in "To Complete the NGINX Plus Web Agent Installation".

Installing NGINX Plus Web 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 complete the configuration by performing the steps in "To Complete the NGINX Plus Web Agent Installation".

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 web agent configuration and module.

OpenAM URL

Enter the full URL of the AM instance the web agents should connect to. 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 "Configuring Apache HTTP Server as a Reverse Proxy Example".

Agent URL

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

Realm

Enter the AM realm containing the agent profile. Realms are case-sensitive.

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 agent silently on an NGINX Plus server:

To Complete Pre-Installation Tasks

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

1. Create at least one policy in AM to protect resources with the agent, as described in the procedure *Configuring Policies*.
2. Configure AM to protect the cross-domain single sign-on (CDSSO) cookie from hijacking. For more information, see *Enabling Restricted Tokens for CDSSO Session Cookies in the ForgeRock Access Management Security Guide*.
3. 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

Note

Windows 2008 R2 users must not create the password file using PowerShell.

Unix example:

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

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

4. (Optional) If AM is configured to perform client authentication, or if the container where the agent will be installed is configured to validate AM's server certificate, set up the required environment variables before installing the agent.

For more information, see *Web Agent Installer Environment Variables*.

To install the NGINX Plus Web Agent 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/nginx17_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/nginx17_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 NGINX Plus web agent installation by performing the steps in "To Complete the NGINX Plus Web Agent Installation".

Complete the NGINX Plus Web Agent Installation

After you have performed the procedures on either "Installing the NGINX Plus Web Agent" or "Installing NGINX Plus Web Agents Silently", perform the steps in the following procedure to complete the installation of the NGINX Plus web agent:

To Complete the NGINX Plus Web Agent Installation

1. Edit the NGINX Plus server configuration file `nginx.conf` to load the web agent module `openamngx_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/nginx17_agent/lib/openamngx_auth_module.so;
...
```

2. Edit the NGINX Plus server configuration file containing the context you want to protect and add web 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 web agent configuration, follow the procedures in this section again to create a new web agent instance for it. The installer will configure the next available web agent instance, for example, `agent_2`.

- Set the `openam_agent` directive to `off` for a context to disable the web 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/nginx17_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/nginx17_agent/instances/agent_2/config/agent.conf;
        root /www/customers
        index index.html
    }

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

The web 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 web agent instance, `agent_2`.

Example 2

```
server {
    listen      80 default_server;
    server_name localhost;
    openam_agent on;
    openam_agent_configuration /web_agents/nginx17_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 web agent instance `agent_1` is protecting the `server` context and the `/` and `/market` location contexts. Protection is disabled for the `/customers` location context.

3. Configure whether the NGINX Plus agent instance should share runtime resources and shared memory, or not.

For more information, see "Configuring Whether Unix Web Agents Should Share Runtime Resources and Shared Memory".

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

Read Permission

- `/web_agents/nginx17_agent/lib`

Read and Write Permission

- `/web_agents/nginx17_agent/instances/agent_nnn`
- `/web_agents/nginx17_agent/log`

Apply execute permissions on the folders listed above, recursively, for the user that runs the NGINX Plus server.

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 web 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".

5. Run the configuration validator for the new agent instance you just created.

The validator will ensure, among other things, that WebSocket communication between your web server and AM is possible.

Perform the following steps to find the agent instance and run the `agentadmin` command:

- a. Change directories to the location where your web agent instances are installed. For example, `/path/to/web_agents/agent_name/instances`.
- b. Find the agent instance you just created, for example, `agent_2`.

- c. Run the **agentadmin --Vi** command. On Unix systems, ensure that you run the command as the user running the web server processes.

Windows Example:

```
C:\web_agents\iis_agent\bin> agentadmin --Vi ^
agent_2 am_user C:\path\to\am_user_password_file /
```

Unix Example:

```
$ sudo -u daemon /path/to/web_agents/agent_name/bin/agentadmin --Vi \
agent_2 am_user /path/to/am_user_password_file /
```

Running configuration validation for agent_2:

```
Agent instance is configured with 1 naming.url value(s):
1. https://openam.example.com:8443/openam is valid
selected https://openam.example.com:8443/openam as naming.url value
validate_bootstrap_configuration: ok
validate_ssl_libraries: ok
validate_agent_login: ok
get_allocator_blockspace_sz(): trying for configured cache size 16777216 bytes
validate_system_resources: ok
validate_session_profile: ok
validate_websocket_connection: ok
validate_worker_init_shutdown: ok
```

Result: 7 out of 7 tests passed, 0 skipped.

Do not use the **--Vi** option to check the instance configuration while the agent is actively protecting a website, as the agent instance may become unresponsive. Instead, use the **--V** option only. For more information about the **--Vi** option, see "Command-Line Tool Reference".

If **validate_websocket_connection** is **not ok**, ensure your web server and the network infrastructure between the web server and the AM servers support WebSockets.

Web agents require WebSocket communication.

6. 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 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 NGINX Web Agents Installation

1. After you start the server, check the server error log to make sure startup completed successfully:

```
2021... [info] 31#31: agent worker startup complete
```

2. Make an HTTP request to a resource protected by the agent, then check the `/web_agents/nginx23_agent/log/system_0.log` file to make sure no startup errors occurred:

```
OpenAM Web Agent Version: 5.8.2.1  
Revision: ab12cde, Container: NGINX Plus 23 Linux 64bit (Ubuntu20),  
Build date: ...
```

3. (Optional) If you have a policy configured, test that the agent is processing requests. For example, make an HTTP request to a resource protected by the agent and check that you are redirected to AM to authenticate. After authentication, AM redirects you back to the resource you tried to access.

Chapter 5

Post-Installation Tasks

This chapter covers tasks to perform after installing web agents in your environment. The following table contains a list of the tasks:

Task	Section
Configure web agents to log audit messages	Section
Configure whether Unix web agents should share runtime resources and shared memory	Section
Configure your environment when communication between clients and agents happens behind load balancers or reverse proxies	Section
Secure communication between the agents and AM	Section

Configuring Audit Logging

Web agents support logging audit events for security, troubleshooting, and regulatory compliance. You can store web agent audit event logs in the following ways:

- **Remotely.** Log audit events to the audit event handler configured in the AM realm. In a site comprised of several AM servers, web agents write audit logs to the AM server that satisfies the web agent's request for client authentication or resource authorization.

Web agents cannot log audit events remotely if:

- AM's Audit Logging Service is disabled.
- No audit event handler is configured in the realm where the web agent is configured.
- All audit event handlers configured in the realm where the web agent is configured are disabled.

For more information about audit logging in AM, see the chapter *Setting Up Audit Logging* in the *ForgeRock Access Management Security Guide*.

- **Locally.** Log audit events in JSON format to a file in the web agent installation directory, `/web_agents/agent_type/logs/audit/`.
- **Locally and remotely.** Log audit events:

- To a file in the agent installation directory.
- To the audit event handler configured in the AM realm in which the agent profile is configured.

The following is an example of an agent log record:

```
{
  "timestamp": "2017-10-30T11:56:57Z",
  "eventName": "AM-ACCESS-OUTCOME",
  "transactionId": "608831c4-7351-4277-8a5f-b1a83fe2277e",
  "userId": "id=demo,ou=user,dc=openam,dc=forgerock,dc=org",
  "trackingIds": [
    "fd5c8ccf-7d97-49ba-a775-76c3c06eb933-82095",
    "fd5c8ccf-7d97-49ba-a775-76c3c06eb933-82177"
  ],
  "component": "Web Policy Agent",
  "realm": "/",
  "server": {
    "ip": "127.0.0.1",
    "port": 8020
  },
  "request": {
    "protocol": "HTTP/1.1",
    "operation": "GET"
  },
  "http": {
    "request": {
      "secure": false,
      "method": "GET",
      "path": "http://my.example.com:8020/examples/",
      "cookies": {
        "am-auth-jwt": "eyJ0eXAiOiJKV1QiLCJhbGciOi[...]\"",
        "i18next": "en",
        "amlbcookie": "01",
        "iPlanetDirectoryPro": "Ts2zDkGUqgtkoxR[...]"
      }
    }
  },
  "response": {
    "status": "DENIED"
  },
  "_id": "fd5c8ccf-7d97-49ba-a775-76c3c06eb933-81703"
}
```

Note

Local audit logs do not have an `_id` attribute, which is an internal AM id.

The audit log format adheres to the log structure shared across the ForgeRock Identity Platform. For more information about the audit log format, see the section [Audit Log Format](#) in the *ForgeRock Access Management Security Guide*.

Web agents support propagation of the transaction ID across the ForgeRock platform using the HTTP header `X-ForgeRock-TransactionId`. For more information about configuring the header, see [Configuring the Trust Transaction Header System Property](#) in the *ForgeRock Access Management Security Guide*.

By default, web agents do not write audit log records. To configure audit logging, perform the following procedure:

To Configure Audit Logging

The procedure assumes the web agent uses centralized configuration. Property names are also provided for local configuration agents.

1. In the AM console, navigate to Realms > *Realm Name* > Applications > Agents > *Web* > *Agent Name* > Global > Audit.
2. In the Audit Access Type property (`com.sun.identity.agents.config.audit.accesstype`), select the type of messages to log. For example, select `LOG_ALL` to log access allowed and access denied events.
3. In the Audit Log Location property (`com.sun.identity.agents.config.log.disposition`), select whether to write the audit logs locally to the agent installation (`LOCAL`), remotely to AM (`REMOTE`), or to both places (`ALL`). For example, keep `REMOTE` to log audit events to the AM instances.
4. (Optional) In the Local Audit Log Rotation Size property (`com.sun.identity.agents.config.local.log.size`), specify the maximum size, in bytes, of the audit log files.

This is a bootstrap property. If you change the value of this property, restart the web server where the agent runs for the changes to take effect.

Configuring Whether Unix Web Agents Should Share Runtime Resources and Shared Memory

By default, the Unix Apache and NGINX Plus agent shared memory, runtime resources, and installation files are shared among the agent instances. For example, the `Agent_1` and `Agent_2` instances write the same session log and audit files (even though each one writes to their own file), use the same agent policy cache, and run a single set of worker processes and background tasks.

You can control how to share the resources, if at all, by configuring *agent groups*, which are groups of agent instances that share runtime resources and shared memory.

Choosing Whether to Share Resources

You can choose to configure several related agent instances to share resources, and configure others to be independent.

Despite sharing resources, agent instances can be started and stopped individually and can run as different users as long as the agent resources can be shared by their effective user and groups.

Choosing whether to share runtime resources and shared memory is an important decision that depends on your environment. Consider the information in the following table before configuring your agents:

Impact of Sharing Resources

Impact	Advantage	Caution
Shared agent policy and session cache	Potentially reduces overhead of requests to AM for authentication and authorization.	Cache may fill with irrelevant entries.
	Reduced memory consumption.	Sharing the cache among different locations or virtual hosts may not be desirable.
	-	Agent instances that are members of the same agent group must be configured in the same Apache or NGINX Plus installation.
Reduced number of background threads. (Single WebSocket connection to AM for notifications)	Reduced system resource usage.	Ensure the <code>AM_MAX_AGENTS</code> environment variable is set to, at least, the total number of agent instances in the installation.
Agent instances share runtime files and semaphores	Reduced system resource usage.	Must ensure files and resources can be accessed by all the agent instances ^a .

^a For example, add the users running the instances to the same group and configure the resources to have `660` permissions. For more information, see [AM_RESOURCE_PERMISSIONS](#).

Configuring Agent Groups

An *agent group* is a group of agent instances that share runtime resources and shared memory. They are defined by adding the `AmAgentID` (Apache agent only) and the `openam_agent_instance` (NGINX Plus agent only) directives to the Apache and NGINX Plus configuration files.

When configuring the agent groups and the directives, take into account the following constraints:

- Neither the Apache or the NGINX Plus agent set the directives during installation.
- The `AmAgentID` directive defaults to `0` and the `openam_agent_instance` directive defaults to `1` when unset.
- The value of the directives must increase by one for each agent group configured. For example, since the default value of the `AmAgentID` directive is `0`, the next agent group must be `1`.
- Agent instances that are members of the same agent group must be part of the same Apache or NGINX Plus installation.
- By default, the maximum number of agent instances in a single installation is `32`. For more information about changing this limit, see [AM_MAX_AGENTS](#).

The following table shows an example of six agent Apache instances split into three different agent groups:

Apache Agent Groups Example

Agent Instances	Directive Configuration	Description
Agent_1 and Agent_2	Not Set (defaults to 0)	The instances share runtime resources and policy cache.
Agent_3, Agent_4, and Agent_5	1	The instances share runtime resources and policy cache.
Agent_6	2	The instance does not share runtime resources and policy cache with any other instance.

To configure the agent group, set the `AmAgentID` or `openam_agent_instance` directives and their value along with the rest of the agent directives in the `httpd.conf` or `nginx.conf` files:

AmAgentID (Apache only)

The following is an example of a `httpd.conf` file with the `AmAgentID` directive configured:

```
<VirtualHost *:80>
ServerName www.site1.com
DocumentRoot /home/www/site1.com
AssignUserID site1 www-data
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
AmAgentID 1
...
</VirtualHost>

<VirtualHost *:8080>
ServerName www.site2.com
DocumentRoot /home/www/site3.com
AssignUserID site2 www-data
LoadModule amagent_module /web_agents/apache24_agent/lib/mod_openam.so
AmAgent On
AmAgentConf /web_agents/apache24_agent/bin/./instances/agent_2/config/agent.conf
AmAgentID 1
...
</VirtualHost>
```

In this example, each virtual host is protected by a different instance of the agent, yet both agent instances belong to the agent group 1. They share runtime resources and shared memory.

openam_agent_instance (NGINX Plus only)

The following is an example of the `nginx.conf` file with the `openam_agent_instance` directive configured:

```
server {
listen      80 default_server;
server_name localhost;
openam_agent on;
openam_agent_configuration /web_agents/nginx17_agent/bin/../../instances/agent_3/config/agent.conf;
openam_agent_instance 2
...
    location /customers {
openam_agent on;
openam_agent_configuration /web_agents/nginx17_agent/bin/../../instances/agent_4/config/agent.conf;
openam_agent_instance 2
root      /www/customers
index    index.html
    }
}
```

In this example, `agent_1` protects the server context while `agent_2` protects a location. Both instances belong to the agent group `1` and share runtime resources and shared memory.

Supporting Load Balancers and Reverse Proxies Between Clients and Agents

When your environment has reverse proxies or load balancers configured between the agents and the clients, you must perform additional configuration in the agents to account for the anonymization of both the clients and the agents.

Failure to do so may cause policy evaluation and other agent features to fail.

For more information, see "*Configuring Environments With Load Balancers and Reverse Proxies*".

Configuring Web Agents to Secure Communication with AM

Your environment may require that the WebSocket communication between AM and the agents happens over SSL. You can configure the agent to validate server certificates (installed in the container where AM runs), or to present a client certificate to AM, or both.

To facilitate integration and testing, web agents are configured by default to trust any server certificate. Test client certificates are not provided or configured.

To send cookies only when the communication channel is secure, configure the property `com.sun.identity.agents.config.cookie.secure` as `true`. For information, see [Cookie Properties](#).

Web agents support OpenSSL and the Windows Secure Channel API to secure communication:

- "Securing Internal Communication with OpenSSL"
- "Securing Communication with the Windows Secure Channel API"

Securing Internal Communication with OpenSSL

Unix-based agents only support OpenSSL libraries. Windows-based agents can choose whether to use OpenSSL or the Windows Secure Channel API.

For more information about the versions of OpenSSL supported and where to locate related libraries, see "Preparing your Environment for Secure Communication Between the Agents and AM".

To Configure Server-Side and Client-Side Validation using OpenSSL

Perform the following steps to configure the agent to validate AM's server certificate chain and to present client certificates if requested:

1. Open the `/web_agents/agent_type/instances/Agent_nnn/config/agent.conf` configuration file.
2. Configure the agent to use OpenSSL:
 - If you are configuring the IIS or the Apache for Windows web agents, set the `org.forgerock.agents.config.secure.channel.disable_bootstrap` property to `true`.

Ensure that the OpenSSL libraries are in the appropriate place, as specified in the "OpenSSL Library Location by Operating System" table.
 - If you are using Unix-based agents, continue to the next step.
3. (Optional) To configure the agent to validate AM's server certificate, perform the following steps:
 - a. Create a Privacy-Enhanced Mail (PEM) file that contains the certificates required to validate AM's server certificate. For example, `ca.pem`.
 - b. Set the `com.sun.identity.agents.config.trust.server.certs` bootstrap property to `false`.
 - c. Set the `com.forgerock.agents.config.cert.ca.file` bootstrap property to the PEM file previously created. For example:

Unix

```
com.forgerock.agents.config.cert.ca.file = /opt/certificates/ca.pem
```

Windows

```
com.forgerock.agents.config.cert.ca.file = C:\Certificates\ca.pem
```

- d. Set the `org.forgerock.agents.config.cert.verify.depth` bootstrap property to the level of certificate validation required in your environment. For more information, see [OpenSSL Certificate Verification Depth](#).
4. (Optional) To configure the agent to present its client certificate when AM is configured to perform client authentication, perform the following steps:
 - a. Create a PEM file that contains the certificate chain for the agent. For example, `client-cert.pem`.

- b. Create a PEM file that contains the private key corresponding to the certificate. For example, `client-private-key.pem`.
- c. Set the `com.forgerock.agents.config.cert.file` property to the file containing the certificate chain. For example:

Unix

```
com.forgerock.agents.config.cert.file = /opt/certificates/client-cert.pem
```

Windows

```
com.forgerock.agents.config.cert.file = C:\Certificates\client-cert.pem
```

- d. Set the `com.forgerock.agents.config.cert.key` bootstrap property to the file containing the client certificate private key. For example:

Unix

```
com.forgerock.agents.config.cert.key = /opt/certificates/client-private-key.pem
```

Windows

```
com.forgerock.agents.config.cert.key = C:\Certificates\client-private-key.pem
```

- e. (Optional) If the private key is password-protected, obfuscate the password by using the `agentadmin --p` command and configure it in the `com.forgerock.agents.config.cert.key.password` bootstrap property. For example:

Unix

```
$ /path/to/web_agents/agent_type/bin/> agentadmin --p "Encryption Key" ``cat
certificate_password.file`"
Encrypted password value: zck+6RKqjtc=
com.forgerock.agents.config.cert.key.password = zck+6RKqjtc=
```

Windows

```
C:\path\to\web_agents\agent_type\bin> agentadmin.exe --p "Encryption_Key"
"Certificate_File_Password"
Encrypted password value: zck+6RKqjtc=
com.forgerock.agents.config.cert.key.password = zck+6RKqjtc=
```

Encryption Key is the value of the `com.sun.identity.agents.config.key` bootstrap property.

5. Review your configuration. It should look similar to the following:

Unix

```
//Server-side
com.sun.identity.agents.config.trust.server.certs = false
com.forgerock.agents.config.cert.ca.file = /opt/certificates/ca.pem
//Client-side
com.forgerock.agents.config.cert.file = /opt/certificates/client-cert.pem
com.forgerock.agents.config.cert.key = /opt/certificates/client-private-key.pem
com.forgerock.agents.config.cert.key.password = zck+6RKqjtc=
```

Windows

```
//General
org.forgerock.agents.config.secure.channel.disable=true
//Server-side
com.sun.identity.agents.config.trust.server.certs = false
com.forgerock.agents.config.cert.ca.file = C:\Certificates\ca.pem
//Client-side
com.forgerock.agents.config.cert.file = C:\Certificates\client-cert.pem
com.forgerock.agents.config.cert.key = C:\Certificates\client-private-key.pem
com.forgerock.agents.config.cert.key.password = zck+6RKqjtc=
```

6. Restart the web agent or the container where it runs.

Securing Communication with the Windows Secure Channel API

The IIS and Apache for Windows web agents use the Windows built-in Secure Channel API by default. If you rather use OpenSSL, see "Securing Internal Communication with OpenSSL".

To Configure Server-Side and Client-Side Validation using the Windows built-in Secure Channel API

Perform the following steps to configure the agent to validate AM's certificate chain and to present client certificates if requested:

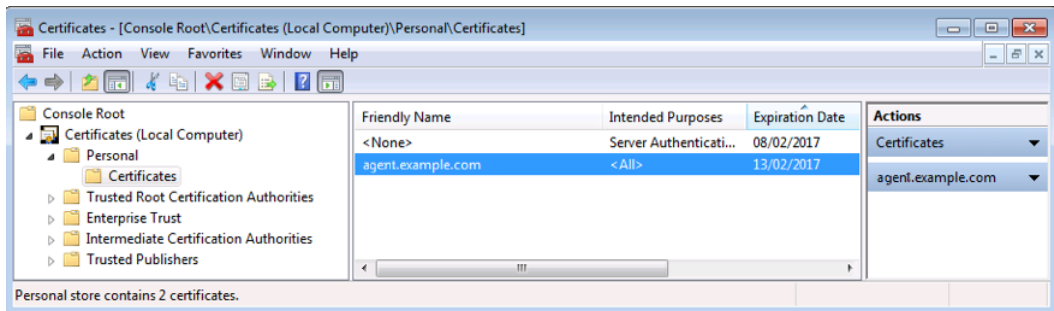
1. Open the `/web_agents/agent_type/instances/Agent_nnn/config/agent.conf` configuration file.
2. Configure the agent to use the Windows built-in Secure Channel API:
 - If this is a new installation, continue to the next step. Windows-based agents use the Windows built-in Secure Channel API by default.
 - If you ever configured the IIS or Apache for Windows web agent to use OpenSSL libraries, set the `org.forgerock.agents.config.secure.channel.disable` bootstrap property to `false`.
3. (Optional) To configure the agent to validate AM certificate chain, perform the following steps:
 - a. Add the certificates required to validate AM's server certificate to the Windows certificate store. For example, if you are using PowerShell, add any root certificates to the `Cert:\LocalMachine\Root` location, and CA certificates to the `Cert:\LocalMachine\Ca` location.
 - b. Set the `com.sun.identity.agents.config.trust.server.certs` bootstrap property to `false`.

4. (Optional) To configure the agent to present client certificates when the container where AM runs is configured to perform client authentication, perform the following steps:
 - a. Import the client certificate chain and private key in the Windows certificate store. For example, if you are using PowerShell, import them to the `Cert:\LocalMachine\My` location.
 - b. Set the `com.forgerock.agents.config.cert` bootstrap property to the friendly name of the client certificate chain.

For example:

```
com.forgerock.agents.config.cert = client-private-friendly-name
```

Friendly Name in the Windows Certificate Store



Note

For compatibility purposes, the agent supports an alternative configuration that does not use the Windows certificate store.

1. Create a Personal Information Exchange (PFX) file containing the certificate chain for the agent and its private key. For example, `client.pfx`.
2. Set the `com.forgerock.agents.config.cert.file` bootstrap property to the previously created PFX file. For example:

```
com.forgerock.agents.config.cert.file = C:\Certificates\client.pfx
```

3. Obfuscate the certificate password by using the `agentadmin --p` command. For example:

```
C:\path\to\web_agents\agent_type\bin> agentadmin.exe --p "Encryption_Key"  
"Certificate_File_Password"  
Encrypted password value: zck+6RKqjtc=
```

`Encryption_Key` is the value of the `com.sun.identity.agents.config.key` bootstrap property.

4. Set the `com.forgerock.agents.config.cert.key.password` bootstrap property to the encrypted password previously created. For example:

```
com.forgerock.agents.config.cert.key.password = zck+6RKqjtc=
```

5. Restart the web agent or the container where it runs.

5. Review your configuration. It should look similar to the following:

Windows Cert Store

```
//Server-side
com.sun.identity.agents.config.trust.server.certs = false
//Client-side
com.forgerock.agents.config.cert = client-private-friendly-name
```

Windows PFX File

```
//Server-side
com.sun.identity.agents.config.trust.server.certs = false
//Client-side
com.forgerock.agents.config.cert.file = C:\Certificates\client.pfx
com.forgerock.agents.config.cert.key.password = zck+6RKqjtc=
```

6. Restart the web agent or the container where it runs.

Chapter 6

Upgrading Web Agents

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

To upgrade web agents, perform the following procedure:

To Upgrade Web Agents

1. Refer to the [Release Notes](#) for information about changes in support and functionality.
2. Back up the web 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 web agent is installed.
5. Remove the old web agent.

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

6. Install the new web agent.

For example, to install the new Apache web agent, see "Installing the Apache Web Agent".

If your policy agent runs in local mode, 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.

7. Review the agent configuration:
 - If the agent configuration is stored in the AM configuration store, review the [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 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.

Important

Ensure the `agent.conf` file contains all required properties. Failure to configure the `agent.conf` properly can result in unexpected agent errors or a crash. For a list of required properties, see "Configuration Location".

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 "YWM00ThLMTQtMzMxOS05Nw==" "`cat newpassword.file`"  
Encrypted password value: 07bJ0SeM/G8yd04=
```

Windows example:

```
$ agentadmin.exe --p "YWM00ThLMTQtMzMxOS05Nw==" "newpassword"  
Encrypted password value: 07bJ0SeM/G8yd04=
```

- d. Set the encrypted password as the value of the `com.sun.identity.agents.config.password` property in the new `agent.conf` file.
9. (NGINX Plus and Unix Apache agents only) Consider whether the agents should share runtime resources and shared memory. For more information, see "Configuring Whether Unix Web Agents Should Share Runtime Resources and Shared Memory".
 10. Ensure the communication between AM and the web agent is secured with the appropriate keys. For more information, see "Configuring AM to Sign Authentication Information".
 11. Start the web server where the web agent is installed.

Note

Web Agents 5 changed the default size of the web agent's session and policy cache from 1 GB to 16 MB. In the unlikely case that an old Apache agent could not release the shared memory, the new Apache agent may not start. For more information, see "*Troubleshooting*".

12. Validate that the web 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.

Tip

You can run the **agentadmin** command with the **--V** option to troubleshoot agent configuration issues in your environment. For more information, see "Command-Line Tool Reference".

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

Chapter 7

Removing Web Agents

The following table contains a list of sections containing information about removing web agents on supported platforms:

Task	Section
Remove web agents on Apache HTTP server or IBM HTTP Server	Section
Remove web agents on Microsoft Internet Information Services (IIS)	Section
Remove web agents on NGINX Plus	Section

Removing the Apache Web Agent

Complete the following steps to remove an Apache agent:

To remove the Apache Web Agent

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

Removing the IIS Web Agent

This section contains a procedure to remove an IIS web agent from a site of an application, which does not remove its libraries, and a procedure to remove the IIS web agent completely from an IIS installation.

To Remove the IIS Web Agent from an IIS Site or Application

Perform the steps in this procedure to remove a single instance of the IIS web agent:

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

```
c:\web_agents\iis_agent\bin> agentadmin.exe --l
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 remove.

3. Run **agentadmin.exe --r**, and specify the ID of the web agent configuration instance to remove.

```
c:\web_agents\iis_agent\bin> agentadmin.exe --r agent_1

Removing agent_1 configuration...
Removing agent_1 configuration... Done.
```

Important

The **--r** option does not remove the web agent libraries. To remove all web agent instances and libraries, see "To Remove Web Agents from IIS".

To Remove Web Agents from IIS

Perform the steps in this procedure to remove all web agents from the IIS installation, for example, during an upgrade.

1. Log on to Windows as a user with administrator privileges.
2. Run **agentadmin --g**. A warning is displayed. Type **yes** to proceed with removing the configuration instance.

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

Warning! This procedure will remove all OpenAM Web Agent references from
IIS Server configuration.

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

Removing agent module from IIS Server configuration...
Removing agent module from IIS Server configuration... Done.
```

Removing the NGINX Plus Web Agent

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

To remove the NGINX Plus Web Agent

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

```
$ ./agentadmin --l

OpenAM Web Agent configuration instances:

id:          agent_1
configuration: /web_agents/nginx17_agent/instances/agent_1
server/site:  /etc/nginx/nginx.conf

id:          agent_2
configuration: /web_agents/nginx17_agent/instances/agent_2
server/site:  /etc/nginx/nginx.conf

id:          agent_3
configuration: /web_agents/nginx17_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 wen 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 web 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 `openamngx_auth_module.so` library.

6. Restart the NGINX Plus server.

Chapter 8

Troubleshooting

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

Tip

The `agentadmin` command offers a validation mode for the agent that can help you troubleshoot issues in your environment; for example, after an agent upgrade or a network change. For more information, see the `--V[!]` option in `agentadmin(1)`.

Solutions to Common Issues

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

Q: I am trying to install a web 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 web 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 web 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: Why are logs not being written to the `/log/system_0.log` and `/log/monitor_0.pipe` files? I am seeing this error:

```
unable to open event channel
```

- A:** It is likely that the agent does not have permission to be able to write to the `/log/system_0.log` and `/log/monitor_0.pipe` log files.

This can occur if you used the `agentadmin --V[i]` validator command using a user account that is different than the account used to run your web server.

You should run the validator command as the same user that runs the web server, for example, by using the `sudo` command.

To fix the issue, change the ownership of these files to match the user or group that is running your web server.

- Q:** My Apache HTTP server is not using port 80. But when I install the web 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 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:** If the web server is running with a non-root user, for example, the `daemon` user, you must ensure that user has the following permissions:

Read Permission

- `/web_agents/agent_name/lib`

Read and Write Permission

- `/web_agents/agent_name/instances/agent_nnn`
- `/web_agents/agent_name/log`

Apply execute permissions on the folders listed above, recursively, for the user that runs the web server.

For IIS web agents, change the ownership of the files using the **agentadmin --o** command. For more information, see "Command-Line Tool Reference".

Tip

You may also see similar issues if SELinux is enabled in **enforcing** mode and it is not configured to allow access to agent directories.

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, *Configuring Success and Failure Redirection URLs*.

Q: I have installed the Unix Apache web agent and now neither Apache nor the web agent start up or log any message. If I remove the web agent, the Apache server starts again. What can be the problem?

A: To troubleshoot a web agent or web server that does not start, set the web agent logging level to the maximum by performing the following steps:

1. Set the environment variable **AM_SYSTEM_LOG_LEVEL** to **All** in your command line session. For example:

```
$ export AM_SYSTEM_LOG_LEVEL=ALL
```

2. Restart the Apache server.
3. Check the logs generated in the **web_agent/apache_24_agent/log/system_n.log** file.

Web agents reserve memory for the policy and session cache based on the **AM_MAX_SESSION_CACHE_SIZE** environment variable. If the server where the web agent is installed does not have enough shared memory available, the web agent may log messages like the following:

```
017-11-10 12:06:00.492 +0000  DEBUG [1:7521][source/shared.c:1451]am_shm_create2() about to create
block-clusters_0, size 1074008064
2017-11-10 12:06:00.492 +0000  ERROR [1:7521]am_shm_create2(): ftruncate failed, error: 28
```

The error message means the web agent tries to reserve 1074008064 bytes of memory, but there is not enough shared memory available. Several reasons may explain why the shared memory is running low, such as:

- A new application or additional workload may be stretching the server resources to the limit.

In this case, ensure that the server has enough shared memory available to satisfy the need of all the applications.

- A web agent may not have been able to release its shared memory after stopping. Therefore, even if the shared memory is technically not in use, it is still reserved and cannot be reassigned unless freed.

Different operating systems manage the shared memory in different ways. Refer to your operating system documentation for information about checking shared memory usage.

You can reduce the amount of memory the web agent reserves for the session and policy cache by setting the `AM_MAX_SESSION_CACHE_SIZE` environment variable to a value between 1048576 (1 MB) and 1074008064 bytes (1 GB). For more information, see "Configuring Web Agent Environment Variables".

Troubleshooting a component that does not start and does not generate logs may be difficult to diagnose. Contact the ForgeRock Support team for more help and information.

- Q:** I have client-based (stateless) sessions configured in AM, and I am getting infinite redirection loops. In the `debug.log` file I can see messages similar to the following:

```
2018-03-15 16:23:10.538 +0000 ERROR [c5319caa-beeb-5a44-a098-d5575e768348]state identifier not
present in authentication state
2018-03-15 16:23:10.538 +0000 WARNING [c5319caa-beeb-5a44-a098-d5575e768348]unable to verify pre-
authentication cookie
2018-03-15 16:23:10.538 +0000 WARNING [c5319caa-beeb-5a44-a098-
d5575e768348]convert_request_after_authn_post(): unable to retrieve pre-authentication request data
2018-03-15 16:23:10.538 +0000 DEBUG [c5319caa-beeb-5a44-a098-d5575e768348] exit status: forbidden
(3), HTTP status: 403, subrequest 0
```

What is happening?

- A:** In this case, the redirection loop happens because the client-based (stateless) session cookie is surpassing the maximum supported browser header size. Since the cookie is incomplete, AM cannot validate it.

To ensure the session cookie does not surpass the browser supported size, configure either signing and compression or encryption and compression.

For more information, see the *ForgeRock Access Management Security Guide*.

- Q:** I have upgraded my agent and, in the logs, I can see errors similar to the following::


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

What should I do?

- A:** Web agents 5.8.2.1 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:** My web agent is not protecting my website. In the logs, I can see errors similar to the following:

```
2018-02-14 13:10:52.816 -0500 ERROR [86169084-5648-6f4d-a706-30f5343d9220]config_fetch(): failed to
load configuration for agent: myagent myagent, error -24
2018-02-14 13:10:52.816 -0500 ERROR [86169084-5648-6f4d-a706-30f5343d9220]amagent_auth_handler():
failed to get agent configuration instance, error: invalid agent session*
```

What is happening?

- A:** The web agent is unable to log in to AM. Possible causes are:

- Network connection between the agent and AM is unavailable.
- The `com.sun.identity.agents.config.naming.url` property, which specifies the URL of AM, may be misconfigured. For more information, see "Bootstrap Properties".

- Q:** I have upgraded my Unix Apache or IBM HTTP Server web agent and even though notifications are enabled, the web agent does not update its configuration. What is happening?

- A:** To troubleshoot this issue, set the web agent logging level to the maximum by performing the following steps:

1. Set the environment variable `AM_SYSTEM_LOG_LEVEL` to `ALL` in your command line session. For example:

```
$ export AM_SYSTEM_LOG_LEVEL=ALL
```

2. Restart the Apache or IBM HTTP server.
3. Check the logs generated in the `web_agent/agent_type/log/system_n.log` file.

Sometimes stopping or upgrading an agent may not clean up the pipe file the agent uses to communicate with AM. If the newly started agent cannot create the pipe to communicate with AM because it already exists, the agent would log messages like the following:

```
2017-12-05 17:12:07.324 UTC   DEBUG [1:10551398][source/monitor.c:503]monitor startup
2017-12-05 17:12:07.325 UTC   ERROR [102:10551398]monitor unable to get semaphore
2017-12-05 17:12:56.552 UTC   DEBUG [304:10551398][source/config.c:295]config_initialise(): agent
configuration read from cache, agent: / wpa-aix7-Httpd7-32bit
```

If you see similar error messages, perform the following steps to delete the pipe file:

1. Stop the Apache or IBM HTTP server.
2. Change directories to the `/tmp` directory.
3. Delete the `monitor.pipe` file.
4. Retart the Apache or IBM HTTP server.

Q: After upgrading, the default Apache welcome page appears 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 Agent Environment Variables](#).

Q: My web agent is not protecting my website. In the `debug.log` file I can see messages similar to the following:

```

2019-06-27 01:54:25 GMT DEBUG [162ba6eb-cf88-3d7f-f92c-ee8b21971b4c]: (source/oidc.c:265) agent_realm
does not have the expected value: JWT
{
  "sub": "demo",
  "auditTrackingId": "267d1f56-0b97-4830-ae91-6be4b8b7099f-5840",
  "iss": "https://openam.example.com:8443/openam/oauth2/Customers",
  "tokenName": "id_token",
  "nonce": "D3AE96656D6D634489AF325D90C435A2",
  "aud": "webagent",
  "s_hash": "rxwIoqDFiwt4MxSwiBa-w",
  "azp": "webagent",
  "auth_time": 1561600459,
  "forgerock": {
    "ssotoken": "wi8tHql..MQAA*",
    "suid": "267d1f56-0b97-4830-ae91-6be4b8b7099f-5647"
  },
  "realm": "/Customers",
  "exp": 1561607661,
  "tokenType": "JWTToken",
  "iat": 1561600461,
  "agent_realm": "/Customers"
}
2019-06-27 01:54:25 GMT WARNING [162ba6eb-cf88-3d7f-f92c-ee8b21971b4c]: redirect_after_authn():
unable to validate JWT

```

What is happening?

- A:** If you configured the agent profile in a realm other than AM's top-level realm (/), you must configure the web agent `com.sun.identity.agents.config.organization.name` bootstrap property with the realm where the agent profile is located. For example, `/Customers`.

Realm names are case-sensitive. Failure to set the realm name exactly as configured in AM will cause the agent to fail to recognize the realm.

- Q:** I am getting HTTP 403 Forbidden messages when accessing protected resources and I can see errors similar to the following in the `debug.log` file:

```

2019-06-27 11:10:04 GMT WARNING [69d4632c-82af-b853-0f340vb7b754]: too many pending authentications
2019-06-27 11:10:04 GMT ERROR [69d4632c-82af-76da-b853-0f340vb7b754]: save_pre_authn_state(): unable
to save state for request

```

What is happening?

- A:** Agents store the progress of authentication with AM in the pre-authentication cookie, `agent-authn-tx`. This cookie, which has a maximum size of 4096 bytes, can fill up if the agent receives a large number of parallel unauthenticated requests to access protected resources.

For more information about how to deal with this issue, see [Multivalue for Pre-Authn Cookie](#).

- Q:** I am getting HTTP 403 Forbidden messages when accessing Web Agents.

- A:** Make sure that the agent is executable:

- In the terminal where the agent is running, go to `/opt/web_agents`.

- Review and, if necessary, change the permissions for the directory:

Chapter 9

Reference

Configuring Web Agent Properties

When you create a web 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 [Custom Properties](#), or locally in the agent properties configuration file, `agent.conf`.

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

Bootstrap Properties

Web Agents use bootstrap properties to start up and connect to AM.

Bootstrap properties are defined exclusively in the local configuration `agent.conf` file, and do not work if set up using the AM console.

Bootstrap properties are not hot-swappable. If you change a bootstrap property, restart the agent or the container where it runs .

Miscellaneous Bootstrap Properties

+ `com.sun.identity.agents.config.organization.name`

The AM realm where the agent profile is located. For example, `/Customers`.

Realm names are case-sensitive. Failure to set the realm name exactly as configured in AM will cause the agent to fail to recognize the realm.

Default: `/`

+ `com.sun.identity.agents.config.username`

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.

The name of the agent profile in AM.

Default: not set

+ `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`.

To encrypt an agent profile password, run the **agentadmin** command with the `--p` option. For an example, see "Command-Line Tool Reference".

Default: not set

+ `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`.

To create an encryption key, run the **agentadmin** command with the `--k` option. For an example, see "Command-Line Tool Reference".

Default: not set

+ `org.forgerock.agents.config.tls`

A space-separated list of security protocols preceded by a dash - that are *not* used when connecting to AM.

The following protocols are supported:

- `SSLv3`
- `TLSv1`
- `TLSv1.1`
- `TLSv1.2` (Enabled)
- `TLSv1.3` (Enabled)

This property is relevant to all web agents using OpenSSL libraries.

Default: `-SSLv3 -TLSv1 -TLSv1.1`

To change the default value, set an environment variable, `AM_SSL_OPTIONS`. For more information, see "Configuring Web Agent Environment Variables".

Note

SSLv2 is always disabled, regardless of setting.

+ `org.forgerock.agents.init.retry.max`

The maximum number of consecutive agent initialization retries.

Default: 0 (when not set)

+ `org.forgerock.agents.init.retry.wait`

The number of seconds to wait between retries.

Default: 0 (when not set)

+ `com.sun.identity.agents.config.connect.timeout`

The number of seconds to wait for a connection to AM before timing out and cancelling the connection. Applies to TCP *connect* operations.

Default: 4

+ `com.sun.identity.agents.config.receive.timeout`

The number of seconds to wait for a response from AM before timing out and dropping the connection. Applies to TCP *receive* operations.

Default: 4

+ `sun.identity.agents.config.naming.url`

A space-separated list of AM URLs to which the agent connects. Set this property to the URL of the load balancer in front of the AM instances (or load balancers, in case of disaster-recovery configurations).

When the web agent cannot connect to the first URL in the list, it automatically connects to the next available URL. The agent stays connected to the new URL until the URL fails, or the agent is restarted.

Default: AM_URL/openam/

+ *org.forgerock.agents.config.connection.pool.enable*

When **true**, the agent uses connection pooling.

Use connection pooling to improve performance when AM is available over low bandwidth connections, or to throttle the maximum number of connections made by the agent.

When AM is available over high bandwidth connections, connection pooling can reduce performance.

Type: Boolean

Default: **true**

Hot-swap: No

Property: `org.forgerock.agents.config.connection.pool.enable`, introduced in Web Agents 5.8

Forward Proxy Bootstrap 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.

Default: not set

+ *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.

Default: not set

+ *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.

Default: not set

+ *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.

Default: not set

Encryption Bootstrap Properties

For more information, see "Configuring Web Agents to Secure Communication with AM".

+ CA Certificate File Name

When the agent is configured to validate server certificates (the `Trust Server Certificates` property is `false`), set this property to the file name that contains a certificate or chain of certificates.

The file should be *Privacy-Enhanced Mail* (PEM) encoded. For example:

```
com.forgerock.agents.config.cert.ca.file = /opt/certificates/openam_ca.pem
com.sun.identity.agents.config.trust.server.certs = false
```

Set this property only when the agent is using OpenSSL libraries. For Web Agents using the Windows built-in Secure Channel API, add the appropriate certificates to the Windows certificate store.

Default: Empty

Property: `com.forgerock.agents.config.cert.ca.file`

+ OpenSSL Certificate Verification Depth

(OpenSSL only) Specifies how deeply the agent verifies AM's server certificate before deciding the certificate is not valid. The depth is the maximum number of CA certificates that will be followed while verifying the server certificate.

If the certificate chain is longer than allowed, the certificates above the limit are ignored.

The property accepts the following values:

- `0`: Only self-signed certificates are accepted.
- `1`: Client certificates can be self-signed or must be signed by a CA which is directly known to the agent container.
- `2` or more: A chain of the specified number of certificates, including the previous ones. For example, the value `5` allows certificates from level `0` to level `5`.

This property is relevant only when server certificate validation is enabled (Trust Server Certificates is `false`).

Default: `9`

Property: `org.forgerock.agents.config.cert.verify.depth`

+ *Public Client Certificate File Name*

When AM is configured to perform client certificate validation, set this property to the name of the file that contains the client certificate chain.

Agents using OpenSSL libraries must specify the certificate chain as a PEM file. For example:

```
com.forgerock.agents.config.cert.file = /opt/certificates/pub_client.pem
```

Agents using the Windows built-in Secure Channel API must choose one of the following options:

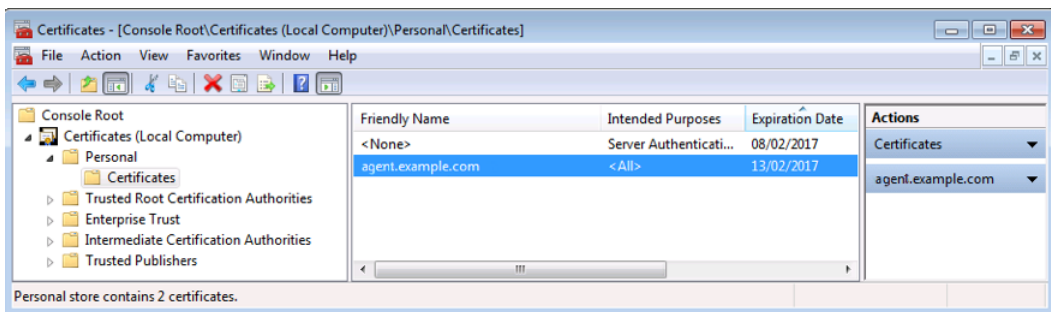
- Store the certificate chain and its private key as a Personal Information Exchange Format (PFX) file, then configure it in the agent property. For example:

```
com.forgerock.agents.config.cert.file = C:\Certificates\client.pfx
```

In this case, you must also configure the `com.forgerock.agents.config.cert.key.password` property.

- Store the certificate locally in the Windows certificate store and configure the friendly name of the client certificate as it shows in Windows, in the agent property:

Friendly Name in the Windows Certificates Snap-in



For example:

```
com.forgerock.agents.config.cert = client-private-friendly-name
```

Default: not set

Property: `com.forgerock.agents.config.cert.file`

+ *Private Client Certificate File Name*

When AM is configured for client-side certificate verification, set this property to the file that contains the client certificate private key.

Web Agents using OpenSSL must specify the private key as a PEM file. For example:

```
com.forgerock.agents.config.cert.key = /opt/certificates/client_key.pem
```

Web Agents using the Windows built-in Secure Channel API should not configure this property.

Default: not set

Property: `com.forgerock.agents.config.cert.key`

+ *Private Key Password*

When AM is configured for client-side certificate verification, and the PEM file containing the client certificate private key is password-protected, set this property to the obfuscated password.

Obfuscate the password by using **agentadmin --p** command, as follows:

Unix

```
$ cd /web_agents/agent-type/bin
$ /web_agents/agent-type/bin> ./agentadmin --p "Encryption Key" "`cat cert_password.file`"
Encrypted password value: zck+6RKqjtc=
```

Windows

```
C:\>cd C:\web_agents\agent-type\bin
C:\path\to\web_agents\bin\> agentadmin.exe --p "Encryption Key" "certpassword"
Encrypted password value: zck+6RKqjtc=
```

Encryption Key is the value of the `com.sun.identity.agents.config.key` bootstrap property.

Default: not set

Property: `com.forgerock.agents.config.cert.key.password`

+ *Trust Server Certificates*

Whether the agent should validate the certificate presented during SSL handshakes by the container where AM runs:

- **true**. The agent trusts any server certificate. By default, and to facilitate integration and testing, agent is configured to trust any server certificate.
- **false**. The agent trusts AM's certificate only if found to be correct and valid.

Important

If the agent cannot connect to AM, it does not allow access to any protected resource. Ensure the agent is properly configured before setting this property to **false**. For more information, see "Configuring Web Agents to Secure Communication with AM".

Default: **true**

Property: `com.sun.identity.agents.config.trust.server.certs`

+ *Use OpenSSL to Secure Internal Communications*

Whether Windows-based agents should use the Windows built-in Secure Channel API or OpenSSL to secure internal communication with AM:

- **true**. The agent uses OpenSSL.
- **false**. The agent uses the Windows built-in Secure Channel API.

Default: **false**

Property: `org.forgerock.agents.config.secure.channel.disable`

+ *Accept Secure Cookies From AM Over HTTP*

For requests that arrive over a secure channel, by default, AM upgrades cookies to secure. However, during internal communication with the agent, AM can send these secure cookies over HTTP.

The property can take the following values:

- **true**: The agent accepts secure cookies from AM over HTTP.
- **false**: The agent rejects secure cookies from AM over HTTP.

Note

It is best practice to use HTTPS for *all* connections to AM.

Default: **false**

Property: `com.forgerock.agents.plain.channels.insecure`

Global Properties

Set the following agent properties in the AM console at:

Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Global.

Profile Global Properties

+ *Group (agentgroup)*

Assign the agent to a previously configured agent group in order to inherit selected properties from the group. Select `Unassigned` to remove the agent from an agent group.

+ *Password*

Web Agents password used when creating the password file and when installing the agent.

If you change the password, manually update the password in the bootstrap property `com.sun.identity.agents.config.password`.

+ *Status*

Status of the agent configuration.

+ *Location of Agent Configuration Repository*

How to manage the agent configuration:

- `centralized`: Through AM.
- `local`: Locally in the agent configuration file.

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

Default: `centralized`

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

+ *Agent Configuration Change Notification*

Whether AM sends a notification to the web agent to reread the agent profile after a change to a hot-swappable property. This property applies only when you store the agent profile in AM's configuration data store.

Default: `true`

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

Hot-swap: yes

+ *Web Socket Connection Interval*

The time in minutes after which web agents reopen their WebSocket connection to AM. This property helps ensure a balanced distribution of connections across the AM servers on the site.

Default: `30`

Property: `org.forgerock.openam.agents.config.balance.websocket.connection.interval.in.minutes`

Hot-swap: yes

+ *JWT Cookie Name*

The name of the cookie that holds the OpenID Connect ID token on the user's browser.

Before changing the name of this cookie, consider the following points:

- The cookie is only used by the web agent and is never presented to AM.
- The cookie name must be unique across the set of cookies the user's browser receives, since some browsers behave in unexpected ways when receiving several cookies with the same name. For example, you should not set the session ID token cookie name to `iPlanetDirectoryPro`, which is the default name of AM's session cookie.

Default: `am-auth-jwt`

Property: `org.forgerock.openam.agents.config.jwt.name`

Hot-swap: yes

+ *Exchange SSO Token for JWT*

This property is marked as deprecated and will be removed in a later version.

Use Accept SSO Token instead.

Property: `com.forgerock.agents.accept.ipdp.cookie`

+ *Disable validation of the audience claim*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

The claims to validate in the ID token containing the end user's session:

- `0`: Validate the `aud` and `nonce` claim.
- `1`: Validate the `nonce` claim; don't validate the `aud` claim.

During an authentication request, AM creates an ID token that contains, among others, the end user's session, and the `aud` claim. The `aud` claim is set to the agent profile of the agent that made the request. When AM returns the ID token to the end user's user-agent, it appends a `nonce` parameter to the request, which is a one-time-usable random string that is understood by both AM and the agent that made the authentication request.

When the agent receives a request to access a protected resource, the agent checks that the audience (the `aud` claim) of the ID token and the value of the `nonce` are appropriate. For example, it checks that the value of the `aud` claim is the name of its own agent profile.

In environments where several agents protect the same application, this validation poses a problem; even if the ID token is valid and contains a valid session, an agent cannot validate a ID token created for a different agent because the audience would not match. Therefore, the agent redirects the end user to authenticate again.

Tip

For security reasons, agents should validate as many claims in the ID token as possible.

Default: `0`

Hot-swap: Yes

Property: `com.forgerock.agents.jwt.aud.disable`

+ *Agent Profile ID Whitelist*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

A comma-separated list of profile IDs that the agent considers as valid values for the `aud` claim. This claim is represented in the ID token containing the end user's session.

When several agents are configured with different agent profiles to protect the same application, set this property to a list of the agent profiles that are protecting the same application.

With the following setting, the Agent considers `agentprofile1` and `agentprofile2` to be valid, and does not validate them:

```
com.forgerock.agents.jwt.aud.whitelist=agentprofile1,agentprofile2
```

Default: Empty

Hot-swap: Yes

Property: `com.forgerock.agents.jwt.aud.whitelist`

+ *Enable Notifications*

Whether AM sends notifications to the agent to:

- Refresh the session cache when a session times out or a client logs out from AM.
- Refresh the policy cache when the administrator changes a policy.

Default: `true`

Hot-swap: No

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

+ *Agent Notification URL (deprecated)*

🔔 *This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.*

When creating a web agent profile, the AM console configures a default value for this property to maintain compatibility with earlier versions of the web agent. The default value should be removed. For more information, see the [Release Notes](#).

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

+ *Agent Deployment URI Prefix*

Overrides the request URL given by the agent when it is configured behind a load balancer or proxy. Use this property when the protocol, hostname, or port of the load balancer or proxy differ from those of the web agent.

For the agent to honor this property, at least one of the following properties must be enabled:

- `Override Request URL Port`
- `Override Request URL Protocol`
- `Override Request URL Host`

Use these properties only if you are not using `x-forwarded` headers from the load balancer or proxy to override the agent's protocol, hostname, and port values.

For more information, see "Regarding Communication Between Clients and Agents".

Default: `agent-root-URL`

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

Hot-swap: yes

+ *Configuration Reload Interval*

Time in minutes after which to fetch the 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 (deprecated)*

⚠ *This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.*

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

+ 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.

Default: `agent-root-URL`

Property: `sunIdentityServerDeviceKeyValue[n]`

General Global Properties

+ SSO Only Mode

When `true`, the agent manages only 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`, the agent manages user authorization, by using the policy engine in AM.

Tip

Consider configuring the Reset Idle Timeout (`com.forgerock.agents.call.session.refresh`) property when configuring the agent in SSO-only mode.

Type: Boolean

Default: `false`

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

+ Reset Idle Timeout

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Specifies whether an agent configured in SSO-only mode should refresh the user's session idle time when the user accesses a protected resource.

Sessions in AM have an idle timeout after which they expire. In general, when users access protected resources through an agent, the agent requests a policy decision on behalf of that user, which resets the idle timeout.

If the agent is configured in SSO-only mode, the session may unexpectedly expire in AM due to idle timeout before the user has finished accessing the application.

Set this property to `true` to refresh the timeout when the user performs an action.

When set to `true`, the agent makes an additional call to AM; this may cause a performance impact. Configure this property only if:

- The agent is configured in SSO-only mode.
- User's sessions are timing out in AM because they are unexpectedly reaching the maximum idle timeout value.

Default: Not set

Property: `com.forgerock.agents.call.session.refresh`

+ *Resources Access Denied URL*

The URL of the customized access denied page. If empty, then the agent returns an HTTP status of 403 (Forbidden). The URL can be absolute or relative.

The following are not permitted:

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

Default: not set

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

+ *Agent Debug Level*

Valid values for the property are:

- All
- Error
- Info
- Message
- Warning

Default: `Error`

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

+ *Agent Debug File Rotation (deprecated)*

⚠ *This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.*

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

Default: `true`

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

+ *Agent Debug File Size*

+ *Not available in the console for AM 7.0.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

File size in bytes beyond which the debug 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).

This is a bootstrap property. Configure it in the `agent.conf` file, even when the agent is configured in centralized mode.

Default: `10000000`

Hot-swap: No

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

+ *Local Agent Debug File Name*

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

This is a bootstrap property. Configure it in the `agent.conf` file, even when the agent is configured in centralized mode.

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

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

Audit Global Properties

+ *Audit Access Types*

The type of audit events to log:

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

Default: `LOG_NONE`

Hot-swap: yes

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

+ *Audit Log Location*

The location where the web agent logs audit messages:

- `REMOTE`. Log audit event messages to the audit event handler configured in the AM realm where the web agent is configured.
- `LOCAL`. Log audit event messages locally to the agent installation.
- `ALL`. Log audit event messages to the audit event handler configured in the AM realm and locally to the agent installation.

Default: `REMOTE`

Hot-swap: yes

Property: `com.sun.identity.agents.config.log.disposition`

+ Remote Log Filename (deprecated)

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

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

+ Remote Audit Log Interval (deprecated)

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

Interval in minutes after which audit log messages are periodically sent to the remote log file.

Default: 5

Hot-swap: no

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

+ Rotate Local Audit Log

+ Not available in the console for AM 7.0.x and earlier versions.

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

This is a bootstrap property. Configure it in the `agent.conf` file, even when the agent is configured in centralized mode.

Hot-swap: No

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

+ Local Audit Log Rotation Size

+ Not available in the console for AM 7.0.x and earlier versions.

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

This is a bootstrap property. Configure it in the `agent.conf` file, even when the agent is configured in centralized mode.

The maximum size in bytes of the local audit log files. Web Agents rotate audit log files when they reach this size.

Default: `52428800`

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

Hot-swap: No

+ *Local Agent Audit File Name*

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

This is a bootstrap property. Configure it in the `agent.conf` file, even when the agent is configured in centralized mode.

Default: `/web_agents/agent_version/instances/agent_nnn/logs/audit/audit.log`

Property: `com.sun.identity.agents.config.local.audit.logfile`

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*

The 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.

Note

If you specify an FQDN in this property, you must also add it to the Agent Root URL for CDSO property.

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 `validn`, 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`

Load Balancing Properties**+ AM Load Balancer Cookie Enabled**

+ *Not available in the console for AM 7.0.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

When `true`, the agent passes the hardcoded `amlbcookie` to AM.

Use this property to improve performance. Load balancer cookies can reduce the number of calls that different AM instances make to the Core Token Service (CTS).

Default: `false`

Type: Boolean

Hot-swap: Yes

Property: `com.forgerock.agents.config.add.amlbcookie`, introduced in Web Agents 5.8

Application Properties

Set the following agent properties in the AM console at:

Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Application

Not-Enforced URL Properties

+ *Ignore Path Info for Not-Enforced URLs*

When `true`, 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, see Ignore Path Info.

Note

The NGINX Plus web agent does not support this setting.

Default: `true`

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

+ *Regular Expressions for Not-Enforced URLs*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

When `true`, enables use of Perl-compatible regular expressions in Not-enforced URL settings.

Default: `false`

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

+ Not-Enforced URLs

List of URLs allowed to bypass authentication and grant immediate access to resources, such as images, stylesheets, or static HTML pages.

Tip

You can also limit not-enforced URL rules to specific HTTP methods. For more information, see "Not-Enforced URL and Client IP Lists".

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=*`.

To match a resource that uses non-ASCII characters, percent-encode the resource when creating the rule.

For example, to match resources under an IRI such as `http://www.example.com/forstå`, specify the following percent-encoded rule:

```
com.sun.identity.agents.config.notenforced.url[n]=/forst%C3%A5/*
```

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.

Default: Not set

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

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 `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 web agent

Default: `false`

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

+ *Fetch Attributes for Not-Enforced URLs*

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

Default: `false`

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

Not-Enforced IP Properties

+ *Not-Enforced Client IP List*

The 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]= 2001:5c0:9168:0:0:0:2/128
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.

Note

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

Default: not set

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

+ Client IP Validation

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

Default: `false`

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

Not-Enforced URL from IP Processing Properties

+ Not-Enforced URL from IP Processing List

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

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[0]=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[1]=192.168.1.1-192.168.1.10|/public/*
org.forgerock.agents.config.notenforced.ipurl[2]=2001:5c0:9168:0:0:0:1-2001:5c0:9168:0:0:0:2|/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[3]=192.168.1.0/24 192.168.100.0/24|/public/*
org.forgerock.agents.config.notenforced.ipurl[4]=2001:5c0:9168:0:0:0:2/128|/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 the following methods:

- **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[5]=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[6]=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.

• Internationalized Resource Identifiers (IRIs)

To match a resource that uses non-ASCII characters, percent-encode the resource when creating the rule.

For example, to match resources under an IRI such as `http://www.example.com/forstå`, specify the following percent-encoded rule:

```
com.sun.identity.agents.config.notenforced.ipurl[7]=192.6.8.0/24|/forst%C3%A5/*
```

Default: not set

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

+ Regular Expressions for Not-Enforced IPs

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

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

Default: `false`

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

Not-Enforced Fallback Mode Properties

+ Not-Enforced Fallback Mode

Specifies whether the web agent should allow traffic to resources specified in the not-enforced lists when AM is not available. The property accepts two values:

- `true`: While AM is unavailable, the web agent:
 1. Reads the cached agent profile configuration until it expires. If you are not familiar with the web agent's caches, see "Caching Capabilities".
 2. After the cache expires, reads the local configuration file `webagents/agent_type/instances/agent_I/config/agent.conf`.

If not-enforced properties are configured in the local configuration file, the web agent allows access to the not-enforced resources. However, response attributes for not-enforced resources are not available until AM is accessible.

- `false`: When AM is unavailable, the web agent prevents access to all resources, including any not-enforced resources.

Configuring this property requires setting up several properties in the the local configuration file `webagents/agent_type/instances/agent_1/config/agent.conf` even if the agent profile is in centralized configuration.

In the `Bootstrap` section, configure the fallback property:

```
com.forgerock.agents.config.fallback.mode = true
```

In the `Configuration` section, configure the not-enforced properties required for your environment. For example:

```
com.sun.identity.agents.config.notenforced.url.attributes.enable = true
com.sun.identity.agents.config.notenforced.url.invert = false
com.sun.identity.agents.config.notenforced.url[0] = http://agenttest.example.com/index.html
```

This is a bootstrap property. Configure it in the `agent.conf` file, even when the agent is configured in centralized mode.

Default: `false`

Property: `com.forgerock.agents.config.fallback.mode`

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.

Default: `NONE`

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, the case of which must exactly match the identity store schema, 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`.

Tip

Make sure the case of your LDAP attribute name matches the case of the LDAP schema, otherwise you may see an error similar to the following:

```
do_header_set(): SM_LOGIN (UiD) is not available in profile attributes
```

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[LDAP_NAME]=[HTTP_HEADER]`

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.

Default: `NONE`

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

+ *Response Attribute Map*

Map 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`.

Default: not set

Property: `com.sun.identity.agents.config.response.attribute.mapping[RESPONSE_ATTR]=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.

Default: `NONE`

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`.

Default: Not set

Property: `com.sun.identity.agents.config.session.attribute.mapping[SESSION_ATTR]=HTTP_HEADER`

Common Attributes Fetching Processing Properties

+ *Attribute Multi-Value Separator*

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`

Continuous Security Properties

+ *Continuous Security Cookies*

Maps cookie values available in inbound resource requests to entries in the environmental conditions map, which web agents send to AM during policy evaluation.

This property has the format `[cookie_name]=map_entry_name`, where:

- `[cookie_name]` specifies the name of the cookie in the inbound request.
- `map_entry_name` specifies the name of the entry within the environmental conditions map that contains the value of `cookie_name`.

Example:

```
org.forgerock.openam.agents.config.continuous.security.cookies[trackingcookie1]=myCookieEntry
```

Web agents add entries from both of the continuous security properties into the environmental conditions map, which AM's authorization framework accesses during policy evaluation.

Use server-side authorization scripts to:

- Access the map's contents
- Write scripted conditions based on cookies and headers in the request

For more information about server-side authorization scripts in AM, see the *ForgeRock Access Management Authorization Guide*.

When you specify continuous security properties, web agents generate environmental condition entries in the map as follows:

Key	Value
<code>requestIp</code> ^a	<p>Contains the inbound request's IP address. The web agent determines the IP as follows:</p> <ul style="list-style-type: none"> • If the <code>com.sun.identity.agents.config.client.ip.header</code> property is configured, the web agent extracts the IP address from the header. • If the <code>com.sun.identity.agents.config.client.ip.header</code> property is not configured, the web agent uses the container's connection information to determine the client ip address.
<code>requestDNSName</code> ^b	<p>Contains the inbound request's host name. The web agent determines the host name as follows:</p> <ul style="list-style-type: none"> • If the <code>com.sun.identity.agents.config.client.hostname.header</code> property is configured, the web agent extracts the host name from the header. • If the <code>com.sun.identity.agents.config.client.hostname.header</code> property is not configured, the web agent uses the the web agent uses the container's connection information to determine the client's host name.
<code>variable_name</code> ^c	Contains an array of cookie or header values.

^aThe `requestIp` entry is created in the map regardless of how the continuous security properties are configured.

^bThe `requestDNSName` entry is created in the map regardless of how the continuous security properties are configured.

^c There may be as many *variable_name* entries as values specified in the continuous security properties.

Consider the following example:

```
org.forgerock.openam.agents.config.continuous.security.cookies[ssid]=mySsid  
org.forgerock.openam.agents.config.continuous.security.headers[User-Agent]=myUser-Agent
```

Assuming the incoming request contains an *ssid* cookie and an *User-Agent* header, the environmental conditions map would contain the following variables:

- *requestIp*, containing the IP address of the client. For example, *192.16.8.0.1*.
- *requestDNSName*, containing the host name of the client. For example, *client.example.com*.
- *mySsid*, containing the value of the *ssid* cookie. For example, *77xe99f4zqi1199z*.
- *myUser-Agent*, containing the value of the *from* header. For example, *Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko*.

Default: not set

Property: `org.forgerock.openam.agents.config.continuous.security.cookies[cookie_name]=map_entry_name`

+ Continuous Security Headers

Maps header values in inbound resource requests to entries in the environmental conditions map, which web agents send to AM during policy evaluation.

This property has the format `[header_name]=map_entry_name`, where:

- `[header_name]` specifies the name of the header in the inbound request.
- `map_entry_name` specifies the name of the entry within the environmental conditions map that contains the value of `header_name`.

Example:

```
org.forgerock.openam.agents.config.continuous.security.headers[User-Agent]=myUserAgentHeaderEntry
```

Web agents add entries from both of the continuous security properties into the environmental conditions map, which AM's authorization framework accesses during policy evaluation.

Use server-side authorization scripts to:

- Access the map's contents
- Write scripted conditions based on cookies and headers in the request

For more information about server-side authorization scripts in AM, see the *ForgeRock Access Management Authorization Guide*.

When you specify continuous security properties, web agents generate environmental condition entries in the map as follows:

Key	Value
<code>requestIp</code> ^a	<p>Contains the inbound request's IP address. The web agent determines the IP as follows:</p> <ul style="list-style-type: none"> • If the <code>com.sun.identity.agents.config.client.ip.header</code> property is configured, the web agent extracts the IP address from the header. • If the <code>com.sun.identity.agents.config.client.ip.header</code> property is not configured, the web agent uses the container's connection information to determine the client ip address.
<code>requestDNSName</code> ^b	<p>Contains the inbound request's host name. The web agent determines the host name as follows:</p> <ul style="list-style-type: none"> • If the <code>com.sun.identity.agents.config.client.hostname.header</code> property is configured, the web agent extracts the host name from the header. • If the <code>com.sun.identity.agents.config.client.hostname.header</code> property is not configured, the web agent uses the the web agent uses the container's connection information to determine the client's host name.
<code>variable_name</code> ^c	Contains an array of cookie or header values.

^aThe `requestIp` entry is created in the map regardless of how the continuous security properties are configured.

^bThe `requestDNSName` entry is created in the map regardless of how the continuous security properties are configured.

^c There may be as many `variable_name` entries as values specified in the continuous security properties.

Consider the following example:

```
org.forgerock.openam.agents.config.continuous.security.cookies[ssid]=mySsid
org.forgerock.openam.agents.config.continuous.security.headers[User-Agent]=myUser-Agent
```

Assuming the incoming request contains an `ssid` cookie and an `User-Agent` header, the environmental conditions map would contain the following variables:

- `requestIp`, containing the IP address of the client. For example, `192.16.8.0.1`.
- `requestDNSName`, containing the host name of the client. For example, `client.example.com`.
- `mySsid`, containing the value of the `ssid` cookie. For example, `77xe99f4zqi1l99z`.
- `myUser-Agent`, containing the value of the `from` header. For example, `Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko`.

Default: not set

Property: `org.forgerock.openam.agents.config.continuous.security.headers[header_name]=map_entry_name`

SSO Properties

Set the following agent properties in the AM console at:

Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > SSO

Cookie Properties

+ *Cookie Name*

Name of the SSO token cookie used for authentication with AM. If empty, the agent retrieves the cookie name from the AM server.

Default: `iPlanetDirectoryPro`

Hot-swap: No

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

+ *Persistent JWT Cookie*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Enable persistence for JWT cookies. If `true` the JWT cookie is not set as a Session Cookie.

Default: `false`

Property:

Hot-swap: yes

+ *Cookie Security*

When `true`, the agent marks cookies as secure, sending them only if the communication channel is secure. Set to `true` when agent connections are over SSL.

Default: `false`

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

Hot-swap: yes

+ *Accept SSO Token*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Specifies whether the agent accepts SSO tokens and ID tokens as session cookies. The SSO token name is given by Cookie Name.

This property requires AM 6 or later.

Possible values are:

- **0**. The agent does not accept SSO tokens as session cookies.
- **1**. The agent accepts both SSO tokens and ID tokens as session tokens during the login flow, and afterwards. SSO tokens *are not converted* to ID tokens.

If the agent receives a request with both an SSO token and an ID token, it checks the ID token first. If invalid, it checks the SSO token. If both are invalid, the agent redirects the user for authentication.

In the same way the agent caches session information when using ID tokens, (see "Caching Capabilities"), the agent also caches session information for SSO tokens.

Set this property to **1** in the following cases:

- (Migration only) Your custom login pages use SSO tokens as session tokens, and the Custom Login property is set to **2**.

+ *Custom Login Mode and Accept SSO Token Matrix*

Custom Login Mode	Accept SSO Token ^a	
	0	1
0	"Default Login Redirection Mode".	"Default Login Redirection Mode". SSO tokens are accepted - used for REST clients.
1	Custom Login Mode, redirecting to the resource requested originally.	Not supported

Custom Login Mode	Accept SSO Token ^a	
	0	1
	Same domain and different domain. SSO token to session ID token conversion.	
2 ^b	Not supported	Custom Login Mode, redirecting with a <code>goto</code> query parameter to the originally requested resource. Same domain only. SSO token accepted.

^aRequires AM 6 or later

^bThis is not a standard flow, and the feature is evolving. It poses limitations, and it is meant only for environments migrating from earlier versions of the agent.

- (Migration only) Your applications, for example, REST or JavaScript clients, can only set SSO tokens.

Default: 0

Hot-swap: Yes

Property: `com.forgerock.agents.accept.sso.token`

+ HTTP Only Mode

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Agents with this property set to `true` mark cookies as `HttpOnly` to prevent scripts and third-party software from accessing them.

Default: `true`

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

+ Multivalue for Pre-Authn Cookie

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Web agents use the `agent-authn-tx` cookie to track the progress of authentication with AM and protect the request from replay attacks.

When this property is set to `true`, the agent creates a single cookie containing records to identify all concurrent authentication requests to AM.

In environments with lots of concurrent requests, or where the protected URLs are long, the cookie can reach the maximum size supported by the browser. When this happens, new authentication requests fail and the agent issues a 403 HTTP message to the user.

When this property is set to `false`, the agent creates a pre-authentication cookie for each authentication request to AM, with the name of `agent-authn-tx-string`. This is the default.

In some environments, this will create a large number of cookies. If you have tests in your environment that make multiple requests to AM from the same browser, you may find intermittent 403 HTTP messages; browsers have a limit of how many cookies they can handle.

Something similar happens to web servers; they have a limit of how many headers (cookies) they can manage at one time. Set the property to `true` if you find that creating too many cookies is having an impact on your environment.

Default: `false`

Property: `org.forgerock.openam.agents.config.multivalue.pre.authn.cookies`

Hot-swap: yes

+ *SameSite Cookie Attribute*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

When configured, the agent sets the `SameSite` attribute on all the cookies that it creates. The value of the `SameSite` attribute is what you configure in this property.

For example, to add the `SameSite` attribute with the value of `Lax` to the cookies, set this property to `Lax`.

See the [Same-Site Cookies Internet Draft](#) for more information about the `SameSite` attribute and the possible values it can take.

Note

The attribute will not be set on certain browsers and circumstances, as per recommendation of The Chromium Project.

Default: Not set

Property: `com.forgerock.agents.cdsso.cookie.samesite`

Hot-swap: yes

Cross Domain SSO Properties

+ *CDSSO Redirect URI*

Renames the endpoint the agent uses to process CDSSO requests. The name you choose for a production environment should not give away its purpose to end users.

The agent uses this endpoint during the default login redirection flow, but not during the custom login redirection flow.

For more information, see "Login Redirection and Login Conditional Redirection".

Default: `agent/cdsso-oauth2`

Hot-swap: yes

Property: `com.sun.identity.agents.config.cdsso.redirect.uri`

+ *Cross Domain SSO (deprecated)*

🔔 *This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.*

CDSSO is always enabled.

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

+ *CDSSO Servlet URL (deprecated)*

🔗 This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

Property: `com.sun.identity.agents.config.cdsso.cdcservlet.url`

+ Cookie Domain List

List of domains, such as `.example.com`, in which cookies have to be set in CDSO. 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
```

Default: Empty

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

+ Session Cookie Reset on Authentication Redirect

+ Not available in the console for AM 6.5.x and earlier versions.

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

When `true`, the agent does not reset the session cookie on an authentication redirect if there is a policy advice present.

When `false`, the agent resets the session cookie in all configured domains on every authentication redirect when a policy advice is present.

Default: `false`

Property: `org.forgerock.agents.config.cdsso.advice.cleanup.disable`

Cookie Reset

+ 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. An example of the header would be similar to the following:

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

If the `com.sun.identity.agents.config.fqdn.default` property is set, the web agent sets the cookie domain to the domain specified by the property. If it is not set, the web agent leaves the cookie domain blank.

Default: `false`

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

+ *Cookie Reset Name List*

List of cookies to reset. For example:

```
com.sun.identity.agents.config.cookie.reset[0]=myCookie  
com.sun.identity.agents.config.cookie.reset[1]=nextCookie
```

Default: not set

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

AM Services Properties

Set the following agent properties in the AM console at:

Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Application

Login URL Properties

+ *Custom Login Mode*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Specifies whether the agent uses a custom login mode to redirect unauthenticated users.

The custom login redirection mode requires AM 6 or later.

Possible values are:

- 0. Disabled. Default login redirection mode enabled.
- 1. Custom login redirection mode is enabled (Non-OIDC compliant login flow, standard flow). The agent tracks the user's authentication, converts the SSO token into an ID token at the end of the authentication flow, and then redirects the user to the originally requested resource. The SSO token name is given by Cookie Name.
- 2. Custom login redirection mode enabled (Non-OIDC compliant login flow, non-standard flow). The agent does not track the user's authentication, and redirects the user with a `goto` query parameter to the originally requested resource.

This mode is used with the Accept SSO Token (`com.forgerock.agents.accept.sso.token`) property.

Caution

This feature is marked as *evolving*. Use it only when migrating from earlier versions of the agents. Contact ForgeRock if you suspect your environment has a similar use case.

For examples, related properties, and information on how to configure this property, see "Login Redirection and Login Conditional Redirection".

Note that if the Custom Login Mode property is set to a value other than 0, but the redirection URL contains the `oauth2/authorize` endpoint, the agent will use the default login redirection mode.

Default: 0

Property: `org.forgerock.openam.agents.config.allow.custom.login`

Hot-swap: yes

+ Public AM URL

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Specifies the full URL of AM when it is behind a proxy during the custom login flow. For example, `protocol://public_am_fqdn:port/openam`.

Use this property both of the following points are true:

- Your environment uses custom login pages (non-OIDC-compliant flows), and the custom login pages are in a different domain than the agent.
- Your custom login pages are in a network that can only access AM using a proxy, a firewall, or any other technology that remaps the AM URL to one accessible by the custom login pages.

Consider an example where the traffic between AM and the agent happens through the *example-internal.com* domain, but the custom login pages are on the *example-external.com* domain. The traffic between the custom pages and AM translates *am.example-internal.com* into *am.example-external.com*.

You would configure *https://am.example-external.com:8443/openam* as the public AM URL.

Default: `not set`

Property: `com.forgerock.agents.public.am.url`

Hot-swap: `yes`

+ AM Login URL

When configured, specifies the URL of a custom login page to which the agent redirects incoming users without sufficient credentials so that they can authenticate.

Important

When redirecting incoming login requests to a custom login page, you must add it to either the not-enforced URL or IP lists.

Before configuring this property, read "Login Redirection and Login Conditional Redirection".

The login URL has the format `URL[?realm=realm_name¶meter1=value1&...]`, where:

- `URL` is the custom SSO-token-compliant login page to where the agent redirects the unauthenticated users.
- `[?realm=realm_name?parameter1=/value1&...]` specifies optional parameters that the agent will pass to the custom login page, for example, the AM realm which the user should log into.

You do not need to specify the realm in the login URL if any of the following conditions is true:

- The custom login page itself sets the `realm` parameter, for example, because it lets the user chose it. In this case, you must ensure the custom login page *always* appends a `realm` parameter to the `goto` URL.
- The realm where the agent must log the user to has DNS aliases configured in AM.

AM will log in the user to the realm whose DNS alias matches the incoming request URL. For example, an inbound request from the `http://marketplace.example.com` URL logs into the `marketplace` realm if the realm alias is set to `marketplace.example.com`.

- The users should always log in to the Top Level Realm.

Even if you decide to specify the realm by default, this parameter can be overwritten by the custom login page if, for example, the user can chose the realm for authentication.

You can specify as many parameters your custom login pages require.

Example:

```
https://login.example.com/login.jsp?realm=marketplace&param1=value1
```

When the agent redirects the user to the custom login page, it appends a `goto` parameter (as configured in the Goto Parameter Name property) with the agent's CDSO endpoint and a `state` parameter.

The example redirects from the agent to a custom login page:

```
http://login.example.com/login.jsp?realm=marketplace&goto=http%3A%2F%2Fagent.example.com%2Fcustom-login-response%3Fstate%3D3Df2fc384a07b7668e05fc6c26c01edf1bac8a3b55%26realm%3Dmarketplace
```

Note that the `goto` parameter is URL encoded. If the `realm` parameter is configured in the redirection rule, it is also appended to the `goto` parameter.

After the user has logged in, the custom login page must redirect back to the agent. To avoid redirection loops and login failures, consider the following constraints:

- Ensure that the custom login page redirects back to the agent using the URL contained in the `goto` parameter, and that the request contains the `state` parameter.
- Set the `realm` parameter in the redirection request to the agent if the users should not log in to AM's Top Level Realm.

For example, you could use the realm specified in the redirection request from the agent to the custom login pages (if configured in the redirection property), or the custom login page can let the user chose to which realm authenticate to and add the `realm` parameter when redirecting to the agent.

The example redirects from a custom login page to the agent with the realm added to it:

```
http://agent.example.com/custom-login-response?  
state=3Df2fc384a07b7668e05fc6c26c01edf1bac8a3b55&realm=marketplace
```

There is one exception; if the realm where the agent should log the user in to has a DNS alias configured, AM will log in the user to the realm whose DNS alias matches the incoming request URL. For example, an inbound request from the `http://marketplace.example.com` URL will be logged in to the `marketplace` realm if the realm alias is set to `marketplace.example.com`, whether there is a `realm` parameter or not.

Default: `AMURL/openam/UI/Login`

Property: `com.sun.identity.agents.config.login.url`

Hot-swap: yes

+ AM Conditional Login URL

- + *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Conditionally redirect users based on the incoming request URL. If the incoming request URL matches a specified domain name, the web agent redirects the request to a specific URL. That specific URL can be an AM instance, site, or a different website.

Important

When redirecting incoming login requests to a custom login page, you must add it to either the not-enforced URL or IP lists.

Before configuring this property, read "Login Redirection and Login Conditional Redirection".

If the FQDN Check property is enabled, the web agent iterates through the list of URLs until it finds an appropriate redirect URL that matches the FQDN check values. Otherwise, the web agent redirects the user to the URL configured in the conditional redirect rules.

Conditional redirects have the following format, with no spaces between values:

```
[String]|[URL, URL...][?realm=value&module=value2&service=value3]
```

String

Incoming login request URLs, with the following values:

- Domain: Web agents match both the domain and its subdomains. For example, `example.com` matches `mydomain.example.com` and `www.example.com`.

When you combine domain and path, you must provide the port number. For example, `www.example.com:8080/market`.

- Subdomain: For example, `example.com`.

When you combine subdomain and path, you must provide the port number. For example, `example.com:8080/market`.

- Path: For example, `/myapp`.
- Anything in the request URL: For example, a port, such as `8080`.
- No value: Nothing is specified before the `|` character. Conditional rules that do not specify the incoming request's domain apply to every incoming request.

Note

To specify the string as a regular expression, configure the following properties instead: Regular Expression Conditional Login Pattern and Regular Expression Conditional Login URL.

URL, URL...

The URL to which to redirect incoming login requests. The URL can be one of the following:

- AM instance or site.

Specify the URL of an AM instance or site in the format `protocol://FQDN[:port]/URI/oauth2/authorize`, where the port is optional if it is 80 or 443. For example, `https://openam.example.com/openam/oauth2/authorize`.

- Website other than AM.

Specify a URL in the format `protocol://FQDN[:port]/URI`, where the port is optional if it is 80 or 443. For example, `https://myweb.example.com/authApp`.

- List of AM instances or sites, or websites other than AM

If the redirection URL is not specified, the web agent redirects the request to the AM instance or site specified by the `com.sun.identity.agents.config.naming.url` bootstrap property.

Important

When using the default redirection login mode, ensure the Custom Login Mode property is set to `0`.

When using the custom redirection login mode, consider the following points:

- The Custom Login Mode property must **not** be set to `0`.
- When the agent redirects the user to the custom login page, it appends a `goto` parameter (as configured in the Goto Parameter Name property) with the agent's CDSSO endpoint and a `state` parameter.

The example redirects from the agent to a custom login page:

```
http://login.example.com/login.jsp?realm=marketplace&goto=http%3A%2F%2Fagent.example.com%2Fcustom-login-response%3Fstate%3D3Df2fc384a07b7668e05fc6c26c01edf1bac8a3b55%26realm%3Dmarketplace
```

Note that the `goto` parameter is URL encoded. If the `realm` parameter is configured in the redirection rule, it is also appended to the `goto` parameter.

After the user has logged in, the custom login page must redirect back to the agent. To avoid redirection loops and login failures, consider the following constraints:

- Ensure that the custom login page redirects back to the agent using the URL contained in the `goto` parameter, and that the request contains the `state` parameter.
- Set the `realm` parameter in the redirection request to the agent if the users should not log in to AM's Top Level Realm.

For example, you could use the realm specified in the redirection request from the agent to the custom login pages (if configured in the redirection property), or the custom login page can let the user chose to which realm authenticate to and add the `realm` parameter when redirecting to the agent.

The example redirects from a custom login page to the agent with the realm added to it:

```
http://agent.example.com/custom-login-response?state=3Df2fc384a07b7668e05fc6c26c01edf1bac8a3b55&realm=marketplace
```

There is one exception; if the realm where the agent should log the user in to has a DNS alias configured, AM will log in the user to the realm whose DNS alias matches the incoming request URL. For example, an inbound request from the `http://marketplace.example.com` URL will be logged in to the marketplace realm if the realm alias is set to `marketplace.example.com`, whether there is a `realm` parameter or not.

?realm=/value

The AM realm to where the agent should log the users. For example, `?realm=/marketplace`.

You do not need to specify the realm in the login URL if any of the following conditions is true:

- The custom login page itself sets the `realm` parameter, for example, because it lets the user chose it. In this case, you must ensure the custom login page *always* appends a `realm` parameter to the `goto` URL.
- The realm where the agent must log the user to has DNS aliases configured in AM.

AM will log in the user to the realm whose DNS alias matches the incoming request URL. For example, an inbound request from the `http://marketplace.example.com` URL logs into the `marketplace` realm if the realm alias is set to `marketplace.example.com`.

- The users should always log in to the Top Level Realm.

Even if you decide to specify the realm by default, this parameter can be overwritten by the custom login page if, for example, the user can chose the realm for authentication.

&module=value2&service=value3

Parameters that can be added to the URL(s), such as:

- **module**, which specifies the authentication module the user authenticates against. For example, `?module=myAuthModule`.
- **service**, which specifies an authentication chain or tree the user authenticates against. For example, `?service=myAuthChain`.
- Any other parameters your custom login pages require.

Chain parameters with an **&** character, for example, `realm=value&service=value`.

When configuring conditional login with multiple URLs, set up the parameters for each of the URLs.

Examples:

```
com.forgerock.agents.conditional.login.url[0]=example.com|https://openam.example.com/openam/oauth2/authorize
com.forgerock.agents.conditional.login.url[1]=myapp.domain.com|https://openam2.example.com/openam/oauth2/authorize?realm=/sales
com.forgerock.agents.conditional.login.url[3]=sales.example.com/marketplace|https://openam1.example.com/openam/oauth2/authorize?realm=/sales, https://openam2.example.com/openam/oauth2/authorize?realm=/marketplace
com.forgerock.agents.conditional.login.url[4]=myapp.domain.com|http://mylogin.example.com?realm=/customers
com.forgerock.agents.conditional.login.url[5]=|https://openam3.example.com/openam/oauth2/authorize?realm=/customers&module=myAuthModule
```

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

Hot-swap: Yes

+ Regular Expression Conditional Login URL

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Conditionally redirect users based on the incoming request URL. If the incoming request URL matches a regular expression, the web agent redirects the request to a specific URL. That specific URL can be an AM instance, site, or a different website.

Before configuring this property, read "Login Redirection and Login Conditional Redirection".

Regular expression conditional login URLs require two properties:

- **Regular Expression Conditional Login Pattern:** Specifies the regular expression that the domain name must match.
- `org.forgerock.agents.config.conditional.login.url`. Specifies the redirection URL and its parameters. Configure this property in the same way you would configure the AM Conditional Login URL property, except you do not specify the string or pipe | character.

Example:

```
org.forgerock.agents.config.conditional.login.pattern[0] = .*shop
org.forgerock.agents.config.conditional.login.url[0] = http://openam.example.com/openam/
oauth2/authorize?realm=sales
```

Default: not set

Properties:

```
org.forgerock.agents.config.conditional.login.pattern[n]
org.forgerock.agents.config.conditional.login.url[n]
```

Hot-swap: yes

+ *Regular Expression Conditional Login Pattern*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

See Regular Expression Conditional Login URL.

+ *Agent Connection Timeout (deprecated)*

🔔 *This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.*

Property: `com.sun.identity.agents.config.auth.connection.timeout`

+ *Polling Period for Primary Server (deprecated)*

🔗 This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

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

Before configuring these properties, read "Logout Redirection".

+ *AM Logout URL*

Specifies the page the agent will redirect the end user to for log out. It can be a page in AM, such as `https://openam.example.com:8443/openam/UI/Logout`, or a page in the application.

The AM logout page will invalidate the user session in AM, but the page in the application may not be able to. See the Invalidate Logout Session property for configuration options.

Default: `AM_URL/openam/UI/Logout`

Property: `com.sun.identity.agents.config.logout.url[n]`

Hot-swap: yes

+ *Disabled Logout Redirection*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

During the logout flow and after logging out the user, this property specifies whether the agent should redirect the end user to another page. For example, to the landing page of the application, or to a login page.

true: Logout redirection is disabled - the agent does not perform the last redirection, and the web client is left on the logout page.

false: Logout redirection is enabled - the agent appends a goto parameter to the logout URL with the value of the Logout Redirect URL.

Default: **true**

Property: `com.forgerock.agents.config.logout.redirect.disable`

Hot-swap: yes

Agent Logout URL Properties

Before configuring these properties, read "Logout Redirection".

+ Logout URL List

Specifies one or more of application logout URLs, such as `http://www.example.com/logout.html`. The agent will trigger a logout flow when the end user accesses one of these pages. Therefore, these pages must be handled by your web server.

Configure alongside the Logout Redirect URL or Agent Logout URL Regular Expression properties.

Default: not set

Property: `com.sun.identity.agents.config.agent.logout.url[n]`

Hot-swap: yes

+ Agent Logout URL Regular Expression

+ Not available in the console for AM 6.5.x and earlier versions.

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Specifies a Perl-compatible regular expression that matches logout URLs.

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.

Default: not set

Property: `com.forgerock.agents.agent.logout.url.regex`

Hot-swap: yes

+ 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`

Default: not set

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

Hot-swap: yes

+ Logout Redirect URL

The end user gets redirected to this URL after logout, unless the Disabled Logout Redirection property is set to `true`. Configure alongside the Logout URL List property.

The URL you define in this property must be handled by your web server.

Default: not set

Property: `com.sun.identity.agents.config.logout.redirect.url`

Hot-swap: yes

+ Invalidate Logout Session

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

During the logout flow, this property specifies whether the agent is responsible for invalidating the end user's session.

Possible values are:

- **true**. The agent invalidates the session in AM when redirecting to the logout URL.
Use this value when the `Logout URL List` property is set to a page in your application, and your application *does not handle* the session invalidation process.

- **false**. The agent does not invalidate the session in AM when redirecting to the logout URL.

Use this value in the following scenarios:

- When the OpenAM Logout URL property is set to a SAML v2.0 single logout page in AM.
- When the OpenAM Logout URL property is set to AM's end user pages.
- When the OpenAM Logout URL property is set to a page in your application, and your application handles the session invalidation process.

Default: true

Property: `org.forgerock.agents.config.logout.session.invalidate`

Hot-swap: yes

+ *Enable Regex for Logout URL List*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Allow regular expressions in Logout URL List.

Default: false

Property: `org.forgerock.agents.config.logout.regex.enable`

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.

Default: 3

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.

Default: 3

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 `true`, 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.

Default: `false`

Property: `com.sun.identity.agents.config.fetch.from.root.resource`

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.

Default: `false`

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 - AM ServerTime.

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.

Default: `0`

Property: `com.sun.identity.agents.config.policy.clock.skew`

Hot-swap: no

+ *Policy Evaluation Realm*

In AM 6.5 and earlier versions, this property was named `Realm`.

Realm where AM evaluates policies for policy decision requests from the agent.

By default, AM evaluates policies in the top-level realm. Set this property to cause AM to use a different realm. This property is recognized by AM, not the agent.

Important

The policy set configured by Policy Set must exist in the realm configured by Policy Evaluation Realm. Otherwise, policy evaluation produces `DENY` results without writing warnings to the logs.

The default policy set exists only in the top-level realm. If you are using a different realm for policy evaluation, do one of the following:

- Create the `iPlanetAMWebAgentService` policy set in that realm.
- Create a different policy set in that realm, and configure Policy Set to use it.

Default: `/` (top-level realm)

Type: Map of `application name:realm`

Property: `org.forgerock.openam.agents.config.policy.evaluation.realm`

Hot-swap: Yes

+ *Policy Set*

In AM 6.5 and earlier versions, this property was named `Application`.

The name of the policy set where AM evaluates policies for policy decision requests from the agent.

By default, AM evaluates policies in `iPlanetAMWebAgentService`. Set this property to cause AM to use a different policy set. This property is recognized by AM, not the agent.

Important

The policy set configured by Policy Set must exist in the realm configured by Policy Evaluation Realm. Otherwise, policy evaluation produces `DENY` results without writing warnings to the logs.

The default policy set exists only in the top-level realm. If you are using a different realm for policy evaluation, do one of the following:

- Create the `iPlanetAMWebAgentService` policy set in that realm.
- Create a different policy set in that realm, and configure Policy Set to use it.

Default: `iPlanetAMWebAgentService`

Type: Map of *application name:policy set*

Property: `org.forgerock.openam.agents.config.policy.evaluation.application`

Hot-swap: Yes

Miscellaneous Properties

Set the following agent properties in the AM console at:

Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Miscellaneous

Advice Handling Properties

+ *Composite Advice Handling*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

When set to `true`, the agent sends composite advice in the query (GET request) instead of sending it through a POST request.

Default: `false`

Property: `com.sun.am.use_redirect_for_advice`

+ Composite Advice Encode

+ Not available in the console for AM 6.5.x and earlier versions.

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Whether to base64 URL-encode composite advices before sending them to custom login endpoints. Set to `true` to increase the security, and protect against cross-site scripting attacks.

The following example is a request from the Agent to AM for a custom login page:

- When this property is `false`, advices are not encoded:

```
http://mylogin.dom.com:80/mylogin?agent/custom-login-response=
https://www.mydomain.com:443/agent/custom-login-response?state=b2e...bff
&composite_advice=<Advices><AttributeValuePair><Attribute name="TransactionConditionAdvice"/
><Value>05c...f6a</Value></AttributeValuePair></Advices>
&original_request_url=https://www.mydomain.com:443/superweb
```

- When this property is `true`, advices are encoded as follows:

```
http://mylogin.dom.com:80/mylogin?agent/custom-login-response=
https://www.mydomain.com:443/agent/custom-login-response?state=b2e...bff
&composite_advice=PEFkdm...zPg
&original_request_url=https://www.mydomain.com:443/superweb
```

The following example is an OIDC request from the Agent to AM:

- When this property is `false`, advices are not encoded:

```
http://am/server/oauth2/authorize?claims=
{"id_token":{"acr":{"essential":true,"values":
["composite_advice: <Advices><AttributeValuePair><Attribute name="TransactionConditionAdvice"/
><Value>05c...f6a</Value></AttributeValuePair></Advices>"]}
,"TxId":{"value":"05c...f6a"}}}
&response_mode=form_post&state=b2e...bff
&redirect_uri=https://www.mydomain.com:443/agent/cdsso-oauth2
&response_type=id_token&scope=openid
&client_id=myagent&agent_provider=true&agent_realm=/&nonce=6C6...C47
```

- When this property is `true`, advices are encoded:

```
http://am/server/oauth2/authorize?claims=
{"id_token":{"acr":{"essential":true,"values":
["composite_advice:PEFkdm...zPg"]
},"TxId":{"value":"05c...f6a"}}}
&response_mode=form_post&state=b2e...bff
&redirect_uri=https://www.mydomain.com:443/agent/cdsso-oauth2
&response_type=id_token&scope=openid
&client_id=myagent&agent_provider=true&agent_realm=/&nonce=54B...909
```

Available: AM versions 5.5.3, 6.0.1, 6.5.3, and later

Default: `false`

Property: `com.forgerock.agents.advice.b64.url.encode`

Hot-swap: Yes

Locale

+ *Agent Locale (deprecated)*

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

The default locale for the agent.

Default: `en_US`

Property: `com.sun.identity.agents.config.locale`

Hot-swap: no

Anonymous User

+ *Anonymous User*

Enable or disable `REMOTE_USER` processing for anonymous users.

Default: `false`

Property: `com.sun.identity.agents.config.anonymous.user.enable`

Cookie Processing

+ *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`.

Default: `false`

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*

Number of seconds before the expiry of custom cookies or the pre-authentication cookie, `agent-authn-tx`.

If POST data preservation is enabled, the request expires after the time specified in POST Data Entries Cache Period, which is by default 10 minutes. In this case, consider increasing the value of this property to at least 600 seconds.

Default: `300`

Property: `com.sun.identity.agents.config.profile.attribute.cookie.maxage`

URL Handling

+ *URL Comparison Case Sensitivity Check*

When enabled, enforces case insensitivity in both policy and not-enforced URL evaluation.

Default: `true`

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.

Default: `false`

Property:

Ignore Naming URL

+ Ignore Preferred Naming URL in Naming Request (deprecated)

🔔 This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

When enabled, do not send a preferred naming URL in the naming request.

Default: `true`

Property: `com.sun.identity.agents.config.ignore.preferred.naming.url`

Invalid URL

+ Invalid URL Regular Expression

+ Not available in the console for AM 6.5.x and earlier versions.

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

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

Property: `com.forgerock.agents.agent.invalid.url.regex`

Ignore Server Check

+ Ignore Server Check (deprecated)

🔔 This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

When `true`, do not check whether AM is up before doing a 302 redirect.

Default: `false`

Property: `com.sun.identity.agents.config.ignore.server.check`

Ignore Path Info

+ *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 Java 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/servlet/*` and the request URL is `http://host/webapp/servlet/example.jsp`, the path info is `/servlet/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 Plus web agent does not support this setting.

Default: `false`

Property: `com.sun.identity.agents.config.ignore.path.info`

Multi-Byte Enable Properties

+ *Native Encoding of Profile Attributes (deprecated)*

This property does not apply to Web Agents 5.8, although it may appear in the AM console.

When `true`, the agent encodes the LDAP header values in the default encoding of operating system locale. When disabled, the agent uses UTF-8.

Default: `false`

Property: `com.sun.identity.agents.config.convert.mbyte.enable`

Goto Parameter Name

+ *Goto Parameter Name*

Renames the `goto` parameter. The web agent appends the requested URL to the renamed parameter during redirection after logout or after reaching an access denied page. Rename the parameter when your application requires a parameter other than `goto`.

Consider the following example:

```
com.sun.identity.agents.config.redirect.param=goto2
```

A valid redirection URL using the `goto2` parameter may look similar to the following:

```
https://www.example.com:8443/accessDenied.html?goto2=http%3A%2F%3Awww.example.com%3A8020%3Amanagers%2Findex.jsp
```

In this example, the URL appended to the `goto2` parameter is the URL that the user tried to access when the web agent redirected the request to the `accessDenied.html` page. Note that you configure the access denied page using the Resources Access Denied URL (`com.sun.identity.agents.config.access.denied.url`) property.

The Goto Parameter Name property also affects the OpenAM Logout URL (`com.sun.identity.agents.config.logout.url`) property.

Default: `goto`

Property: `com.sun.identity.agents.config.redirect.param`

Hot-swap: `yes`

JSON-Formatted Response

+ *URLs to Receive JSON-Formatted Responses*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Use wildcard patterns to specify a list of resource URLs that will trigger a JSON-formatted response from the agent, and optionally, override the default HTTP status code.

For more information on wildcard usage, see [Specifying Resource Patterns with Wildcards](#).

Returning the responses in JSON format is useful for non-browser-based, or AJAX applications, that may not want to redirect users to the AM user interface for authentication.

Tip

You should set the HTTP Return Code for JSON-Formatted Responses property to a supported HTTP code, for example **202**, to prevent applications that do not support redirects, for example, from displaying a default error page.

For example, you could specify the following settings:

```
org.forgerock.agents.config.json.url[0]=http://*.example.com:*/api/*
org.forgerock.agents.config.json.response.code=202
```

Performing a GET operation on a protected resource covered by the wildcard pattern would trigger a JSON response, as follows:

```
$ curl --include https://www.example.com/api/
HTTP/1.1 202 Accepted
Date: Tue, 29 Jan 2019 15:10:09 GMT
Server: Apache/2.4.6 (CentOS) OpenAM Web Agent/5.5.1.0
Set-Cookie: am-auth-jwt=; Path=/; Max-Age=0; Expires=Thu, 01-Jan-1970 00:00:00 GMT
Set-Cookie: agent-authn-tx=eJwN...XI34=; Path=/; HttpOnly; Max-Age=300; Expires=Tue, 29-Jan-2019
15:15:09 GMT
Content-Length: 418
Content-Type: text/html; charset=iso-8859-1

{
  "error": {
    "errors": [
      {
        "message": "redirect",
        "location": "https://openam.example.com:8443/openam/oauth2/authorize
?response_mode=form_post
&state=d38a8b36-894e-4544-a5a1-d9230fb85246
&redirect_uri=https%3A%2F%2Fwww.example.com%3A443%2Fagent%2Fcdsso-oauth2
&response_type=id_token
&scope=openid
&client_id=myApacheAgent
&agent_provider=true
&agent_realm=%2F
&nonce=531F...BB67"
      }
    ],
    "code": 302
  }
}
```

Notice that the HTTP result code is the specified **202 Accepted**, and the JSON response contains the actual result, a **302 Found** (redirect) to the AM server for authentication.

Default: not set

Property: `org.forgerock.agents.config.json.url[n]`

+ Headers and Values to Receive JSON-Formatted Responses

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Specify HTTP headers and associated values that trigger JSON-formatted errors to be returned.

Tip

You should set the HTTP Return Code for JSON-Formatted Responses property to a non-error HTTP code, for example `202`, to prevent user agents displaying their default error pages.

For example, you could specify the following settings:

```
org.forgerock.agents.config.json.header[enableJsonResponse]=true
org.forgerock.agents.config.json.response.code=202
```

Performing a GET operation, and including the specified header, would trigger a JSON response, as follows:

```
$ curl --include --header "enableJsonResponse: true" https://www.example.com/endpoints/
HTTP/1.1 202 Accepted
Date: Tue, 29 Jan 2019 15:10:09 GMT
Server: Apache/2.4.6 (CentOS) OpenAM Web Agent/5.5.1.0
Set-Cookie: am-auth-jwt=; Path=/; Max-Age=0; Expires=Thu, 01-Jan-1970 00:00:00 GMT
Set-Cookie: agent-authn-tx=eJwN...EySX; Path=/; HttpOnly; Max-Age=300; Expires=Tue, 29-Jan-2019
15:30:19 GMT
Content-Type: text/html; charset=iso-8859-1

{
  "error": {
    "errors": [
      {
        "message": "redirect",
        "location": "https://openam.example.com:8443/openam/oauth2/authorize
?response_mode=form_post
&state=d1e8b9e4-53c4-134a-bc6e-7b9c7f22e943
&redirect_uri=https%3A%2F%2Fwww.example.com%3A443%2Fagent%2Fcdsso-oauth2
&response_type=id_token
&scope=openid
&client_id=myApacheAgent
&agent_provider=true
&agent_realm=%2F
&nonce=531F...BB67"
      }
    ],
    "code": 302
  }
}
```

Notice that the HTTP result code is the specified **202 Accepted**, and the JSON response contains the actual result, a **302 Found** (redirect) to the AM server for authentication.

Default: not set

Property: `org.forgerock.agents.config.json.header[Header]=Value`

+ *Invert Properties That Receive JSON-Formatted Responses*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

When `true`, 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 do *not* trigger JSON-formatted responses. Only non-specified values trigger JSON-formatted responses.

Default: `false`

Property: `org.forgerock.agents.config.json.url.invert`

+ HTTP Return Code for JSON-Formatted Responses

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Specifies an HTTP response code to return when a JSON-formatted error is triggered.

Tip

Set this property to a non-error HTTP code, for example `202`, to prevent user agents displaying their default error pages.

Example: `org.forgerock.agents.config.json.response.code=202`

Default: not set

Property: `org.forgerock.agents.config.json.response.code`

Miscellaneous Header-Related Properties

+ Add Cache-Control Headers

When `true`, enables the use of Cache-Control headers that prevent proxies from caching resources accessed by unauthenticated users.

Default: `false`

Property: `com.forgerock.agents.cache_control_header.enable`

+ MIME-Encode HTTP Header Values

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

MIME-encoding of HTTP header values:

- Not set or `0`: The agent MIME-encodes the value of HTTP headers if said value is a multi-byte Unicode string.
- `1`: The agent MIME-encodes the value of every HTTP header.
- `2`: The agent does not MIME-encode the value of any HTTP header.

Default: not set

Property: `com.forgerock.agents.header.mime.encode`

Hot-swap: yes

Deprecated Properties

+ *Anonymous User Default Value (deprecated)*

User ID of unauthenticated users.

Default: `anonymous`

Property: `com.sun.identity.agents.config.anonymous.user.id`

Advanced Properties

Set the following agent properties in the AM console at:

Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced

Client Identification

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.

Default: not set

Property: `com.sun.identity.agents.config.client.ip.header`

+ *Client Hostname Header*

HTTP header name that holds the hostname of the client.

Default: not set

Property: `com.sun.identity.agents.config.client.hostname.header`

Load Balancing

+ *Load Balancer Setup (deprecated)*

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

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`).

Note

The following headers, when defined on the proxy or load-balancer, override the value of the `com.sun.identity.agents.config.agenturi.prefix` property:

- `X-Forwarded-Proto`
- `X-Forwarded-Host`
- `X-Forwarded-Port`

Do not configure the agent to override its hostname, port, or protocol, if you are already using these headers.

Default: `false`

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`).

Note

The following headers, when defined on the proxy or load-balancer, override the value of the `com.sun.identity.agents.config.agenturi.prefix` property:

- `X-Forwarded-Proto`
- `X-Forwarded-Host`
- `X-Forwarded-Port`

Do not configure the agent to override its hostname, port, or protocol, if you are already using these headers.

Default: `false`

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`).

Note

The following headers, when defined on the proxy or load-balancer, override the value of the `com.sun.identity.agents.config.agenturi.prefix` property:

- `X-Forwarded-Proto`
- `X-Forwarded-Host`
- `X-Forwarded-Port`

Do not configure the agent to override its hostname, port, or protocol, if you are already using these headers.

Default: `false`

Property: `com.sun.identity.agents.config.override.port`

+ *Override Notification URL (deprecated)*

🔗 This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

Default: `false`

Property: `com.sun.identity.agents.config.override.notification.url`

Fragment Redirect

+ *Fragment Redirect Enabled*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Manage the browser's URL fragment during authentication, as follows:

- `false`: Remove the browser's URL fragment during authentication.

For example, a request to `http://my.domain.com:8080/myapp/index.html#chapter-1` is authenticated and redirected to `http://my.domain.com:8080/myapp/index.html`. The fragment `#chapter-1` is lost.

- `true`: Save the browser's URL fragment during authentication.

For example, a request to `http://my.domain.com:8080/myapp/index.html#chapter-1` is authenticated and redirected to the same URL. The fragment is not lost.

An extra redirect is incurred for all unauthenticated requests, to capture and process the URL fragment.

Fragment redirect is not possible in the following cases:

- The request URL is marked for JSON responses, usually for non-browser clients, such as JavaScript or other coded clients.
- The agent is configured to use **Custom Login Mode** with the value `2`, for non-OIDC compliant login flow, non-standard flow. The agent does not track the user's authentication, and redirects the user with a `goto` query parameter to the originally requested resource.

Default: `false`

Property: `org.forgerock.agents.config.fragment.redirect.enable`

Post Data Preservation

+ *POST Data Preservation*

Enables HTTP POST data preservation.

Default: `false`

Property: `com.sun.identity.agents.config.postdata.preserve.enable`

+ *POST Data Entries Cache Period*

Number of minutes before expiry of the Post Data Preservation cache.

Consider increasing Profile Attributes Cookie Maxage to at least the value of this property.

Default: `10`

Property: `com.sun.identity.agents.config.postcache.entry.lifetime`

+ *POST Data Storage Directory*

The directory local to the agent installation where the agent writes preserved POST data while requesting authorization to AM.

This is a bootstrap property. Configure it in the `agent.conf` file, even when the agent is configured in centralized mode.

Default: `/web_agents/agent_type/log`

Property: `org.forgerock.agents.config.postdata.preserve.dir`

+ *POST Data Sticky Load Balancing Mode*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

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.

Default: not set

Property: `com.sun.identity.agents.config.postdata.preserve.stickysession.mode`

+ POST Data Sticky Load Balancing Value

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

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 to the same value configured in the `com.sun.identity.agents.config.postdata.preserve.lbcookie` property.

Default: not set

Property: `com.sun.identity.agents.config.postdata.preserve.stickysession.value`

+ POST Data Sticky Load Balancing Cookie Name

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Specifies the name of a cookie to use for enabling sticky load balancing when the `com.sun.identity.agents.config.postdata.preserve.stickysession.mode` property is set to `COOKIE`.

Set the cookie name to the same value configured in the `com.sun.identity.agents.config.postdata.preserve.stickysession.value` property.

Default: not set

Property: `com.sun.identity.agents.config.postdata.preserve.lbcookie`

+ *Submit POST Data using JavaScript*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

When `true`, preserved POST data is resubmitted to the destination server after authentication by using JavaScript.

Default: `false`

Property: `org.forgerock.agents.pdp.javascript.repost`

+ *URLs Ignored by the Agent POST Data Inspector*

+ *Not available in the console for AM 6.5.x and earlier versions.*

To set as a custom property in AM, go to Realms > *Realm Name* > Applications > Agents > Web > *Agent Name* > Advanced > Custom Properties.

To set for local configurations, add the property to the `agent.conf` file.

Specifies a list of URLs that will not be processed by the web 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 Properties.

Default: not set

Property: `org.forgerock.agents.config.skip.post.url[n]`

Sun Java System Proxy Server Properties

+ *Override Proxy Server's Host and Port (deprecated)*

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

When `true` ignore the host and port settings.

Default: `false`

Property: `com.sun.identity.agents.config.proxy.override.host.port`

Hot-swap: no

Microsoft IIS Server Properties

+ *Authentication Type (deprecated)*

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

The agent should normally perform authentication, so this is not required. If necessary, set to `none`.

Default: not set

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.

Default: not set

Property: `com.sun.identity.agents.config.replaypasswd.key`

+ *Filter Priority (deprecated)*

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

The loading priority of filter, DEFAULT, HIGH, LOW, or MEDIUM.

Default: **HIGH**

Property: `com.sun.identity.agents.config.iis.filter.priority`

+ *Filter configured with OWA (deprecated)*

🔔 This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

Property: `com.sun.identity.agents.config.iis.owa.enable`

+ *Change URL Protocol to HTTPS (deprecated)*

🔔 This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

Property: `com.sun.identity.agents.config.iis.owa.enable.change.protocol`

+ *Idle Session Timeout Page URL (deprecated)*

🔔 This property does not apply to Web Agents 5.8.2.1, 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`.

Default: **false**

Property: `com.sun.identity.agents.config.iis.password.header`

+ *Logon and Impersonation*

When **true**, the agent should do Windows Logon and User Impersonation.

Default: **false**

Property: `com.sun.identity.agents.config.iis.logonuser`

IBM Lotus Domino Server Properties

+ *Check User in Domino Database (deprecated)*

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

Default: `false`

Property: `com.sun.identity.agents.config.domino.check.name.database`

+ Use LTPA token (deprecated)

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

Default: `false`

Property: `com.sun.identity.agents.config.domino.ltpa.enable`

+ LTPA Token Cookie Name (deprecated)

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

Default: `LtpaToken`

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

+ LTPA Token Configuration Name (deprecated)

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

Default: `LtpaToken`

Property: `com.sun.identity.agents.config.domino.ltpa.config.name`

+ LTPA Token Organization Name (deprecated)

⚠ This property does not apply to Web Agents 5.8.2.1, although it may appear in the AM console.

Default: not set

Property: `com.sun.identity.agents.config.domino.ltpa.org.name`

Custom Properties

+ *Custom properties*

Additional properties to augment the set of properties supported by agent. Custom properties can be specified as follows:

- `customproperty=custom-value1`
- `customlist[0]=customlist-value-0`
- `customlist[1]=customlist-value-1`
- `custommap[key1]=custommap-value-1`
- `custommap[key2]=custommap-value-2`

Add any property that is not yet in the AM console as a custom property.

Property: `com.sun.identity.agents.config.freeformproperties`

Configuring Web Agent Environment Variables

This section covers web agent properties that are configured by using environment variables.

Configure the environment variables to affect the user that is running the web server, virtual host, or location that the agent protects.

Tip

For information on allowing environment variables to be used in NGINX, see the `env` directive in the *NGINX Core functionality documentation*.

You *must* restart the container in which web agents are running to apply changes to these settings.

- Web Agent Environment Variables
- Web Agent Installer Environment Variables

Web Agent Environment Variables

AM_IPC_BASE

(Unix only) Specifies the base number for IPC identifiers used by the agent. The shared memory semaphore ID range used by the agent starts at the specified value. Set this variable only if you detect that the agent semaphores are clashing with those of other processes in your environment.

The default is an arbitrary value.

AM_MAX_AGENTS

Specifies the maximum number of agent instances in the installation. The higher the number, the more shared memory the agent reserves. The default value is `32`.

Once the number is met, any additional agent instances that start will log an error, and will not protect resources.

AM_MAX_SESSION_CACHE_SIZE

Specifies the maximum size of shared memory for the session and policy cache, in bytes. When unset, the session cache size is 16777216 bytes (16 MB). The maximum size the cache can grow is 1073741824 bytes (1 GB), and the minimum size is 1024 bytes (1 MB).

Setting the variable to `0` configures a cache size of 16777216 bytes (16 MB).

You may need to increase the size of the session and policy cache if you plan to hold many active sessions at any given time.

AM_NET_TIMEOUT

Specifies the timeout in seconds for the agent installer to contact AM during agent configuration validation. The default is 4 seconds.

If the installer takes longer than the default, or configured value, to contact AM and validate the configuration; then installation will fail.

AM_POLICY_CACHE_MODE

Set to `on` to enable the policy cache.

You must also specify a directory in which to store the policies in the `AM_POLICY_CACHE_DIR` environment variable.

For more information, see Policy Cache.

AM_POLICY_CACHE_DIR

Specifies a directory in which to store the policy cache. The agent must be able to write to this directory. For example, `/path/to/web_agents/<agent_type>/log`.

For more information, see Policy Cache.

AM_RESOURCE_PERMISSIONS

(Unix only) Specifies the permissions that the agent sets for its runtime resources. Possible values are:

- `0600`
- `0660`

- 0666

The `AM_RESOURCE_PERMISSIONS` environment variable requires the `umask` value to allow these permissions for the files.

Consider an example where the Apache agent is running with the `apache` user. The `umask` value is set to `0022` and the `AM_RESOURCE_PERMISSIONS` environment variable is set to `0666`. The agent runtime resources will have the following permissions:

Resource Permissions Example in Linux

Resource	Permission	Owner
<code>/path/to/web_agents/agent_type/log/system_n.log</code>	644	apache
<code>/path/to/web_agents/agent_type/log/monitor_n.log</code>	644	apache
<code>/path/to/web_agents/agent_type/instances/agent_n/conf/agent.conf</code>	640	apache
<code>/path/to/web_agents/agent_type/instances/agent_n/logs/debug/debug.log</code>	644	apache
<code>/dev/shm/am_cache_0</code>	644	apache
<code>/dev/shm/am_log_data_0</code>	644	apache

Any semaphores owned by the `apache` user have `644` permissions as well.

Consider another example where `umask` is set to `0002` and the `AM_RESOURCE_PERMISSIONS` environment variable is set to `0666`. The files would be created with `664` permissions, which would allow the files to be read and written by the members of the group, as well.

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
- TLSv1.2 (Enabled)
- TLSv1.3 (Enabled)

For example, to configure `TLSv1.1`, set the environment variable to `AM_SSL_OPTIONS = -SSLv3 -TLSv1 -TLSv1.2`.

AM_SYSTEM_LOG_LEVEL

Specifies the log level of garbage collector statistics for all web agent instances in the container. The logs are written into the `system_n.log` file, where `n` indicates the agent group number.

Consider an environment with two Apache server installations:

- `Apache_1` has two agent instances configured, `agent_1` and `agent_2`, configured to share runtime resources (AmAgentId is set to 0). Both agent instances will write to the `system_0.log` file.
- `Apache_2` has one agent instance configured, `agent_3`, with AmAgentId set to 1. The instance will write to the `system_1.log` file.

By default, the `system_n.log` file is stored in the `/path/to/web_agents/agent_type/log` directory. To modify its path or its size, configure the `AM_SYSTEM_LOG_PATH` and `AM_SYSTEM_LOG_SIZE` environment variables.

The `system_n.log` file can contain the following information:

- Agent version information, written when the agent instance starts up.
- Logs for the agent background processes.
- WebSocket connection errors.
- Cache stats and removal of old POST data preservation files.
- Agent notifications.

The default value of the `AM_SYSTEM_LOG_LEVEL` variable is `Error`. Increase it to `Message` or `All` for fine-grained detail.

Valid values for the variable are:

- All
- Message
- Warning
- Error
- Info

AM_SYSTEM_LOG_PATH

Specifies the directory where the `system_n.log` file is stored.

By default, this environment variable is configured to `/path/to/web_agents/agent_type/log`.

AM_SYSTEM_LOG_SIZE

Specifies the size, in bytes, of the `/path/to/web_agents/agent_type/log/system_n.log` file.

The default is 0, which specifies that the log file size is unlimited. Valid ranges for this variable go from 0 to 4294967295 bytes (4GB).

AM_SYSTEM_PIPE_DIR

(Unix only) Specifies the directory where the agent instances will store their temporary pipe files.

The default is `/path/to/web_agents/agent_type/log/`.

Web Agent Installer Environment Variables

Use the following properties during agent installation as needed.

For example, use the `AM_SSL_*` environment properties to install agents when SSL is already configured in AM or in the agent container.

The install process will set the values in the environment variables as part of the agent configuration if the process is successful.

You may also use these settings with the `agentadmin -V[i]` command if you want to test values different from those already configured for the agent instance.

AM_PROXY_HOST

When AM and the agent communicate through a proxy configured in forward proxy mode, this environment variable specifies the proxy FQDN.

AM_PROXY_PASSWORD

When AM and the agent communicate through a proxy configured in forward proxy mode and proxy requires that the agent authenticates using Basic Authentication, this environment variable specifies the password of the agent.

AM_PROXY_USER

When AM and the agent communicate through a proxy configured in forward proxy mode and the proxy requires the agent authenticates using Basic Authentication, this environment variable specifies the user name of the agent.

AM_PROXY_PORT

When AM and the agent communicate through a proxy configured in forward proxy mode, this variable specifies the proxy port number.

APACHE_RUN_USER

Specifies the user running the Apache HTTP or IBM HTTP Server.

Use this variable if there is no Apache user defined in the `httpd.conf` file.

APACHE_RUN_GROUP

Specifies the group to which the user running the Apache HTTP or IBM HTTP Server belongs.

Use this variable if there is no Apache group defined in the `httpd.conf` file.

AM_SSL_SCHANNEL

(Windows only) Specifies whether the agent installation process should use the Windows Secure Channel API. Possible values are:

- `0`. Disable Windows Secure Channel API support. The agent will use OpenSSL libraries instead.

Ensure that the OpenSSL libraries are in the appropriate place, as specified in the "OpenSSL Library Location by Operating System" table.

- `1`. Enable Windows Secure Channel API support.

AM_SSL_KEY

(OpenSSL only) When AM is configured to perform client authentication, this environment variable specifies a PEM file that contains the private key corresponding to the certificate specified in the `AM_SSL_CERT` environment variable. For example, `/opt/certificates/client-private-key.pem` or `C:\Certificates\client-private-key.pem`.

AM_SSL_PASSWORD

(OpenSSL only) When AM is configured to perform client authentication, this environment variable specifies the obfuscated password of the private key configured in the `AM_SSL_KEY` variable. Configure this variable only if the private key is password-protected.

To obfuscate the password, use the `agentadmin --p` command. For example:

Unix

```
$ /path/to/web_agents/agent_type/bin/> agentadmin --p "Encryption Key" "`cat
certificate_password.file`"
Encrypted password value: zck+6RKqjtc=
com.forgerock.agents.config.cert.key.password = zck+6RKqjtc=
```

Windows

```
C:\path\to\web_agents\agent_type\bin> agentadmin.exe --p "Encryption_Key" "Certificate_File_Password"
Encrypted password value: zck+6RKqjtc=
```

AM_SSL_CIPHERS

(OpenSSL only) Specifies 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.htm>.

For example, `HIGH:MEDIUM`.

AM_SSL_CERT

When AM is configured to perform client authentication, this environment variable specifies a PEM file that contains the certificate chain for the agent.

For example, `/opt/certificates/client-cert.pem`, `C:\Certificates\client-cert.pem` (Windows with OpenSSL), or `Cert:\LocalMachine\My location` (Windows with the Windows Secure Channel API).

AM_SSL_CA

When configuring the agent to validate AM's certificate, this environment variable specifies a PEM file that contains the certificates required to validate AM's server certificate. For example, `/opt/certificates/ca.pem`, `C:\Certificates\ca.pem` (Windows with OpenSSL), or or `Cert:\LocalMachine\Ca` (Windows with the Windows Secure Channel API).

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.

Command-Line Tool Reference

Name

agentadmin — manage web agent installation

Synopsis

```
agentadmin {options}
```

Description

This command manages web 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 the Apache Web Agent"
- "Installing the IIS Web Agent"
- "Installing the NGINX Plus Web 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 agent configuration and module.

When installing in Microsoft IIS, enter the ID number of the IIS site in which to install the web 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 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

(Apache web agent for Unix only) Use this option to set the ownership of created directories to the user and group as specified in the Apache configuration file.

Note that this option will only change the ownership of the files and directories if you run the **agentadmin** command as the **root** user or using the **sudo** command.

If you cannot run the **agentadmin** as the **root** user or using the **sudo** command, you must change the ownership manually.

--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 **--acceptLicense** 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 the Apache Web Agent Silently"
- "Installing IIS Web Agents Silently"
- "Installing NGINX Plus Web Agents Silently"

--n

(IIS web agent only) List the sites available in an IIS server.

Example:

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

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
```

--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
```


--g

(IIS web agent only) Remove all web agent instances and libraries from an IIS installation.

Usage: **agentadmin.exe --g**

For more information, see "To Remove Web Agents from IIS".

--e

(IIS web agent only) Enable an existing agent instance.

Usage: **agentadmin.exe --e *agent-instance***

For more information, see "To Disable and Enable Web Agents".

--d

(IIS web agent only) Disable an existing agent instance.

Usage: **agentadmin.exe --d *agent-instance***

For more information, see "To Disable and Enable Web Agents".

--o

(IIS web agent only) Modify Access Control Lists (ACLs) for files and folders related to a web agent instance.

Usage: **agentadmin.exe --o "*identity_or_siteID*" "*directory*" [--siteId]**

Usage: **agentadmin.exe --o "*directory*" --addAll --removeAll**

"*identity_or_siteID*"

Specifies the identity to be added to the directory's ACLs. When used with the `--siteId` option, it specifies an IIS site ID.

"*directory*"

Specifies the directory that would be modified.

[--siteId]

Specifies that the **agentadmin** should use `identity_or_siteID` as an IIS site ID.

--addAll

Add all IIS application pool identities to the directory's ACLs. This option is not compatible with the `--removeAll` option.

--removeAll

Remove all IIS application pool identities from the directory's ACLs. This option is not compatible with the **--addAll** option.

Examples:

```
C:\web_agents\iis_agent\bin>agentadmin.exe --o "IIS_user1" "C:\web_agents\iis_agent\lib"
```

```
C:\web_agents\iis_agent\bin>agentadmin.exe --o "2" "C:\web_agents\iis_agent\lib" --siteId
```

```
C:\web_agents\iis_agent\bin>agentadmin.exe --o "C:\web_agents\iis_agent\lib" --addAll
```

--r

Remove an existing agent instance.

Usage: **agentadmin --r agent-instance**

agent-instance

The ID of the web agent configuration instance to remove.

Respond **yes** when prompted to confirm removal.

On IIS web agents, the **--r** option does not remove the web agent libraries since they can be in use by other web agent instances configured on the same site. To remove all web agent instances and libraries, use the **--g** option instead.

For more information, see:

- "Removing the Apache Web Agent"
- "Enabling and Disabling IIS Web Agents"
- "Removing the NGINX Plus Web Agent"

--k

Generate a new signing key.

Usage: **agentadmin --k**

Examples:

- Unix:

```
$ cd /web_agents/apache24_agent/bin/  
$ ./agentadmin --k  
Encryption key value: YWM00ThLMTQtMzMxO0S05Nw==
```

- Windows:

```
C:\> cd web_agents\apache24_agent\bin
C:\web_agents\apache24_agent\bin> agentadmin --k
Encryption key value: YWM00ThLMTQtMzMx0S05Nw==
```

For more information, see Encryption Bootstrap 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 "YWM00ThLMTQtMzMx0S05Nw==" "`cat newpassword.file`"
Encrypted password value: 07bJ0SeM/G8yd04=
```

- Windows:

```
C:\path\to\web_agents\apache24_agent\bin>agentadmin.exe --p "YWM00ThLMTQtMzMx0S05Nw==" "newpassword"
Encrypted password value: 07bJ0SeM/G8yd04=
```

For more information, see Encryption Bootstrap Properties.

--V[i]

Validate an agent instance.

Usage:

agentadmin --V[i] *agent_instance* [user name] [password file] [realm]

[i]

(Optional) Ensures that the core init and shutdown agent sequences are working as expected.

Do not use this option while the agent is actively protecting a website. Doing so may make the agent instance unresponsive, causing unexpected service outages.

agent_instance

(Required) The agent instance where to run the validation tests. For example, `agent_1`.

user name

(Optional) A user ID that exists in the AM server. Required only for the `validate_session_profile` test. For example, `demo`.

password file

(Optional) A file containing the password of the user ID used for the `validate_session_profile` test. For example, `/tmp/passwd.txt`

realm

(Optional) The realm of the user ID used for the `validate_session_profile` test. For example, `/customers`

The validation mode performs the following tests:

- Ensures that the agent can reach the AM server(s) configured in the `com.sun.identity.agents.config.naming.url` property.
- Ensures that critical bootstrap properties are set. For more information, see "Configuration Location".
- Ensures that SSL libraries are available and that SSL configuration properties are set, if the agent is configured for SSL communication.
- Ensures the agent can log in to AM to fetch the agent profile.
- Ensures the system has enough RAM and shared memory.
- Ensures the agent can log in to AM with the provided user and password credentials.
- Ensures WebSocket connections are available between the agent and AM.
- Ensures that the core init and shutdown agent sequences are working as expected.

Note: This validation requires the `--vi` flag. *Do not use this option while the agent instance is actively protecting a website.* Doing so may make the agent instance unresponsive, causing unexpected service outages.

- (IIS agent only) Ensures that IIS is configured for running application pools in Integrated mode.

Important

On Unix, you should run the `agentadmin --V[i]` validator command as the same user that runs the web server.

For example, to use the Apache HTTP Server `daemon` user:

```
$ sudo -u daemon ./bin/agentadmin --V agent_1
```

Running the command as a different user may cause the `log/system_0.log` and `log/monitor_0.pipe` files to be created with permissions that prevent the agent from writing to them. In this case, you may see an error such as:

```
2018-09-19 13:54:52 GMT ERROR [0x7f0c9cf05700:22420]: unable to open event channel
```

If you have run the validator command as any other user, ensure the agent files have the correct permissions. For more information, review the installation instructions for your web agent in "*Installing Web Agents*".

Example:

```
$ ./agentadmin --Vi agent_1 demo passwd.txt /
```

```
Saving output to /web_agents/apache24_agent/bin/../../log/validate_20180831121402.log
```

```
Running configuration validation for agent_1:
```

```
Agent instance is configured with 1 naming.url value(s):  
1. https://openam.example.com:8443/openam is valid  
selected https://openam.example.com:8443/openam as naming.url value  
validate_bootstrap_configuration: ok  
validate_ssl_libraries: ok  
validate_agent_login: ok  
get_allocator_blockspace_sz(): trying for configured cache size 16777216 bytes  
validate_system_resources: ok  
validate_session_profile: ok  
validate_websocket_connection: ok  
validate_worker_init_shutdown: ok
```

```
Result: 7 out of 7 tests passed, 0 skipped.
```

--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: 5.8.2.1  
Revision: ab12cde  
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

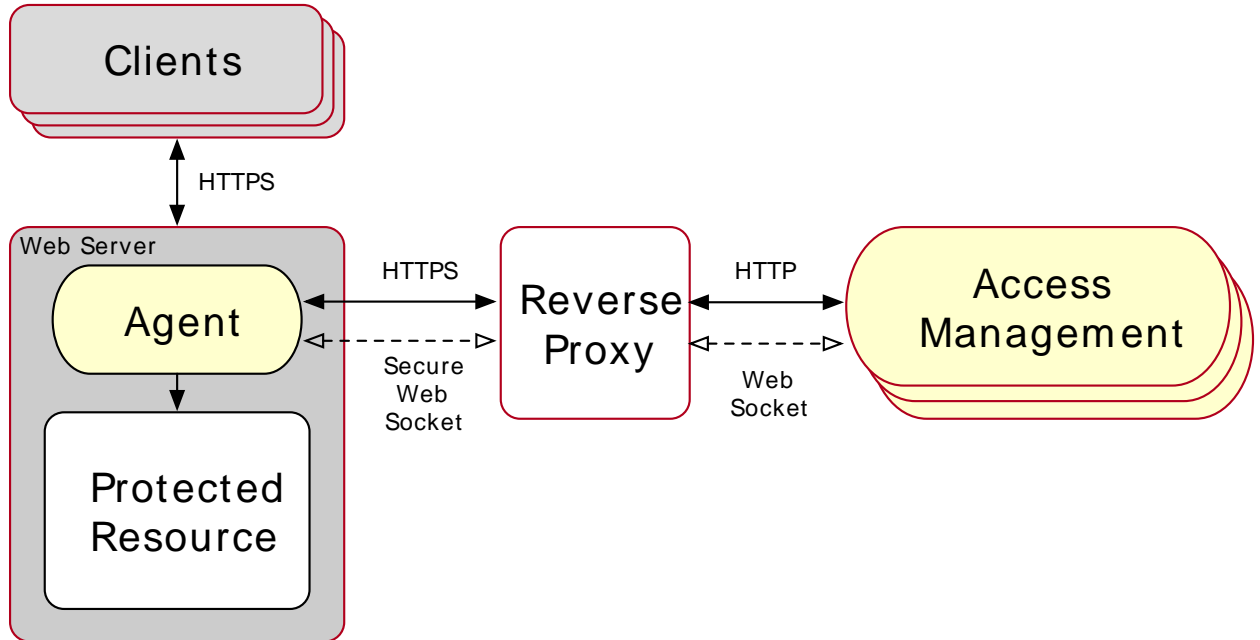
The **agentadmin** command returns **EXIT_SUCCESS** when an operation has completed successfully, and **EXIT_FAILURE** if the operation failed to complete.

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**.

Configuring Apache HTTP Server as a Reverse Proxy Example

This section demonstrates a possible configuration of Apache as a reverse proxy between AM and the agent, but you can use any reverse proxy that supports the WebSocket protocol.

Reverse Proxy Configured Between the Agent and AM



Note that the communication protocol changes from HTTPS to HTTP.

To Configure Apache as a Reverse Proxy Example

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 communication between AM and the agents.

3. Add the proxy configuration inside the `VirtualHost` context. Consider the following directives:

```
<VirtualHost 192.168.1.1>
...
# Proxy Config
RequestHeader set X-Forwarded-Proto "https" ❶
ProxyPass "/openam/notifications" "ws://openam.example.com:8080/openam/notifications"
  Upgrade=websocket ❷
ProxyPass "/openam" "http://openam.example.com:8080/openam" ❸
ProxyPassReverseCookieDomain "openam.internal.example.com" "proxy.example.com" ❹
ProxyPassReverse "/openam" "http://openam.example.com:8080/openam" ❺
...
</VirtualHost>
```

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 a future step 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 recover the forwarded header you configured in the reverse proxy. Also, review other configurations that may be required in an environment that uses reverse proxies. For more information, see "Regarding Communication Between AM and Agents"

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 identities 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 agent profile configuration information, typically created to delegate agent profile creation to the user installing a web or Java 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	<p>In general terms, a service exposing protected resources.</p> <p>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.</p>
Application type	<p>Application types act as templates for creating policy applications.</p> <p>Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.</p>

	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.
Authentication Session	The interval while the user or entity is authenticating to AM.
Session	The interval that starts after the user has authenticated 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 CTS-based sessions and Client-based sessions.
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.

Client-based OAuth 2.0 tokens	After a successful OAuth 2.0 grant flow, AM returns a token to the client. This differs from CTS-based OAuth 2.0 tokens, where AM returns a <i>reference</i> to token to the client.
Client-based sessions	<p>AM <i>sessions</i> for which AM returns session state to the client after each request, and require it to be passed in with the subsequent request. For browser-based clients, AM sets a cookie in the browser that contains the session information.</p> <p>For browser-based clients, AM sets a cookie in the browser that contains the session state. When the browser transmits the cookie back to AM, AM decodes the session state from the cookie.</p>
Conditions	<p>Defined as part of policies, these determine the circumstances under which which a policy applies.</p> <p>Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved.</p> <p>Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.</p>
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.
CTS-based OAuth 2.0 tokens	After a successful OAuth 2.0 grant flow, AM returns a <i>reference</i> to the token to the client, rather than the token itself. This differs from <i>client-based</i> OAuth 2.0 tokens, where AM returns the entire token to the client.
CTS-based sessions	AM <i>sessions</i> that reside in the Core Token Service's token store. CTS-based sessions might also be cached in memory on one or more AM servers. AM tracks these 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.
Delegation	Granting users administrative privileges with AM.
Entitlement	Decision that defines which resource names can and cannot be accessed for a given identity 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.
Identity store	Data storage service holding principals' profiles; underlying storage can be an LDAP directory service or a custom IdRepo implementation.
Java agent	Java web application installed in a web container that acts as a policy enforcement point, 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	Java, web, or custom 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	<p>Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities.</p> <p>When a Subject successfully authenticates, AM associates the Subject with the Principal.</p>
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified identities in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	<p>AM unit for organizing configuration and identity information.</p> <p>Realms can be used for example when different parts of an organization have different applications and identity stores, and when different organizations use the same AM deployment.</p> <p>Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.</p>
Resource	<p>Something a user can access over the network such as a web page.</p> <p>Defined as part of policies, these can include wildcards in order to match multiple actual resources.</p>
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 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 CTS-based sessions, 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	<p>Group of AM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability.</p> <p>The load balancer can also be used to protect AM services.</p>
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateless Service	<p>Stateless services do not store any data locally to the service. When the service requires data to perform any action, it requests it from a data store. For example, a stateless authentication service stores session state for logged-in users in a database. This way, any server in the deployment can recover the session from the database and service requests for any user.</p> <p>All AM services are stateless unless otherwise specified. See also Client-based sessions and CTS-based sessions.</p>
Subject	<p>Entity that requests access to a resource</p> <p>When an identity successfully authenticates, AM associates the identity with the Principal that distinguishes it from other identities. An identity can be associated with multiple principals.</p>
Web Agent	Native library installed in a web server that acts as a policy enforcement point with policies based on web page URLs.