Web Agents

June 30, 2025

WEB AGENTS
Version: 2023.11

Table of Contents

Installation

This guide describes how to install ForgeRock Access Management Web Agents Documentation.

About ForgeRock Identity Platform™ Software

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

Example installation for this guide

Unless otherwise stated, the examples in this guide assume the following installation:

- Web Agents Documentation installed on `http://agent.example.com:80`.
- Access Management installed on `http://am.example.com:8088/am`.
- Work in the top-level realm `/`.

If you use a different configuration, substitute in the procedures accordingly.

Prepare for installation

Before you install

Consider the following points before you install:

- Install AM and Web Agents Documentation in different servers.
- Make sure AM is running, so that you can contact AM from the agent web server.
- Install the web server before you install the agent.
- Make sure OpenSSL or the Windows Secure Channel API (Schannel) is available before installing the agent. Unix-based agents support OpenSSL. Windows-based agents support OpenSSL and Schannel.

  Learn more about SSL requirements in SSL requirements and required OpenSSL libraries in OpenSSL libraries.
- Install only one Web Agents Documentation for each web server and configure as many agent instances as necessary.
- For environments with load balancers or reverse proxies, consider the communication between the agent and the AM servers, and between the agent and the client. Configure both AM and the environment **before** you install the agent.

  Learn more in Configure load balancers and reverse proxies.

OpenSSL libraries

Before installing, make sure the OpenSSL libraries are located or referenced as follows:

Operating System	OpenSSL Library	Location or Variable
Windows 32-bit	• `libeay32.dll` • `ssleay32.dll` • `libcrypto-1_1.dll` (1) • `libssl-1_1.dll` (1)	`\windows\syswow64`
Windows 64-bit(2)	• `libeay64.dll` • `ssleay64.dll` • `libcrypto-1_1-x64.dll` (1) • `libssl-1_1.dll` (1)	`\windows\system32`
Linux	• `libcrypto.so` • `libssl.so`	`$LD_LIBRARY_PATH` `$LD_LIBRARY_PATH_64`
AIX	• `libcrypto.so` • `libssl.so`	`$LIBPATH`

(1) OpenSSL 1.1.0+ only
(2) Windows 64-bit servers require both 32-bit and 64-bit OpenSSL libraries.

Download and unzip Web Agents Documentation

Go to the Backstage download site and download an agent based on your architecture, and operating system requirements. 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 agent configuration and log files. The following directories are extracted:

Installation directories

Directory	Description
`bin/`	The installation and configuration program `agentadmin`.
`config/`	Configuration templates used by the `agentadmin` command during installation.

Directory	Description
`instances/`	Configuration files, and audit and debug logs for individual instances of the agents. The directory is empty when first extracted. **Important** Agent configuration files are created in `/path/to/web_agents/agent_type/instances/agent_n/config/agent.conf`. Make sure the path, including the parent path, does not exceed 260 characters.
`legal/`	Licensing information including third-party licenses.
`lib/`	Shared libraries used by the agent.
`log/`	Log files written during installation. The directory is empty when first extracted. When the agent is running, the directory can contain the following files: • The `system_n.log` file, where the agent logs information related to agent tasks running in the background. Web Agents Documentation timestamps events in coordinated universal time (UTC). • (IIS Web Agents Documentation only) The backup of the site and application configuration files created after running the `agentadmin -g` command. • (IIS Web Agents Documentation only) Files related to the agent caches.
`pdp-cache/`	POST data preservation cache. The agent stores POST data preservation files temporarily. To change the directory, configure POST Data Storage Directory.

Pre-installation tasks

1. In AM, add an agent profile, as described in Create an agent profile in AM using the console:

   The example in this guide uses an agent profile in the top-level realm, with the following values:

   - **Agent ID**: `web-agent`
   - **Agent URL**: `http://www.example.com:80`
   - **Server URL**: `http://am.example.com:8080/am`
   - **Password**: `password`

2. In AM, add a policy set and policy, to protect resources with the agent, as described in Policies in AM's *Authorization guide*.

   The example in this guide uses a policy set and policy in the top-level realm, with the following values:

   - **Policy set**:
     - **Name**: `PEP`
     - **Resource Types**: `URL`

- **Policy**:
  - **Name**: `PEP-policy`
  - **Resource Type**: `URL`
  - **Resource pattern**: `*://*:*/*`
  - **Resource value**: `*://*:*/*`
  - **Actions** tab: Allow HTTP `GET` and `POST`
  - **Subjects** tab: All Authenticated Users.

> **Tip**
>
> When you use your own policy set instead of the default policy set, `iPlanetAMWebAgentService`, update the following properties in the agent profile:
> - Policy Set
> - Policy Evaluation Realm

3. Configure AM to protect the CDSSO cookie from hijacking. For more information, refer to Enabling restricted tokens for CDSSO session cookies in AM's *Security guide*.
4. Create a text file for the agent password, and protect it. For example, use commands similar to these, but use a strong password and store it in a secure place:

Unix

```
$ cat > /secure-directory/pwd.txt
password
CTRL+D

$ chmod 400 /secure-directory/pwd.txt
```

Windows

```
'password' | Out-File -Encoding ascii pwd.txt
```

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

> **Tip**
>
> Although the agent accepts any password length and content, you are strongly encouraged to generate secure passwords. This can be achieved in various ways, for example, by using a password manager.

5. If either of the following are true, set up the required environment variables :
    - AM is configured to perform client authentication
    - The agent web server is to configured to validate AM's server certificate

    For more information, refer to Environment variables.

Configure AM to sign authentication information

AM communicates all authentication and authorization information to Web Agents Documentation, using OpenID Connect ID tokens. For security, configure AM and the agent to use signed tokens. For more information, refer to RFC 7518: JSON Web Algorithms (JWA).

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 one of the following procedures to create new key aliases and configure them in AM.

Configure AM secret IDs for the agents' OAuth 2.0 provider

By default, AM 6.5 and later versions are 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 `rsajwtsigningkey` 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 `hmacsigningtest` key alias available in AM's JCEKS keystore.

    1. Create the following aliases in one of the secret stores configured in AM, for example, the default JCEKS keystore:
        1. Create an RSA key pair.
        2. Create an HMAC secret.
    2. In the AM admin UI, go to **Configure** > **Secret Stores** > Keystore Secret Store Name > **Mappings**.
    3. Configure the following secret IDs:
        1. Configure the new RSA key alias in the `am.global.services.oauth2.oidc.agent.idtoken.signing` secret ID.
        2. 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, because it is also used for signing certain OpenID Connect ID tokens and remote consent requests. For more information, refer to Secret ID default mappings in AM's *Security guide*.
        3. Save your changes.

    For more information about secret stores, refer to Secret stores in AM's *Security guide*.

No further configuration is required in the agents.

Create agent profiles

Use Web Agents Documentation profiles to connect to and communicate with AM.

Create an agent profile for a single agent instance

This section describes how to create an agent profile in the AM admin UI. Alternatively, create agent profiles by using the `/realm-config/agents/WebAgent/{id}` endpoint in the REST API. For more information, refer to REST API explorer in AM's *Getting started with REST*.

1. In the AM admin UI, select **Realms** > Realm Name > **Applications** > **Agents** > **Web**, and add an agent using the following hints:

Agent ID

The ID of the agent profile. This ID resembles a username in AM and is used during the agent installation. For example, `MyAgent`.

> **Tip**
>
> When AM is not available, the related error message contains the agent profile name. Consider this in your choice of agent profile name.

Agent URL

The URL where the agent resides. For more information, refer to Example installation for this guide.

In centralized configuration mode, the Agent URL populates the agent profile for services, such as notifications.

Server URL

The full URL to an authorization server, such as Identity Cloud or AM. For more information, refer to Example installation for this guide.

If the authorization server is deployed in a site configuration (behind a load balancer), enter the site URL.

In centralized configuration mode, the Server URL populates the agent profile for use with login, logout, naming, and cross-domain SSO.

Password

The password the agent uses to authenticate to an authorization server, such as Identity Cloud or AM. Use this password when installing an agent.

> **Tip**
>
> Although the agent accepts any password length and content, you are strongly encouraged to generate secure passwords. This can be achieved in various ways, for example, by using a password manager.

Create an agent profile for multiple agent instances when post data preservation is enabled

By default, the POST data preservation load balancer cookie name and value is set by the agent profile. Therefore, each agent instance behind a load balancer requires its own agent profile.

In scalable environments, such as deployments with load balancing, or environments that run Kubernetes, resources are dynamically created and destroyed.

To facilitate the rapid creation and destruction of agent instances when post data preservation is enabled, set the POST data preservation configuration in `agent.conf` to map one agent profile to multiple agent instances.

The configuration in `agent.conf` overrides the configuration in AM for the following properties:

- POST Data Sticky Load Balancing Mode
- POST Data Sticky Load Balancing Value

For an example, refer to Map one agent profile to multiple agent instances when POST data preservation is enabled.

Create an agent profile group

Use agent profile groups when you set up multiple agents, and want to inherit settings from the group.

1. In the AM admin UI, go to **Realms** > Realm Name > **Applications** > **Agents** > **Web**.
2. Select the **Groups** tab, and add a group with the following settings:
    - **Group ID**: A name for the profile group.
    - **Server URL**: The URL of the AM server in which to store the profile.

Inherit properties from an agent profile group

1. Set up an agent profile and agent profile group, as described in Create an agent profile for a single agent instance and Create an agent profile group.
2. In the AM admin UI, select your agent profile.
3. On the **Global** tab, select **Group**, and select a group from the drop-down menu. The agent profile is added to the group.
4. For each setting in the **Global** tab, select or deselect the 🔒 icon:
    - 🔒: Inherit this setting from the group
    - 🔓: Do not inherit this setting from the group

Authenticate agents to the identity provider

Authenticate agents to Identity Cloud

> **Important**
>
> Web Agents Documentation is automatically authenticated to Identity Cloud by a non-configurable authentication module. Authentication chains and modules are deprecated in Identity Cloud and replaced by journeys.
> You can now authenticate Web Agents Documentation to Identity Cloud with a journey. The procedure is currently optional, but will be required when authentication chains and modules are removed in a future release of Identity Cloud.
> For more information, refer to Identity Cloud's Journeys.

This section describes how to create a journey to authenticate Web Agents Documentation to Identity Cloud. The journey has the following requirements:

- It must be called `Agent`
- Its nodes must pass the agent credentials to the Agent Data Store Decision node.

When you define a journey in Identity Cloud, that same journey is used for all instances of Identity Gateway, Java Agent, and Web Agent. Consider this point if you change the journey configuration.

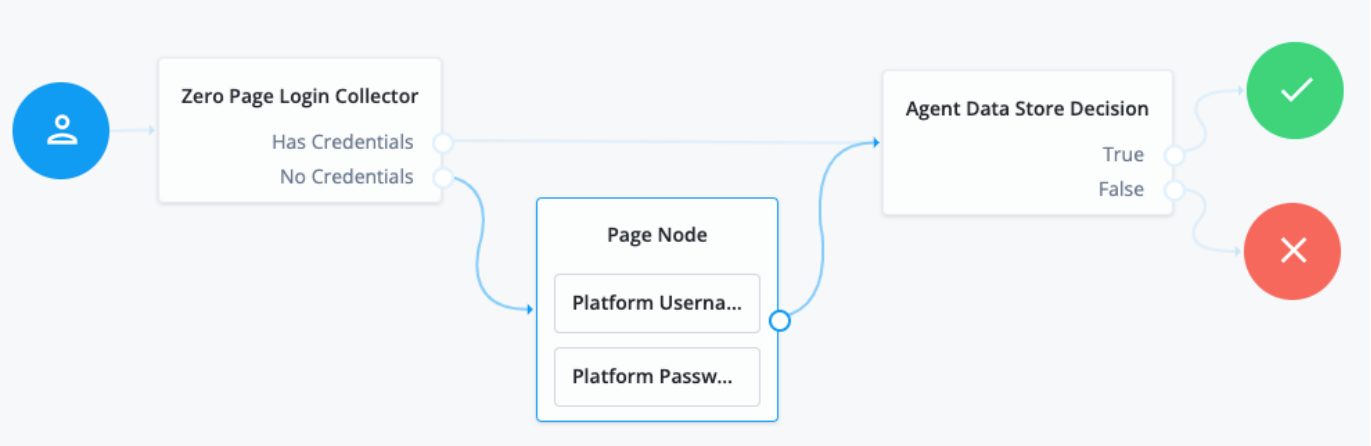
1. Log in to the Identity Cloud admin UI as an administrator.
2. Click **Journeys** > **New Journey**.
3. Add a journey with the following information and click **Create journey**:
   - **Name**: `Agent`
   - **Identity Object**: The users to authenticate.

     This must be a user type object, such as `managed/alpha_user`.
   - (Optional) **Description**: Authenticate an agent to Identity Cloud

   The journey designer is displayed, with the `Start` entry point connected to the `Failure` exit point, and a `Success` node.
4. Using the **Filter nodes** bar, find and then drag the following nodes from the **Components** panel into the designer area:
   - Zero Page Login Collector node to check whether the agent credentials are provided in the incoming authentication request and use their values in the following nodes.

     This node is required for compatibility with Java agent and Web agent.
   - Page node to collect the agent credentials if they are not provided in the incoming authentication request and use their values in the following nodes.
   - Agent Data Store Decision node to verify that the agent credentials match the registered Web Agents Documentation agent profile.

   > **Important**
   >
   > Many nodes can be configured in the panel on the right side of the page. Unless otherwise stated, do not configure the nodes and use only the default values.
5. Drag the following nodes from the **Components** panel into the Page node:
   - Platform Username node
   - Platform Password node
6. Connect the nodes as follows and save the journey:



Authenticate agents to AM

Important

From AM 7.3

When AM 7.3 is installed with a default configuration, as described in Evaluation, Web Agents Documentation is automatically authenticated to AM by an authentication tree. Otherwise, Web Agents Documentation is authenticated to AM by an AM authentication module.
Authentication chains and modules were deprecated in AM 7. When they are removed in a future release of AM, it will be necessary to configure an appropriate authentication tree when you are not using the default configuration.

For more information, refer to AM's Authentication Nodes and Trees.

This section describes how to create an authentication tree to authenticate Web Agents Documentation to AM. The tree has the following requirements:

- It must be called `Agent`
- Its nodes must pass the agent credentials to the Agent Data Store Decision node.

When you define a tree in AM, that same tree is used for all instances of Identity Gateway, Java Agent, and Web Agent. Consider this point if you change the tree configuration.

1. On the **Realms** page of the AM admin UI, choose the realm in which to create the authentication tree.
2. On the **Realm Overview** page, click **Authentication** > **Trees** > **Create tree**.
3. Create a tree named `Agent`.

   The authentication tree designer is displayed, with the `Start` entry point connected to the `Failure` exit point, and a `Success` node.

   The authentication tree designer provides the following features on the toolbar:

Button	Usage
	Lay out and align nodes according to the order they are connected.

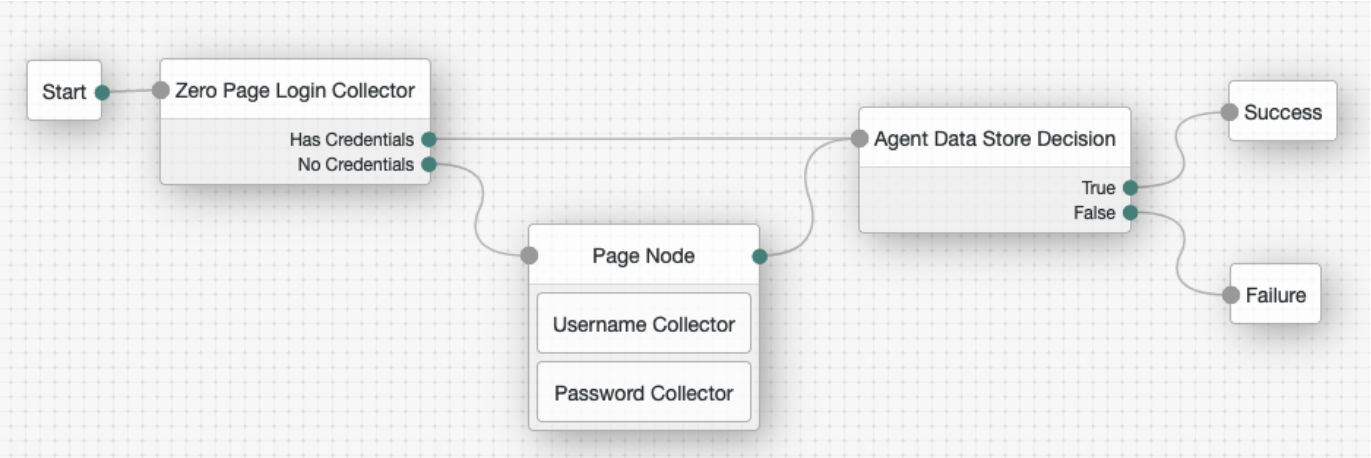
Button	Usage
	Toggle the designer window between normal and full-screen layout.
	Remove the selected node. Note that the `Start` entry point cannot be deleted.

4. Using the **Filter** bar, find and then drag the following nodes from the **Components** panel into the designer area:
   - Zero Page Login Collector node to check whether the agent credentials are provided in the incoming authentication request and use their values in the following nodes.

     This node is required for compatibility with Java agent and Web agent.
   - Page node to collect the agent credentials if they are not provided in the incoming authentication request and use their values in the following nodes.
   - Agent Data Store Decision node to verify that the agent credentials match the registered Web Agents Documentation profile.

> **Important**
>
> Many nodes can be configured in the panel on the right side of the page. Unless otherwise stated, do not configure the nodes and use only the default values.

5. Drag the following nodes from the **Components** panel into the Page node:
   - Username Collector node, to prompt the user to enter their username
   - Password Collector node,to prompt the user to enter their password
6. Connect the nodes as follows and save the tree:



Install Apache or IBM HTTP Web Agents Documentation

Consider the following points before installing Apache or IBM HTTP Web Agents Documentation:

- SELinux can prevent the web server from accessing agent libraries, and the agent from being able to write to audit and debug logs. For more information, refer to Troubleshooting.
- By default, 32 agent instances can run at the same time in a single installation. For information about changing the limit, refer to *AM_MAX_AGENTS* in Environment variables.
- (For Apache Web Agents Documentation) By default, the agent replaces authentication functionality provided by Apache, for example, the `mod_auth_*` modules. Configure Use Built-in Apache HTTPD Authentication Directives to use built-in Apache authentication directives such as `AuthName`, `FilesMatch`, and `Require` for specified not-enforced URLs.

Tune multi-processing modules

Apache and IBM HTTP server include Multi-Processing Modules (MPMs) that extend the functionality of a web server to support a wide variety of operating systems and customizations for a site.

Before installation, configure and tune MPMs, as follows:

- Configure one of the following modules:
  - `mpm-event` for Unix-based servers
  - `mpm-worker` for Unix-based servers
  - `mpm_winnt` for Windows servers

  The `prefork-mpm` module isn't adapted to high-traffic deployments. It can cause performance issues to both the agent and AM.
- Make sure that there are enough processes and threads available to service the expected number of client requests.

  MPM-related performance is configured in the file `conf/extra/http-mpm.conf`:

```
<IfModule mpm_worker_module>
StartServers            2
MaxRequestWorkers     150
MinSpareThreads        25
MaxSpareThreads        75
ThreadsPerChild        25
MaxConnectionsPerChild  0
</IfModule>
```

  `MaxRequestWorkers` and `ThreadsPerChild` control the maximum number of concurrent requests. The default configuration allows 150 concurrent clients across 6 processes of 25 threads each.

  Configure `MaxRequestWorkers` and `ServerLimit` to get a high level of concurrent clients.

  To prevent problems registering the notification queue listener, don't change the default value of `MaxSpareThreads`, `ThreadLimit`, or `ThreadsPerChild`.

For information about Apache configuration properties, refer to Apache MPM worker.

Install interactively

1. Review the information in Before you install and complete the Preinstallation tasks.
2. (Optional) In environments where a user isn't defined in the Apache or IBM HTTP server configuration file `httpd.conf`, set the following environment variables in your command line session to change ownership of created directories.

   The following examples change ownership to the user `user`:

   ```
   $ export APACHE_RUN_USER=user
   $ export APACHE_RUN_GROUP=user
   ```

   For more information, refer to Installation environment variables
3. Shut down the Apache or IBM HTTP server where you plan to install the agent.
4. Make sure AM is running.
5. Run `agentadmin --i` to install the agent:

   Apache on Linux

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

   Apache on Windows

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

   IBM HTTP Server on Linux

   ```
   $ cd /web_agents/httpservern_agent/bin/
   $ ./agentadmin --i
   ```
6. When prompted, enter information for your deployment:

> **Tip**
>
> To cancel the installation at any time, press `CTRL-C`.

1. Enter the complete path to the Apache or IBM HTTP server configuration file:

Apache on Linux

```
Configuration file [/opt/apache/conf/httpd.conf]:/etc/httpd/conf/httpd.conf
```

Apache on Windows

```
Configuration file [/opt/apache/conf/httpd.conf]:/etc/httpd/conf/httpd.conf
```

IBM HTTP Server on Linux

```
Configuration file [/opt/apache/conf/httpd.conf]: /opt/IBM/HTTPServer/conf/httpd.conf
```

2. (Optional) When installing the agent as the root user, consider changing directory ownership to the same user and group specified in the server configuration:

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

This step appears only if environment variables are set as described in step 2, and `User` and `Group` are not defined in `httpd.conf`, such as in non Red Hat Enterprise Linux-based distributions.

> **Tip**
>
> See which user or group is running the server by viewing the `Group` and `User` directives in `httpd.conf`.

The following errors can occur when the permissions are wrong:

- Server fails to start up
- Requests to a protected resource return a blank page
- Log rotation errors

3. 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: /config/agent.conf
```

The installer can import settings from an existing agent on the new installation and skip prompts for values present in the existing configuration file. You must re-enter the agent profile password.

4. Enter the full URL for the AM instance that the agent will use, including the deployment URI:

```
AM server URL: http://am.example.com:8080/am
```

> **Note**
>
> If a reverse proxy is configured between AM and the agent, set the AM URL to the proxy URL, for example, `https://proxy.example.com:443/am`. For information about setting up an environment for reverse proxies, refer to Apache as a reverse proxy.

5. Enter the full URL of the agent:

```
Agent URL: http://www.example.com:80
```

6. Enter the name of the agent profile created in AM:

```
Agent Profile name: web-agent
```

7. Enter the agent profile realm:

```
Agent realm/organization name: [/]:  /
```

> **Note**
>
> Realms are case-sensitive.

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

```
The path to the password file: /secure-directory/pwd.txt
```

9. Review the configuration:

```
Installation parameters:
   AM URL: http://am.example.com:8080/am
   Agent URL: http://www.example.com:80
   Agent Profile name: web-agent
   Agent realm/organization name: /
   Agent Profile password source: /secure-directory/pwd.txt

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

10. Accept or update the configuration:
    - To accept the configuration type `yes`.
    - To change the configuration 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.

    On successful completion, the installer adds the agent as a module to the server configuration file `httpd.conf`. The agent adds a backup configuration file with the installation datestamp: `http.conf_amagent_yyyymmddhhmmss`.

7. (Unix only) Make sure the user or group running the Apache or IBM HTTP server has appropriate permissions for the following directories:

Apache on Linux

```
Read permission:
* /web_agents/apache24_agent/lib

Read and write permission:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log

Execute permission to validate an installation by using the agentadmin --V[i\] command:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log
```

Apache on Windows

```
Read permission:
* /web_agents/apache24_agent/lib

Read and write permission:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log

Execute permission to validate an installation by using the agentadmin --V[i\] command:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log
```

IBM HTTP Server on Linux

```
Read permission:
* /web_agents/httpservern_agent/lib

Read and write permission:
* /web_agents/httpservern_agent/instances/agent_n
* /web_agents/httpservern_agent/log

Execute permission to validate an installation by using the agentadmin --V[i\] command:
* /web_agents/httpservern_agent/instances/agent_n
* /web_agents/httpservern_agent/log
```

Tip

See which user or group is running the server by viewing the `Group` and `User` directives in `httpd.conf`.

The following errors can occur when the permissions are wrong:

- Server fails to start up
- Requests to a protected resource return a blank page
- Log rotation errors

Note

The same issues can occur if SELinux is enabled in `enforcing` mode, and not configured to allow access to agent directories. For more information, refer to Troubleshooting.

8. Start the Apache or IBM HTTP server.
9. Check the installation, as described in Check the installation.

Install on a virtual host

Web Agents Documentation instances can operate with multiple virtual hosts. 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.

Installing on a virtual host is a manual process that involves copying an instance directory created by the `agentadmin` installer and adding it to the configuration file of the virtual host.

1. Install an agent in the default root configuration, as described in Install Apache or IBM HTTP Web Agents Documentation. This agent is referred to as the *root agent*.
2. Create a profile for the agent on the virtual host, as described in Creating agent profiles. This agent is referred to as the *virtual host agent*.
3. Create at least one AM policy to protect resources on the virtual host, as described in Policies in AM's *Authorization guide*.
4. Shut down the Apache or IBM HTTP server where you plan to install the agent.
5. Locate an agent configuration instance to duplicate, and make a copy. For example, copy `agent_1` to `agent_2`:

Apache on Linux

```
$ cd /web_agents/apache24_agent/instances
$ cp -r agent_1 agent_2
```

Apache on Windows

```
c:\> cd c:\web_agents\apache24_agent\instances
c:\path\to\web_agents\apache24_agent\instances> xcopy /E /I agent_1 agent_2
```

IBM HTTP Server on Linux

```
$ cd /web_agents/httpservern_agent/instances
$ cp -r agent_1 agent_2
```

6. Assign modify privileges to the new instance folder for the user that runs the virtual host. The following examples assign privileges for `agent_2` to a user named *user*:

Apache on Linux

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

Apache on Windows

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

IBM HTTP Server on Linux

```
$ cd /web_agents/httpservern_agent/instances
$ chown -hR user agent_2
```

7. In the new instance folder, edit the `agent.conf` file as follows:
    1. Set the value of Agent Profile Name to the name of the profile you created for the virtual host agent. For example, set the value to `agent_2`.
    2. Configure the encryption key and password for the virtual host agent, using a scenario that suits your environment:
        - Scenario 1: The password of the virtual host agent profile is the same as the password of the root agent profile[1].

          The encryption key and encryption password of the root agent and virtual host agent must match. Because you copied the configuration file, you don't need to do anything else.
        - Scenario 2: The password of the virtual host agent profile is different from the password of the root agent profile[2].

          Follow these steps to generate a new encryption key, encrypt the new password, and configure them in the profile of the virtual host agent:
            1. Generate a new encryption key:

               ```
               $ agentadmin --k
               Encryption key value: YWM…5Nw==
               ```
            2. (Unix only) Store the agent profile password in a file, for example, `newpassword.file`.
            3. Encrypt the agent profile password:

Apache on Linux

```
$ ./agentadmin --p "YWM…5Nw==" “cat newpassword.file”
Encrypted password value: 07b…dO4=
```

Apache on Windows

```
$ agentadmin.exe --p "YWM…5Nw==" "newpassword"
Encrypted password value: 07b…dO4=
```

IBM HTTP Server on Linux

```
$ ./agentadmin --p "YWM…5Nw==" “cat newpassword.file”
Encrypted password value: 07b…dO4=
```

4. Set the following properties in `agent.conf`:
   - Agent Profile Password Encryption Key with the value of the generated encryption key:

     ```
     com.sun.identity.agents.config.key = YWM...5Nw==
     ```

   - Agent Profile Password with the value of the encrypted password:

     ```
     com.sun.identity.agents.config.password = 07b...dO4=
     ```

3. Throughout the configuration, replace references to the original instance directory with the new instance directory. For example, replace `agent_1` with `agent_2` in the following properties:
   - Local Agent Debug File Name
   - Local Agent Audit File Name
4. Throughout the configuration, replace 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` in the following properties:
   - Agent Deployment URI Prefix
   - FQDN Default

8. Edit the Apache or IBM HTTP server configuration file, `httpd.conf`:

    1. Find the following lines at the end of the file. The following example is for Apache agent on Linux, but you can adapt it to your configuration:

        ```
        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
        ```

    2. 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>
        ```

    3. 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.

9. Save and close the configuration file.

10. (Unix only) Make sure the user or group running the Apache or IBM HTTP server has appropriate permissions for the following directories:

Apache on Linux

```
Read permission:
* /web_agents/apache24_agent/lib

Read and write permission:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log

Execute permission to validate an installation by using the agentadmin --V[i\] command:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log
```

Apache on Windows

```
Read permission:
* /web_agents/apache24_agent/lib

Read and write permission:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log

Execute permission to validate an installation by using the agentadmin --V[i\] command:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log
```

IBM HTTP Server on Linux

```
Read permission:
* /web_agents/httpservern_agent/lib

Read and write permission:
* /web_agents/httpservern_agent/instances/agent_n
* /web_agents/httpservern_agent/log

Execute permission to validate an installation by using the agentadmin --V[i\] command:
* /web_agents/httpservern_agent/instances/agent_n
* /web_agents/httpservern_agent/log
```

Tip

See which user or group is running the server by viewing the `Group` and `User` directives in `httpd.conf`.

The following errors can occur when the permissions are wrong:

- Server fails to start up
- Requests to a protected resource return a blank page
- Log rotation errors

> **Note**
>
> The same issues can occur if SELinux is enabled in `enforcing` mode, and not configured to allow access to agent directories. For more information, refer to Troubleshooting.

11. Start the Apache or IBM HTTP server.
12. Check the installation, as described in Check the installation.

Install silently

Use the `agentadmin --s` command for silent installation. For information about the options, refer to agentadmin command.

1. Review the information in Before you install and complete the Preinstallation tasks.
2. Shut down the Apache or IBM HTTP server where you plan to install the agent.
3. Make sure AM is running.
4. Run the `agentadmin --s` command with the required arguments. The following example is for Apache agent on Linux, but you can adapt it to your configuration:

```
$ sudo agentadmin --s \
  "/etc/httpd/conf/httpd.conf" \
  "http://am.example.com:8080/am" \
  "http://www.example.com:80" \
  "/" \
  "webagent" \
  "/secure-directory/pwd.txt" \
  --changeOwner \
  --acceptLicence
Web Agents Documentation for Apache Server installation.

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

5. (Unix only) Make sure the user or group running the Apache or IBM HTTP server has appropriate permissions for the following directories:

Apache on Linux

```
Read permission:
* /web_agents/apache24_agent/lib

Read and write permission:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log

Execute permission to validate an installation by using the agentadmin --V[i\] command:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log
```

Apache on Windows

```
Read permission:
* /web_agents/apache24_agent/lib

Read and write permission:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log

Execute permission to validate an installation by using the agentadmin --V[i\] command:
* /web_agents/apache24_agent/instances/agent_n
* /web_agents/apache24_agent/log
```

IBM HTTP Server on Linux

```
Read permission:
* /web_agents/httpservern_agent/lib

Read and write permission:
* /web_agents/httpservern_agent/instances/agent_n
* /web_agents/httpservern_agent/log

Execute permission to validate an installation by using the agentadmin --V[i\] command:
* /web_agents/httpservern_agent/instances/agent_n
* /web_agents/httpservern_agent/log
```

Tip

See which user or group is running the server by viewing the `Group` and `User` directives in `httpd.conf`.

The following errors can occur when the permissions are wrong:

- Server fails to start up
- Requests to a protected resource return a blank page
- Log rotation errors

> **Note**
>
> The same issues can occur if SELinux is enabled in `enforcing` mode, and not configured to allow access to agent directories. For more information, refer to Troubleshooting.

6. Start the Apache or IBM HTTP server.
7. Check the installation, as described in Check the installation.

Check the installation

1. After you start Apache or IBM HTTP server, check the error log to make sure startup was successful:

```
[Tue Sep …] AH00163:
Apache/2.4.6 (CentOS) Web Agents Documentation/2023.11 configured — resuming normal operations
```

2. Make an HTTP request to a resource protected by the agent, then check the `/log/system_0.log` file to verify that no errors occurred on startup. The log should contain a message similar to this:

```
[0x7fb89e7a6700:22]: Web Agents Documentation Version: 2023.11
Revision: ab12cde, Container: Apache 2.4 Linux 64bit (Centos6),
Build date: Mar …
```

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

Install in a subrealm

Examples in this document install the agent in the top-level realm. To install the agent in a subrealm during interactive or silent installation, use the subrealm during the installation or in the response file.

For example, instead of:

```
Agent realm/organization name: [/]: /
```

specify:

```
Agent realm/organization name: [/]: /myrealm
```

Even though the agent is installed in a subrealm, the default login redirect requires the user realm to be the top-level realm. For information about how to change the user realm, refer to Login redirect.

Configure error logs

Edit the server configuration file `httpd.conf` to log errors.

The following line, present by default in `httpd.conf`, logs warning conditions for the container:

```
LogLevel warn
```

The following example line includes the agent error logs at debug-level:

```
LogLevel warn amagent:debug
```

1. The root agent profile refers to the agent installation performed in Install Apache or IBM HTTP Web Agents Documentation and required for installation on virtual hosts.
2. The root agent profile refers to the agent installation performed in Install Apache or IBM HTTP Web Agents Documentation and required for installation on virtual hosts.

Configure Apache or IBM HTTP Web Agents Documentation

The examples in this section are for Apache agent on Linux, but you can adapt them to your configuration.

> **Important**
>
> IBM HTTP server 9 supports Apache directives; IBM HTTP server 8,5 does not.

`AmAgent` directive to switch the agent on or off

Switch the agent on or off globally or independently for different server locations. Server locations include the global environment, a virtual host, a specific location, or a set of directory blocks. Use the following settings:

`AmAgent On`

The agent protects server locations. It allows or denies requests based on AM policy configuration and not-enforced rules.

`AmAgent Off`

Apache or IBM HTTP server protects server locations; the agent plays no part in protecting the server locations.

Default: `AmAgent` is set to `On` at a global level in the `httpd.conf` configuration file as follows:

```
AmAgent On
AmAgentConf /opt/web_agents/apache24_agent/instances/agent_1/config/agent.conf
AmAuthProvider Off
```

The `AmAgent` configuration is hierarchical; when it is `On` or `Off` globally it is set for all server locations except those explicitly specified otherwise.

Tip

Consider setting `AmAgent` to `Off` for the following situations:

- For server locations that need no AM authentication or policy, such as the public face of a website, or `/css` or `/images` directories.
- When Apache or IBM HTTP server is acting as a reverse proxy to AM or Identity Cloud, and you don't want the agent to take part in protecting AM or Identity Cloud.

Example where `AmAgent` is `On` globally and `Off` for specific directories

In the following example `httpd.conf`, the agent is `On` globally and `Off` for the `/var/www/transaction` directory:

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

<Directory /var/www/transaction>
    AmAgent Off
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

AmAgent On
AmAgentConf /opt/web_agents/apache24_agent/instances/agent_1/config/agent.conf
AmAuthProvider Off
```

Accessing a resource in `/var/www/`

The agent protects the resource, and overrides the `Require all granted` directive.

To access the resource, the request must match a not-enforced rule in the agent configuration or be allowed by an AM policy evaluation.

Accessing a resource in `/var/www/transaction`

Apache or IBM HTTP server manages the access and applies the `Require all granted` directive. The agent plays no part in protecting the resource.

`AmAgent` is `Off` globally and `On` for specific server locations

Important

When `AmAgent` configuration is `Off`, configure the server location `/agent` as `On`. This allows AM to redirect requests to the `/agent` endpoint after authentication.

In the following example `httpd.conf`, the agent is `Off` globally but `On` for the `/var/www/transaction` and `/agent` locations:

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

<Directory /var/www/transaction>
    AmAgent On
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

<Location /agent>
    AmAgent On
</Location>

AmAgent Off
AmAgentConf /opt/web_agents/apache24_agent/instances/agent_1/config/agent.conf
AmAuthProvider Off
```

Accessing a resource in `/var/www/`

Apache or IBM HTTP server manages the access and applies the `Require all granted` directive.The agent plays no part in protecting the resource.

Accessing a resource in `/var/www/transaction`

The agent protects the resource, and overrides the `Require all granted` directive.

To access the resource, the request must match a not-enforced rule in the agent configuration or be allowed by an AM policy evaluation.

`AmAuthProvider` directive to use Apache as the enforcement point

When `AmAgent` is `On`, combine AM policy with Apache `Require` directives to control access globally or independently for different server locations. Server locations include the global environment, a virtual host, a specific location, or a set of directory blocks.

> **Caution**
>
> Using multiple authorization sources increases complexity. To reduce the risk of an invalid security configuration, test and validate the directives.

Use the following settings:

`AmAuthProvider Off`

The agent acts as the enforcement point, allowing or denying requests based on not-enforced rules and AM policies.

`AmAuthProvider On`

Apache or IBM HTTP server acts as the enforcement point, allowing or denying requests based on AM policy and Apache `Require` directives

For information about `Require` directives, refer to Require Directive on the Apache website. `Require AmAuth` is a directive specifically for Web Agents Documentation. When the directive is specified, users must be authenticated with AM. Otherwise, the agent redirects them to AM for authentication.

Default: `AmAuthProvider` is `Off`

The `AmAuthProvider` configuration is hierarchical; when it is `On` or `Off` globally it is set for all server locations except those explicitly specified otherwise.

For simplicity, it is recommended to leave `AmAuthProvider` as `Off` globally and set it to `On` for specific locations where you want Apache to act as the enforcement point.

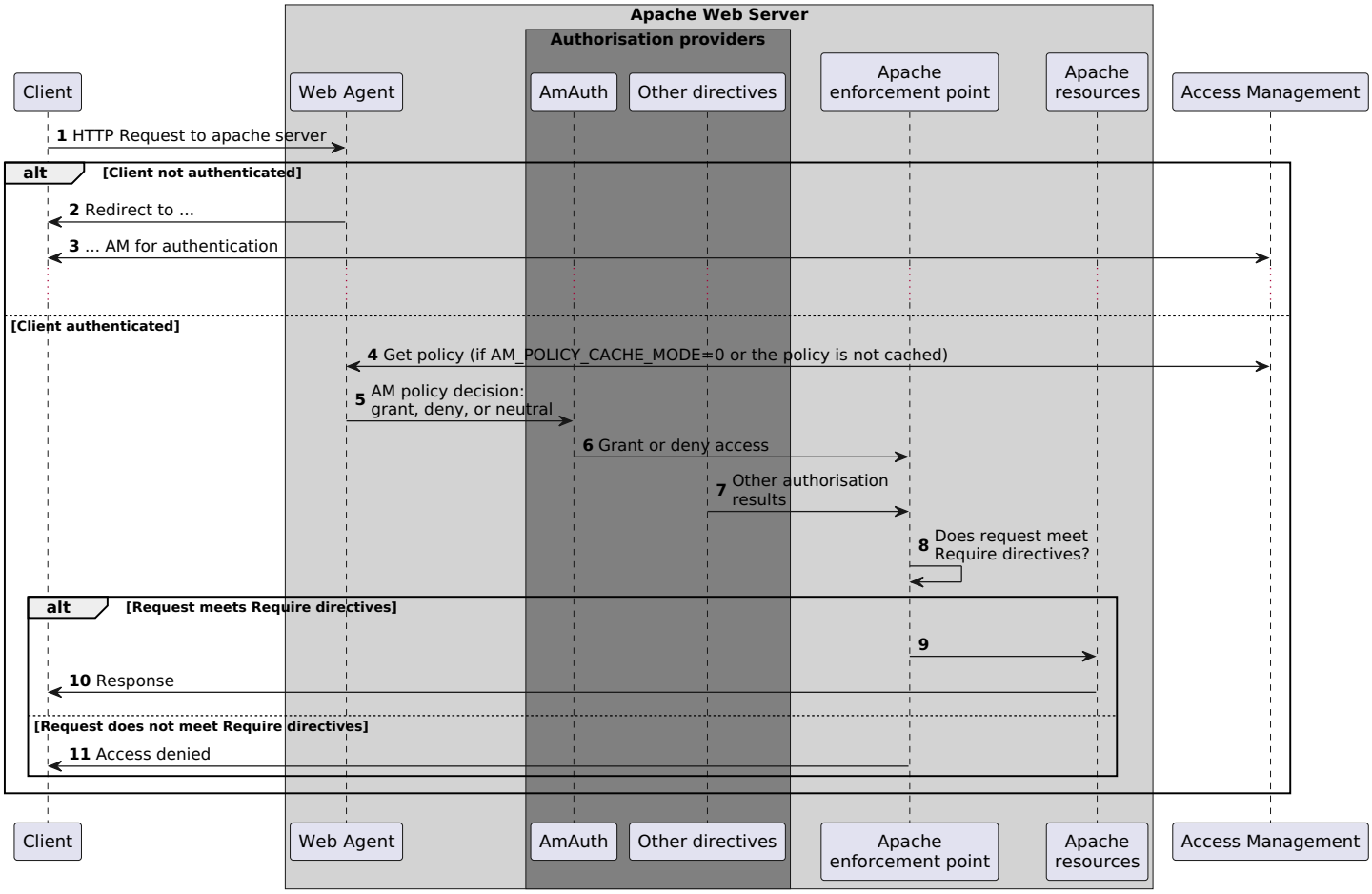
When `AmAuthProvider` is `On` and the request doesn't match a not-enforced rule

When a request doesn't match a not-enforced rule, the agent does the following:

- Checks that the user is authenticated with AM, and redirects the user for authentication if not.
- Requests policy information from AM for the request.
- Relays the policy information to the Apache `Require AmAuth` directive.

Apache or IBM HTTP server uses the `Require AmAuth` directive and other `Require` directives to allow or deny access to resources.

The following image shows the flow of requests:

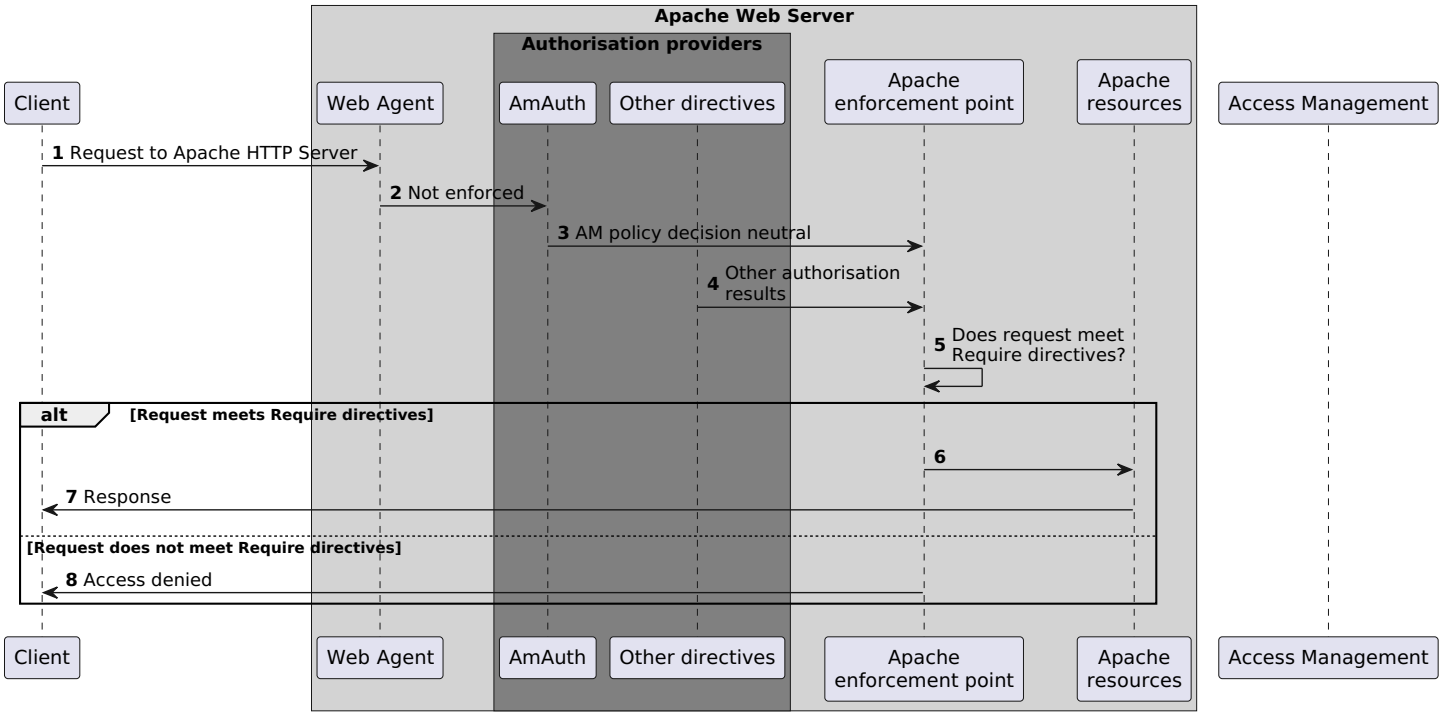


When `AmAuthProvider` is `On` and the request matches a not-enforced rule

When a request matches a not-enforced rule, the agent does not require the user to be authenticated with AM or request policy information from AM. The `Require AmAuth` directive returns a neutral value.

Apache or IBM HTTP server uses the other `Require` directives to allow or deny access to resources.

The following image shows the flow of requests:



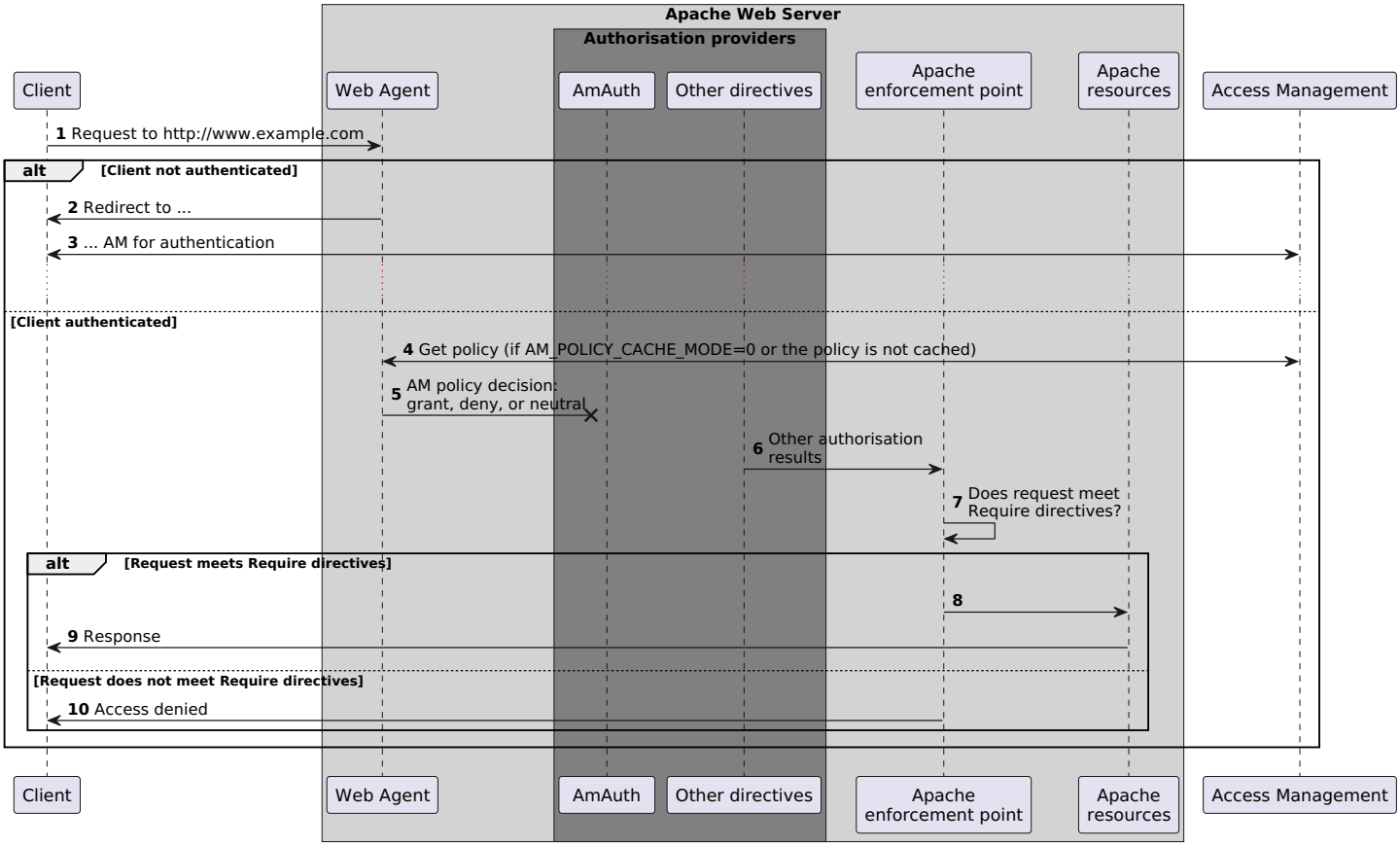
Consider the following points for using not-enforced rules when `AmAuthProvider` is `On`:

- Instead of using not-enforced rules to provide caveats to AM policy enforcement, use Apache `Require` directives.
- In server locations where the agent is configured with not-enforced rules, set `AmAuthProvider` to `Off` to let the agent do the enforcement.
- If you use not-enforced rules when `AmAuthProvider` is `On`, remember that the agent drops out of authorisation decisions for requests that match a rule. Apache `Require` directives are used to allow or deny requests.

When `AmAuthProvider` is `On` and `Require AmAuth` is not specified

When `AmAuthProvider` is `On`, the `Require AmAuth` directive should always be specified. If `AmAuthProvider` is `On` but the `Require AmAuth` directive is not specified, users are still required to authenticate with AM but Apache does not use policy information from AM in its decision.

The following image shows the flow of requests:



The following example has this configuration:

- The request doesn't match a not-enforced rule.
- `AmAuthProvider` is `On` for the `/var/www/transaction` directory.
- `Require AmAuth` is not specified

```
//Not a recommended configuration

<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

<Directory /var/www/transaction>
    AmAuthProvider On
    Options Indexes FollowSymLinks
    AllowOverride None
    <RequireAll>
        Require ip 19.168.2
    </RequireAll>
</Directory>

AmAgent On
AmAgentConf /opt/web_agents/apache24_agent/instances/agent_1/config/agent.conf
AmAuthProvider Off
```

Accessing a resource in `/var/www/transaction`

Apache or IBM HTTP server uses the `Require ip` directive to allow or deny the request. The user must be authenticated with AM and a valid user must be set, but AM policy information is ignored.

Example where `AmAuthProvider` is `Off` globally and `On` for specific directories

The example is configured as follows:

- The request doesn't match a not-enforced rule
- `AmAuthProvider` is `Off` globally
- `AmAuthProvider` is `On` for the `/var/www/transaction` directory:
- `Require AmAuth` is specified

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

<Directory /var/www/transaction>
    AmAuthProvider On
    Options Indexes FollowSymLinks
    AllowOverride None
    <RequireAll>
        Require AmAuth
        Require ip 19.168.2
    </RequireAll>
</Directory>

AmAgent On
AmAgentConf /opt/web_agents/apache24_agent/instances/agent_1/config/agent.conf
AmAuthProvider Off
```

Accessing a resource in `/var/www/`

The agent acts as the enforcement point, allowing or denying requests based on not-enforced rules and AM policies.

Accessing a resource in `/var/www/transaction`

The agent provides AM policy information to the `Require AmAuth` directive. Apache uses that and the `Require ip` directive to allow or deny the request.

To access the resource, the user must be authenticated with AM, and the request must meet AM policy requirements and come from the specified IP address.

Apache as a reverse proxy

This section has an example configuration of Apache HTTP Server as a reverse proxy between AM and Web Agents Documentation. You can use any reverse proxy that supports the WebSocket protocol.

For information about how to configure Apache for load balancing, and other requirements for your environment, refer to the Apache documentation.

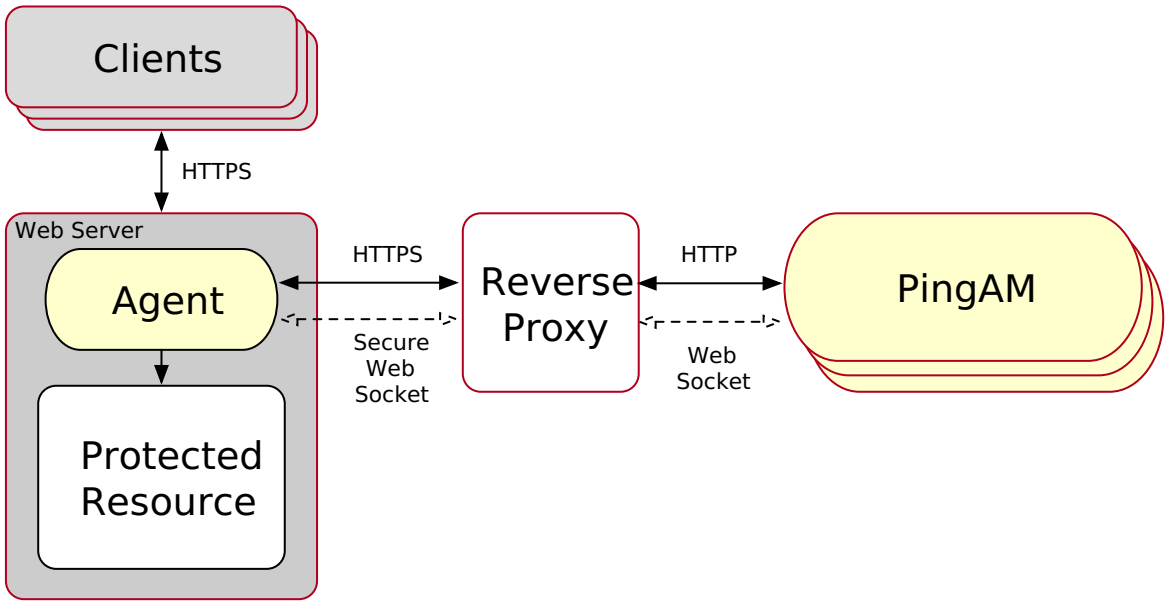


Figure 1. Apache HTTP Server reverse proxy configured between the agent and AM

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" (1)
ProxyPass "/openam/notifications" "ws://am.example.com:8080/am/notifications" Upgrade=websocket (2)
ProxyPass "/openam" "http://am.example.com:8080/am" (3)
ProxyPassReverseCookieDomain "openam.internal.example.com" "proxy.example.com" (4)
ProxyPassReverse "/openam" "http://am.example.com:8080/am" (5)
...
</VirtualHost>
```

(1) RequestHeader: Set to `https` or `http`, depending on the proxy configuration. If the proxy is configured for https, as in the above example, set to `https`. Otherwise, set `http`. In a later step, you configure AM to recognize the forwarded header and use it in the `goto` parameter for redirecting back to the agent after authentication.

(2) ProxyPass: Set to allow WebSocket traffic between AM and the agent. If HTTPS is configured between the proxy and AM, set to use the `wss` protocol instead of `ws`.

(3) ProxyPass: Set to allow HTTP traffic between AM and the agent.

(4) ProxyPassReverseCookieDomain: Set to rewrite the domain string in `Set-Cookie`headers in the format internal domain (AM's domain) public domain (proxy's domain).

(5) ProxyPassReverse: Set to the same value configured for the `ProxyPass` directive.

For more information about configuring Apache HTTP Server 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, refer to Agent connection to AM through a load balancer/reverse proxy

Install IIS Web Agents Documentation

Web Agents Documentation instances can be configured to operate with multiple websites in IIS. 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.

Consider the following points:

- Web Agents Documentation requires IIS to be run in Integrated mode.
- A Web Agents Documentation configured for a site or parent application protects any application configured within. The same is true for protected applications containing applications within.

Consider the following restrictions:

- 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 *do 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 Agents Documentation 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**
>
> Install Web Agents Documentation on the child application before installing it in the parent. Trying to install an agent on a child that is already protected results in error.

- You can disable the agent protection at any level of the IIS hierarchy, with the following constraints:
  - Disabling the agent in a parent application disables the protection on all children applications that do not have a specific agent instance installed on them.
  - Disabling the agent in a child application does not disable protection on its parent application.
- 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.

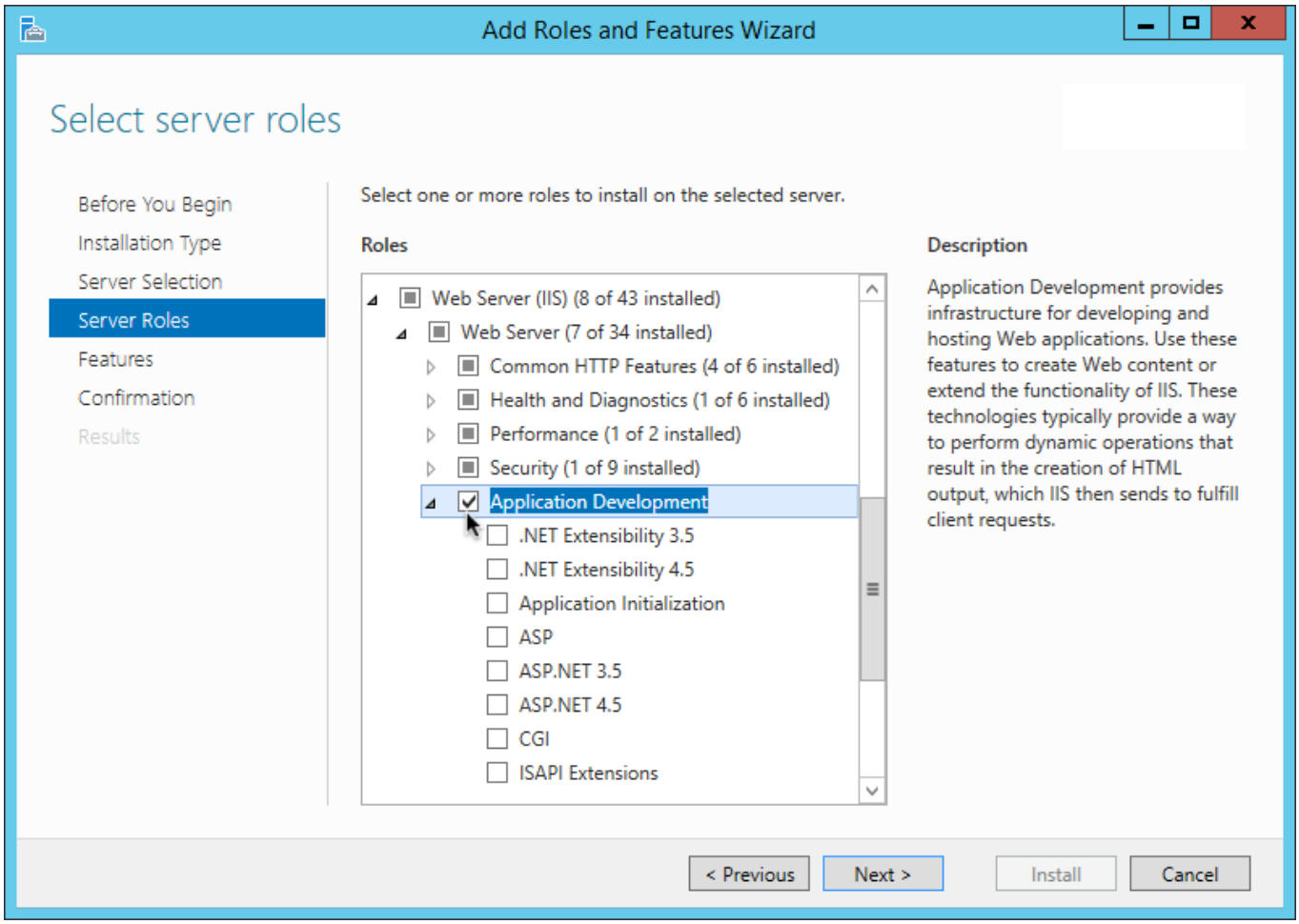


Figure 1. Adding the application development component to IIS

Install IIS Web Agents Documentation interactively

1. Review the information in Before you install and complete the Preinstallation tasks.
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.

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

5. When prompted, enter information for your deployment.

> **Tip**
>
> To cancel the installation at any time, press `CTRL-C`.

1. 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 dot (`.`) 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
         application path:/, pool DefaultAppPool
1.1.1    virtualDirectory path:/, configuration: C:\inetpub\wwwroot\web.config

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

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

   - 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.
   - 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.

- 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`.

2. 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:
```

3. Enter the full URL of the AM instance the agent will use. Make sure the deployment URI is specified.

> **Note**
>
> If a reverse proxy is configured between AM and the agent, set the AM URL to the proxy URL, for example, `https://proxy.example.com:443/am`. For information about setting up an environment for reverse proxies, refer to Apache as a reverse proxy.

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

4. 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
```

5. 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
```

6. Enter the agent profile realm. Realms are case-sensitive.

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

7. 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
```

8. 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:
   AM URL: https://am.example.com:8443/am
   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.
```

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

> **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.

   Each agent instance has a numbered configuration and logs directory. The first agent configuration and logs are located in `web_agents\iis_agent\instances\agent_1\`.

6. Make sure 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, refer to agentadmin command.

When permissions aren't set correctly, errors such as getting a blank page when accessing a protected resource can occur.

7. If you installed Web Agents Documentation in an application, set CDSSO Redirect URI to the application path, as follows:
    1. Go to **Realms** > Realm Name > **Agents** > **Web** > Agent Name > **SSO** > **Cross Domain SSO**.
    2. Add the application path to the default value of CDSSO Redirect URI. For example, if you installed Web Agents Documentation in an application such as `MyApp1/MyApp2`, set the property to `MyApp1/MyApp2/agent/cdsso-oauth2`.
    3. Save your changes.

Install IIS Web Agents Documentation silently

Use the `agentadmin --s` command for silent installation. For information about the options, refer to agentadmin command.

1. Review the information in Before you install and complete the Preinstallation tasks.
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://am.example.com:8443/am" ^
  "http://iis.example.com:80" ^
  "/" ^
  "iisagent" ^
  "c:\pwd.txt" ^
  --acceptLicence

AM Web Agents Documentation for IIS Server installation.

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

4. Make sure 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, refer to agentadmin command.

When permissions aren't set correctly, errors such as getting a blank page when accessing a protected resource can occur.

5. (Optional) If you installed the agent in a parent application, enable it for its child applications by following the steps in Disable and enable agent protection for child applications.

Enable and disable IIS Web Agents Documentation

Disable and enable Web Agents Documentation on an IIS site or application

The `agentadmin` command shows only instances of the web agent; to enable or disable the protection of children applications, refer to Disable and enable agent protection for child applications.

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

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

AM Web Agents Documentation configuration instances:

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

Make a note of the ID value of the configuration instance you want to disable or enable.

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

Disable and enable agent protection for child applications

1. Edit the child application's `web.config` configuration.
2. Decide whether to enable or disable web agent protection:
   - To disable 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.
   - To enable agent protection, understand that 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 agent's protection for a child application by following the steps in this procedure, remove the lines added to the `web.config` file to enable protection again.

Enable support for IIS basic authentication and password replay

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 `login_USER`. The agent either sets all three, or does not set any of them.

When Logon and Impersonation is enabled, the agent performs Windows login and sets the user impersonation token in the IIS session context.

When Show Password in HTTP Header is enabled, the agent adds the password in the `USER_PASSWORD` header.

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

The agent requires IIS to run in Integrated mode.

Microsoft issues

Apply workarounds for the following Microsoft issues:

- Prompt for credentials when you access WebDav-based FQDN sites in Windows
- Office applications open blank from SharePoint WebDAV or sites

To Configure IIS basic authentication and password replay support

1. Use the `openssl` tool to generate a suitable encryption key:

   ```
   $ openssl rand -base64 32
   e63…sw=
   ```

2. In the AM admin UI, go 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. Go to **Realms** > Realm Name > **Authentication** > **Settings** > **Post Authentication Processing**, and in **Authentication Post Processing Classes**, add the class `com.sun.identity.authentication.spi.ReplayPasswd`.

4. Restart AM.

5. In the AM admin UI go to **Realms** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name > **Advanced**

   1. In Replay Password Key, enter the encryption key generated in a previous step.

   2. For Windows login for user token impersonation, enable Logon and Impersonation.

   3. Save your changes.

6. (Optional) To set the encrypted password in the IIS `AUTH_PASSWORD` server variable, go to **Realms** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name > **Advanced**, and enable Show Password in HTTP Header.

7. (Optional) If you require Windows login, or you need to use basic authentication with SharePoint or OWA, then you must do the following so that the agent requests AM to provide the appropriate account information from Active Directory in its policy response:

   - Configure Active Directory as a user data store
   - Configure the IIS web agent profile **User ID Parameter** and **User ID Parameter Type**.

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

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

     In the AM admin UI under **Realms** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name > **AM Services**, set User ID Parameter and User ID Parameter Type.

     For example, if the real username for Windows domain login 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 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 agent can set the HTTP headers `REMOTE_USER`, `AUTH_USER`, or `login_USER` and `USER_PASSWORD` with Active Directory attribute values suitable for Windows login, setting the remote user, and so forth.

8. (Optional) To access Microsoft Office from SharePoint pages, configure AM to persist the authentication cookie. For information, refer to "Persistent cookie module" or "Persistent cookie decision node" in AM's *Authentication and SSO guide*.

Install in a subrealm

Examples in this document install the agent in the top-level realm. To install the agent in a subrealm during interactive or silent installation, use the subrealm during the installation or in the response file.

For example, instead of:

```
Agent realm/organization name: [/]: /
```

specify:

```
Agent realm/organization name: [/]: /myrealm
```

Even though the agent is installed in a subrealm, the default login redirect requires the user realm to be the top-level realm. For information about how to change the user realm, refer to Login redirect.

Install NGINX Plus Web Agents Documentation

Examples use the NGINX Plus R29 agent path. For other supported versions, replace the R29 agent path with the required version. For information about supported versions of NGINX, refer to Other requirements.

Note that 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.

Install NGINX Plus Web Agents Documentation interactively

1. Review the information in Before you install and complete the Preinstallation tasks.
2. Shut down the server where you plan to install the agent.
3. Make sure AM is running.
4. Run the `agentadmin --i` command to install the agent:

```
$ cd /web_agents/nginx29_agent/bin/
$ ./agentadmin --i
```

5. When prompted, enter information for your deployment.

> **Tip**
>
> To cancel the installation at any time, press `CTRL-C`.

1. 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
```

2. 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 p assword.

   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:
```

3. Enter the full URL of the AM instance that the agent should connect to:

> **Note**
>
> If a reverse proxy is configured between AM and the agent, set the AM URL to the proxy URL, for example, `https://proxy.example.com:443/am`. For information about setting up an environment for reverse proxies, refer to Apache as a reverse proxy.

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

4. 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
```

5. Enter the name of the agent profile created in AM:

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

6. Enter the agent profile realm. Realms are case-sensitive:

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

7. 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:/secure-directory/pwd.txt
```

8. 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 setting is correct, type `yes` to proceed with installation:

```
Installation parameters:
 AM URL: https://am.example.com:8443/am
 Agent URL: http://www.example.com:80
 Agent Profile name: nginx_agent
 Agent realm/organization name: /
 Agent Profile password source: /secure-directory/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/nginx29_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/nginx29_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.
```

Each agent instance has a numbered configuration and logs directory. The first agent configuration and logs are located in `/web_agents/nginx29_agent/instances/agent_1/`.

6. Finish installation as described in Complete the NGINX Plus Web Agents Documentation Installation.

Install NGINX Plus Web Agents Documentation silently

Use the `agentadmin --s` command for silent installation. For information about the options, refer to agentadmin command.

1. Review the information in Before you install and complete the Preinstallation tasks.
2. Shut down the 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://am.example.com:8443/am" \
 "http://www.example.com:80" \
 "/" \
 "nginx_agent" \
 "/secure-directory/pwd.txt" \
 --acceptLicence
Web Agents Documentation 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/nginx29_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/nginx29_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 installation as described in Complete the NGINX Plus Web Agents Documentation Installation.

Complete the NGINX Plus Web Agents Documentation installation

After interactive or silent installation, follow these steps to complete the installation.

1. Edit the NGINX Plus server configuration file `nginx.conf` to load the agent module `openam_ngx_auth_module.so`:

```
$ 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/nginx29_agent/lib/openam_ngx_auth_module.so;
…
```

2. Add and `openam_agent` directive at the global level of `nginx.conf` to set the agent as `on`. For more information, refer to openam_agent.
3. Give the user or group running the NGINX Plus server appropriate permissions for the following directories:
    - Read permission: `/web_agents/nginx29_agent/lib`
    - Read and write permission:
        - `/web_agents/nginx29_agent/instances/agent_nnn`
        - `/web_agents/nginx29_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, refer to Troubleshooting.
4. Start the server.

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

Check the NGINX Web Agents Documentation 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 verify that no startup errors occurred:

```
Web Agent Version: ${wpa.vers.ext}
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.

Install in a subrealm

Examples in this document install the agent in the top-level realm. To install the agent in a subrealm during interactive or silent installation, use the subrealm during the installation or in the response file.

For example, instead of:

```
Agent realm/organization name: [/]: /
```

specify:

```
Agent realm/organization name: [/]: /myrealm
```

Even though the agent is installed in a subrealm, the default login redirect requires the user realm to be the top-level realm. For information about how to change the user realm, refer to Login redirect.

Configure NGINX Plus Web Agents Documentation

NGINX directives

Add NGINX directives to the `/etc/nginx/nginx.conf` configuration file to configure the global environment or individual HTTP servers and HTTP locations.

Directives are applied hierarchically. When set at the global level in `nginx.conf`, they apply to all HTTP servers and HTTP locations except those explicitly specified otherwise. Similarly, when set for an HTTP server or HTTP location, they are set for all child locations except those explicitly specified otherwise.

`openam_agent`

A flag to set the agent on or off:

`openam_agent on`

The agent protects the resource. It allows or denies requests based on AM policy configuration and not-enforced rules.

`openam_agent off`

NGINX protects the resource. The agent plays no part in protecting the server locations.

Default: None.

After installation, add `openam_agent on` to `/etc/nginx/nginx.conf` at the global level.

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log notice;
openam_agent on
```

Tip

Consider setting `openam_agent` to `off` for the following situations:

- For HTTP servers or HTTP locations that need no AM authentication or policy, such as the public face of a website, `/css` directories, or `/images` directories.
- When an NGINX HTTP Server is acting as a reverse proxy to AM or Identity Cloud, if you don't want the agent to take part in protecting URLs in AM or Identity Cloud.

openam_configuration

The path to the local bootstrap file for the agent:

`openam_configuration <path to nginx.conf>`

Default: None, but during agent installation you must provide the path to `/etc/nginx/nginx.conf`.

openam_threadpool

The name of the AM threadpool:

`openam_threadpool <name>`

Default: The NGINX default threadpool

Caution

Before setting this directive, consider the consequence of changing the threadpool name.

openam_agent_instance

A number to identify an instance of NGINX Plus:

`openam_agent_instance <number>`

Default: `1`

In deployments with multiple instances of NGINX Plus, use a unique number for each instance.

Examples

`openam_agent` *is* on *globally but off` for one HTTP location*

Important

When `openam_agent` configuration is `off`, configure the server location `/agent` as `on`. This allows AM to redirect requests to the `/agent` endpoint after authentication.

In the following example `nginx.conf`:

- `agent_1` in the `server` context protects the `/` and `/marketplace`location` contexts
- No agent instance protects the `/customers`location` context.

```
server {
  listen       80 default_server;
  server_name  localhost;
  openam_agent on;
  openam_agent_configuration /web_agents/nginx29_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
  }
}
```

Different agent instances protect different parts of the deployment

In the following example `nginx.conf`:

- `agent_1` at the `server` context protects the `/` and `/marketplace`location` contexts
- `agent_2` protects the `/customers`location` context

```
server {
  listen       80 default_server;
  server_name  localhost;
  openam_agent on;
  openam_agent_configuration /web_agents/nginx29_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/nginx29_agent/instances/agent_2/config/agent.conf;
      root   /www/customers
      index  index.html
    }

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

NGINX as a reverse proxy

This section contains an example configuration of NGINX as a reverse proxy between AM and Web Agents Documentation. You can use any reverse proxy that supports the WebSocket protocol.

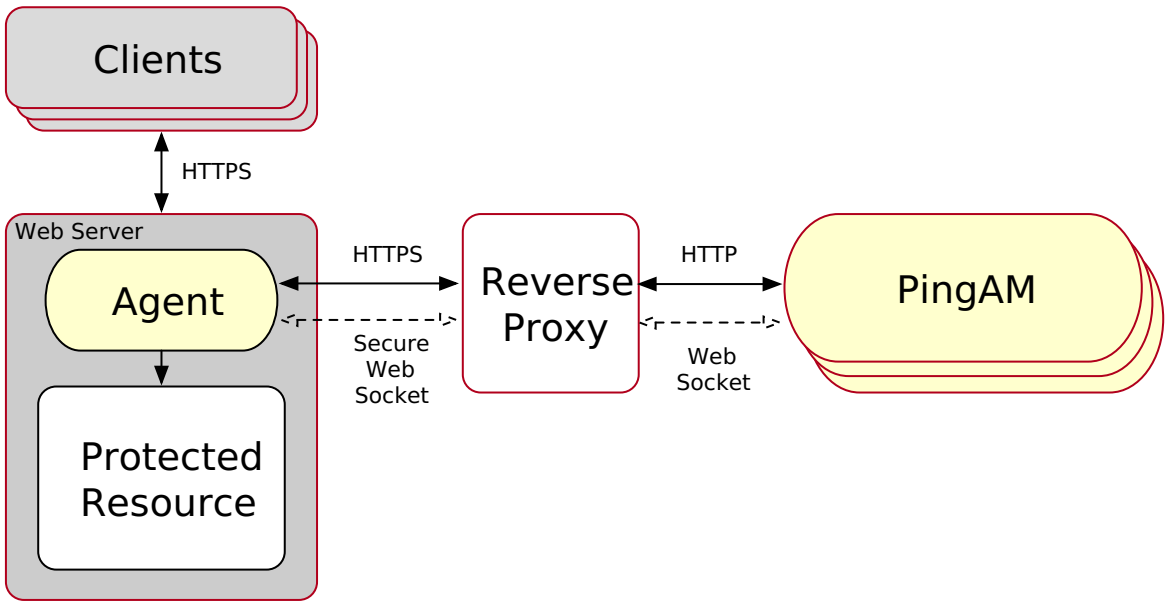


Figure 1. NGINX reverse proxy configured between the agent and AM

For information about how to configure NGINX for load balancing, and for other environment requirements, refer to the NGINX documentation at NGINX as a WebSocket Proxy.

After interactive or silent installation, follow these steps to configure NGINX as a reverse proxy.

1. Locate the NGINX Plus server configuration file `/etc/nginx/nginx.conf`.
2. Edit `nginx.conf` to add directives to the context you want to protect:

```
server {
  ...
  location /am
    {
      proxy_set_header Host $host proxy_pass http://hostname:port/am;
      proxy_http_version 1.1;
      proxy_set_header Connection ""; # to allow keep alives to work #
    }
  location /am/notifications/
    {
      proxy_pass http://hostname:port/am/notifications;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "Upgrade";
      proxy_set_header Host $host;
    }
  ...
}
```

3. Ensure the user or group running the NGINX Plus server has the appropriate permissions over the following directories:
   - Read Permission: `/web_agents/nginx29_agent/lib`
   - Read and Write Permission:
     - `/web_agents/nginx29_agent/instances/agent_nnn`
     - `/web_agents/nginx29_agent/log`
4. Restart the reverse proxy instance.
5. Configure AM to recover the forwarded header configured in the reverse proxy.
6. Review other configuration that a reverse proxy environment can require. For more information, refer to Agent connection to AM through a load balancer/reverse proxy

Post-installation tasks

Note the location of configuration files and logs

Each agent instance has a numbered configuration and logs directory, starting with `agent_1`. The first agent configuration and logs are located at `web_agents/agent_type/instances/agent_1/`.

The following configuration files and logs are created:

- `web_agents/agent_type/instances/agent_1/config/` : Bootstrap properties to connect to AM and download the configuration. This directory contains properties that are used only in local configuration mode.
- `web_agents/agent_type/instances/agent_1/logs/audit/` : Audit log directory. Used only if Audit Log Location is `LOCAL` or `ALL` .
- `web_agents/agent_type/instances/agent_1/logs/debug/` : The directory where the agent writes debug log files after startup.

During agent startup, the location of the logs can be based on the agent web server, or defined in the site configuration file for the server. For example, bootstrap logs for NGINX Plus Web Agents Documentation can be written to `/var/log/nginx/error.log` .

Validate the agent instance

Validate the agent instance by using the `agentadmin --V[i]` command. For information about the options and requirements for this command, refer to agentadmin.

Linux

```
$ sudo -u web-server-user
$ cd /web_agents/agent_type/bin/
$ ./agentadmin --Vi agent_name am_identity_name /path/to/am_identity_password
```

Windows

```
C:\web_agents\agent-type\bin> agentadmin --Vi ^
agent_name am_identity_name C:/path/to/am_identity_password /
```

A result similar to this is displayed:

```
Running configuration validation for agent_1:

  Agent instance is configured with 1 naming.url value(s):
  1. https://am.example.com:8443/am is valid
  selected https://am.example.com:8443/am 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.
```

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

Configure shared runtime resources and memory

Consider using agent resource groups in atypical deployments, where multiple independent web servers are deployed on the same machine. Agent resource groups apply only to Apache HTTP server or NGNIX, because IIS runs only as a single instance.

Agent resource groups allow server processes to share resources and memory, such as background tasks, log files, runtime resources including pipes, caches, and notification channels to AM.

An agent resource group is determined by the AmAgentID directive in a web server configuration. The value is numeric and defaults to 0 for a typical, single-server deployment. By default, up to 32 agent instances can be in a single installation. For information about changing this limit, refer to *AM_MAX_AGENTS* in Environment variables.

Choose whether to share resources

Consider the information in the following table before configuring your agent resource groups:

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.

Impact	Advantage	Caution
Reduced number of background threads. (Single WebSocket connection to AM for notifications)	Reduced system resource usage.	Ensure that the `AM_MAX_AGENTS` 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.	Ensure that files and resources can be accessed by all agent instances. For example, add the users running the instances to the same group and configure the resources to have `660` permissions. For more information, refer to *AM_RESOURCE_PERMISSIONS* in Environment variables.

Configure Apache agent groups

To create an Apache agent group, edit the Apache configuration file, `/etc/httpd/conf/httpd.conf`, to add an `AmAgentId` directive.

To isolate agents in different web servers hosted on the same machine, set `AmAgentId` to a different value in each server configuration.

The following example `httpd.conf` file configures one group with **AmAgentId 1**, including two virtual hosts. Each virtual host is protected by a different instance of the agent, but both agent instances belong to the agent group `1`.

> **Important**
>
> The `AmAgentId` configuration must be outside the `VirtualHost` section.

```
AmAgentId 1
<VirtualHost *:80>
ServerName www.site1.com
DocumentRoot /home/www/site1.com
AssignUserID site1 www-data
AmAgent On
AmAgentConf /web_agents/apache24_agent/bin/../instances/agent_1/config/agent.conf
…
</VirtualHost>

<VirtualHost *:8080>
ServerName www.site2.com
DocumentRoot /home/www/site3.com
AssignUserID site2 www-data
AmAgent On
AmAgentConf /web_agents/apache24_agent/bin/../instances/agent_2/config/agent.conf
…
</VirtualHost>
```

When `AmAgentId` is not specified for an agent instance, it uses the default value of `0`.

To create multiple agent groups in an Apache agent installation, use different values for `AmAgentId`. In the previous example, you could specify two groups by using `AmAgentId 1` and `AmAgentId 2`.

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

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.

Configure NGINX Plus agent groups

To add NGINX Plus agent instances to a group, add the `openam_agent_instance` directive to each instance in the NGINX Plus server configuration file `/etc/nginx/nginx.conf`.

The following example `nginx.conf` file configures one agent group, **openam_agent_instance 2**, containing `agent_3` and `agent_4`:

```
server {
  listen       80 default_server;
  server_name  localhost;
  openam_agent on;
  openam_agent_configuration /web_agents/nginx29_agent/bin/../instances/agent_3/config/agent.conf;
openam_agent_instance 2
  …
  location /customers {
    openam_agent on;
    openam_agent_configuration /web_agents/nginx29_agent/bin/../instances/agent_4/config/agent.conf;
openam_agent_instance 2
    root   /www/customers
    index  index.html
  }
…
```

When `openam_agent_instance` is not specified for an agent instance, the instance uses the default value of `1`.

To create multiple agent groups in an NGINX Plus agent installation, use different values for `openam_agent_instance`. In the previous example, you could specify two groups by using `openam_agent_instance 2` and `openam_agent_instance 3`.

Support load balancers and reverse proxies between clients and agents

When your environment has reverse proxies or load balancers configured between agents and 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, refer to Configure load balancers and reverse proxies.

Configure audit logging

Web Agents Documentation supports the logging of audit events for security, troubleshooting, and regulatory compliance. Store 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, agents write audit logs to the AM server that satisfies the agent request for client authentication or resource authorization.

Agents cannot log audit events remotely if:

- AM's audit logging service is disabled.
- No audit event handler is configured in the agent profile realm.
- All audit event handlers configured in the agent profile realm are disabled.

For more information about audit logging in AM, refer to Setting up audit logging in AM's *Security guide*.

Locally

Log audit events in JSON format to a file in the 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 agent profile realm.

The example is 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, refer to Audit log format in AM's *Security guide*.

Web Agents Documentation supports propagation of the transaction ID across the ForgeRock Identity Platform using the HTTP header `X-ForgeRock-TransactionId`. For more information about configuring the header, refer to Configuring the trust transaction header system property in AM's *Security guide*.

By default, Web Agents Documentation does not write audit log records. To configure audit logging, perform the following procedure:

To configure audit logging

This procedure assumes that Web Agents Documentation is in centralized configuration mode. Property names are provided for local configuration mode.

1. In the AM admin UI, go to **Realms** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name > **Global** > **Audit**.
2. In Audit Access Types, select the type of messages to log. For example, select `LOG_ALL` to log access allowed and access denied events.
3. In Audit Log Location, 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. In Local Audit Log Rotation Size, specify the maximum size, in bytes, of the audit log files.

   This is a bootstrap property. After changing this property, restart the web server where the agent runs.

Secure connections

Secure communication between the agent and AM

> **Caution**
>
> Be aware of security breaches and vulnerabilities. Make sure your environment isn't using outdated, insecure protocols, such as SSL 3.0, TLS 1.0, and others.

To secure communications, configure the agent to validate server certificates installed in the server where AM runs and to present a client certificate to AM. Learn more in AM's Secure HTTP and LDAP connections.

To facilitate integration and test, Web Agents Documentation is configured by default to trust any server certificate. Test client certificates aren't provided or configured.

To send cookies only when the communication channel is secure, set Enable Cookie Security to `true` .

Secure communication with OpenSSL

Unix-based agents support OpenSSL libraries. Windows-based agents can use OpenSSL or the Windows Secure Channel API (Schannel).

> **Note**
>
> If you want to use OpenSSL for the IIS or Windows Apache agent, configure the agent to use OpenSSL before continuing:
>
> - Make sure the OpenSSL libraries are in the correct location.
> - Disable Schannel by setting the Enable OpenSSL to Secure Internal Communications property to `true` .

Configure server certificate validation using OpenSSL

Perform the following steps to configure the agent to validate AM's or Identity Cloud's server certificate:

1. Set the Server Certificate Trust property to `false` .

> **Important**
>
> The Server Certificate Trust property should always be `false` in production.

2. Set the CA Certificate File Name property to the filename of the CA bundle for your system. The exact location and name of the CRT file varies by operating system. For example, for Ubuntu, it's `/etc/ssl/certs/ca-certificates.crt`.
3. Set the OpenSSL Certificate Verification Depth property to the level of certificate validation required in your environment.
4. Restart the agent.

Configure client certificate authentication using OpenSSL

When AM or Identity Cloud are configured to perform client authentication, you must configure the agent to present its client certificates as follows:

1. Create a PEM file that contains the certificate chain for the agent. For example, `client-cert.pem`.
2. Create a PEM file that contains the private key corresponding to the certificate. For example, `client-private-key.pem`.
3. Set the Public Client Certificate File Name 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
```

4. Set the Private Client Certificate File Name 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
```

5. If the private key is password-protected:

    1. Obfuscate the password using 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=
    ```

    Windows

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

    Where encryption-Key is the value of the Agent Profile Password Encryption Key property.

    2. Set the Private Key Password property to the encrypted password value. For example:

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

6. Restart the agent.

Tip

Use the `agentadmin --Vi` command to validate the TLS connection settings between the agent and AM or Identity Cloud.

Secure communication with Schannel

By default, the IIS and Apache for Windows agents use the Windows built-in Secure Channel API (Schannel). Alternatively, you can use OpenSSL as described in Secure internal communication with OpenSSL.

> **Note**
>
> Before continuing, make sure the agent is configured to use Schannel:
>
> - If this is a new installation, Windows-based agents use Schannel by default.
> - If you've previously configured the IIS or Apache agent to use OpenSSL libraries, set the Enable OpenSSL to Secure Internal Communications property to `false`.

Configure server certificate validation using Schannel

Perform the following steps to configure the agent to validate AM's or Identity Cloud's server certificate:

1. Set the Server Certificate Trust property to `false`.

   > **Important**
   >
   > The Server Certificate Trust property should always be `false` in production.

2. If you're using self-signed certificates or the server certificate is issued from a new CA, add the certificates required to validate AM's or Identity Cloud's server certificate to the Windows certificate store. For example, to use PowerShell, add certificates to the following locations:
   - **Root CA certificates**: add them to the `Cert:\LocalMachine\Root` location.
   - **Intermediate CA certificates**: add them to the `Cert:\LocalMachine\Ca` location.
3. Restart the agent.

Configure client certificate authentication using Schannel

When AM or Identity Cloud are configured to perform client authentication, you must configure the agent to present its client certificates using one of the following methods.

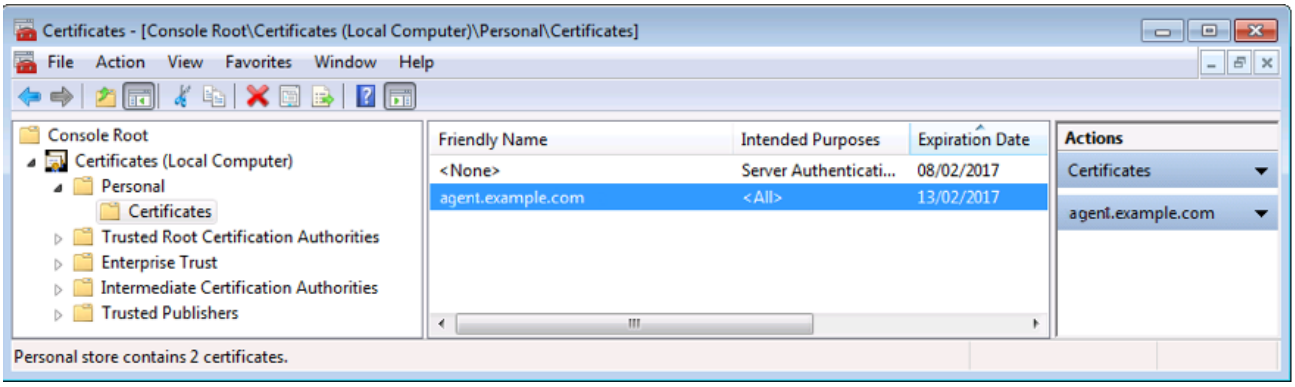
The method you use depends on whether you are loading the client certificate through the Windows certificate store or through a PFX certificate:

Agent authenticates using a client certificate stored in the Windows Certificate store

Configure the agent to present its client certificate as follows:

1. Import the client certificate chain and private key into the Windows certificate store. For example, for PowerShell, import them to `Cert:\LocalMachine\My`.
2. Set the Public Client Certificate File Name property to the friendly name of the client certificate chain. For example:

```
com.forgerock.agents.config.cert.file = agent.example.com
```



3. Restart the agent.

Agent authenticates using a PFX file that contains the certificate chain

Configure the agent to present its client certificate as follows:

1. Create a Personal Information Exchange (PFX) file that contains the certificate chain for the agent and its private key. For example, `client.pfx` .
2. Set the Public Client Certificate File Name property to the PFX file you just created. For example:

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

3. Obfuscate the certificate password 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=
```

Where encryption-Key is the value of the Agent Profile Password Encryption Key property.

4. Set the Private Key Password property to the encrypted password value. For example:

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

5. Restart the agent.

> **Tip**
>
> Use the `agentadmin --Vi` command to validate the TLS connection settings between the agent and AM or Identity Cloud.

Remove Web Agents Documentation

Remove Apache Web Agents Documentation

1. Shut down Apache HTTP Server where the agent is installed.
2. Run `agentadmin --l` to output a list of the installed web agent configuration instances.

   Note the ID of the Web Agents Documentation instance 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_1

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

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

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

4. Start the Apache HTTP Server.

Remove a single instance of IIS Web Agents Documentation

Perform the steps in this procedure to remove :

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

```
c:\web_agents\iis_agent\bin> agentadmin.exe --l
agentadmin.exe --l
Web Agent configuration instances:

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

   Note the ID of the Web Agents Documentation instance to remove.
3. Run `agentadmin.exe --r`, specifying the ID of the Web Agents Documentation 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 agent libraries. To remove all agent instances and libraries, refer to Remove all instances of IIS Web Agents Documentation.

Remove all instances of IIS Web Agents Documentation

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 Web Agents Documentation 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.
```

Remove NGINX Plus Web Agents Documentation

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

```
$ ./agentadmin --l
OpenAM Web Agents Documentation configuration instances:

id:            agent_1
configuration: /web_agents/nginx29_agent/instances/agent_1
server/site:   /etc/nginx/nginx.conf

id:            agent_2
configuration: /web_agents/nginx29_agent/instances/agent_2
server/site:   /etc/nginx/nginx.conf

id:            agent_3
configuration: /web_agents/nginx29_agent/instances/agent_3
server/site:   /etc/nginx/nginx.conf
```

   Note the ID of the Web Agents Documentation instance to remove.

3. Run the `agentadmin --r` command, specifying the ID of the agent instance to remove. A warning is displayed. Type `yes` to remove the instance.

```
$ ./agentadmin --r agent_1
Warning! This procedure will remove the Web Agents Documentation configuration for agent_1
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_1 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 `openam_ngx_auth_module.so` library.
6. Restart the NGINX Plus server.

agentadmin command

The `agentadmin` command manages Web Agents Documentation installation. It returns `EXIT_SUCCESS` (or `0`) when it completes successfully, and `EXIT_FAILURE` (or a code greater than zero) when it fails.

The following options are supported:

`--i`

Install a new agent instance.

Usage: `agentadmin --i`

`--s`

Silently, non-interactively, install 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

(Apache HTTP Server) The full path to the server configuration file. The installer modifies this file to include the agent configuration and module.

(Microsoft IIS) 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`.

am-url

The full URL of the AM instance that the agent will use. Ensure the deployment URI is specified.

Example: `https://am.example.com:8443/am`

> **Note**
>
> If a reverse proxy is configured between AM and the agent, set the AM URL to the proxy URL, for example, `https://proxy.example.com:443/am`. For information about setting up an environment for reverse proxies, refer to Apache as a reverse proxy.

agent-url

The full URL of the server on which the agent is running.

Example: `http://www.example.com:80`

realm

The AM realm containing the agent profile.

agent-profile-name

The name of the agent profile in AM.

agent-profile-password

The full path to the agent profile password file.

`--changeOwner`

Apache web agent for Unix only: Change the ownership of created directories to the user and group as specified in the Apache configuration file.

To use this option, you must run the `agentadmin` command as the `root` user or with the `sudo` command. If you cannot run the `agentadmin` command as the `root` user or with the `sudo` command, you must change the ownership manually.

`--acceptLicense`

Do not display the license during installation.

`--forceInstall`

If the agent cannot connect to the specified AM server during installation, proceed with a silent installation instead of exiting.

`--n`

(IIS web agent only) List the sites available in an IIS server.

Example:

```
c:\web_agents\iis_agent\bin> agentadmin.exe --nIIS 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
AM 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, refer to 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, refer to 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, refer to 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"

Specify the identity to be added to the directory's ACLs. When used with the `--siteId` option, this option specifies an IIS site ID.

"directory"

Specify the directory that would be modified.

`[--siteId]`

Specify 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.

Example:

```
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 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.

`--k`

Generate a new signing key.

Usage: `agentadmin --k`

Example:

Unix

```
$ cd /web_agents/apache24_agent/bin/
$ ./agentadmin --k
Encryption key value: YWM…5Nw==
```

Windows

```
C:\> cd web_agents{apache_agent_version}\bin
C:\web_agents{apache_agent_version}\bin> agentadmin --k
Encryption key value: YWM…5Nw==
```

`--p`

Use a generated encryption key to encrypt a new password.

Usage: `agentadmin --p encryption-key password`

encryption-key

An encryption key, generated by the `agentadmin --k` command.

password

The password to encrypt.

Examples:

Unix

```
$ ./agentadmin --p "YWM0OThlMTQtMzMxOS05Nw==" "cat newpassword.file"
Encrypted password value: 07b…dO4=
```

Windows

```
C:\path\to\web_agents{apache_agent_version}\bin>
agentadmin.exe --p "YWM0OThlMTQtMzMxOS05Nw==" "newpassword"
Encrypted password value: 07b…dO4=
```

`--V[i]`

Validate the installation. Use this command in conjunction with sustaining to troubleshoot installations.

This command validates the following points:

- The agent can reach the AM server(s) configured in AM Connection URL.
- Critical bootstrap properties are set. For more information, see Configuration location.
- TLS/SSL libraries are available and that SSL configuration properties are set, if the agent is configured for SSL communication.
- The agent can log in to AM to fetch the agent profile.
- The system has enough RAM and shared memory.
- The agent can log in to AM with the provided user and password credentials.
- The agent can decrypt the agent profile password using the encryption key in the `agent.conf` file.
- WebSocket connections are available between the agent and AM.
- The core init and shutdown agent sequences are working as expected. This validation requires the `--Vi` flag.
- (IIS agent only) IIS is configured for running application pools in Integrated mode.

Important

- To prevent service outage or an unresponsive agent, run the command only when the agent instance is not actively protecting a website.
- On Unix, run the command as the same user or group 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 can 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, causing an error such as:

  ```
  … GMT ERROR   [0x7f0c9cf05700:22420]: unable to open event channel
  ```

- Make sure the user running the command has execute permission on the following directories:
  - `/web_agents/apache24_agent/instances/agent_nnn`
  - `/web_agents/apache24_agent/log`

Usage:

```
agentadmin --V[i] agent_instance [user name] [password file] [realm]
```

[i]

(Optional) Ensure that the core init and shutdown agent sequences are working as expected.

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, `/secure-directory/passwd.txt`

realm

(Optional) The realm of the user ID used for the `validate_session_profile` test. For example, `/customers`.

Example:

```
$ ./agentadmin --Vi agent_1 demo passwd.txt /
Saving output to /web_agents/apache24_agent/bin/../log/validate_xxx.log

Running configuration validation for agent_1:

Agent instance is configured with 1 naming.url value(s):
1. https://am.example.com:8443/am is valid
selected https://am.example.com:8443/am 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.

Example:

```
AM Web Agent for IIS Server
     Version: 2023.11
     Revision: xxx
     Build machine: xxx
     Build date: xxx

     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).
```

Installation environment variables

This section lists Web Agents Documentation properties that are configured by environment variables, and set during installation.

Use installation environment variables with the `agentadmin -V[i]` command to validate the installation with different parameters:

Linux

```
$ AM_PROXY_HOST=proxy.host.net AM_PROXY_PORT=8080 AM_PROXY_USER=user AM_PROXY_PASSWORD=pass ./agentadmin --
Vi.
```

Windows

```
C:\>set AM_PROXY_HOST=proxy.host.net
C:\>set AM_PROXY_PORT=8080
C:\>set AM_PROXY_USER=user
C:\>set AM_PROXY_PASSWORD=pass
C:\>agentadmin.exe --Vi agent_1
```

For information about other environment variables, refer to Environment variables.

AM_PROXY_HOST

The proxy FQDN, when AM and the agent communicate through a proxy configured in forward proxy mode.

AM_PROXY_PASSWORD

The agent password, when AM and the agent communicate through a proxy configured in forward proxy mode, and the proxy requires that the agent authenticates using Basic Authentication.

AM_PROXY_USER

The agent username, when AM and the agent communicate through a proxy configured in forward proxy mode, and the proxy requires that the agent authenticates using Basic Authentication.

AM_PROXY_PORT

The proxy port number, when AM and the agent communicate through a proxy configured in forward proxy mode.

APACHE_RUN_USER

The user running the Apache HTTP or IBM HTTP Server. Set this variable before installation when an Apache user is not defined in `httpd.conf`. This can be the case in non Red Hat Enterprise Linux-based distributions.

APACHE_RUN_GROUP

The group to which the user running the Apache HTTP Server or IBM HTTP Server belongs. Set this variable before installation when an Apache group is not defined in `httpd.conf`. This can be the case in non Red Hat Enterprise Linux-based distributions.

AM_SSL_SCHANNEL

Use for Windows only, when TLS/SSL is configured in AM or the agent web server.

A flag for whether the agent installation process should use the Windows Secure Channel API (Schannel):

- `0`: Disable Schannel support. The agent uses OpenSSL libraries instead.

  Make sure the OpenSSL libraries are in the correct location.

- `1`: Enable Schannel support.

AM_SSL_KEY

Use for OpenSSL only, when TLS/SSL is configured in AM or the agent web server.

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:

Unix

```
/opt/certificates/client-private-key.pem
```

Windows

```
C:\Certificates\client-private-key.pem
```

AM_SSL_PASSWORD

Use for OpenSSL only, when TLS/SSL is configured in AM or the agent web server.

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:

Unix

```
$ /path/to/web_agents/agent_type/bin/> agentadmin --p "Encryption Key" “cat certificate_password.file”

Encrypted password value: zck...jtc=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

Use for OpenSSL only, when TLS/SSL is configured in AM or the agent web server.

The 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 at http://www.openssl.org.

For example, `HIGH:MEDIUM`.

AM_SSL_CERT

Use when TLS/SSL is configured in AM or the agent web server.

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 Schannel).

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 `Cert:\LocalMachine\Ca` (Windows with Schannel).

Deploy Web Agents Documentation with Docker

The example in this section provides a Dockerfile and instructions to deploy Apache Web Agents Documentation to extend and protect an application. Adapt the information for other agent containers.

Consider the following limitations:

- The Dockerfile doesn't manage logs, so agent logs are lost when the Docker container is killed. Manage logs independently of the Dockerfile in the following ways, according to your environment:
  - Store logs persistently to a volume
  - Store logs to a host machine
  - Tail logs into STDOUT or STDERR so that Docker can collect the data

- The Dockerfile isn't suitable for local configuration mode and doesn't update bootstrap properties. The agent must be configured to operate in the default Centralized configuration mode. For more information, refer to Location of Agent Configuration Repository.

1. Build a Docker image of your application. This example uses a sample application called `fr-sample-app:1.0`.
2. In Identity Cloud or AM, set up an agent profile and policy. For more information, refer to Identity Cloud's Prepare for installation or AM's Prepare for installation.

   This example uses the following configuration:

   - AM URL: `https://am.example.com:8443/am`
   - AM realm: top-level
   - Agent URL: `http://agent.example.com:80`
   - Agent profile name: `web-agent`
   - Agent profile password: `password`
   - Policy set and policy: Allow HTTP `GET` and `POST` for all authenticated users.
3. Create a local folder for the agent .zip file, the Dockerfile, and the agent profile password—they must be in the same folder. This example uses `/path/to/docker`.
4. Download the agent .zip file to the local folder.
5. Create a file containing the agent profile password. The filename in this example is `agent_secret` and the password is `password`.

   ```
   /path/to/docker$ cat > agent_secret
   password
   CTRL+D
   ```

   Tip

   Although the agent accepts any password length and content, you are strongly encouraged to generate secure passwords. This can be achieved in various ways, for example, by using a password manager.
6. Create the following Dockerfile in `/path/to/docker/Dockerfile`. Arguments are provided by the build command.

```
# Application Docker image
ARG BASE_DOCKER_IMAGE
FROM ${BASE_DOCKER_IMAGE}

# Install and unzip the application, required for unpacking the agent build.
# Not required if the base image is already unzipped.
# For non-Debian Linux distributions, use the appropriate package manager.
RUN apt-get update && \
        apt-get install unzip --no-install-recommends -y && \
        apt-get clean

# Define the build arguments.
# Arguments without default values must be specified in the build command.
ARG AGENT_VERSION
ARG AGENT_ZIP_FILE=web-agent-${AGENT_VERSION}-Apache_v24_Linux_64bit.zip
ARG AGENT_HOME=/opt
ARG AM_URL
ARG APACHE_CONF=/usr/local/apache2/conf/httpd.conf
ARG AGENT_URL=http://agent.dummy.url:80
ARG AGENT_REALM=/
ARG AGENT_PROFILE

# Copy the agent .zip file to the Docker directory where the agent is installed.
COPY ${AGENT_ZIP_FILE} ${AGENT_HOME}/${AGENT_ZIP_FILE}

# Unzip the agent and delete the .zip file
RUN cd ${AGENT_HOME} && \
        unzip ./${AGENT_ZIP_FILE} && \
        rm -rf ./${AGENT_ZIP_FILE}

# Install the agent and mount the file containing the agent password
RUN --mount=type=secret,id=agent_secret,required=true \
        "${AGENT_HOME}"/web_agents/apache24_agent/bin/agentadmin --s \
        "${APACHE_CONF}" \
        "${AM_URL}" \
        "${AGENT_URL}" \
        "${AGENT_REALM}" \
        "${AGENT_PROFILE}" \
        "/run/secrets/agent_secret" \
        --changeOwner \
        --forceInstall
```

7. Find values for the following arguments that correspond to your application and environment:
    - `agent_secret`: The name of the file containing the agent profile password.
    - `BASE_DOCKER_IMAGE`: The name and path to the base image of your application.
    - `AGENT_VERSION`: The agent version in the Docker image.
    - `AGENT_ZIP_FILE`: Name of the agent .zip file. Default: Derived from `AGENT_VERSION`.
    - `AGENT_HOME`: Docker directory where the agent is installed. Default: `/opt`.
    - `AM_URL`: Identity Cloud or AM server URL including port number.
    - `AGENT_URL`: Agent URL. Default: `http://agent.dummy.url:80`.

- `APACHE_CONF` : Path to the Apache server configuration. Default: `/usr/local/apache2/conf/httpd.conf` .
- `AGENT_REALM` : Identity Cloud or AM realm containing the agent profile.
- `AGENT_PROFILE` : Agent profile name. Default `/` .

8. With a Docker daemon running, build the Docker image with the following command, replacing the example values with your own values:

```
/path/to/docker$ docker build --secret id=agent_secret \
  --build-arg BASE_DOCKER_IMAGE=fr-sample-app:1.0 \
  --build-arg AGENT_VERSION=${wpa.vers.ext} \
  --build-arg AGENT_ZIP_FILE=web-agent-${wpa.vers.ext}-Apache_v24_Linux_64bit.zip \
  --build-arg AGENT_HOME=/opt \
  --build-arg AM_URL=https://am.example.com:8443/am \
  --build-arg AGENT_URL=http://agent.example.com:80 \
  --build-arg APACHE_CONF=/etc/httpd/conf/httpd.conf \
  --build-arg AGENT_REALM=/ \
  --build-arg AGENT_PROFILE=web-agent \
  --tag agent-image:${wpa.vers.ext} .

...
 => => writing image sha256:803...ada  0.0s
 => => naming to docker.io/library/web-agent:{wpa_vers_ext}
```

9. Run the container:

```
/path/to/docker$ docker run -it --name apache24-agent -p 80:80 web-agent:${wpa.vers.ext}

... Apache/2.4.58 (Unix) AM Web Agent/${wpa.vers.ext} configured -- resuming normal operations
... Command line: 'httpd -D FOREGROUND'
```

10. Access your application through the agent at http://agent.example.com:80. Access is managed by Identity Cloud or AM according to the policy configured for the agent profile.

    This example displays the Identity Cloud or AM login in page. When you log in as a user, you access the sample application.

Upgrade and rollback

To upgrade or roll back an agent Docker container to a different agent version:

1. Build a new Docker container with the different agent version, using a tag name that corresponds to the version.
2. Replace the Docker image tag in your environment.

Upgrade

Web Agents Documentation supports the following types of upgrade:

- Drop-in software update:

  Usually, an update from a version of Web Agents Documentation to a newer minor version, as defined in Ping Identity Product Support Lifecycle Policy | PingGateway and Agents. For example, update from 2023.9 to 2023.11 can be a drop-in software update.

  Drop-in software updates can introduce additional functionality and fix bugs or security issues. Consider the following restrictions for drop-in software updates:

  - Don't require any update to the configuration
  - Can't cause feature regression
  - Can change default or previously configured behavior **only** for bug fixes and security issues
  - Can deprecate **but not remove** existing functionality

- Major upgrade:

  Usually, an upgrade from a version of Web Agents Documentation to a newer major version, as defined in Ping Identity Product Support Lifecycle Policy | PingGateway and Agents. For example, upgrade from 5.10 to 2023.3 is a major upgrade.

  Major upgrades can introduce additional functionality and fix bugs or security issues. Major upgrades do not have the restrictions of drop-in software update. Consider the following features of major upgrades:

  - Can require code or configuration changes
  - Can cause feature regression
  - Can change default or previously configured behavior
  - Can deprecate **and** remove existing functionality

This guide describes how to upgrade a single ForgeRock Access Management Web Agents Documentation instance. To upgrade sites with multiple Web Agents Documentation instances, one by one, stop, upgrade, and then restart each server individually, leaving the service running during the upgrade.

For information about upgrade between supported versions of Web Agents Documentation, refer to Ping Identity Product Support Lifecycle Policy | PingGateway and Agents.

Example installation for this guide

Unless otherwise stated, the examples in this guide assume the following installation:

- Web Agents Documentation installed on `http://agent.example.com:80`.
- Access Management installed on `http://am.example.com:8088/am`.
- Work in the top-level realm `/`.

If you use a different configuration, substitute in the procedures accordingly.

Drop-in software update

Perform a drop-in software update

1. Read the release notes for information about changes in Web Agents Documentation.
2. Download the agent binaries from the Backstage download site.
3. Redirect client traffic away from the protected website.
4. Stop the web server where the agent is installed.
5. Replace the following executable files in the current installation with the corresponding files in the downloaded binaries, and make sure that they have the same permissions as the original files:
    - Apache Web Agents Documentation:
        - `web_agents/apache24_agent/lib/mod_openam.so`
        - `web_agents/apache24_agent/bin/agentadmin`
    - IIS Web Agents Documentation:
        - `web_agents/iis_agent/lib/mod_iis_openam_64.dll`
        - `web_agents/iis_agent/lib/mod_iis_openam_64.pdb`
        - `web_agents/iis_agent/lib/mod_iis_openam_32.dll`
        - `web_agents/iis_agent/lib/mod_iis_openam_32.pdb`
        - `web_agents/iis_agent/bin/agentadmin.exe`
        - `web_agents/iis_agent/bin/agentadmin.pdb`
    - NGINX Plus Web Agents Documentation:
        - `web_agents/nginx<version-number>_agent/lib/openam_ngx_auth_module.so`
        - `web_agents/nginx<version-number>_agent/bin/agentadmin`

          Use the module in the directory for your NGINX version. The following example is for NGINX Plus 29: `web_agents/nginx29_agent/lib/openam_ngx_auth_module.so`
6. Start the web server where the agent is installed.
7. Validate that the agent is performing as expected in the following ways:
    - Check in `/path/to/web_agents/agent_type/log/system_n.log` that the new version of the agent is running.
    - Go to a protected page on the website and confirm whether you can access it according to your configuration.
    - Check logs files for errors.

> **Tip**
>
> To troubleshoot your environment, run the agentadmin command with the `--v` option.

8. Allow client traffic to flow to the protected website.

Roll back from a drop-in software update

> **Important**
>
> Before you roll back to an earlier version of Web Agents Documentation, consider whether any change to the configuration during or since upgrade could be incompatible with the earlier version.

To roll back from a drop-in software update, run through the procedure in Drop-in software update, but replace the executables with the earlier files, or with those from an earlier version of the agent.

Major upgrade

Perform a major upgrade

1. Read the release notes for information about changes in Web Agents Documentation.
2. Download the agent binaries from the Backstage download site.
3. Plan for server downtime.

   Plan to route client applications to another server until the process is complete and you have validated the result. Make sure the owners of client application are aware of the change, and let them know what to expect.
4. Back up the directories for the agent installation and web server configuration and store them in version control so that you can roll back if something goes wrong:
   - In local configuration mode:

     ```
     $ cp -r /path/to/web_agents/apache24_agent /path/to/backup
     $ cp -r /path/to/apache/httpd/conf /path/to/backup
     ```
   - In centralized configuration mode, back up as described in AM's Maintenance guide.
5. Redirect client traffic away from the protected website.
6. Stop the web server where the agent is installed.
7. Remove the old Web Agents Documentation, as described in Remove Web Agents Documentation.
8. Delete the following shared memory files:
   - `/dev/shm/am_cache_0`
   - `/dev/shm/am_log_data_0`

Depending on your configuration, the files can be named differently.

9. Install the new agent.

   In local configuration mode, provide the `agent.conf` file. For more information, refer to Local configuration (`agent.conf`).

10. Review the agent configuration:

    - In local configuration mode, use the backed-up copy of `agent.conf` file for guidance, the agent's release notes, and AM's Release notes to check for changes. Update the file manually to include properties for your environment.

      > **Important**
      >
      > To prevent errors, make sure the `agent.conf` file contains all required properties. For a list of required properties, refer to Configuration location.

    - In centralized configuration mode, review the agent's release notes and AM's Release notes to check for changes. If necessary, change the agent configuration using the AM admin UI.

11. (If you provided the `agent.conf` file to the installer **and** you are upgrading from an agent version earlier than 4.1.0 hotfix 23)

    Re-encrypt the password specified in the Agent Profile Password:

    1. Obtain the encryption key from the bootstrap property Agent Profile Password Encryption Key in the new `agent.conf` file.
    2. (Unix only) Store the agent profile password in a file; for example, `newpassword.file`.
    3. Encrypt the agent profile password with the encryption key by running the agentadmin command with the `--p` option.

       Unix

       ```
       $ ./agentadmin --p "YWM…Nw=" “cat newpassword.file”
       Encrypted password value: 07b…dO4=
       ```

       Windows

       ```
       $ agentadmin.exe --p "YWM…5Nw=" "newpassword"
       Encrypted password value: 07b…dO4=
       ```

    4. Set the encrypted password as the value of the Agent Profile Password property in the new `agent.conf` file.

12. (NGINX Plus and Unix Apache agents only) Configure shared runtime resources and shared memory. For more information, refer to Configure shared runtime resources and memory.
13. Ensure the communication between AM and the web agent is secured with the appropriate keys. For more information, refer to Configuring AM to sign authentication information.
14. Start the web server where the agent is installed.

    > **Note**
    >
    > Web Agents Documentation 5 changed the default size of the agent 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, refer to Troubleshooting.

15. Validate that the agent is performing as expected in the following ways:
    - Check in `/path/to/web_agents/agent_type/log/system_n.log` that the new version of the agent is running.
    - Go to a protected page on the website and confirm whether you can access it according to your configuration.
    - Check logs files for errors.

    > **Tip**
    >
    > To troubleshoot your environment, run the agentadmin command with the `--v` option.

16. Allow client traffic to flow to the protected website.

Roll back from a major upgrade

> **Important**
>
> Before you roll back to an earlier version of Web Agents Documentation, consider whether any change to the configuration during or since upgrade could be incompatible with the earlier version.

To roll back from a major upgrade, run through the procedure in Major upgrade, but use the backed up directories for the agent installation and web server configuration.

Post update and upgrade tasks

After upgrade, review the what's new section in the release notes and consider activating new features and functionality.

For information about other post-installation options, refer to Post-installation tasks.

User guide

This guide describes how to use ForgeRock Access Management Web Agents Documentation.

About ForgeRock Identity Platform™ Software

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

About Web Agents Documentation

Web Agents Documentation is an Access Management add-on component that operates as a Policy Enforcement Point (PEP) or policy agent for applications deployed in a web server.

Web Agents Documentations intercept inbound requests to applications. Depending on the *filter mode* configuration, Web Agents Documentations interact with AM to:

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

  For information about how to enforce user authentication only, refer to SSO-only mode.

This chapter covers how Web Agents Documentation works and how it protects applications.

Agent components

Web Agents Documentation includes the following main components:

Agent Modules

Intercepts and processes inbound requests to protected resources.

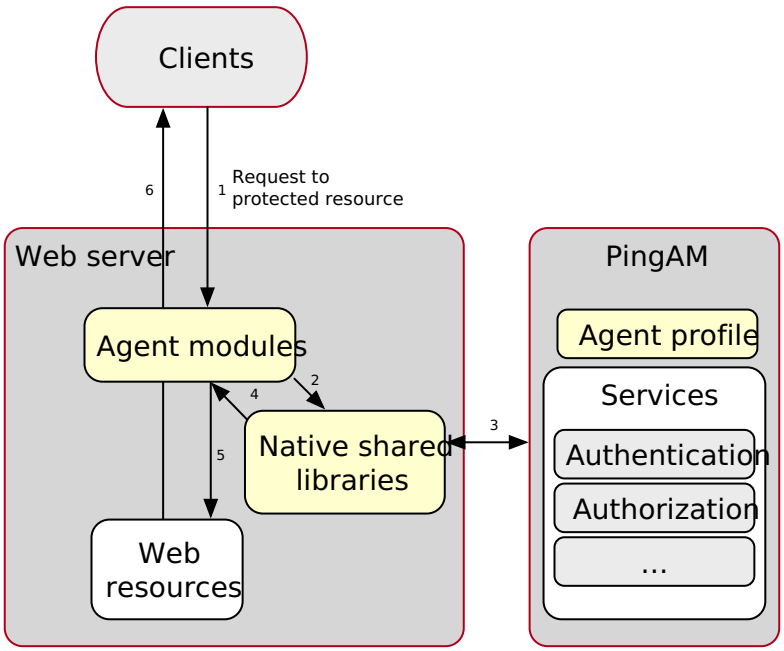
Native Shared Libraries

Enables agents to interact with AM.

Agent Profile

The agent profile is not strictly part of Web Agents Documentation, but plays an important part in the agent operation. It contains a set of configuration properties that define the agent behavior.

The following image shows the Web Agents Documentation components when the agent profile is stored in the AM configuration store:



Configuration location

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

Centralized configuration

AM stores the agent properties in the AM configuration store. Storing the agent configuration centrally allows you to configure your agents using the AM admin UI, the `ssoadm` command, and the REST API.

To access the centralized web agent configuration, on the AM admin UI go to **Realms** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.

Configure properties that are not present in the AM admin UI as *custom properties*, on the **Advanced** tab of the console. For a list of property names, refer to the Properties reference.

When properties and value pairs are defined as custom properties, they take precedence for that property. Therefore, to prevent configuration errors, do not configure a property as a custom property if it has a UI counterpart.

For more information about creating centrally-stored agent profiles, refer to Creating agent profiles.

Local configuration (`agent.conf`)

The Web Agents Documentation installer creates the following file to store agent bootstrap and configuration properties: `/web_agents/agent_type/instances/agent_nnn/config/agent.conf`

The installer populates `agent.conf` with enough information to make the agent start. To manage the configuration, edit `agent.conf` to add properties, remove properties, and change values. You cannot update `agent.conf` using the AM admin UI, the `ssoadm` command, or the REST API.

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

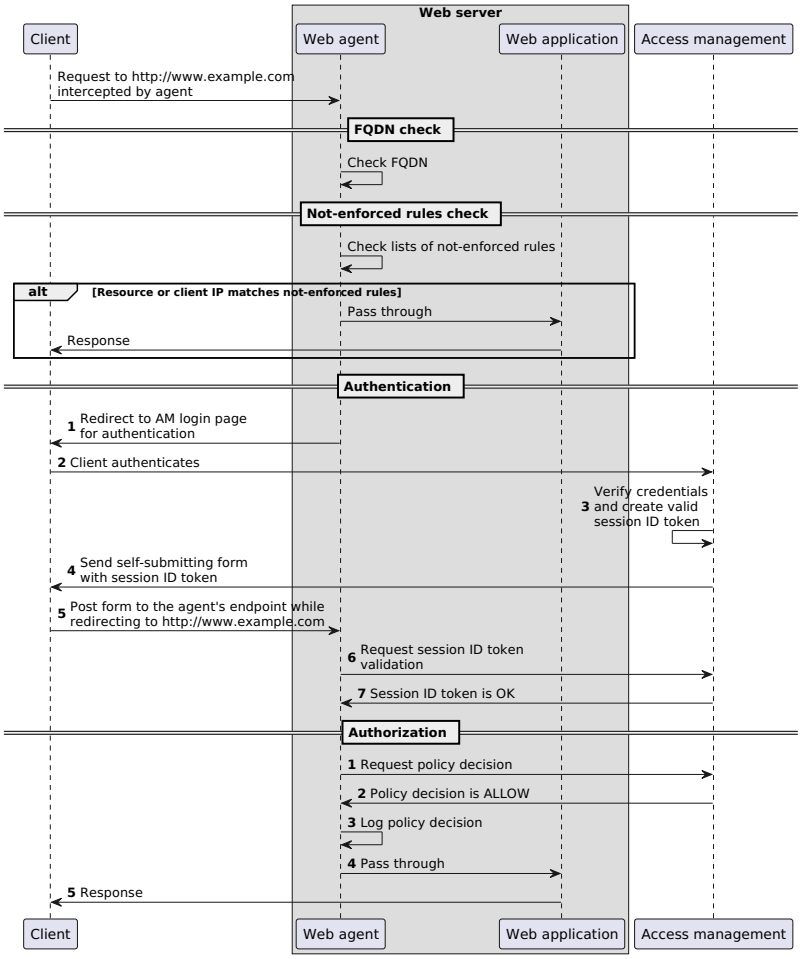
```
# Bootstrap properties
com.sun.identity.agents.config.organization.name = /
com.sun.identity.agents.config.username = ApacheAgentProfile
com.sun.identity.agents.config.password = o7OuvnaDnQ==
com.sun.identity.agents.config.key = OGM...Yg=
com.sun.identity.agents.config.naming.url = https://am.example.com:8443/am

# 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 = 0
com.sun.identity.agents.config.local.logfile = /web_agents/agent_type/instances/agent_name/logs/debug/debug.log
com.sun.identity.agents.config.local.audit.logfile = /web_agents/agent_type/instances/agent_name/logs/audit/audit.log
com.sun.identity.agents.config.debug.level = Error
```

The properties are provided with an example value. For information about each of these properties, refer to the Properties reference.

Request process flow

When a client requests access to an application resource, the Web Agents Documentation intercepts the request. AM then validates the identity of the client, and their authorization to access the protected resource.

The following simplified data flow occurs when an unauthenticated client requests a resource protected by a Web Agents Documentation and AM. The flow assumes that requests must meet the requirements of an AM policy. For a detailed diagram, refer to Single sign-on in AM's *Authentication and SSO guide*.



FQDN check

When FQDN checking is enabled, the agent can redirect requests to different domains, depending on the hostname of the request. For more information, refer to FQDN checking.

Not-enforced rules check

The agent evaluates whether the requested resource or the client IP address matches a not-enforced rule.

- *Alternate flow*. The requested resource or the client IP address matches a not-enforced rule. The agent allows access to the resource.
- *Alternate flow*. The client receives a response from `www.example.com`. The flow ends.
- 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.

For more information, refer to Not-enforced rules.

Authentication

1-2: The client authenticates to AM.

During client authentication, and to protect against reply attacks, the agent issues a pre-authentication cookie, named `agent-authn-tx`. The agent uses the cookie to track the authentication request to AM, and deletes it immediately after authentication.

Depending on the configuration, the agent can either issue a single cookie to track all concurrent authentication requests, or one cookie for each request.

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.

3: AM's authentication service verifies the client credentials and creates a valid OIDC JWT, with session information.

4: AM sends the client a self-submitting form with the OIDC JWT.

5: The client posts the self-submitting form to the agent endpoint, and the Web Agents Documentation consumes it.

6: The agent contacts AM to validate the session contained in the ID token.

7: AM validates the session.

Authorization

1: The agent contacts AM's policy service, requesting a decision about whether the client is authorized to access the resource.

2: AM's policy service returns `ALLOW`.

3: The agent writes the policy decision to the audit log.

4: The agent enforces the policy decision. Because the Policy Service returned `ALLOW`, the agent performs a pass-through operation to return the resource to the client.

5: The client accesses the resource.

Realms

Agent profile realm

The *agent profile realm* is the AM realm in which the agent profile is stored. The agent profile stores a set of configuration properties that define the behavior of the agent.

During agent installation, the installer prompts for the agent profile realm, and populates the property Agent Profile Realm in the bootstrap properties file. By default, the agent profile realm is set to the top-level realm.

The agent profile realm can be different to the user realm and policy evaluation realm. Groups of agents can use the same agent profile realm, which can be separate from the user realm and policy evaluation realm.

For information about creating agent profiles in the top-level realm or other realms, refer to Create agent profiles.

Policy evaluation realm

The *policy evaluation realm* is the realm that the agent uses to request policy decisions from AM. In most circumstances, the policy evaluation realm is the same as the user realm.

The policy evaluation realm is configured by Policy Evaluation Realm, and defaults to the top-level realm. The policy set to use is configured by Policy Set.

In AM, only the top-level realm has a default policy set, called iPlanetAMWebAgentService. If you use a policy evaluation realm that is in a subrealm of the top-level realm, you must also define a policy set and policies in the equivalent realm in AM.

User realm

The *user realm* is the realm in which a user is authenticated. In most circumstances, the user evaluation realm is the same as the policy evaluation realm.

By default, users authenticate to AM in the top-level realm, however, the agent can authenticate users in different realms depending on the request domain, path, or resource.

When a user logs out, the agent maintains the user realm. The agent obtains the realm info from the JWT, if one is available, or by calling `sessioninfo`. When the user logs out, the stored realm is passed to the logout endpoint automatically.

The first time an authenticated user requests a resource from the agent, the agent establishes the user realm from the session. It permanently associates the realm with the session in the session cache. When the session ends, the agent automatically passes the realm to the logout endpoint.

For more information about changing the user realm, refer to Login redirect.

Sessions

On startup, Web Agents Documentation uses the following properties to obtain a session from AM:

- Agent Profile Name
- Agent Profile Password
- Agent Profile Realm

The agent session lifetime is defined by the AM version and configuration, and is essentially indefinite.

For the security of your deployment, set the agent session lifetime as described in Manage Web Agents Documentation sessions.

If you clear agent sessions in the AM admin UI, you can accidentally kill an active agent session. If this happens, the agent detects that its session has expired and automatically obtains a new one.

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 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 (`am.example.com`) is in a different DNS domain than the web agent (`myapp.website.com`), single sign-on would not be possible.

Web Agents Documentation works 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, refer to Single sign-on and Implementing CDSSO in AM's *Authentication and SSO guide*.

Policy enforcement

The agent evaluates policies as defined by the Policy evaluation mode (AM_POLICY_CACHE_MODE) environment variable. For information about caching policy decisions, refer to Caching.

This example sets up AM as a policy decision point for requests processed by Web Agents Documentation. Before you start, install a Web Agents Documentation as described in the Installation, with the following values:

- AM server URL: `http://am.example.com:8088/am`
- Agent URL: `http://agent.example.com:80`
- Agent profile name: `web-agent`
- Agent profile realm: `/`
- Agent profile password: `/secure-directory/pwd.txt`

Enforce a policy decision from AM

1. Using the ForgeRock Access Management docs for information, log in to AM as an administrator, and make sure you are managing the `/` realm.
2. Add a Web Agents Documentation profile:
    1. In the AM admin UI, select **Applications** > **Agents** > **Web**.
    2. Add an agent with the following values:
        - Agent ID: `web-agent`
        - Agent URL: `http://agent.example.com:80`
        - Server URL: `http://am.example.com:8088/am`
        - Password: `password`
3. Add a policy set and policy:
    1. In the AM admin UI, select **Authorization** > **Policy Sets**, and add a policy set with the following values:
        - **Id** : `PEP`
        - **Resource Types** : `URL`
    2. In the policy set, add a policy with the following values:
        - **Name** : `PEP-policy`
        - **Resource Type** : `URL`
        - **Resources** : `*://*:*/*`
    3. On the **Actions** tab, add actions to allow HTTP `GET` and `POST`.

4. On the **Subjects** tab, remove any default subject conditions, add a subject condition for all `Authenticated Users`.

4. Assign the new policy set to the agent profile:

    1. In the AM admin UI, Select **Applications** > **Agents** > **Web**, and select your agent.
    2. On the agent page, select the **AM Services** tab.
    3. Set **Policy Set** to `PEP`, and then click **Save**.

5. Test the setup:

    1. In the AM admin UI, select **Identities** > **Add Identity**, and add a user with the following values:
        - **Username** : `demo`
        - **First name** : `demo`
        - **Last name** : `user`
        - **Email Address** : `demo@example.com`
        - **Password** : `Ch4ng31t`
    2. Log out of AM, and clear any cookies.
    3. Go to `http://agent.example.com:80`. The AM login page is displayed.
    4. Log in to AM as user `demo`, password `Ch4ng31t`, to access the web page protected by the Web Agents Documentation.

Retrieve advice or response attributes from policy decisions

When AM makes a policy decision, it communicates an entitlement to the agent, which can optionally include advice and response attributes.

When AM denies a request with advice, the agent uses the advice to take remedial action. For example, when AM denies a request because the authentication level is too low, it can send advice to increase the authentication level. The agent then prompts the user to reauthenticate at a higher level, for example, by using a one-time password.

When AM allows a request it can include the following types of response attributes in the entitlement:

- Subject response attributes: Any LDAP user attribute configured for the identity store where AM looks up the user's profile. For more information, refer to Identity stores in AM's *Setup guide*.

  The agent adds the listed attributes to the response.

- Static response attributes: Any key:value pair, for example, `FrequentFlyerStatus` : `gold`.

  Depending on the value of Response Attribute Map, and Response Attribute Fetch Mode, the agent adds the listed attributes to HTTP headers or HTTP cookies in the response.

This example builds on the example in Enforce a policy decision from AM. Set up and test that example first.

1. Configure subject response attributes and static response attributes in the AM policy you created earlier:

    1. In the AM admin UI, select the `PEP-policy`, and go to the **Response Attributes** tab.

2. In the **SUBJECT ATTRIBUTES** frame, select one or more of the available attributes. For example, select `cn`.
3. In the **STATIC ATTRIBUTES** frame, add a response attribute pair. For example, add the following pair:
   - **PROPERTY NAME**: `FrequentFlyerStatus`
   - **PROPERTY VALUE**: `gold`
4. Click **Save Changes**.

2. In the AM admin UI, select the `web-agent` you created earlier.

   The agent must use the AM policy set and realm where the response attributes are configured.

   If the response attributes are not present in the policy decision from AM, the agent does not create the corresponding HTTP header or cookie.

3. In the **Application** tab, set **Response Attribute Fetch Mode** to `HTTP-HEADER` or `HTTP-COOKIE` to select whether to map response attribute names to HTTP header names or HTTP cookie names.

   For more information, refer to Response Attribute Fetch Mode.

4. In the **Response Attribute Map** field, map the subject response attributes you selected in AM:
   - **Key**: `cn`
   - **Value**: `CUSTOM-name`

     The name of the AM response attribute `cn` is mapped to the HTTP header or cookie called `CUSTOM-name`. The value is taken from the user profile.

     For more information, refer to Response Attribute Fetch Mode.

5. In the **Response Attribute Map** field, map the static response attributes you added in AM:
   - **Key**: `FrequentFlyerStatus`
   - **Value**: `CUSTOM-flyer-status`

     The name of the AM response attribute `FrequentFlyerStatus` is mapped to the HTTP header or cookie called `CUSTOM-flyer-status`. The value is `gold`.

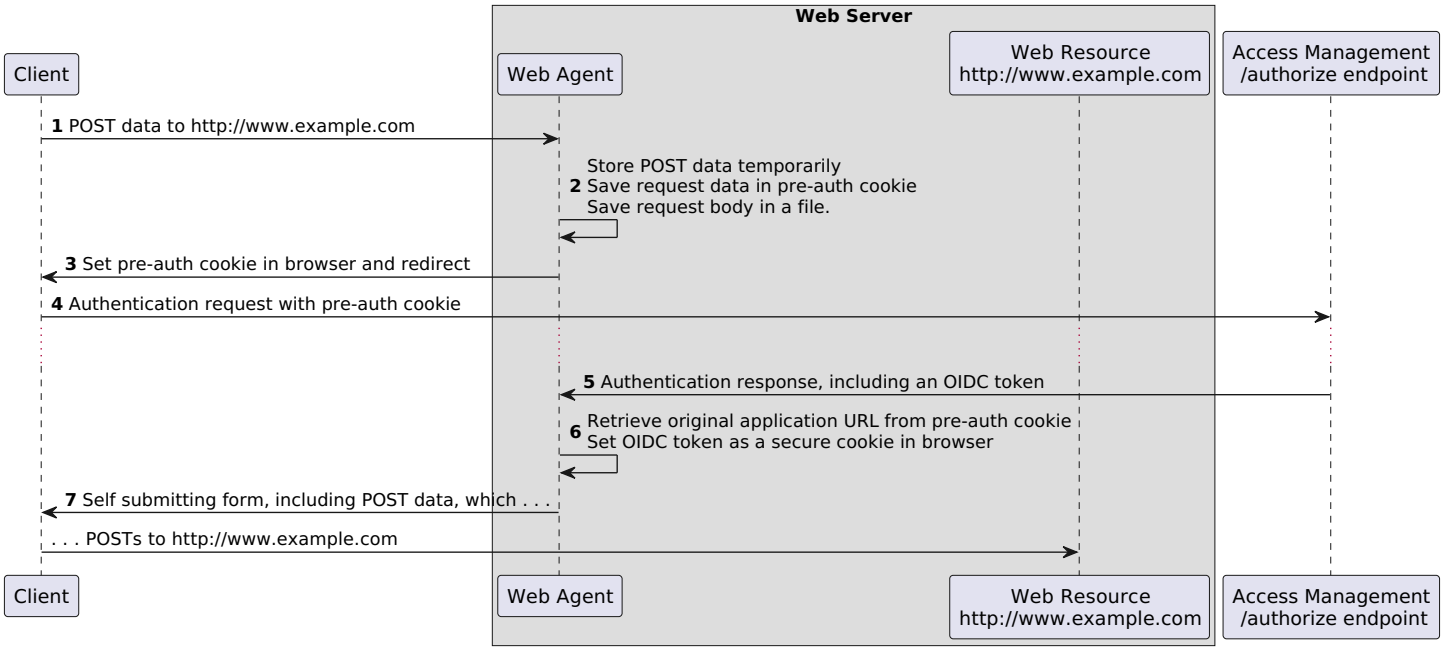
     For more information, refer to Response Attribute Map

POST data preservation

Use POST data preservation in environments where clients submit form data, and have short-lived sessions.

When POST data preservation is enabled, and an unauthenticated client posts data to a protected resource, the agent stores the POST data temporarily, and redirects the client to the login screen. The data can be any POST content, such as HTML form data or a file upload. After successful authentication, the agent recovers the stored POST data, and automatically submits it to the protected resource.

The following image shows a simplified data flow, when an unauthenticated client POSTs data to a protected web application:



Web Agents Documentation guarantees the integrity of the data, and the authenticity of the client as follows:

1. An unauthenticated client requests a POST to a protected resource.
2. The agent stores the POST data temporarily in the directory defined by POST Data Storage Directory, and saves data about the request in a standard pre-authentication cookie.
3. The agent sets a pre-authentication cookie in the browser, and redirects to the `/authorize` endpoint in AM.
4. The client authenticates with AM.
5. AM sends an authentication response to the registered redirect URI.
6. The agent retrieves the original application URL from the pre-authentication cookie, and replays the request with its body content to the server. The authentication response includes an OIDC token, which the agent sets as a secure cookie in the browser.
7. The agent sends a self-submitting form to the client browser, that includes the form data the user attempted to post in step 1. The self-submitting form POSTs to the protected resource.

For information about configuration properties, refer to POST data preservation.

Security considerations for POST data preservation

POST data is stored temporarily in the agent file system before a user is authenticated. Therefore, any unauthenticated user can POST a file that is then stored by the agent. Consider the following points when you configure POST data preservation:

- Payloads from unauthenticated users are stored in the agent files system. If your threat evaluation does not accept this risk, do not use POST data preservation; set Enable POST Data Preservation to `false`.
- By default, POST data is stored in the installation directory, `/path/to/web_agents/agent_type/instances/agent_n/pdp-cache`. To store POST data in a dedicated directory, set POST Data Storage Directory. Make sure that the new directory has the correct read/write permissions for the ID that the server uses.

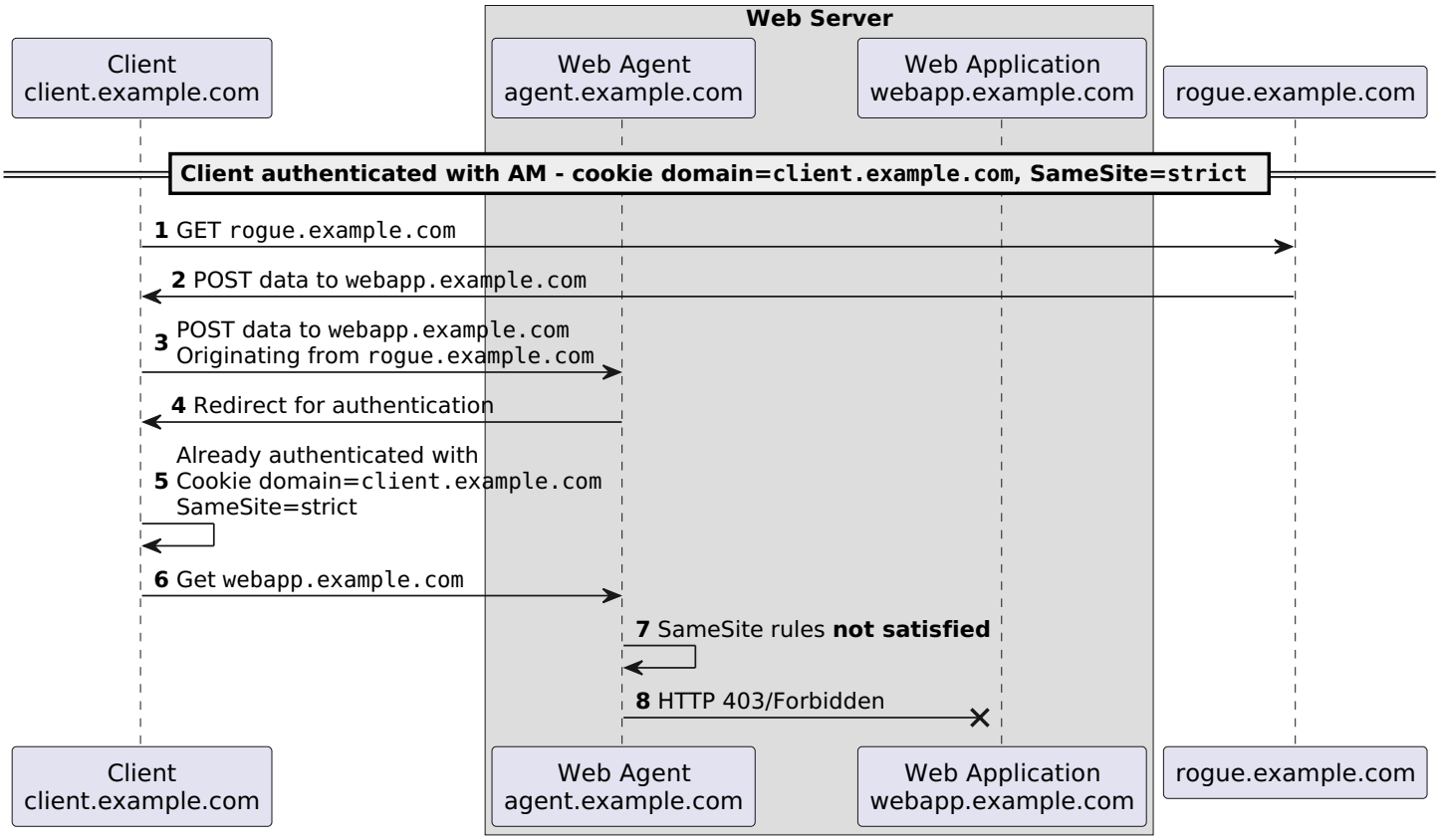
- Set the directory permissions to minimize the following risks:
  - Permissive access to POST data.
  - Leakage of personally identifiable information (PII).
- POST data is stored for the time defined by POST Data Entries Cache Period and then deleted. To identify threats in POST data before it is deleted, make sure Intrusion Detection Systems inspect the data within the specified time.

Defend against CSRF attacks when using POST data preservation

> **Warning**
>
> Cross-site request forgery attacks (CSRF or XSRF) can be a cause of serious vulnerabilities in web applications. It is the responsibility of the protected application to implement countermeasures against such attacks, because Web Agents Documentation cannot provide generic protection against CSRF. ForgeRock recommends following the latest guidance from the OWASP CSRF Prevention Cheat Sheet.
> When POST data preservation is enabled, captured POST data that is replayed appears to come from the same origin as the protected application, not from the site that originated the request. Therefore, CSRF defenses that rely solely on checking the origin of requests, such as SameSite cookies or Origin headers, are not reliable.
> To defend against CSRF attacks when POST data preservation is enabled, the agent uses a secure cookie and a nonce. The nonce must correspond to the authentication response from AM. This defense during authentication is specified in Cross-Site Request Forgery.
> ForgeRock strongly recommend using token-based mitigations against CSRF, and relying on other measures only as a defense in depth, in accordance with OWASP guidance.

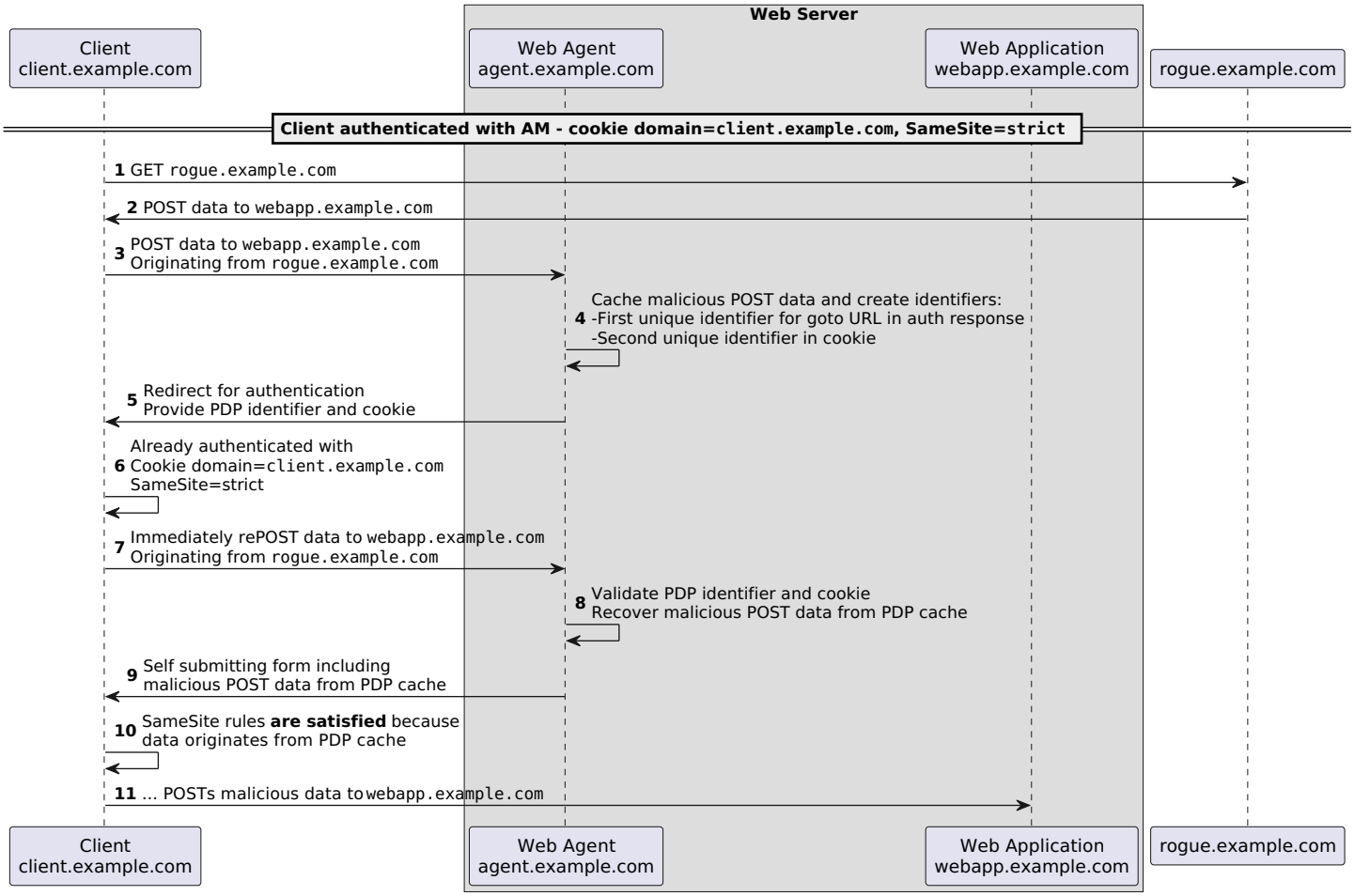
CSRF attack when POST data preservation is disabled

The following image shows a simplified data flow during a CSRF attack on an authenticated client when POST data preservation is disabled. In this limited scenario, the agent SameSite setting is enough to defend the web application:



CSRF attack when POST data preservation is enabled

The following image shows a simplified data flow during a CSRF attack on an authenticated client when POST data preservation is enabled. In this scenario, the SameSite setting **is not** enough to defend the web application:



Login redirect

When an unauthenticated user requests access to a protected resource, the agent redirects the user to log in. The choice of the login endpoint, and the parameters it receives, is defined by the login redirect mode and whether the agent accepts SSO tokens and ID tokens as session cookies.

Configure login redirect options as follows:

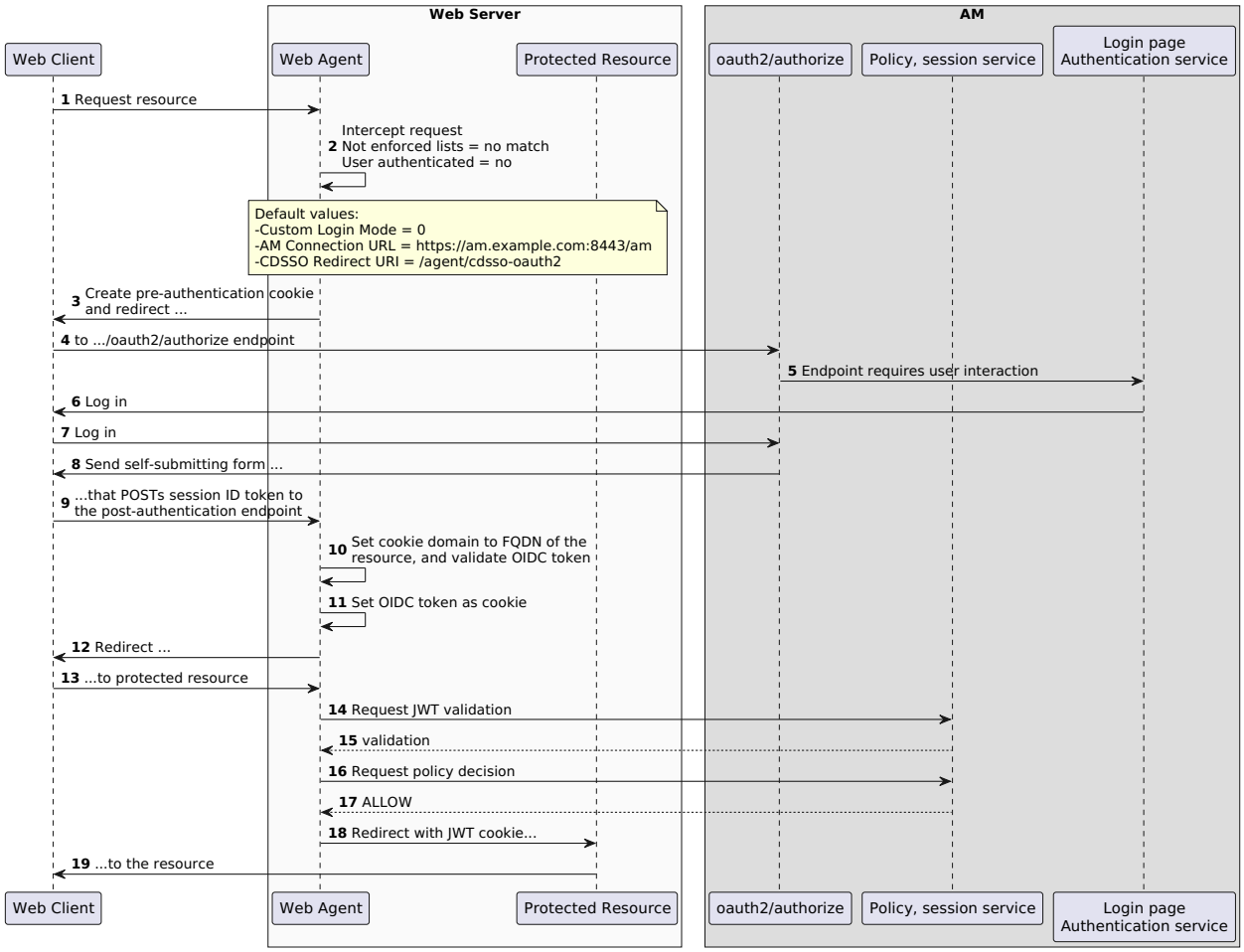
		Accept SSO Token	
		0	1
Enable Custom Login Mode	0	Default login redirect: • Don't configure AM Login URL. • (Optional) Configure AM Conditional Login URL to point to the AM admin UI or the AM `oauth2/authorize` endpoint.	

		• Doesn't accept SSO tokens as session cookies.	• Accepts SSO tokens and ID tokens as session tokens during and after the login flow. • Doesn't convert SSO tokens to ID tokens.
	1	Same domain custom login redirect or Cross domain custom login redirect: • Redirects login to the originally requested resource. • Converts SSO tokens to ID tokens. • Configure AM Login URL to point to a custom login page. • (Optional) Configure AM Conditional Login URL to point to the same URL as AM Login URL with additional parameters such as the login realm. Requests are logged conditionally to this URL. • Don't configure AM Conditional Login URL or AM Login URL to point to the AM admin UI or the AM `oauth2/authorize` endpoint.	• Not supported.
	2	• Not supported.	Same domain custom login redirect: • This non-standard flow has limitations. Use only for environments migrating from earlier versions of the agent. • Redirects login with a `goto` query parameter to the originally requested resource • Accepts SSO tokens • Configure AM Login URL to point to a custom login page • (Optional) Configure AM Conditional Login URL to point to the same URL as AM Login URL with additional parameters such as the login realm. Requests are logged conditionally to this URL. • Don't configure AM Conditional Login URL or AM Login URL to point to the AM admin UI or the AM `oauth2/authorize` endpoint.

Default login redirect

In default login redirect, the agent redirects users for authentication to a page on the AM admin UI. The agent uses OpenID Connect ID JTWs as session tokens. Default login always redirects users to the top-level realm, irrespective of the user realm.

The following image shows the flow of data during a default login redirect. When an unauthenticated user requests access to a protected resource. The agent wraps the SSO session token inside an OpenID Connect JWT. Authentication requires access to the `/oauth2/authorize` endpoint, which invokes the AM admin UI and other endpoints such as `oauth2/authorize`, `json/authenticate`, `json/sessions`, `json/serverinfo`, and `XUI/*`.



Custom login redirect

In custom login redirect, the agent can redirect login in the following ways:

- Redirect login to custom login pages, in the same or a different realm
- Conditionally redirect login to different AM instances, AM sites, or authentication realms, based on the request URL.
- Use AM-specific SSO tokens as session tokens

Same domain custom login redirect

The agent redirects unauthenticated users to a custom login page in the same domain, adding the `original_request_url` parameter to the redirect. The parameter records the requested URL, which can then be used by custom login application or page.

The custom login page posts an SSO token to `agent/custom-login-response`, with the realm as an optional parameter, and sets an SSO token in the same domain as the agent.

The agent attempts to validate the SSO token against the AM endpoints.

At the end of the login flow, the agent can do the final redirect to the protected resource, or to the originally requested resource, with a `goto` query parameter

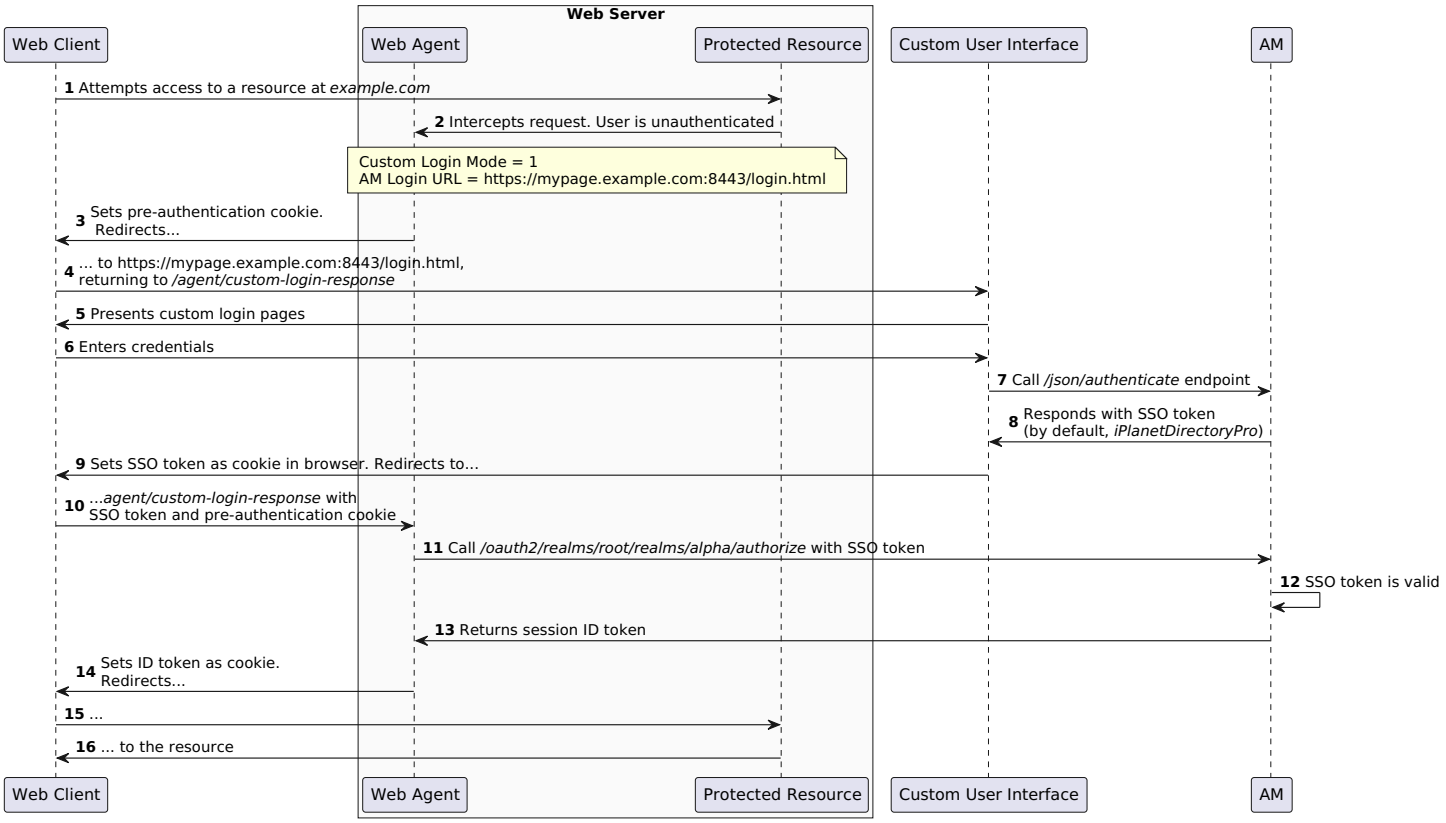
Same domain custom login redirect with final redirect to the protected resource

Set the following properties:

- Enable Custom Login Mode as `1`, to use the OIDC-compliant custom login redirection mode
- AM Login URL, to the URL of the custom login page

If the custom login page is a part of the agent's protected application, add the custom login pages to the not-enforced lists.

The following image shows the data flow for a custom login redirect when the custom login pages are in the same domain as the agent, and the agent redirects the the request to the originally requested resource.



Cross-domain custom login redirect

The agent redirects unauthenticated users to a custom login page in a different domain, including the `original-request-uri` parameter in the redirect. The parameter records the requested URL, which can then be used by custom login application or page.

The custom login page provides a `custom-login-response`, and sets an SSO token, which can be accessed only in that domain. Because the agent can't access the cookie, it redirects to AM for the Default login redirection mode.

Depending on your environment, the agent can contact AM to validate the cookie even if it can't detect it. In other cases, you need to configure an additional property.

If AM can validate the SSO token, it returns an ID token as part of the default redirection login flow.

Consider the following points:

- Ensure the login pages **don't** set the SSO token cookie with the `SameSite=Strict` attribute.
- If AM can't validate the SSO token (for example, because it can't recognize the domain set for the cookie), it redirects 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:

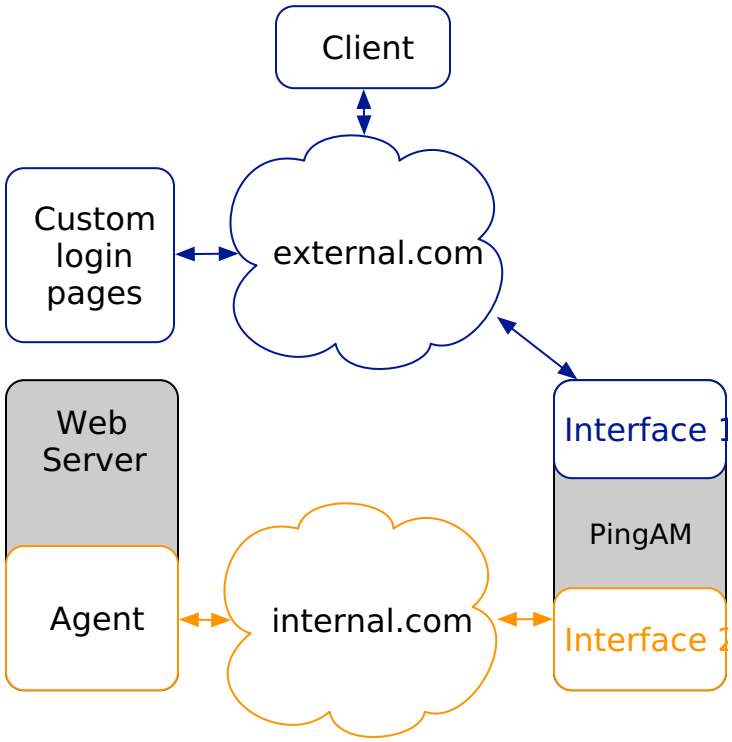
Cross-domain custom login redirect on a shared network

On a 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.

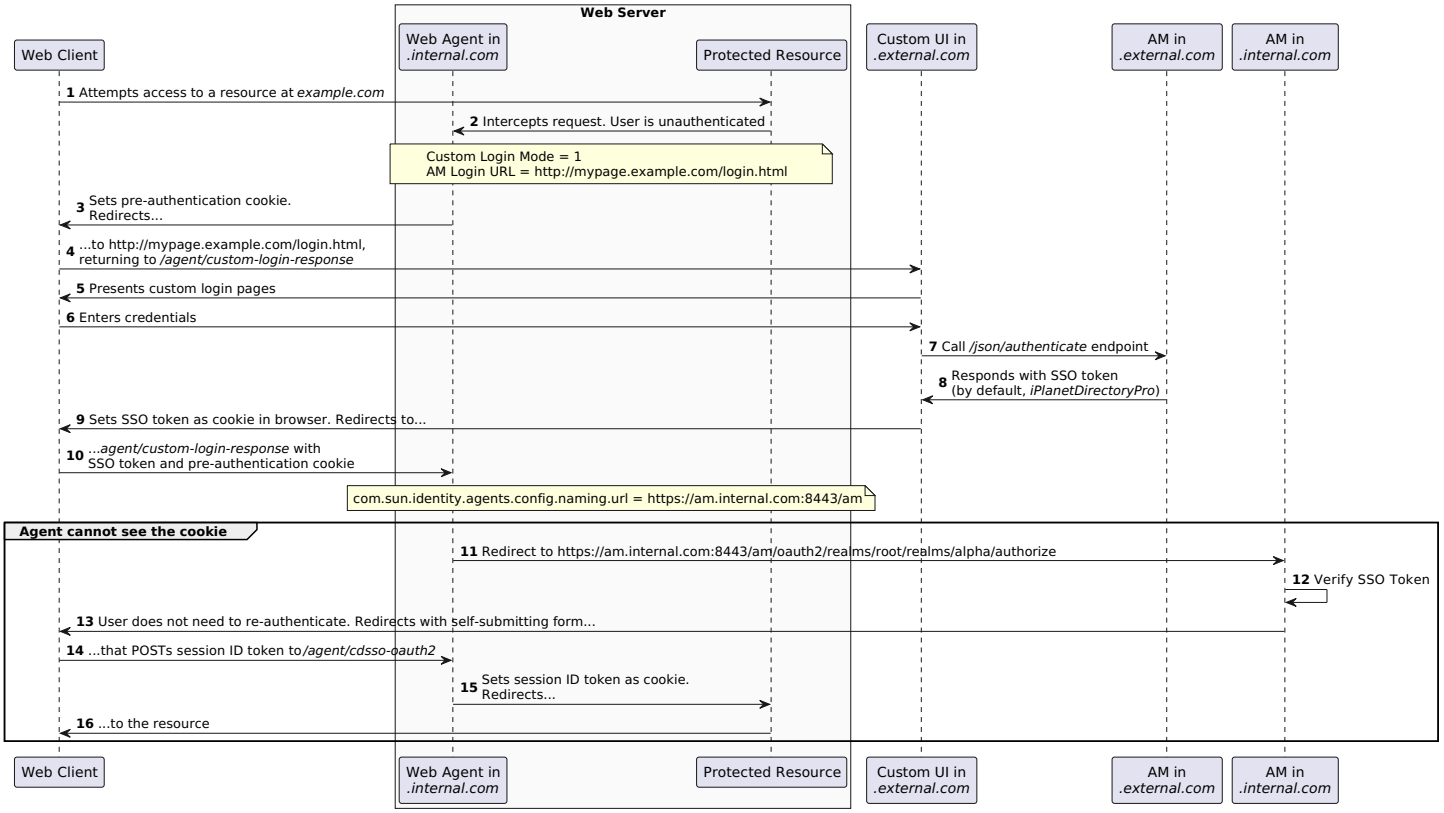
Use the following properties to configure this scenario:

- Enable Custom Login Mode
- AM Login URL

The following image illustrates the environment. The web server housing the protected resources can be connected to the external network in different ways; with two interfaces, or through a proxy. It isn't important for custom login, so it isn't shown.



The following image shows the data flow:



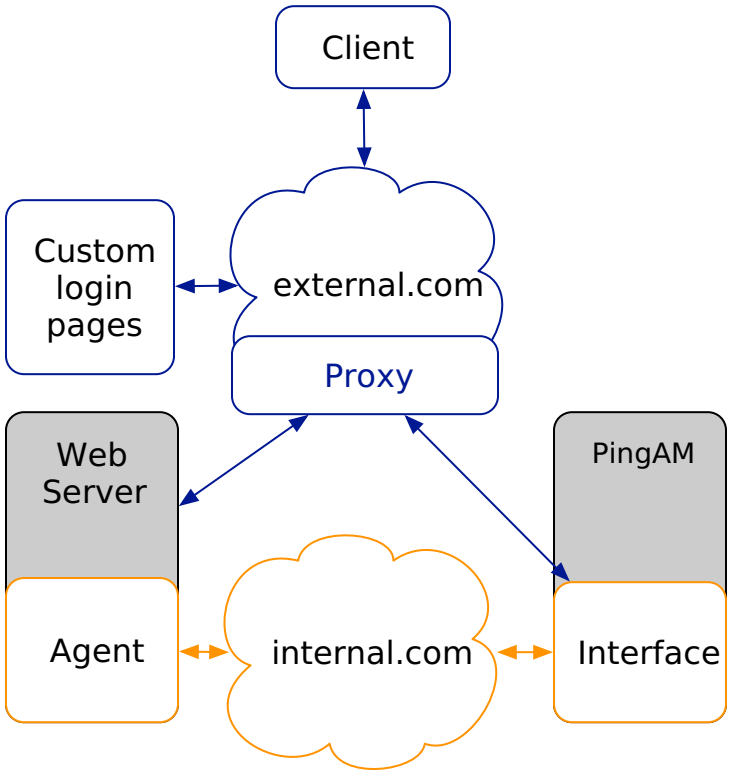
Cross-domain custom login redirect with AM behind a proxy

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.

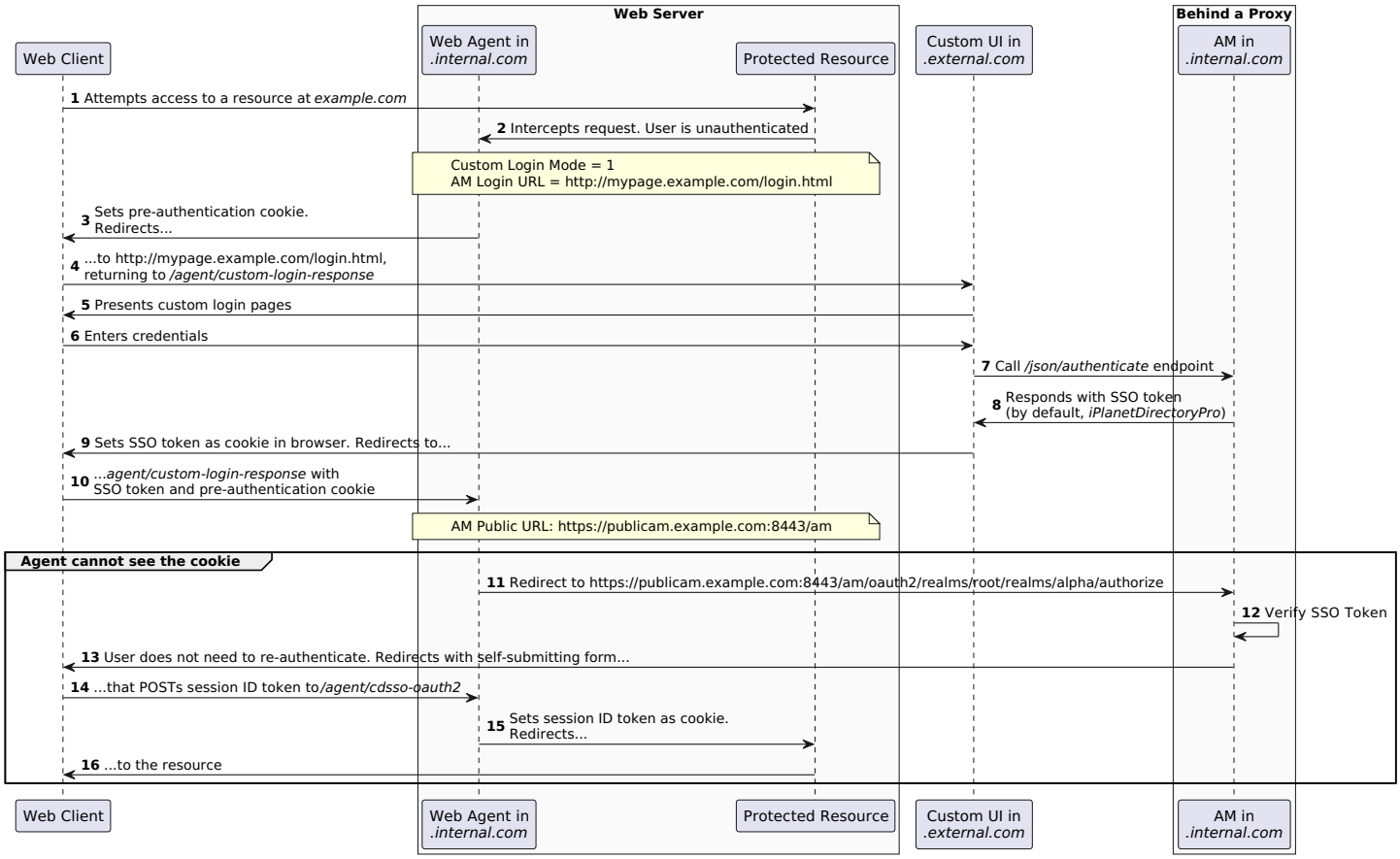
Use the following properties to configure this scenario:

- Enable Custom Login Mode
- AM Login URL
- Public AM URL

The following image illustrates the environment. The web server where the protected resources are can be connected to the external network in different ways; with two interfaces, or through a proxy. It isn't important for custom login, so it isn't shown in the following diagram:



The following image shows the data flow:



Conditional login redirect

Use conditional redirects to redirect the end user to different AM instances or sites, or to different custom pages, depending on the incoming request URL.

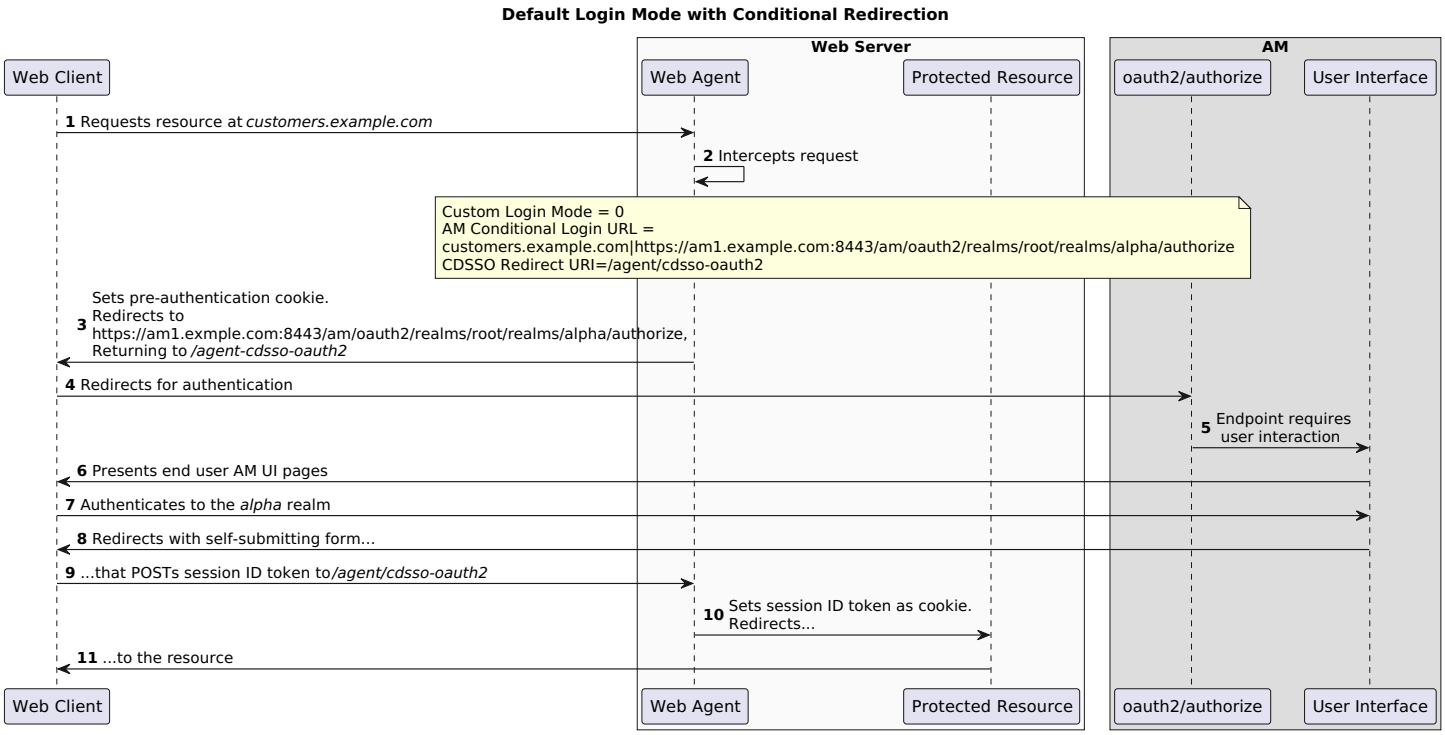
To redirect login to a specific authentication tree, add the `service` parameter, for example: `http://am.example.com:8088/am/oauth2/realms/root/realms/alpha/authorize?service=myTree` would authenticate users using an authentication tree called `myTree`.

Conditionally redirect login by domain name

When the incoming request URL matches a specified domain name, configure the following properties to redirect to a specified URL:

- Enable Custom Login Mode = 0
- AM Conditional Login URL
- CDSSO Redirect URI

The following image shows the data flow when requests to the domain `customers.example.com` are redirected to the `alpha` realm. Other requests are redirected to the top-level realm.



Conditionally redirect login by matching regular expressions

When the incoming request URL matches a regular expression, configure the following properties to redirect to a specified URL:

- Enable Custom Login Mode = 0
- Regular Expression Conditional Login Pattern
- Regular Expression Conditional Login URL

In the following example, when the request matches the regular expression `.*shop`, the agent redirects it to the `alpha` realm for authentication:

```
org.forgerock.openam.agents.config.allow.custom.login = 0
org.forgerock.agents.config.conditional.login.pattern[0] = .*shop
org.forgerock.agents.config.conditional.login.url[0] = https://am.example.com/am/oauth2/realms/root/realms/alpha/
authorize
```

Redirect login to a custom URL configured in AM

AM's **OAuth2 Provider** service can be configured to use a custom URL to handle login, to override the default AM login page. When a custom login page is configured in AM, configure the agent to ensure that it redirects the login to that page.

1. In the AM admin UI, go to **Services** > **OAuth2 Provider** > **Advanced** > **Custom Login URL Template**, and note the custom URL.
2. Go to **Applications** > **Agents** > **Web**, and select your Web Agents Documentation.

3. On the **AM Services** tab set the following properties:
    - Enable Custom Login Mode: Set to `1`.
    - AM Conditional Login URL: Set to the custom URL in step 1.

Logout

This section describes how to trigger a logout based on the properties of a request, and how to redirect users after logout to a specified logout resource.

The agent maintains the user realm for each session, obtaining it from the JWT or `sessioninfo` endpoint. When a user logs out, the agent automatically passes the stored realm to the logout endpoint.

Web Agents Documentation provides the following properties to configure logout:

Task	Property	Description
Trigger logout	• Enable Regex for Logout URL List	A flag to evaluate expressions in Logout URL List as regular expressions instead of as wildcard expressions.
	• Logout URL List	An expression that resolves to one or more application logout URLs. When the end user accesses a logout URL, the agent triggers a logout flow. The web server must be able to handle the logout URLs. Expressions can be wildcard expressions, Perl-compatible regular expressions, or ECMAScript-compatible (IIS) regular expressions.
	• Agent Logout URL Regular Expression (deprecated)	A Perl-compatible or ECMAScript-compatible (IIS) regular expression that resolves to one or more application logout URLs. This property is deprecated; use Logout URL List instead. If this property is used, it is evaluated before Enable Regex for Logout URL List in the logout flow.
Manage logout	• AM Logout URL	A URL to manage the logout.
	• Enable Invalidate Logout Session	A flag to kill the AM session when the value of Logout URL List is a page in your application and your application doesn't handle the session invalidation process.

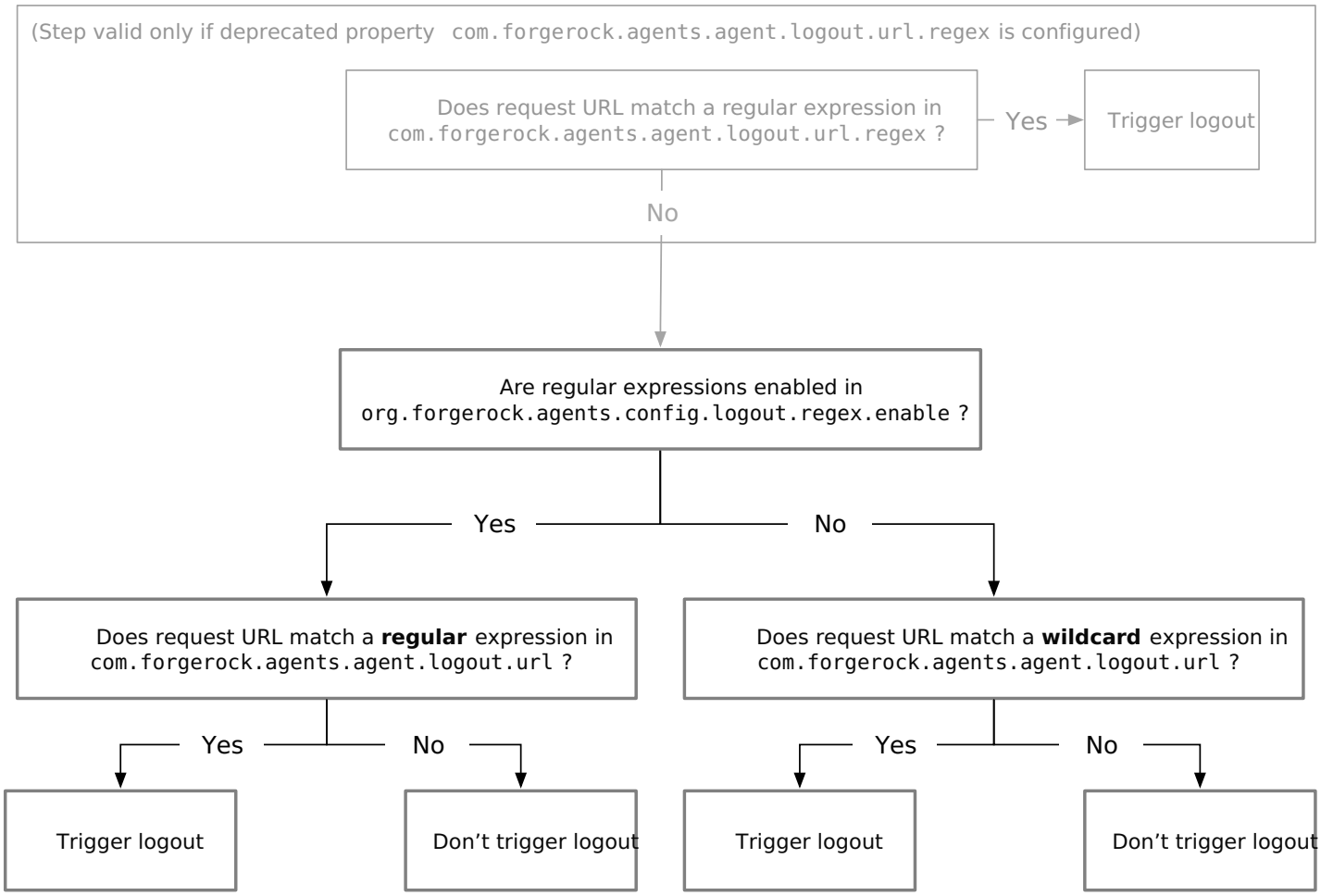
Task	Property	Description
	• Reset Cookies on Logout List	A list of cookies to reset on logout.
Redirect after logout	• Logout Redirect URL	A URL to which the user is redirected after logout.
	• Disable Logout Redirection	A flag to disallow redirect after logout. When `true`, the agent performs session logout in the background and continues processing access to the current URL.

Trigger logout with a URL

The agent triggers logout according to the configuration of the following properties:

- Logout URL List
- Agent Logout URL Regular Expression (deprecated)
- Enable Regex for Logout URL List

The following image shows how the properties are applied:



Examples

- The following example triggers logout when the request URL is from `*/bank/log-me-out`:

```
org.forgerock.agents.config.logout.regex.enable=false
com.forgerock.agents.agent.logout.url=*//*:*/bank/log-me-out
```

- The following example triggers logout when the request URL is anywhere in the path `*/logout/*`:

```
org.forgerock.agents.config.logout.regex.enable=false
com.forgerock.agents.agent.logout.url=*//*:*/*/logout/*
```

- The following example triggers logout when:
  - The request URL is on the path `*/protectedA/*` or `*/protectedB/*`,
  - The request URL contains a second query section that includes `op=logout` anywhere in the parameter list

```
org.forgerock.agents.config.logout.regex.enable=true
com.forgerock.agents.agent.logout.url=https:\/\/example.domain.com:443\/(protectedA|protectedB)\?
(.*\&)*op=logout(\&.*)*$
```

Redirect logout to a landing page

The agent redirects users to a specified resource after logout when the following properties are configured:

- Disable Logout Redirection
  - Set to `false` to allow redirect on logout. The agent appends a goto parameter to the logout URL with the value of the Logout Redirect URL.
  - Set to `true` to disable redirect in logout. The agent doesn't perform the last redirection and leaves the web client on the logout page.

    Consider setting Enable Invalidate Logout Session to `true` when this property is `true`.
- Logout Redirect URL

  Specify an HTML page to which the agent redirects the end user on logout. The page must be available in your web server.

Depending on the redirect URL, perform this additional configuration:

- Add the URL to the Not-Enforced URL List.
- If the URL doesn't perform a REST logout to AM, set Enable Invalidate Logout Session to `true`.
- If the URL isn't relative to AM, or in the same scheme, FQDN, and port, add it to the AM validation service.

  For more information, refer to Identity Cloud's Configure trusted URLs or AM's Configure trusted URLs.

End AM sessions on logout

Configure one of the following properties to manage logout:

- AM Logout URL to redirect the request to AM's `/am/UI/Logout` endpoint. This is the default value.
- Enable Invalidate Logout Session
  - Set to `true` when Logout URL List is configured with a page in your application, but your application *doesn't handle* the session invalidation process.

    The agent doesn't add the `goto` parameter to the URL, and the web client remains in the logout page.

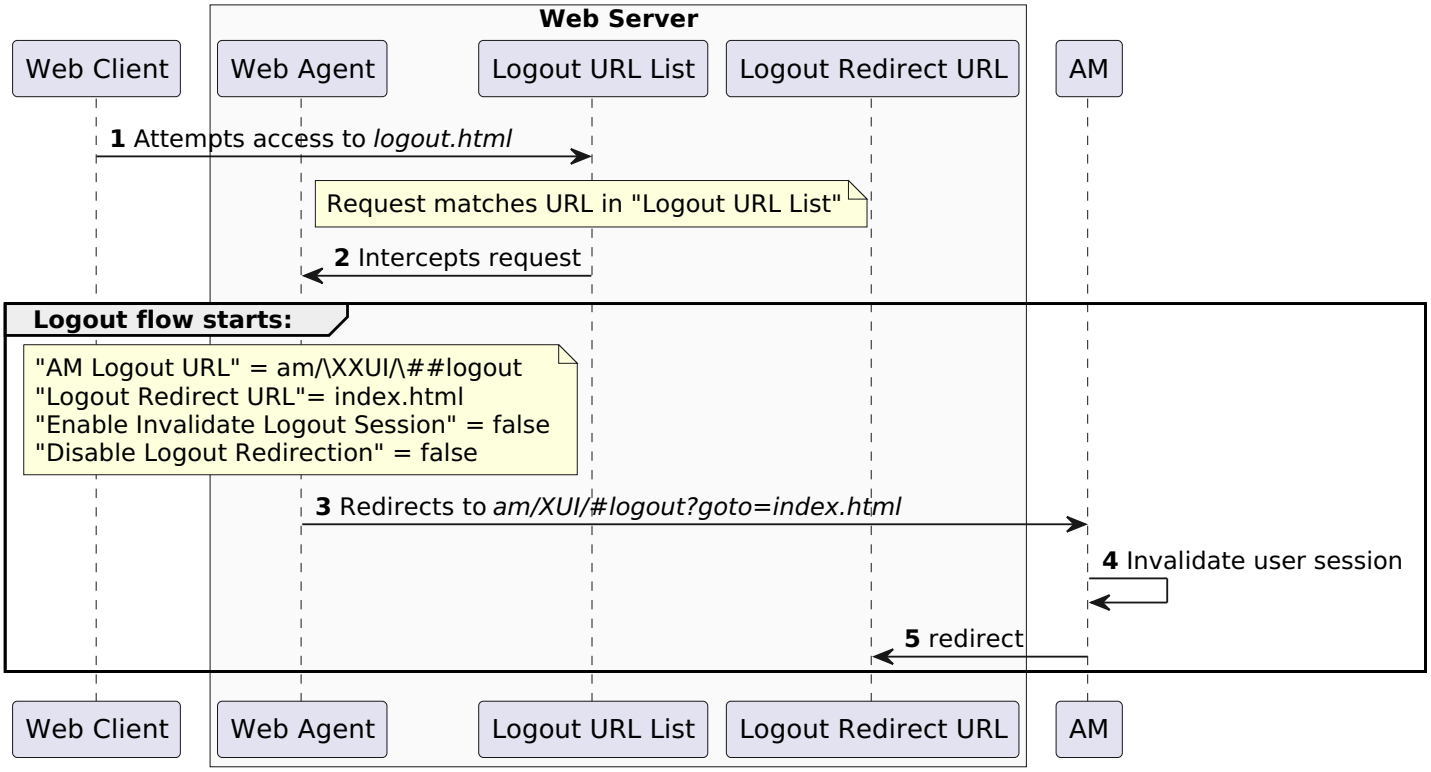
    The agent deletes its own JWT cookie and invalidates the AM session.
  - Set to `false` when Logout URL List has any of the following values:
    - A SAML v2.0 logout page.
    - An AM logout page.
    - A page in your application, and your application *does handle* the session invalidation process.

The agent deletes its own JWT cookie but doesn't invalidate the AM session.

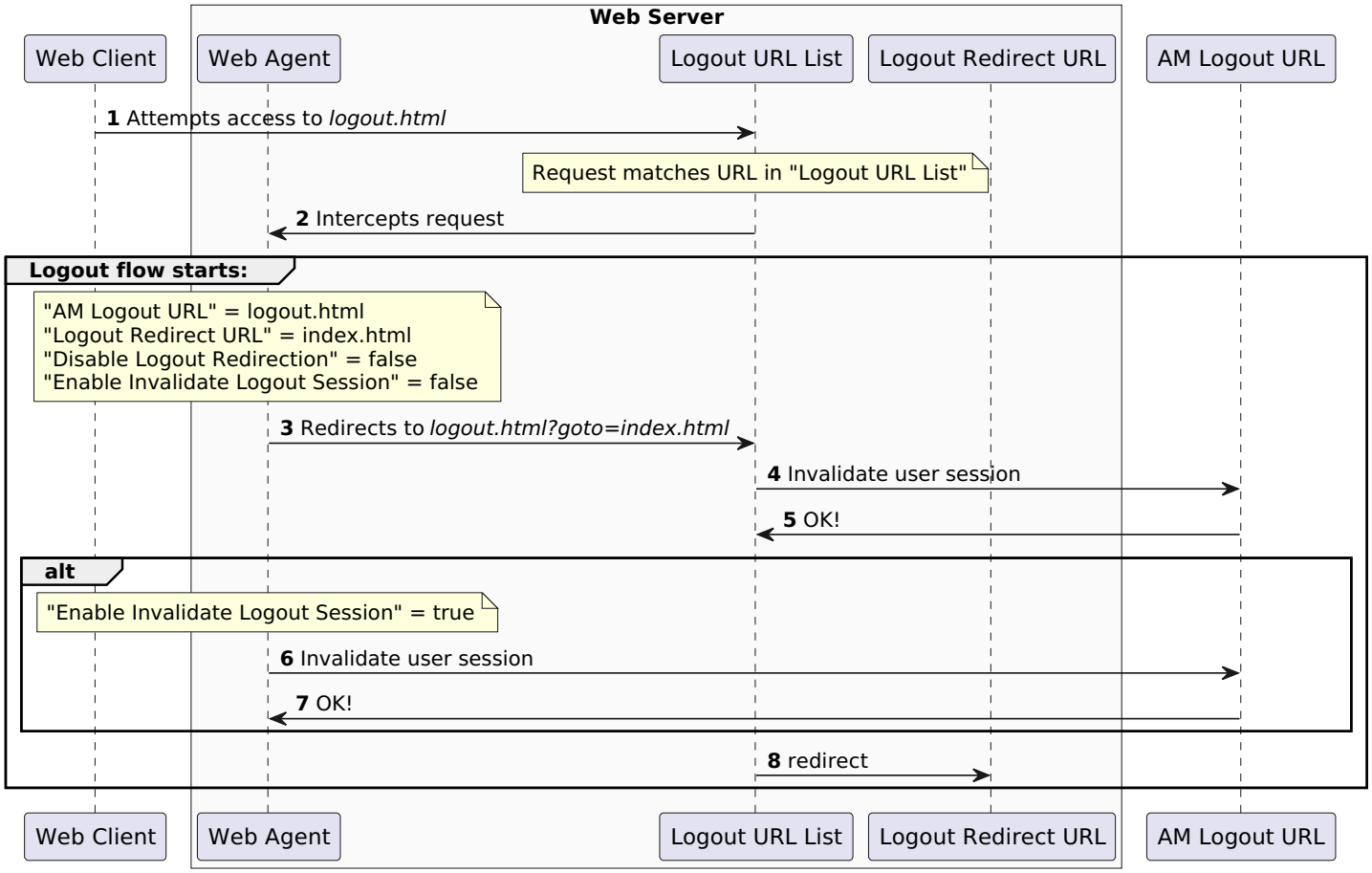
Reset cookies on logout

To reset specified cookies during logout, configure Reset Cookies on Logout List.

Example logout flow with AM as the logout page



Example logout flow with the application serving the logout page



Not-enforced rules

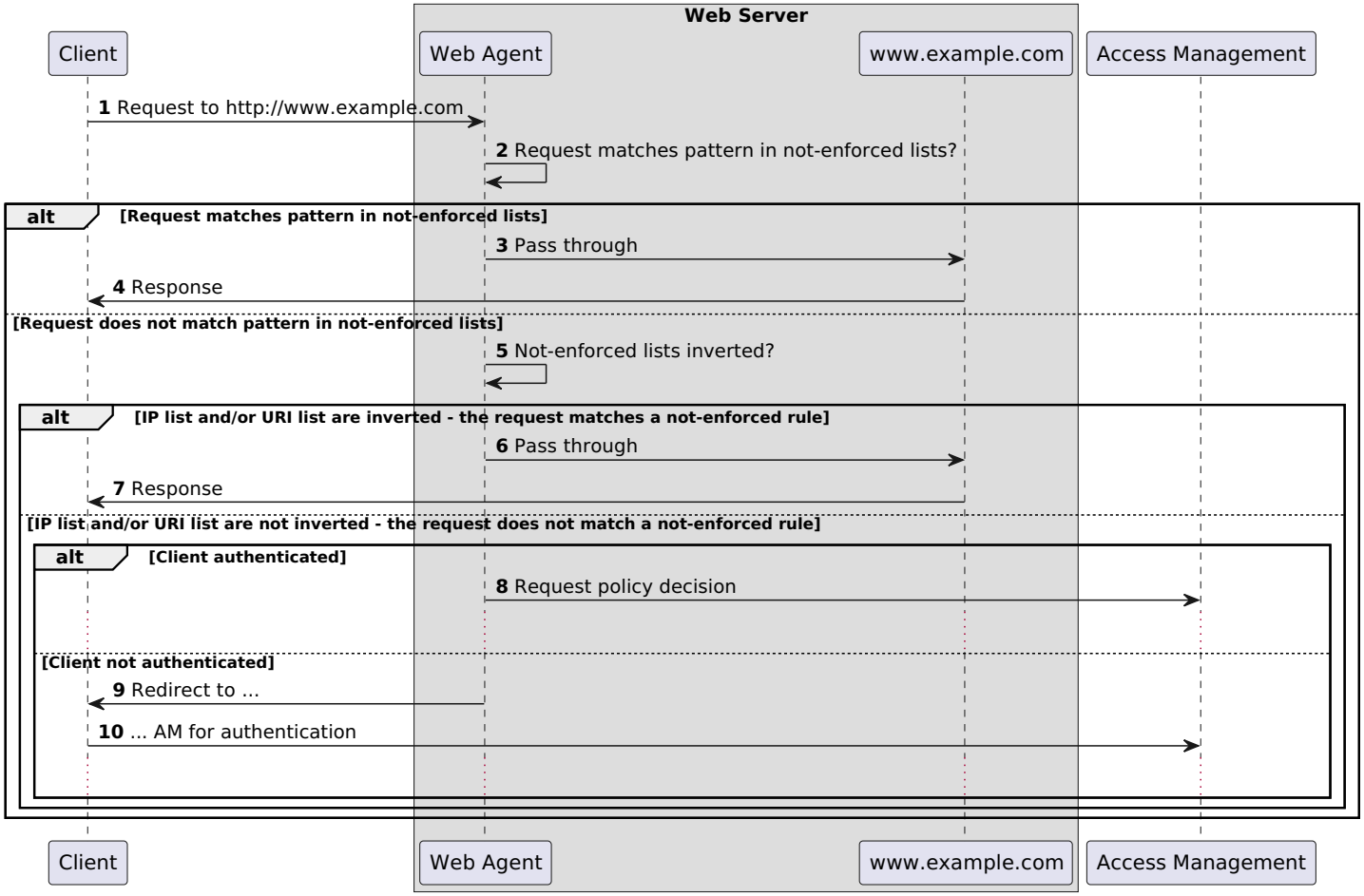
Some resources, such as the "public" directory of a web application, contain data that is not sensitive. It can be accessed by any, authenticated or unauthenticated, clients. The agent uses lists of *not-enforced rules* to identify these resources in the web application.

The agent matches incoming requests to the lists of not-enforced rules. When a request matches a not-enforced rule, the agent bypasses the call to AM:

- If an unauthenticated user sent the request, the agent does not redirect the user to log in.
- If an authenticated user sent the request, the agent does not request a policy evaluation from AM.

Use not-enforced rules to reduce the number of unnecessary calls to AM, and therefore improve the performance and speed of your application.

The following image shows the data flow when Web Agents Documentation evaluates not-enforced rules for a request:



1-2. A client requests a resource and the agent checks whether the request matches a rule in a not-enforced list.

3-5. If the request matches a rule, the agent passes the request without requiring authentication or policy decisions. Otherwise, the agent checks whether rules are inverted.

6-10. If the request matches an inverted rule, the agent passes the request without requiring authentication or policy decisions. Otherwise, the agent enforces authentication and policy decisions.

Conventions for not-enforced rules

Consider the following points about not-enforced rules:

- Web servers normalize request URLs as described in RFC 3986: Normalization and comparison before passing them to the agent. The agent compares the normalized URL to the not-enforced rule.
- Trailing forward-slashes `/` can represent a directory. Therefore, `/images/` does not match `/images`, but does match `/images/index.html`

Invert not-enforced URL rules

Invert all rules in a not-enforced URL list by setting Invert Not-Enforced URLs to `true`.

Consider the following points when you invert all rules:

- If Not-Enforced URL List is empty, all URLs are enforced.
- At least one URL must be enforced. To allow access to any URL without authentication, consider disabling the agent.

Wildcards

For more information about using wildcards, refer to Wildcards.

Multi-level wildcard (*)

The following list summarizes the behavior of the multi-level wildcard (`*`):

- Matches zero or more occurrences of any character except for the question mark (`?`).
- Spans multiple levels in a URL.
- Cannot be escaped. Therefore, the backslash (`\`) or other characters cannot be used to escape the asterisk, as such `\*` .
- Cannot be used in the same rule as the one-level wildcard (`-*-`) or regular expression.
- Explicit patterns are required to match URL parameters. For example:
  - URL patterns ending with `/foo*` do not match URLs with parameters
  - URL patterns ending with `/foo*?*` match any parameter

Multi-level wildcard for not-enforced IP rules

Rules in Not-Enforced IP List	Matches request IP	Does not match request IP
`192.168.1.*`	`192.168.1.0` `192.168.1.0/24`	`192.168.0.1`

Multi-level wildcard for not-enforced URI rules

Rules in Not-Enforced URL List	Matches request URL	Does not match request URL

`http://A-examp.com:8080/*`	`http://A-examp.com:8080/`
	`http://A-examp.com:8080/index.html`
	`http://A-examp.com:8080/x.gif`
		`http://B-examp.com:8080/`
		`http://A-examp.com:8090/index.html`
		`http://A-examp.com:8080/a?b=1`

`http://A-examp.com:8080/*.html`	`http://A-examp.com:8080/index.html`
	`http://A-examp.com:8080/pub/ab.html`
	`http://A-examp.com:8080/pri/xy.html`
		`http://A-examp.com/index.html`
		`http://A-examp.com:8080/x.gif`
		`http://B-examp.com/index.html`

`http://A-examp.com:8080/*/ab`	`http://A-examp.com:8080/pri/xy/ab/xy/ab`
	`http://A-examp.com:8080/xy/ab`
		`http://A-examp.com/ab`
		`http://A-examp.com/ab.html`
		`http://B-examp.com:8080/ab`

Rules in Not-Enforced URL List	Matches request URL	Does not match request URL

`http://A-examp.com:8080/ab/*/de`	`http://A-examp.com:8080/ab/123/de`
	`http://A-examp.com:8080/ab/ab/de`
	`http://A-examp.com:8080/ab/de/ab/de`
		`http://A-examp.com:8080/ab/de`
		`http://A-examp.com:8090/ab/de`
		`http://B-examp.com:8080/ab/de/ab/de`

One-level wildcard (`-*-`)

The following list summarizes the behavior of the one-level wildcard (`-*-`):

- Matches zero or more occurrences of any character except for the forward-slash (`/`) and the question mark (`?`).
- Does not span across multiple levels in a URL.
- Cannot be escaped. Therefore, the backslash (`\`) or other characters cannot be used to escape the hyphen-asterisk-hyphen, like this `\-*-` .
- Cannot be used in the same rule as the multi-level wildcard (`*`) or regular expression.

One-level wildcard for not-enforced URI rules

Rules in Not-Enforced URL List	Matches request URL	Does not match request URL

`http://A-examp.com:8080/b/-*-`	`http://A-examp.com:8080/b/`
	`http://A-examp.com:8080/b/cd`
		`http://A-examp.com:8080/b`
		`http://A-examp.com:8080/b/cd/` (This URL should match the rule, but does not because of the known issue AMAGENTS-4672.)
		`http://A-examp.com:8080/b/c?d=e`
		`http://A-examp.com:8080/b/cd/e`
		`http://A-examp.com:8090/b/`

`http://A-examp.com:8080/b/-*-/f`	`http://A-examp.com:8080/b/c/f` `http://A-examp.com:8080/b/cde/f`	`http://A-examp.com:8080/b/c/e/f` `http://A-examp.com:8080/f/`

`http://A-examp.com:8080/b/c-*-/f`	`http://A-examp.com:8080/b/cde/f`
	`http://A-examp.com:8080/b/cd/f`
	`http://A-examp.com:8080/b/c/f`
		`http://A-examp.com:8080/b/c/e/f`
		`http://A-examp.com:8080/b/c/`
		`http://A-examp.com:8080/b/c/fg`

Multiple wildcards

When multiple wildcards are included in the same rule of a Not-Enforced URL List, the agent matches the parameters in any order that they appear in a resource URI.

For example, the following rule applies to any resource URI that contains a `member_level` and `location` query parameter, in any order:

```
com.sun.identity.agents.config.notenforced.url[1]=http://www.example.com:8080/customers/*?
*member_level=*&location=*
```

In following example, the requests would be not-enforced:

```
https://www.example.com/customers/default.jsp?member_level=silver&location=fr
https://www.example.com/customers/default.jsp?location=es&member_level=silver
https://www.example.com/customers/default.jsp?location=uk&vip=true&member_level=gold
```

Regular expressions

> **Important**
>
> Regular expressions are evaluated differently by different engines. When you use regular expressions in not-enforced lists, make sure that the expressions are evaluated in the way you expect. Double check that the correct URLs are enforced and not enforced.

Set Regular Expressions for Not-Enforced URLs to `true`, and consider the following points for using regular expressions in not-enforced rules:

- Wildcards cannot be used. The asterisk `*` is not treated as a wildcard, but is treated as part of the expression, representing repetition of the last character 0-n times.
- The following formats cannot be used:
  - Netmask CIDR notation
  - IP address ranges

  However, regular expressions can match a range of IP addresses, such as:

  ```
  com.sun.identity.agents.config.notenforced.ip[1]=192\.168\.10\.(10|\d)
  ```

- If an invalid regular expression is specified in a rule, the rule is dropped, and an error message is logged.

HTTP Methods

Rules that apply an HTTP method filter are configured as custom properties in AM.

Add one or more HTTP method keywords followed by an index value. The not-enforced rule is applied when the incoming request uses the HTTP method. Keywords include but are not restricted to `GET`, `HEAD`, `POST`, `PUT`, `PATCH`, `DELETE`, and `OPTIONS`.

If the same method is used in multiple rules, increment the index to make the rule unique:

```
com.sun.identity.agents.config.notenforced.url[PATCH,1]=http://www.example.com:8080/scripts/*
com.sun.identity.agents.config.notenforced.url[PATCH,2]=http://www.other.com:8080/scripts/*
```

By default, no HTTP method is specified for a rule, and all methods are not-enforced for that rule. When one or more HTTP methods are specified, only those methods are not-enforced; methods that are not specified are enforced.

The following example does not enforce OPTIONS requests to the `scripts` directory, but does enforce other HTTP methods:

```
com.sun.identity.agents.config.notenforced.url[OPTIONS,1]=http://www.example.com:8080/scripts/*
```

To specify a list of methods, add multiple rules:

```
com.sun.identity.agents.config.notenforced.url[OPTIONS,1]=http://www.example.com:8080/scripts/*
com.sun.identity.agents.config.notenforced.url[PATCH,2]=http://www.other.com:8080/scripts/*
com.sun.identity.agents.config.notenforced.url[TRACE,3]=http://www.example.com:8080/scripts/*
```

Unrecognized methods can invalidate a rule.

Compound rules

Configure compound rules in Not-Enforced URL from IP Processing List.

In the following example, the agent does not enforce HTTP requests from the IP addresses `192.6.8.0/24` to any file in `/public`, or any files or directories that start with the string `login` in the directory `/free_access URI`:

```
org.forgerock.agents.config.notenforced.ipurl[1]=192.6.8.0/24|http://www.example.com:8080/public/* */
free_access/login*
```

Encoding non-ASCII characters in rules

Percent-encode resources that use non-ASCII characters.

For example, to match resources to the URI `http://www.example.com/forstå`, specify the following percent-encoded rule:

```
/forst%C3%A5/*
```

Continuous security

When a user requests a resource through AM, excluding proxies and load balancers, the Web Agents Documentation is usually the first point of contact. Because Web Agents Documentation is closer to the user than AM, and outside the firewalls that separate the user and AM, the Web Agents Documentation can sometimes gather information about the request, which AM cannot access.

When Web Agents Documentation requests a policy decision from AM, it can include the additional information in an *environment map*, a set of name/value pairs that describe the request IP and DNS name, along with other, optional, information. The additional information can then be included in the policy, for example, to allow only incoming requests that contain the `InternalNetwork`.

In AM, use server-side authorization scripts to access the environment map, and write scripted conditions based on cookies and headers in the request. For information about server-side authorization scripts, refer to Scripting a policy condition in AM's *Authorization guide*.

Environment maps with customizable keys

In Web Agents Documentation, use the continuous security properties Continuous Security Cookie Map and Continuous Security Header Map to configure an environment map with the following parts:

requestIp

The IP address of the inbound request, determined as follows:

- If Client IP Address Header is configured, Web Agents Documentation extracts the IP address from the header.
- Otherwise, Web Agents Documentation uses the web server connection information to determine the client IP address.

This entry is always created in the map.

requestDNSName

The host name address of the inbound request, determined as follows:

- If Client Hostname Header is configured, Web Agents Documentation extracts the host name from the header.
- Otherwise, Web Agents Documentation uses the web server connection information to determine the client's host name.

This entry is always created in the map.

Other variable names

An array of cookie or header values. An entry is created for each value specified in the continuous security properties.

In the following example, the continuous security properties are configured to map values for the `ssid` cookie and `User-Agent` header to fields in an environment map:

```
org.forgerock.openam.agents.config.continuous.security.cookies[ssid]=mySsid
org.forgerock.openam.agents.config.continuous.security.headers[User-Agent]=myUser-Agent
```

If the incoming request contains an `ssid` cookie and a `User-Agent` header, the environment map takes the value of the cookie and header, as shown in this example:

```
requestIp=192.16.8.0.1
requestDnsName=client.example.com
mySsid=77xe99f4zqi1l99z
myUser-Agent=Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
```

Caching

Web Agents Documentation supports the following caches to speed up agent operations:

Configuration cache

The configuration cache stores web agent configuration properties.

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

- AM notifies the agent of changes to hot-swappable agent configuration properties. This only applies to deployments that use centralized configuration mode.
- The information in the cache reaches the expiration time specified by Configuration Reload Interval.

  When a configuration property in the cache is invalid, the 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, refer to 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 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 agent does not specify the reason why the client needs to authenticate.

    After the client authenticates, and the session is cached, the 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

Event	What is invalidated?
Session reaches the expiration time specified by SSO Cache Polling Period.	Session
Policy decision reaches the expiration time specified by Policy Cache Polling Period.	Policy decision
Administrator makes a change to policy configuration (and policy notifications are enabled)	All sessions and all policy decisions

Important

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

Policy cache

The policy cache builds upon the session and 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 Documentation uses 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 a pattern of policy rules downloaded locally. In this case, the agent denies access.
- 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 session property change notifications, as described in Notifications.
- Reduce the amount of time that sessions can remain in the agent session cache.

Consider the following caveats 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 supports only 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, refer to Environment variables.

Attribute fetch modes

For information about properties to configure attribute fetching, refer to Attribute processing.

Web Agents Documentation can fetch user information, and inject it into HTTP request headers 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.

> **Important**
>
> When injecting information into HTTP headers, do not use underscores (_) in the header name. Underscores are incompatible with systems that run CGI scripts, and the header can be silently dropped.

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 web server. The 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 website consumes and displays it.

SSO-only mode

Web Agents Documentation intercepts all inbound client requests to access a protected resource and processes the request based on the Enable SSO Only Mode property.

The configuration setting determines the mode of operation that should be carried out on the intercepted inbound request, as follows:

- When `true`, the agent manages user authentication only. The filter invokes the AM Authentication Service to verify the identity of the user. If the identity is verified, the user is issued a session token through AM's session service.
- When `false`, which is the default, the agent also manages user authorization, by using the policy engine in AM.

FQDN checking

When FQDN checking is enabled, the agent checks the FQDN of a request before it evaluates the authentication or authorization of the request, as follows:

- If the request matches the default domain, or the value of a mapped domain, the agent passes the request on to the next step without changing the domain.
- Otherwise, the agent redirects the request to the specified domain before passing it on to the next step.

Use this feature to prevent the redirect of requests in the following scenarios:

- Where resource URLs differ from the FQDNs in AM policies, for example, in load balanced and virtual host environments.
- Where hostnames are virtual, allocated dynamically, or match a pattern, for example in a Kubernetes deployment.
- Where hostnames are partial.

FQDN checking requires Enable FQDN Check to be `true`, FQDN Default to be set to a suitable value, and optionally, FQDN Virtual Host Map to map incoming URLs to valid outgoing domains.

The agent maps FQDNs as follows:

1. If the request matches the domain in FQDN Default, the agent passes the request to the next step without changing the request domain.
2. Otherwise, if the request matches a mapped domain (map value) in FQDN Virtual Host Map, the agent passes the request to the next step without changing the request domain.
3. Otherwise, if the request matches a mapped host (map key) in FQDN Virtual Host Map, the agent redirects the request to the mapped domain before passing it to the next step.
4. Otherwise, the agent redirects the request to the domain in FQDN Default before passing it to the next step.

Examples

The following example configuration and requests illustrate how the agent checks and remaps FQDNs:

Example configuration

- Agent Root URL for CDSSO:

  `sunIdentityServerDeviceKeyValue=agent.example.com`

- Not-Enforced URL List

  `com.sun.identity.agents.config.notenforced.url[0]=http://www.agent1*.example.com`

- Enable FQDN Check:

  `com.sun.identity.agents.config.fqdn.check.enable=true`

- FQDN Default:

  `com.sun.identity.agents.config.fqdn.default=agent.default.com`

- FQDN Virtual Host Map:

  `com.sun.identity.agents.config.fqdn.mapping[agent.example.com]=agent.example.com`

  `com.sun.identity.agents.config.fqdn.mapping[agent.example.com]=agent-*`

  `com.sun.identity.agents.config.fqdn.mapping[any.value.com]=ag*.example.com`

  `com.sun.identity.agents.config.fqdn.mapping[agent.othertest.com]=other.example.com`

Example requests

- `http://agent.default.com/app` : The request URL matches the domain of the default mapping, so the agent does not redirect the request.
- `https://agent.example.com/app` : The request URL matches the value (domain) in the first mapping, so the agent does not redirect the request.
- `http://agent-4738294739287492/foo/bar/` : The request URL matches the value (domain) in the second mapping, using the wildcard, so the agent does not redirect the request. Note that the value of the key is irrelevant in this match.
- `https://agent123.example.com/app` : The request URL matches the value (domain) in the third mapping, so the agent does not redirect the request.
- `https://agent.othertest.me/app` : The request URL matches the key (host) in the fourth mapping, so the agent redirects the request to `https://other.example.com/app` .
- `https://agent.othertest2.me/app` : The request URL doesn't match any mapping, so the agent redirects the request to the default domain, `https://agent.example.com/app` .

Cookie reset

Web Agents Documentation can reset cookies before redirecting the client to a login page, 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 agent can reset the cookies set by the other mechanism before redirecting the client to a login page.

> **Note**
>
> To set and reset secure or HTTP Only cookies, in addition to the cookie reset properties, 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.

If you have enabled attribute fetching by 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.

Configure load balancers and reverse proxies

Most environments deploy a load balancer and reverse proxy between the agent and clients, and another between the agent and AM, as shown in the following diagram:

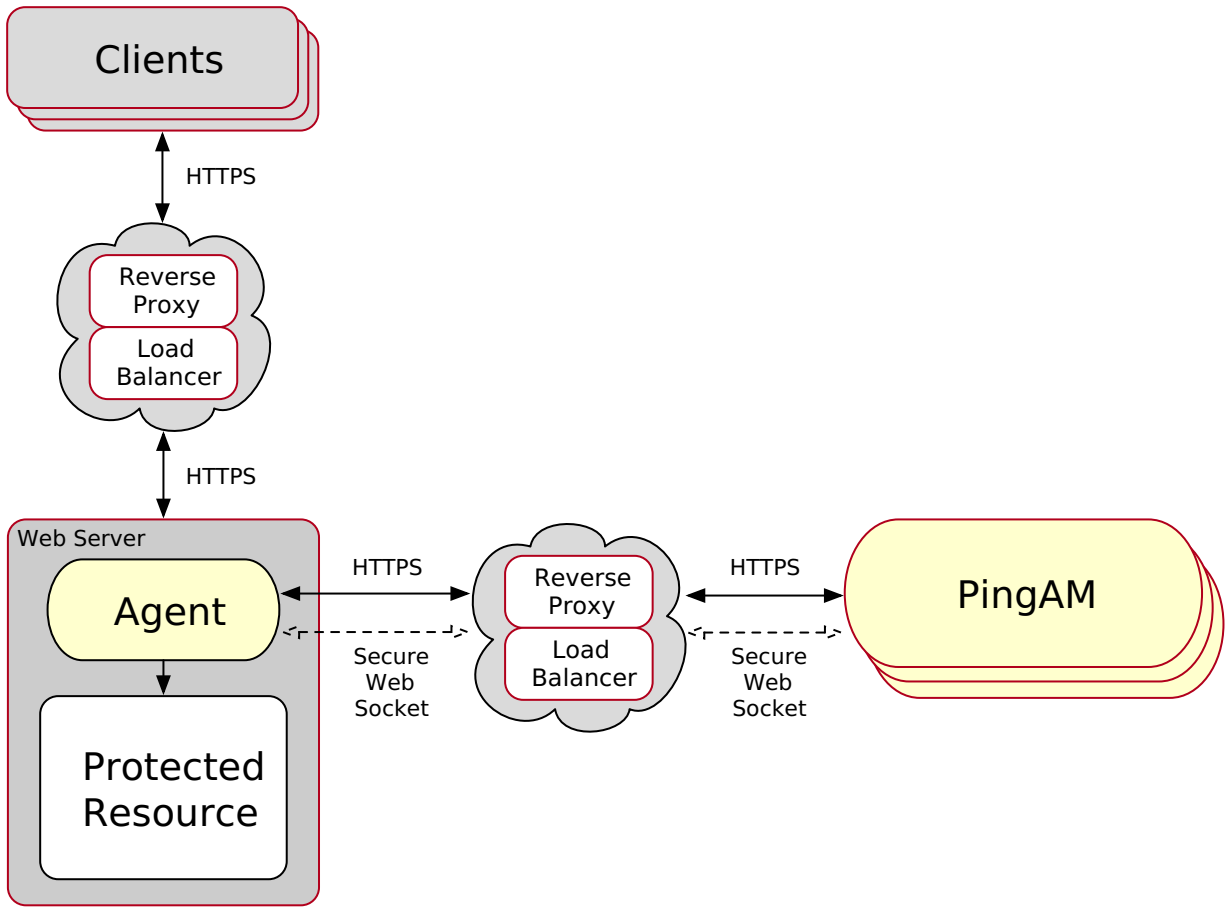


Figure 1. Environments with load balancers and reverse proxies

The reverse proxy and the load balancer can be the same entity. In complex environments, multiple load balancers and reverse proxies can be deployed in the network.

Identifying clients behind load balancers and reverse proxies

When a load balancer or reverse proxy is situated in the request path between the agent and a client, the agent does not have direct access to the IP address or hostname of the client. The agent cannot identify the client.

For load balancers and reverse proxies that support provision of the client IP and hostname in HTTP headers, configure the following properties:

- Client IP Address Header
- Client Hostname Header

When there are multiple load balancers or reverse proxies in the request path, the header values can include a comma-separated list of values, where the first value represents the client, as in `client,next-proxy,first-proxy`.

Agent connection to AM through a load balancer/reverse proxy

When a reverse proxy is situated between the agent and AM, it can be used to protect the AM APIs.

When a load balancer is situated between the agent and AM, it can be used to regulate the load between different instances of AM.

Consider the points in this section when installing Web Agents Documentation in an environment where AM is behind a load balancer or a reverse proxy.

Agent's IP address and/or FQDN

The load balancer or reverse proxy conceals the IP addresses and FQDNs of the agent and of AM. Consequently, AM cannot determine the agent base URL.

To prevent problems during installation or redirection, do the following:

- Configure the load balancer or reverse proxy to forward the agent IP address and/or FQDN in a header.
- Configure AM to recover the forwarded headers. For more information, see Configuring AM to Use Forwarded Headers.
- Install the agent using the IP address or FQDN of the load balancer or reverse proxy as the point of contact for the AM site.

AM sessions and session stickiness

Improve the performance of policy evaluation by setting AM's sticky cookie (by default, `amlbcookie`) to the AM's server ID. For more information, refer to Configuring site sticky load balancing in AM's *Setup guide*.

When configuring multiple agents, consider the impact on sticky load balancer requirements of using one or multiple agent profiles:

- If agents are configured with multiple agent profiles, configure sticky load balancing. The agent profile name is contained in the OpenID Connect JWT, used by the agent and AM for communication. Without session stickiness, it is not possible to make sure the appropriate JWT ends in the appropriate agent instance.

  To have multiple agent profiles without sticky load balancing, disable validation of the `aud` claim in the session ID token. Either enable Disable Audience Claim Validation, or configure Agent Profile ID Allow List.

  For security reasons, agents should validate all claims in session ID tokens. Therefore, use this approach sparingly and mostly for migrations.

- If multiple agents are configured with the same agent profile, decide whether to configure sticky load balancing depending on other requirements of your environment.

WebSockets

For communication between the agents and AM servers, the load balancers and reverse proxies must support the WebSocket protocol. For more information, refer to the load balancer or proxy documentation.

> **Tip**
>
> For configuration examples, refer to Apache as a reverse proxy and NGINX as a reverse proxy.

Configuring AM to use forwarded headers

When a load balancer or reverse proxy is situated between the agent and AM, configure AM to recover the forwarded headers that expose the agents' real IP address or FQDN.

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

1. Log in to the AM admin UI as an administrative user, such as `amAdmin`.

2. Select **Realms** > Realm Name > **Services**.
3. Select **Add a Service** > **Base URL Source**, and create a default service, leaving the fields empty.
4. Configure the service with the following properties, leaving the other fields empty:
    - **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 specified in RFC 7239: Forwarded HTTP Extension.
    - **Context path**: AM's deployment URI. For example, `/am`.

   For more information, refer to Base URL source in AM's *Reference*.
5. Save your changes.

Agent connection to clients through a load balancer/reverse proxy

When a reverse proxy is situated between the agent and client, it can be used to anonymize the client traffic that enters the network.

When a load balancer is situated between the agent and client, it can be used to regulate the load between the agents and the web application servers.

Consider the points in this section when installing Web Agents Documentation in an environment where clients are behind a load balancer or a reverse proxy.

Client's IP Address and/or FQDNs

The load balancer or reverse proxy conceals the IP addresses and FQDNs of the agent and clients. Consequently, the agent cannot determine the client base URL.

Configure the load balancer or reverse proxy to forward the client IP address and/or the client FQDN in a header. Failure to do so prevents the agent from performing policy evaluation, and applying not-enforced and conditional login/logout rules.

For more information, refer to Configuring client identification properties.

POST data preservation and sticky load balancing

For POST data preservation, use sticky load balancing to ensure that after login the client hits the same agent as before and can therefore get their saved POST data.

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

For more information, refer to Configure POST Data Preservation for Load Balancers or Reverse Proxies.

Mapping agent host name to a load balancer or reverse proxy

In the following diagram, the agent and load balancer/reverse proxy use the same protocol and port, but different FQDNs:

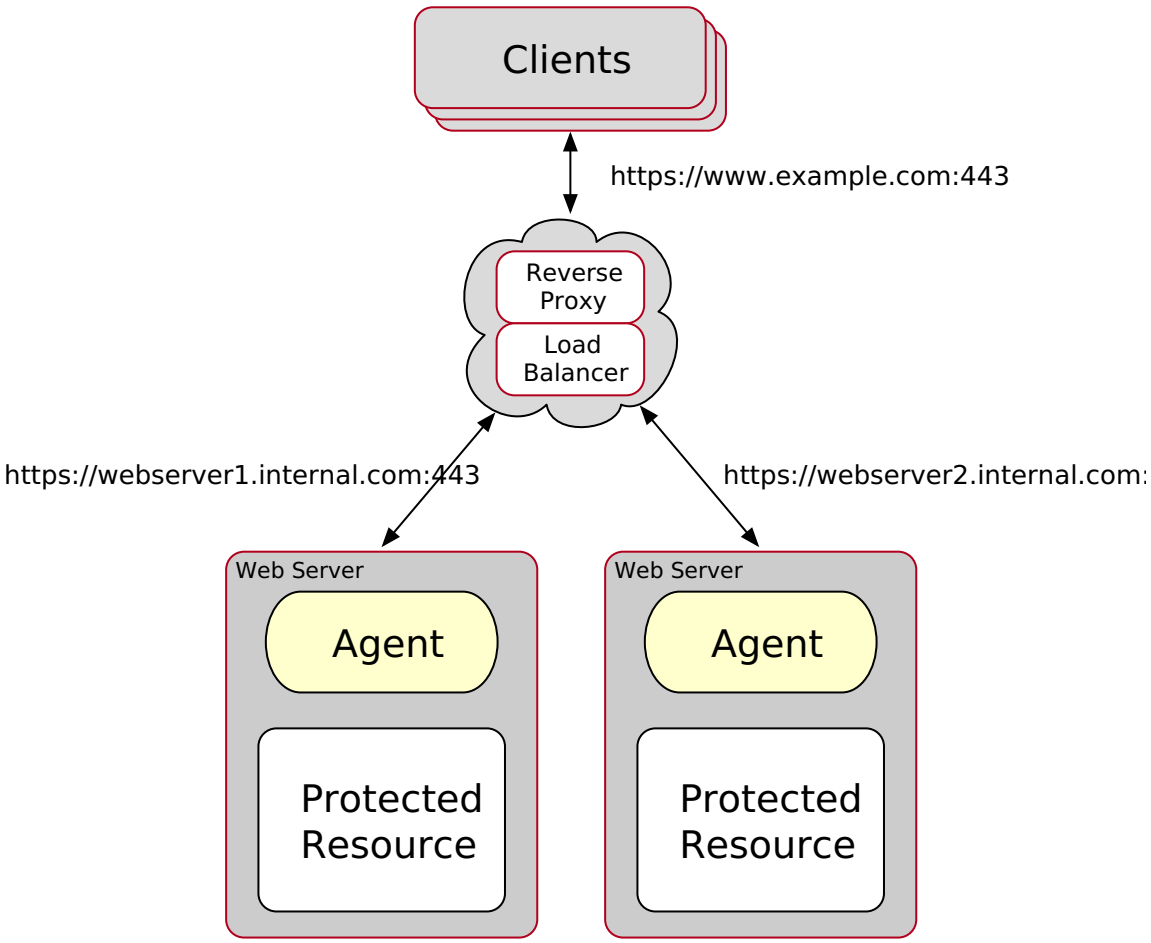


Figure 2. Same protocol and port, different FQDN

Map the host name of the agent to that of the load balancer or reverse proxy.

1. Log in to the AM admin UI as an administrative user with rights to modify the web agent profile.
2. Go to **Realms** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.
3. Set the following options in the **Global** tab:
   - **FQDN Check**: Enable

     The equivalent property setting is Enable FQDN Check = `true`.
   - **FQDN Default**: Set to the FQDN of the load balancer or proxy, for example `lb.example.com`. Do not set it to the protected server FQDN where the agent is installed.

     The equivalent property setting is FQDN Default = `lb.example.com`.
   - **Agent Root URL for CDSSO**: Set to the FQDN of the load balancer or proxy, for example `https://lb.example.com:80/`.

     The equivalent property setting is Agent Root URL for CDSSO = `lb.example.com`.

- **FQDN Virtual Host Map**: Map the load balancer or proxy FQDN to the FQDN where the agent is installed. For example,
  - **Key**: `agent.example.com` (protected server)
  - **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`.

4. Save your work.

Overriding protocol, host, and port

The load balancer or reverse proxy forwards requests and responses between clients and protected web servers. In the following diagram, the protocol, port, and FQDN configured on the load balancer and reverse proxy are different than those on the protected web server:

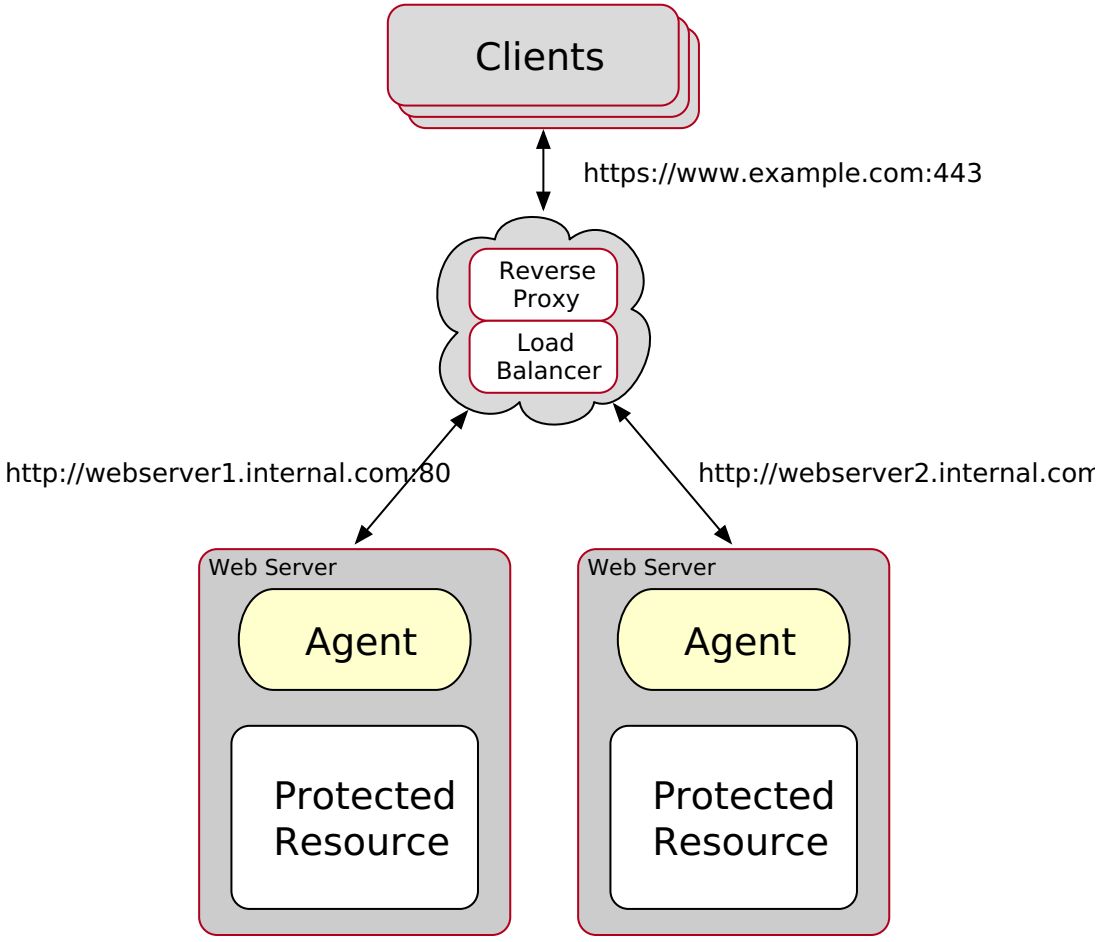


Figure 3. Different protocol, port, and FQDN

Important

Agent configuration for TLS offloading prevents FQDN checking and mapping. Consequently, URL rewriting and redirection do not work correctly when the agent is accessed directly, instead of through the load balancer or proxy. This should not be a problem for client traffic, but could be a problem for web applications accessing the protected web server directly, from behind the load balancer.

> **Note**
>
> When the following headers are defined on the proxy or load-balancer, they override the value of Agent Deployment URI Prefix:
>
> - `X-Forwarded-Proto`
> - `X-Forwarded-Host`
> - `X-Forwarded-Port`
>
> If you are using these headers, do not configure the agent to override its hostname, port, or protocol.

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

1. Log in to the AM admin UI as an administrative user with rights to modify the agent profile.
2. Go to **Realms** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.
3. Set the following options in the **Global** tab:
   - **Agent Deployment URI Prefix**: Set to the URI of the load balancer or proxy.

     This value is used to override the protocol, host, and port of the protected server.

     The equivalent property setting is Agent Deployment URI Prefix = `external.example.com`.
   - **Agent Root URL for CDSSO**: Set to the URL of the load balancer or proxy.

     The equivalent property setting is Agent Root URL for CDSSO = `https://external.example.com:443`.
4. Enable the following options in the **Advanced** tab:
   - **Override Request URL Protocol**

     The equivalent property setting is Enable Override Request URL Protocol = `true`.
   - **Override Request URL Host**

     The equivalent property setting is Enable Override Request URL Host = `true`.
   - **Override Request URL Port**

     The equivalent property setting is Enable Override Request URL Port = `true`.
5. Save your work.

Configuring client identification properties

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

This procedure explains how to configure the client identification properties for a centralized web agent profile configured in the AM admin UI. The steps also mention the properties for web agent profiles that rely on local, file-based configurations:

1. Log in to the AM admin UI with a user that has permissions to modify the web agent profile.
2. Go to **Realms** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.

3. Set the following options in the **Advanced** tab:
    - **Client IP Address Header**: Configure the name of the header containing the IP address of the client. For example, `X-Forwarded-For`.

      The equivalent property setting is Client IP Address Header = `X-Forwarded-For`.

      Configure this property if any of the following points are true:
        - AM policies are IP address-based.
        - The agent is configured for not-enforced IP rules.
        - The agent is configured take any decision based on the client's IP address.
    - **Client Hostname Header**: Configure the name of the header containing the FQDN of the client. For example, `X-Forwarded-Host`.

      The equivalent property setting is Client Hostname Header = `X-Forwarded-Host`.

      Configure this property if any of the following points are true:
        - AM policies are URL address-based.
        - The agent is configured for not-enforced URL rules.
        - The agent is configured take decisions based on the client's URL address.
4. Save your changes.

Configuring POST Data Preservation for Load Balancers or Reverse Proxies

Use one of the following procedures to configure post data preservation for load balancers or reverse proxies.

Map one agent profile to one agent instance when POST data preservation is enabled

In this procedure, a separate agent profile must created in AM for each agent instance. For scalable deployments, where resources are dynamically created and destroyed, use Map one agent profile to multiple agent instances when POST data preservation is enabled instead.

1. Configure your load balancer or reverse proxy to ensure session stickiness when the cookie or URL query parameter are present.
2. Log in to the AM admin UI as a user that has permissions to modify the agent profile.
3. Go to **Realms** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.
4. Set the following options in the **Advanced** tab:
    - POST Data Sticky Load Balancing Mode:
        - **COOKIE**: The agent creates a cookie for POST data preservation session stickiness. The content of the cookie is configured in the next step.
        - **URL**: The agent appends to the URL a string specified in the next step.

The equivalent property setting is `com.sun.identity.agents.config.postdata.preserve.stickysession.mode=COOKIE` or `com.sun.identity.agents.config.postdata.preserve.stickysession.mode=URL`.

- POST Data Sticky Load Balancing Value: Configure a key-pair value separated by the `=` character.

  The agent creates the value when it evaluates POST Data Sticky Load Balancing Mode. 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.

  The equivalent property setting is `com.sun.identity.agents.config.postdata.preserve.stickysession.value=lb=myserver`.

5. Save your changes.

Map one agent profile to multiple agent instances when POST data preservation is enabled

Use this procedure for scalable deployments, where resources can be dynamically created or destroyed. For example, use it in deployments with load balancing, or environments running Kubernetes.

1. Configure your load balancer or reverse proxy to ensure session stickiness when the cookie or URL query parameter are present.
2. For each agent instance, configure post data preservation in the agent configuration file `agent.conf`. This configuration overrides the configuration in AM.

   In the following example, the settings in `agent.conf` configure two agents behind a load balancer to use the same agent profile, and provide uniqueness to the load balancer:

   - Agent 1:

     ```
     com.sun.identity.agents.config.postdata.preserve.stickysession.mode = COOKIE
     com.sun.identity.agents.config.postdata.preserve.stickysession.value = EXAMPLE=Agent1
     ```

   - Agent 2:

     ```
     com.sun.identity.agents.config.postdata.preserve.stickysession.mode = COOKIE
     com.sun.identity.agents.config.postdata.preserve.stickysession.value = EXAMPLE=Agent2
     ```

   For information about the values to use, refer to the following properties:

   - POST Data Sticky Load Balancing Mode
   - POST Data Sticky Load Balancing Value

3. Restart the web server where the agent is installed.

Environment variables

Configure environment variables to affect the user that is running the web server, virtual host, or location that the agent protects.

This section describes Web Agents Documentation properties that are configured by environment variables. After setting an environment variable, restart Web Agents Documentation.

For information about environment variables for installation, refer to Installation environment variables.

For information about allowing environment variables to be used in NGINX, refer to the env directive in the *NGINX Core functionality documentation*.

AM_IPC_BASE

(Unix only) 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.

Default: Arbitrary value

AM_MAX_AGENTS

The maximum number of agent instances in the installation. The higher the number, the more shared memory the agent reserves.

When the maximum is reached, if another agent instance starts, an error is logged and the agent won't protect any resources.

Default: `32`

AM_MAX_SESSION_CACHE_SIZE

The maximum size in bytes of the shared memory for the session and policy cache:

- Not set, or set to `0`: `16777216` (16 MB)
- Maximum value: `1073741824` (1 GB)
- Minimum value `1024` (1 MB)

For multiple concurrent sessions, consider using a higher value.

AM_NET_TIMEOUT

The number of seconds for which the agent installer can contact AM during agent configuration validation.

If the installer takes longer than this value to contact AM and validate the configuration, installation fails.

Default: 4 seconds

Policy evaluation mode (AM_POLICY_CACHE_MODE)

Policy evaluation mode:

- `off` or `0` (default): When a request requires a policy decision, the agent contacts AM for the decision.
- `on`: The agent downloads all policies from AM at startup. When a request requires a policy decision, the agent uses the downloaded policies to make the policy decision.

In both modes, the agent caches the policy decision. If a request requires the same policy decision again, the agent uses the cached decision.

(Optional) Use the `AM_POLICY_CACHE_DIR` environment variable to specify a directory in which to store the policy cache.

AM_POLICY_CACHE_DIR

The 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`.

AM_RESOURCE_PERMISSIONS

(Unix only) The permissions that the agent sets for its runtime resources.

Allowed values:

- `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 `0022` and the `AM_RESOURCE_PERMISSIONS` is `0666`. The agent runtime resources have the following permissions:

Resource Permissions Example in Linux

Resource	Permission	Owner
`/path/to/web_agents/agent_type/log/system_n.log`	644	`apache`
`/path/to/web_agents/agent_type/log/monitor_n.log`	644	`apache`
`/path/to/web_agents/agent_type/instances/agent_n/conf/agent.conf`	640	`apache`
`/path/to/web_agents/agent_type/instances/agent_n/logs/debug/debug.log`	644	`apache`
`/dev/shm/am_cache_0`	644	`apache`
`/dev/shm/am_log_data_0`	644	`apache`

Any semaphores owned by the `apache` user have `644` permissions as well.

Consider another example where `umask` is `0002` and `AM_RESOURCE_PERMISSIONS` is `0666`. The files are created with `664` permissions, which allows them 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 Security Protocol List bootstrap property.

Specifies a space-separated list of security protocols preceded by a dash (`-`) that *won't* be used when connecting to AM.

Supported protocols:

- `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

The log level for messages from the agent startup and background processes. Messages provide information about the agent initialisation, local files that the agent uses, or resources that the agent uses.

By default, messages are written to the file given by AM_SYSTEM_LOG_PATH, by default `/path/to/web_agents/agent_type/log/system_n.log`.

The value `n` in the `/path/to/web_agents/agent_type/log/system_n.log` file indicates the agent group number. Consider an environment with the following Apache HTTP 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 write to the `system_0.log` file.
- `Apache_2` has one agent instance configured, `agent_3`, with AmAgentId set to 1. The instance write to the `system_1.log` file.

The `/path/to/web_agents/agent_type/log/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 following case-insensitive values are valid:

- All
- Message
- Warning
- Error (default)
- Info

AM_SYSTEM_LOG_PATH

The full path and filename to the `/path/to/web_agents/agent_type/log/system_n.log` file.

Default: `/path/to/web_agents/agent_type/log/system_n.log`

`AM_SYSTEM_LOG_SIZE`

The size in bytes of the `/path/to/web_agents/agent_type/log/system_n.log` file.

Valid range: 0 (unlimited log file size) to 4294967295 bytes (4GB)

Default: `0`

`AM_SYSTEM_PIPE_DIR`

(Unix only) The directory where agent instances store temporary pipe files.

Default: `/path/to/web_agents/agent_type/log/`

Troubleshooting

ForgeRock provides support services, professional services, training through ForgeRock University, and partner services to help you set up and maintain your deployments. For information, refer to Getting support.

When you are trying to solve a problem, save time by asking the following questions:

- How do you reproduce the problem?
- What behavior do you expect, and what behavior do you see?
- When did the problem start occurring?
- Are their circumstances in which the problem does not occur?
- Is the problem permanent, intermittent, getting better, getting worse, or staying the same?

If you contact ForgeRock for help, include the following information with your request:

- Description of the problem, including when the problem occurs and its impact on your operation.
- The product version and build information.
- Steps you took to reproduce the problem.
- Relevant access and error logs, stack traces, and core dumps.
- Description of the environment, including the following information:
  - Machine type
  - Operating system and version
  - Web server and version
  - Java version
  - Patches or other software that might affect the problem

The following items are some common issues and solutions:

> **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, refer to the `--V[i]` option in agentadmin command.

Question

During upgrade or installation, what should I do if I get shared memory errors?

Answer

1. Stop the web server where the agent is installed.
2. Delete the following shared memory files:
   - `/dev/shm/am_cache_0`
   - `/dev/shm/am_log_data_0`

     Depending on your configuration, the files can be named differently.
3. Start the agent.

Question

When I map a response or attribute to an HTTP header, using the following properties, why is the header ignored:

- Session Attribute Map
- Response Attribute Map
- Profile Attribute Map

Answer

When injecting information into HTTP headers, do not use underscores (`_`) in the header name. Underscores are incompatible with systems that run CGI scripts, and the header can be silently dropped.

Question

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 error:

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

Answer

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.

Question

I am trying to install Web Agents Documentation 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?

Answer

When installing Web Agents Documentation 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.

For information about configuring SELinux, refer to the Linux documentation.

Question

Why are logs not being written to `/log/system_0.log` and `/log/monitor_0.pipe` files? I am seeing this error:

```
unable to open event channel
```

Answer

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 to the account used to run your web server.

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.

Question

My Apache HTTP server is not using port 80. When I install Web Agents Documentation it defaults to port 80. How do I fix this?

Answer

You probably set `ServerName` in the Apache HTTP Server configuration to the host name, but did not specify the port number.

Instead, 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
```

Question

My web server and Web Agents Documentation 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?

Answer

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, refer to agentadmin command.

> **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.

Question

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

Answer

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 admin UI, and go to **Realms** > Realm Name > **Services** > **Add** > **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 Configuring success and failure redirection URLs.

Question

I have installed the Unix Apache Web Agents Documentation, and neither Apache HTTP Server nor the agent start up or log any message. If I remove the agent, the Apache HTTP Server starts again. What can be the problem?

Answer

To troubleshoot Web Agents Documentation or a web server that does not start, set the 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 HTTP Server.
3. Check the logs generated in the `web_agent/apache24_agent/log/system_n.log`.

   Web Agents Documentation reserves memory for the policy and session cache based on the `AM_MAX_SESSION_CACHE_SIZE` environment variable. If the server where the 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, refer to 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.

Question

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:

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

What is happening?

Answer

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, refer to AM's Security guide.

Question

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.iplanet.sso.SSOException: Invalid Agent Root URL
com.iplanet.sso.SSOException: Goto URL not valid for the agent Provider ID
```

What should I do?

Answer

Web Agents Documentation accepts only 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 Documentation prevents 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/`.

Question

I have upgraded my Unix Apache or IBM HTTP Server Web Agents Documentation, and even though notifications are enabled, the agent does not update its configuration. What is happening?

Answer

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 does not clean 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:

   ```
   ... UTC   DEBUG [1:10551398][source/monitor.c:503]monitor startup
   ... UTC   ERROR [102:10551398]monitor unable to get semaphore
   ... 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. Restart the Apache or IBM HTTP server.

Question

After upgrade, the default Apache welcome page appears instead of my custom error pages. What should I do?

Answer

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

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

For more information about `ErrorDocument`, refer to the Apache documentation.

Question

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

```
[../resources/troubleshooting/troubleshooting.bash:#web-agent-install]
```

Answer

Web Agents Documentation 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.

Question

My Web Agents Documentation is not protecting my website. In the logs, I can see errors similar to the following:

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

What is happening?

Answer

The Web Agents Documentation is unable to log in to AM. Possible causes are:

- Network connection between the agent and AM is unavailable.
- The AM Connection URL property, which specifies the AM URL may be misconfigured.

Question

My Web Agents Documentation is not protecting my website. In the `debug.log` file I can see messages similar to the following:

```
... 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://am.example.com:8443/am/oauth2/Customers",
   "tokenName":"id_token",
   "nonce":"D3AE96656D6D634489AF325D90C435A2",
   "aud":"webagent",
   "s_hash":"rxwxIoqDFiwt4MxSwiBa-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"
   }
... GMT WARNING [162ba6eb-cf88-3d7f-f92c-ee8b21971b4c]: redirect_after_authn(): unable to validate JWT
```

What is happening?

Answer

If you configured the agent profile in a realm other than AM's top-level realm (`/`), you must configure the 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 causes the agent to fail to recognize the realm.

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:

```
... GMT WARNING [69d4632c-82af-b853-0f340vb7b754]: too many pending authentications
... GMT ERROR  [69d4632c-82af-76da-b853-0f340vb7b754]: save_pre_authn_state(): unable to save state for request
```

What is happening?

Answer

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

For more information, refer to Enable Multivalue for Pre-Authn Cookie.

Question

I am getting HTTP 403 Forbidden messages when accessing the Web Agents Documentation.

Answer

Make sure the Web Agents Documentation is executable:

1. In the terminal where the Web Agents Documentation is running, go to `/opt/web_agents`.
2. Review and, if necessary, change the permissions for the directory:

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 group

A group of agent instances that share runtime resources and shared memory.

Agent profile

A set of configuration properties that define the behavior of the agent.

Agent profile realm

The AM realm in which the agent profile is stored.

Application

In general terms, a service exposing protected resources.

In the context of AM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.

Application type

Application types act as templates for creating policy applications.

Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.

Application types also define the internal normalization, indexing logic, and comparator logic for applications.

Attribute-based access control (ABAC)

Access control that is based on attributes of a user, such as how old a user is or whether the user is a paying customer.

Authentication

The act of confirming the identity of a principal.

Authentication chaining

A series of authentication modules configured together which a principal must negotiate as configured in order to authenticate successfully.

Authentication level

Positive integer associated with an authentication module, usually used to require success with more stringent authentication measures when requesting resources requiring special protection.

Authentication module

AM authentication unit that handles one way of obtaining and verifying credentials.

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.

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.

Centralized configuration mode

AM stores the agent properties in the AM configuration store. See also local configuration mode.

The configuration mode is defined by Location of Agent Configuration Repository.

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 *reference* to token to the client.

Client-based sessions

AM sessions 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.

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.

Conditions

Defined as part of policies, these determine the circumstances under which a policy applies.

Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved.

Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.

Configuration datastore

LDAP directory service holding AM configuration data.

Cross-domain single sign-on (CDSSO)

AM capability allowing single sign-on across different DNS domains.

CTS-based OAuth 2.0 tokens

After a successful OAuth 2.0 grant flow, AM returns a *reference* to the token to the client, rather than the token itself. This differs from client-based OAuth 2.0 tokens, where AM returns the entire token to the client.

CTS-based sessions

AM sessions 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 AM.

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 java container that acts as a policy enforcement point, filtering requests to other applications in the container, with policies based on application resource URLs.

Local configuration mode

The Web Agents Documentation installer creates the file `/web_agents/agent_type/instances/agent_nnn/config/agent.conf` to store the agent configuration properties. See also centralized configuration mode.

The configuration mode is defined by Location of Agent Configuration Repository.

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

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 policy decision point.

Policy evaluation realm

The AM realm that the agent uses to request policy decisions from AM.

Policy Information Point (PIP)

Entity that provides extra information, such as user profile attributes, that a policy decision point needs in order to make a decision.

Principal

Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities.

When an AM identity successfully authenticates, AM associates the identity with the Principal.

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

AM unit for organizing configuration and identity information.

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.

Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.

Resource

Something a user can access over the network such as a web page.

Defined as part of policies, these can include wildcards in order to match multiple actual resources.

Resource owner

In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.

Resource server

In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.

Response attributes

Defined as part of policies, these allow AM to return additional information in the form of "attributes" with the response to a policy decision.

Role based access control (RBAC)

Access control that is based on whether a user has been granted a set of permissions (a role).

Security Assertion Markup Language (SAML)

Standard, XML-based language for exchanging authentication and authorization data between identity providers and service providers.

Service provider (SP)

Entity that consumes assertions about a principal (and provides a service that the principal is trying to access).

Session 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 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

Group of AM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability.

The load balancer can also be used to protect AM services.

Standard metadata

Standard federation configuration information that you can share with other access management software.

Stateless Service

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.

All AM services are stateless unless otherwise specified.

Subject

Entity that requests access to a resource

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.

User realm

The AM realm in which a user is authenticated.

Web Agent

Native library installed in a web server that acts as a policy enforcement point with policies based on web page URLs.

Maintenance guide

This guide describes how to perform recurring administrative operations in ForgeRock Access Management Web Agents Documentation.

About ForgeRock Identity Platform™ software

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

Auditing

Web Agents Documentation logs audit events for security, troubleshooting, and regulatory compliance. Logs are written in UTF-8. Store audit event logs in the following ways:

Remotely

Log audit events to the audit event handler configured in the AM realm. In an environment with several AM servers, agents write audit logs to the AM server that satisfies the agent request for client authentication or resource authorization.

Web Agents Documentation cannot log audit events remotely if:

- AM's audit logging service is disabled.
- No audit event handler is configured in the realm where the agent is configured.
- All audit event handlers configured in the realm where the agent is configured are disabled.

For more information about audit logging in AM, refer to Setting up audit logging in AM's *Security guide*.

Locally

Log audit events in JSON format to `/path/to/web_agents/agent_type/instances/agent_n/logs/audit/audit.log`. The following is an example agent log file: `/web_agents/nginx22_agent/instances/agent_1/logs/audit/audit.log`.

Remotely and locally

Log audit events:

- To `/path/to/web_agents/agent_type/instances/agent_n/logs/audit/audit.log`
- To the audit event handler configured in the AM realm in which the agent profile is configured.

The following is an example agent log record:

```
{
   "timestamp":"2017-10-30T11:56:57Z",
   "eventName":"AM-ACCESS-OUTCOME",
   "transactionId":"608...77e",
   "userId":"id=demo,ou=user,dc=example,dc=com",
   "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, refer to Audit log format in AM's *Security guide*.

Web Agents Documentation supports propagation of the transaction ID across the ForgeRock Identity Platform, using the HTTP header `X-ForgeRock-TransactionId`. For more information about configuring the header, refer to Configuring the trust transaction header system property in AM's *Security guide*.

By default, Web Agents Documentation does not write audit log records. To configure audit logging, perform the following procedure:

Configure audit logging

By default, Web Agents Documentation does not write audit log records. To configure audit logging, perform this procedure. The agent in this example is in remote configuration mode.

1. In `agent.conf`, set values for the following properties:
    - Local Agent Audit File Name
    - Local Audit Log Rotation Size

    > **Note**
    >
    > After changing a bootstrap property, restart the web server where the agent runs.

2. On the AM admin UI, select **Realms** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.
3. On the **Global** tab, select the following options to configure audit:
    - Agent Debug Level
    - Audit Access Types
    - Audit Log Location

Monitoring

> **Important**
>
> The monitoring endpoint described in this section is deprecated. Use it only for diagnostics, in conjunction with Forgerock support.

A monitoring endpoint provides access to metrics for operations within the agent and between the agent an AM.

The monitoring endpoint is protected by HTTP Basic Authentication; to access it, provide the agent URL, and the AM agent profile name and password. Always use HTTPS for secure connections to client applications.

Metrics are displayed as a JSON response, with the fields described in Summary of monitoring metrics.

Access the monitoring endpoint

1. Install a Web Agents Documentation as described in the Installation, and use the agent to protect a web application. For example, set up the example in Policy enforcement.
2. Access the agent monitoring endpoint as follows, where `https://agent.example.com:443` is the agent URL, and `web-agent` is the AM agent profile name.

```
$ curl https://agent.example.com:443/agent/monitor -u web-agent

Enter host password for user 'web-agent':
```

3. Enter the AM agent profile password to display the metrics:


```
{
  "cache-invalidation": {
    "policy": 0,
    "profile": 1
  },
  "policy-decisions": {
    "neu": 0,
    "local": 0,
    "remote": 2,
    "cache": 0
    },
  "gc": {
    "runs": 1,
    "released": 0,
    "release-deferred": 0,
    "fill": 0.000000
  },
  "cache-operations": {
    "writes": 0,
    "rewrites": 2,
    "reads": 2,
    "misses": 0,
    "deletes": 0,
    "free-deferred": 0,
    "write-faults": 0,
    "expired": 0
  },
  "connections": {
    "added": 2,
    "reused": 3
  }
}
```


Summary of monitoring metrics

Metric	Submetric	Count of
`cache-invalidation`	`policy`	Number of policy change notifications received from AM.
	`profile`	Number of agent configuration change notifications received from AM.
`policy-decisions`	`neu`	Number of requests that were not enforced by the agent because of the not-enforced URL lists.
	`local`	Number of policy decisions the agent makes locally.
	`remote`	Number of policy decisions the agent requests from AM.
	`cache`	Number of policy decisions the agent takes from the cache.
`gc`	`runs`	Number of garbage collection runs.

Metric	Submetric	Count of
	`released`	Number of cache entries released during garbage collection runs.
	`release-defered`	Number of entries with release deferred until the next garbage collection run.
	`fill`	Floating point value between 0 and 1, representing the proportion of cache that is free after the most the recent garbage collection.
`cache-operations`	`writes`	Number of writes to cache.
	`rewrites`	Number of updates to cache.
	`reads`	Number of reads from cache.
	`misses`	Number of failed searches of the cache.
	`deletes`	Number of deletes from cache.
	`free-deferred`	Number of cache removals deferred until the next garbage collection run.
	`write-faults`	Number of cache writes that fail because the cache is full.
	`rewrites`	Number of expired cache entries.
`connections`	`added`	Number of new connections made.
	`reused`	Number of times existing connections were reused.

Notifications

AM sends the following notifications to Web Agents Documentation through WebSockets:

Configuration notifications

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

Configuration notifications apply when the agent profile is stored in AM's configuration data store.

For more information about the cache, refer to Configuration cache.

Session Notifications

When a client logs out, or a CTS-based session expires, AM sends a notification to the agent to remove the client's entry from the session cache.

For more information about the cache, refer to Session and policy decision cache.

Policy Notifications

When an administrator changes a policy, AM sends a notification to the agent to flush the session and policy decision cache, and the policy cache. Enable Notifications controls whether the AM server sends notifications to connected agents. It is enabled by default.

For more information about the cache, refer to Session and policy decision cache and Policy cache.

In configurations with load balancers and reverse proxies, make sure the load balancers and reverse proxies support WebSockets.

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

Disable notifications

> **Caution**
>
> Notifications are enabled by default. Before disabling notifications, consider the impact on security if the agent is not notified of changes in AM.

1. On the AM admin UI, select **Realms** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.
2. On the **Global** tab, deselect the following options to disable notifications:
   - Enable Notifications

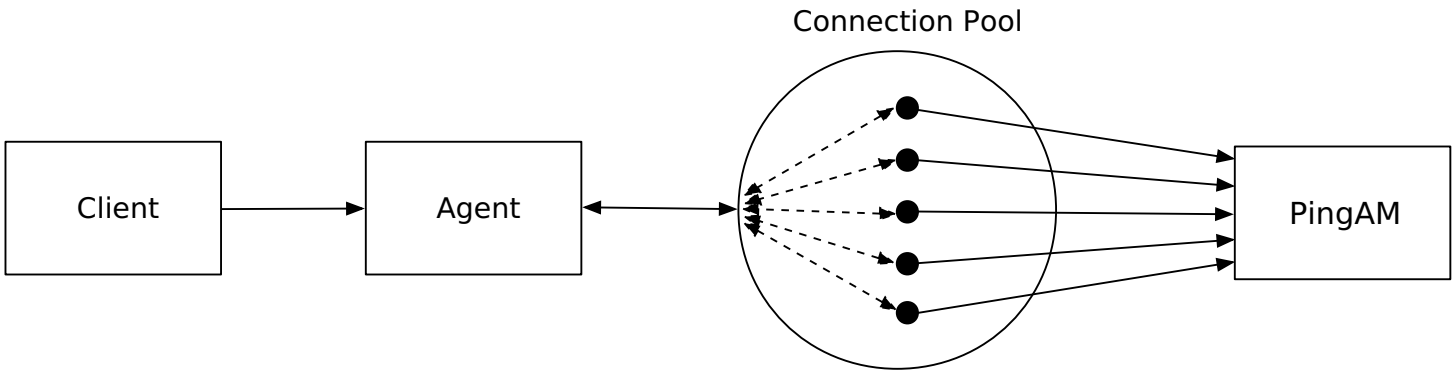
     After changing this property, restart the web server where the agent runs.
   - Enable Notifications of Agent Configuration Change

Connection pooling

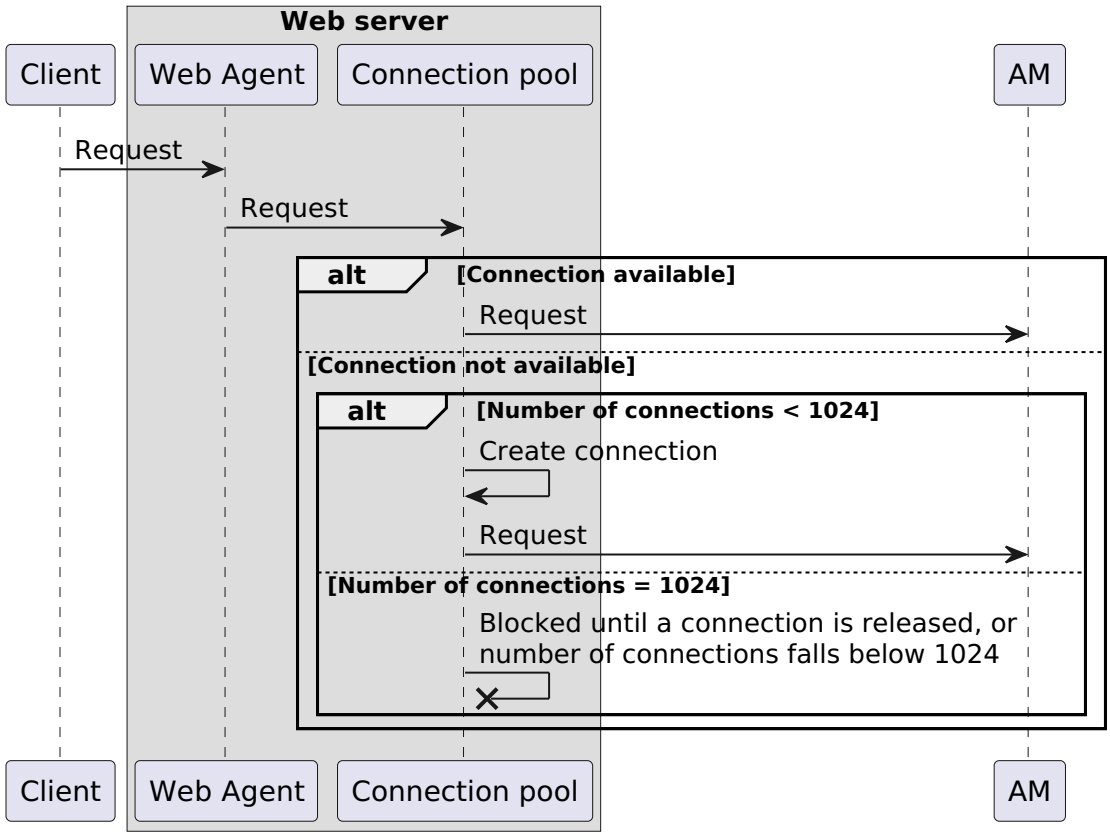
Use a connection pool between Web Agents Documentation and AM to cache and reuse connections, and so reduce the overhead of creating new connections. The agent can use an array of connections concurrently, with multiple request threads.

To enable connection pooling, set Enable Connection Pooling to `true`. Test and tune the performance of your deployment with connection pooling before you use it in a production environment.

The following image shows the architecture of a connection pool:



The following image shows the flow of information when a request is treated in a connection pool:



When a client makes a request, the agent intercepts the request and uses the connection pool to connect to AM. If a connection is available, the agent uses that connection. The client is unaware of the connection reuse.

If a connection is not available, and fewer than 1024 connections are in use, the agent creates and uses a new connection. If 1024 connections are already in use, the request waits until an existing connection is released, or a new connection can be created.

When 1024 connections are in use, the agent creates additional temporary connections. Connections can be closed by AM/IDC, but the agent reopens them when it detects that they are closed.

When the request is complete, the agent closes the connection to the pool, but retains the physical connection. The connection is then available to requests with the same connection parameters.

Consider the following points for connection pooling:

- The connection pool can contain up to 1024 cached connections
- When more than 1024 connections are required, the agent creates temporary connection.
- By default, connections timeout after four seconds of waiting for a response. To change this value, configure Connection Timeout
- Tune Connection Timeout so that it is:
  - Long enough for systems to respond, and therefore prevent unnecessary failures
  - As short as possible to minimize the time to wait after a network failure

- To reduce the overhead of making new connections and SSL handshakes, set the HTTP keep-alive headers for AM containers or reverse proxies to longer than Connection Timeout.

ForgeRock Identity Cloud guide

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

About this guide

This guide is for customers using an agent-based integration model, with ForgeRock Access Management on-premise, or another on-premise access management solution. The guide provides an example of how to transition from on-premise access management to Identity Cloud without changing the architecture of the agent-based model.

The examples in this document are based on an available version of Identity Cloud. As Identity Cloud evolves, the examples in this document will be updated to reflect the changes.

Example installation for this guide

Identity Cloud is described in the Identity Cloud Docs.

Find the value of the following properties:

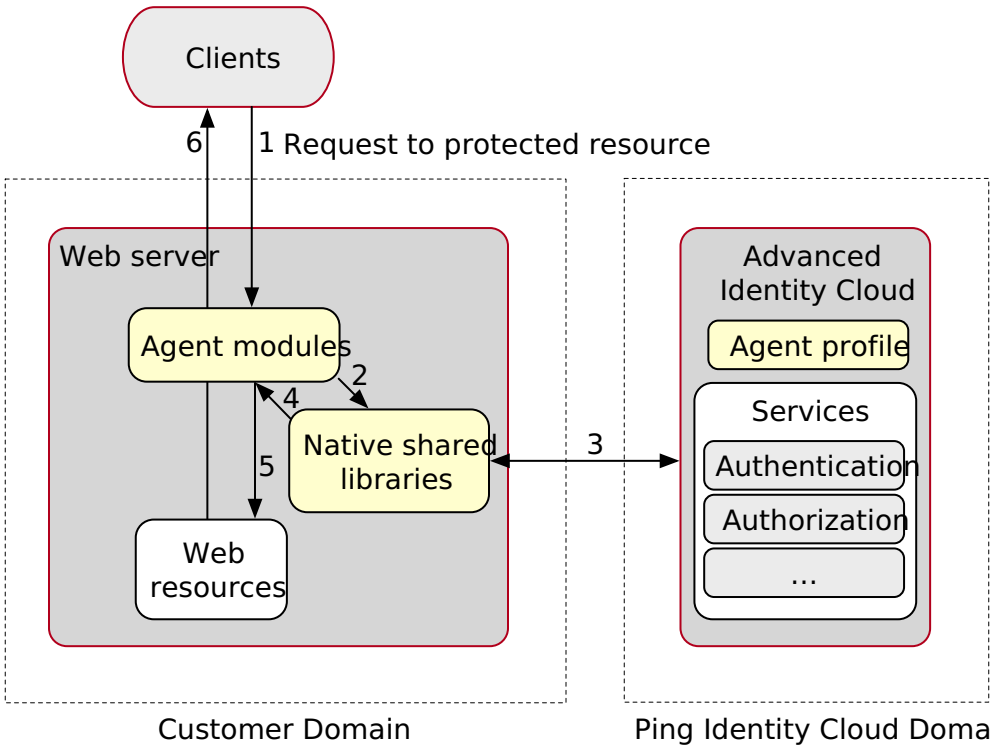
- The agent URL. This guide uses Web Agents Documentation installed on `http://agent.example.com:80`, in the `alpha` realm.
- The root URL of your Identity Cloud. This guide uses `https://tenant.forgeblocks.com`.
- The server URL of the Access Management component of the Identity Cloud. This guide uses `https://tenant.forgeblocks.com/am`.
- The realm where you work. This guide uses `alpha`.

If you use a different configuration, substitute in the procedures accordingly.

About Web Agents Documentation and the Identity Cloud

Identity Cloud simplifies the consumption of ForgeRock as an Identity Platform. However, many organizations have business web applications and APIs deployed across multiple clouds, or on-premise. This guide provides an example of how to use Web Agents Documentation with the Identity Cloud, without changing the architecture of the agent-based model.

The following image illustrates the flow of an inbound request to a website, through a Web Agents Documentation, and the agent's interaction with Identity Cloud to enforce resource-based policies.



For information about the Identity Cloud, refer to the Identity Cloud Docs.

Prepare for installation

For information about installing Web Agents Documentation, refer to the Installation. This section summarizes considerations for using the agent with Identity Cloud:

- Configure Identity Cloud and set up a policy before you install the agent. When you configure the agent in the Identity Cloud admin UI, you can select the policy.
- For environments with load balancers or reverse proxies, consider the communication between the agent and the Identity Cloud servers, and between the agent and the client. Do one of the following:
  - Configure the environment before you install the agent.
  - Install the agent using agentadmin --s --forceInstall to prevent the agent from trying to connect to Identity Cloud before installation.

Example installation for this guide

Unless otherwise stated, the examples in this guide assume the following installation:

- AM server URL: `https://tenant.forgeblocks.com:443/am`
- Agent URL: `http://agent.example.com:80`
- Agent profile name: `web-agent`
- Agent profile realm: `/alpha`

- Agent profile password: `/secure-directory/pwd.txt`

Add a demo user in Identity Cloud

Add a user so you can test the examples in this guide.

1. In the Identity Cloud admin UI, select **Identities** > **Manage** > **Alpha realm - Users**.
2. Add a new user with the following values:
   - **Username** : `demo`
   - **First name** : `demo`
   - **Last name** : `user`
   - **Email Address** : `demo@example.com`
   - **Password** : `Ch4ng3!t`

Create a policy set and policy in Identity Cloud

1. In the Identity Cloud admin UI, select **Native Consoles** > **Access Management**. The AM admin UI is displayed.
2. In the AM admin UI, select **Authorization** > **Policy Sets** > **New Policy Set**, and add a policy set with the following values:
   - **Id** : `PEP`
   - **Resource Types** : `URL`
3. In the policy set, add a policy with the following values:
   - **Name** : `PEP-policy`
   - **Resource Type** : `URL`
   - **Resource pattern** : `*://*:*/*`
   - **Resource value** : `*://*:*/*`
4. On the **Actions** tab, add actions to allow HTTP `GET` and `POST`.
5. On the **Subjects** tab, remove any default subject conditions, add a subject condition for all `Authenticated Users`.

Create an agent profile in Identity Cloud

1. In the Identity Cloud admin UI, go to **Gateways & Agents** > **New Gateway/Agent**, and add a Web Agents Documentation with the following values:
   - **Agent ID** : `web-agent`
   - **Password** : `password`
   - **Application URL** : `http://agent.example.com:80`
2. Click **Save Profile** and **Done**.

3. On the agent profile page, select **Use Policy Authorization**, select a policy set to assign to the profile, and then click **Save**.

   If a suitable policy set isn't available, select **Edit advanced settings** to edit or create one.

Enforce policy decisions from Identity Cloud

This example sets up ForgeRock Identity Cloud as a policy decision point for requests processed by Web Agents Documentations. For more information about Web Agents Documentations, refer to the User guide.

1. Using the ForgeRock Identity Cloud docs, log in to Identity Cloud as an administrator.
2. Make sure you are managing the `alpha` realm. If not, switch realms.
3. Create a policy set and policy.
4. Create an agent profile.

   When a policy set is assigned to the agent profile during creation, the agent uses that policy set. If a suitable policy set isn't available during creation, select **Edit advanced settings** to edit or create one and assign it to the agent profile.

5. Test the setup:

   1. Go to `http://agent.example.com:80`. The Identity Cloud login page is displayed.
   2. Log in to Identity Cloud as user `demo`, password `Ch4ng3!t`, to access the web page protected by the Web Agents Documentation.

Security guide

Use this guide to reduce risk and mitigate threats to Web Agents Documentation security.

Threats

Understand and address security threats.

Operating Systems

Secure your operating systems.

Connections

Secure network connections.

Access

Remove non-essential access and features, update patches, and manage cookies.

Keys and Secrets

Manage keys and secrets.

Audit Trails

Audit events in your deployment.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com. The common REST API provides ForgeRock Identity Platform software common ways to access web resources and collections of resources.

Threats

The following sections describe some possible threats to Web Agents Documentation, which you can mitigate by following the instructions in this guide.

Out-of-date software

Prevent the exploitation of security vulnerabilities by using up-to-date versions of the agent and third-party software.

Review and follow the Ping Identity security advisories. Follow similar lists from all of your vendors.

Cached pages in browsers and web proxies

When browsers and web proxies cache pages that are accessed by a user, the cache can include sensitive information. Caching pages in browsers and web proxies increases the risk of unwanted disclosure, especially in shared browsing environments.

Similarly, when web server responses are cached, sensitive information can be accessed by attackers. Caching web server responses is a common method to improve loading times and reduce server load.

To manage caching in the agent, set Add Cache-Control Headers so that HTTP responses generated by the agent include the Cache-Control HTTP header.

When the Cache-Control HTTP header is present, the response can't be cached by the browser. The header uses the following values: `max-age=0`, `no-cache`, `no-store`, `must-revalidate`.

If you need the Cache-Control header for each page, set it in the application or in the web server. For example, after a session has terminated, browser caching can result in a blank page being displayed or a previously cached page being shown without needing to reauthenticate. Setting the Cache-Control header in the application or in the web server prevents this from happening.

Apache

Use `mod_headers` to add Cache-Control at a global, location or VirtualHost level. For example:

```
<Location "myuri">

    Header set Cache-Control "no-store, no-cache, must-revalidate, max-age=0"
    Header set Pragma "no-cache"

</Location>
```

Learn more in Apache Module mod_headers in the Apache documentation.

NGINX

Use the `ngx_http_headers_module` to add Cache-Control at a global, location or VirtualHost level. For example:

```
add_header Cache-Control "no-store, no-cache, must-revalidate, max-age=0"
add_header Pragma "no-cache"
```

Learn more in Module ngx_http_headers_module in the NGINX documentation.

IIS

Use the `<customHeaders>` element of the `<httpProtocol>` element to add Cache-Control at a global level. For example:

```
<httpProtocol>
  <customHeaders>
    <clear />
    <add name="Cache-Control" value="no-cache, no-store, must-revalidate" />
    <add name="Pragma" value="no-cache" />
  </customHeaders>
</httpProtocol>
```

Learn more in Custom Headers <customHeaders> in the Microsoft IIS documentation.

When deciding whether to include the Cache-Control HTTP header, consider both security and the performance impact on customer applications. Setting this property can reduce performance because browser pages aren't cached.

Learn more in HTTP caching in the Mozilla developer documentation.

Reconnaissance

The initial phase of an attack sequence is often reconnaissance. Limit the amount of information available to attackers during reconnaissance, as follows:

- Avoid using words that help to identify Web Agents Documentation in error messages.

  When AM is not available, the related error message contains the agent profile name. Consider this in your choice of agent profile name.

- Configure Agent Debug Level to use the lowest level of logging necessary. For example, consider logging at the `ERROR` or `WARNING` level, instead of `TRACE` or `MESSAGE`.

Cross-site scripting

Warning

Cross-site request forgery attacks (CSRF or XSRF) can be a cause of serious vulnerabilities in web applications. It is the responsibility of the protected application to implement countermeasures against such attacks, because Web Agents Documentation cannot provide generic protection against CSRF. ForgeRock recommends following the latest guidance from the OWASP CSRF Prevention Cheat Sheet.
When POST data preservation is enabled, captured POST data that is replayed appears to come from the same origin as the protected application, not from the site that originated the request. Therefore, CSRF defenses that rely solely on checking the origin of requests, such as SameSite cookies or Origin headers, are not reliable.
To defend against CSRF attacks when POST data preservation is enabled, the agent uses a secure cookie and a nonce. The nonce must correspond to the authentication response from AM. This defense during authentication is specified in Cross-Site Request Forgery.
ForgeRock strongly recommend using token-based mitigations against CSRF, and relying on other measures only as a defense in depth, in accordance with OWASP guidance.

For more protection against cross-site scripting attacks, configure Composite Advice Encode.

POST data preservation

POST data is stored temporarily in the agent file system before a user is authenticated. Therefore, any unauthenticated user can POST a file that is then stored by the agent. Consider the following points when you configure POST data preservation:

- Payloads from unauthenticated users are stored in the agent files system. If your threat evaluation does not accept this risk, do not use POST data preservation; set Enable POST Data Preservation to `false`.
- By default, POST data is stored in the installation directory, `/path/to/web_agents/agent_type/instances/agent_n/pdp-cache`. To store POST data in a dedicated directory, set POST Data Storage Directory. Make sure that the new directory has the correct read/write permissions for the ID that the server uses.
- Set the directory permissions to minimize the following risks:
  - Permissive access to POST data.
  - Leakage of personally identifiable information (PII).
- POST data is stored for the time defined by POST Data Entries Cache Period and then deleted. To identify threats in POST data before it is deleted, make sure Intrusion Detection Systems inspect the data within the specified time.

For information about configuration properties, refer to POST data preservation.

Compromised passwords

Use secure passwords for server administration. For information about how to create the agent profile password, refer to Preinstallation tasks.

> **Tip**
>
> Although the agent accepts any password length and content, you are strongly encouraged to generate secure passwords. This can be achieved in various ways, for example, by using a password manager.

Misconfiguration

Misconfiguration can arise from bad or mistaken configuration decisions, and from poor change management. Depending on the configuration error, features can stop working in obvious or subtle ways, and potentially introduce security vulnerabilities.

For example, if a configuration change prevents the server from making HTTPS connections, many applications can no longer connect, and the problem is detected immediately. However, if a configuration change allows insecure TLS protocol versions or cipher suites for HTTPS connections, some applications negotiate insecure TLS, but appear to continue to work properly.

- Access policy is not correctly enforced.

  Incorrect parameters for secure connections and incorrect Access Control Instructions (ACI) can lead to overly permissive access to data, and potentially to a security breach.

- The server fails to restart.

  Although failure to start a server is not directly a threat to security, it can affect service availability.

To guard against bad configuration decisions, implement good change management:

- For all enabled features, document why they are enabled and what your configuration choices mean. This implies a review of configuration settings, including default settings that you accept.
- Validate configuration decisions with thorough testing.
- Maintain a record of your configurations and the changes applied.

  For example, use a filtered audit log. Use version control software for any configuration scripts and to record changes to configuration files.
- Maintain a record of external changes to the system, such as changes to operating system configuration, and updates to software, such as the JVM that introduces security changes.

Path traversal attempts

Some Java servers, such as those based on J2EE and Spring, use path parameters (also known as matrix parameters) in URL paths. These servers can process requests in a non-standard way that is unsafe for the agent forwarding the request. As a result, bad actors can make use of these parameters for path traversal attempts to access protected resources.

The agent requires the URL path to be normalized consistently to be able to effectively enforce rules.

Web Agents Documentation rejects unsafe uses of path parameters with an `HTTP 400` in the following scenarios:

- The request contains one or more `%2F` or `%2f` (encoded forward slash) characters in the path parameters.
- The request includes empty path segments or dot path segments with path parameters. Some example unsafe uses include:
  - `/;/`
  - `/..;`
  - `/.;`
  - `/..;parameter/`

  Legitimate uses of `;` as a path parameter are still permitted. For example, the agent won't reject this request with the `jessionid` parameter: `/segment1/segment2/;jsessionid=1234`

Unauthorized access

Data theft can occur when access policies are too permissive, and when the credentials to gain access are too easily cracked. It can also occur when the data is not protected, when administrative roles are too permissive, and when administrative credentials are poorly managed.

Poor risk management

Threats can arise when plans fail to account for outside risks. To mitigate risk, develop appropriate answers to at least the following questions:

- What happens when a server or an entire data center becomes unavailable?

- How do you remedy a serious security issue in the service, either in the Web Agents Documentation software or the connected systems?
- How do you validate mitigation plans and remedial actions?
- How do client applications work when the Web Agents Documentation offline?

  If client applications require always-on services, how do your operations ensure high availability, even when a server goes offline?

For critical services, test expected operation and disaster recovery operation.

Operating systems

When you deploy Web Agents Documentation, familiarize yourself with the recommendations for the host operating systems that you use. For comprehensive information about securing operating systems, refer to the CIS Benchmark documentation.

System updates

Over the lifetime of a deployment, the operating system might be subject to vulnerabilities. Some vulnerabilities require system upgrades, whereas others require only configuration changes. All updates require proactive planning and careful testing.

For the operating systems used in production, put a plan in place for avoiding and resolving security issues. The plan should answer the following questions:

- How does your organization become aware of system security issues early?

  This could involve following bug reports, mailing lists, forums, and other sources of information.
- How do you test security fixes, including configuration changes, patches, service packs, and system updates?

  Validate the changes first in development, then in one or more test environments, then in production in the same way you would validate other changes to the deployment.
- How do you roll out solutions for security issues?

  In some cases, fixes might involve both changes to the service, and specific actions by those who use the service.
- What must you communicate about security issues?
- How must you respond to security issues?

Software providers often do not communicate what they know about a vulnerability until they have a way to mitigate or fix the problem. Once they do communicate about security issues, the information is likely to become public knowledge quickly. Make sure you can expedite resolution of security issues.

To resolve security issues quickly, make sure you are ready to validate any changes that must be made. When you validate a change, check that the fix resolves the security issue. Validate that the system and Web Agents Documentation software continue to function as expected in all the ways they are used.

System audits

System audit logs make it possible to uncover system-level security policy violations that are not recorded in Web Agents Documentation, such as unauthorized access to Web Agents Documentation files. Such violations are not recorded in Web Agents Documentation logs or monitoring information.

Also consider how to prevent or at least detect tampering. A malicious user violating security policy is likely to try to remove evidence of how security was compromised.

Unused features

By default, operating systems include many features, accounts, and services that Web Agents Documentation software does not require. Each optional feature, account, and service on the system brings a risk of additional vulnerabilities. To reduce the surface of attack, enable only required features, system accounts, and services. Disable or remove those that are not needed for the deployment.

The features needed to run and manage Web Agents Documentation software securely include the following:

- Software to secure access to service management tools; in particular, when administrators access the system remotely.
- Software to secure access for remote transfer of software updates, backup files, and log files.
- Software to manage system-level authentication, authorization, and accounts.
- Firewall software, intrusion-detection/intrusion-prevention software.
- Software to allow auditing access to the system.
- System update software to allow updates that you have validated previously.
- If required for the deployment, system access management software such as SELinux.
- Any other software that is clearly indispensable to the deployment.

Consider the minimal installation options for your operating system, and the options to turn off features.

Consider configuration options for system hardening to further limit access even to required services.

For each account used to run a necessary service, limit the access granted to the account to what is required. This reduces the risk that a vulnerability in access to one account affects multiple services across the system.

Make sure you validate the operating system behavior every time you deploy new or changed software. When preparing the deployment and when testing changes, maintain a full operating system with Web Agents Documentation software that is not used for any publicly available services, but only for troubleshooting problems that might stem from the system being *too* minimally configured.

Network connections

Protect network traffic by using HTTPS where possible.

Recommendations For Incoming Connections (From Clients to Web Agents Documentation)

Protocol	Recommendations
HTTP	HTTP connections that are not protected by SSL/TLS use cleartext messages. When you permit insecure connections, you cannot prevent client applications from sending sensitive data. For example, a client could send unprotected credentials in an HTTP Authorization header. Even if the server were to reject the request, the credentials would already be leaked to any eavesdroppers. Always use HTTPS for connections up to a load-balancer or proxy in front of the web application or server.
HTTPS	Use HTTPS for secure connections. Follow industry-standard TLS recommendations for Security/Server Side TLS. When using an HTTP connection handler, use HTTPS to protect client connections. Some client applications require a higher level of trust, such as clients with additional privileges or access. Client application deployers might find it easier to manage public keys as credentials than to manage username/password credentials. Client applications can use SSL client authentication.

Recommendations For Outgoing Connections (From Web Agents Documentation to Another Service)

Client	Recommendations
Common Audit event handlers	Configure ForgeRock Common Audit event handlers to use HTTPS when connecting to external log services.

Message-level security

Server protocols such as HTTP, LDAP, and JMX rely on TLS to protect connections. To enforce secure communication, refer to Configure SSL communication between the agent and AM.

Communication between the agent and clients is managed by the web server in which the agent runs. For information about how to secure connections, refer to the web server documentation.

Access

The following sections describe how to restrict non-essential access to your deployment, and reduce the amount of non-essential information that it provides.

Remove non-essential features

The more features you have turned on, the more features you need to secure, patch, and audit. If something is not being used, uninstall it, disable it, or protect access to it.

Remove non-essential access

Make sure only authorized people can access your servers and applications through the appropriate network, using the appropriate ports, and presenting strong-enough credentials.

Make sure users connect to systems through the latest versions of TLS, and audit system access periodically.

Provide access only as necessary; restrict to required users, and limit their access to the information they need.

Update patches

Prevent the exploitation of security vulnerabilities by using up-to-date versions of the agent and third-party software.

Review and follow the Ping Identity security advisories. Follow similar lists from all of your vendors.

Manage agent sessions

On startup, Web Agents Documentation uses the following properties to obtain a session from AM:

- Agent Profile Name
- Agent Profile Password
- Agent Profile Realm

The session lifetime is defined by the AM version and configuration, and is essentially indefinite. Consider the following points when you configure the agent session lifetime in AM:

- If the lifetime is too short, the agent has to re-authenticate with AM too frequently, using network bandwidth and delaying user requests.
- If the lifetime is too long, the CTS can be cluttered with zombie sessions that are no longer in use.
- A value between 60 minutes and 1440 minutes (24 hours) is suitable for many use cases.

To set the agent session lifetime in AM, add the property `com.iplanet.am.session.agentSessionIdleTime` to the JVM properties in the container where the agent runs, and restart the container. The following example sets the agent session lifetime to 1440 minutes (24 hours):

```
JAVA_OPTS="$JAVA_OPTS -Dcom.iplanet.am.session.agentSessionIdleTime=1440"
```

Expire Identity Cloud and AM sessions

To minimize the time an attacker can attack an active session, set expiration timeouts for every Identity Cloud and AM session. Set timeouts according to the context of the deployment, balancing security and usability, so that the user can complete operations without the session frequently expiring.

For more information, refer to OWASP's Session Management Cheat Sheet.

Set a maximum session lifetime and idle time in Identity Cloud:

- In the Identity Cloud admin UI, select **Native Consoles** > **Access Management**.
- In the AM admin UI, select **Services** > **Add a Service** and add a **Session** service.
- Specify the following properties in minutes:
  - **Maximum Session Time**
  - **Maximum Idle Time**

Set a maximum session lifetime and idle time in Access Management:

- In the AM admin UI, select **Services** > **Add a Service** and add a **Session** service.
- Specify the following properties in minutes:
  - **Maximum Session Time**
  - **Maximum Idle Time**

Manage cookies

Increase the security of cookies generated by Web Agents Documentation or the protected application in the following ways:

- To prevent cookies from being easily associated with an application, change the default name of key cookies. For example, change the SSO cookie in Cookie Name.
- To transmit securely all cookies written by the agent, set Enable Cookie Security.
- To reduce the risk of cross-site request forgery (CSRF) attacks, set the SameSite attribute of cookies in SameSite Cookie Attribute.
- To ensure that cookies cannot be accessed through client-side scripts, and to mitigate any XSS attacks, set Enable HTTP Only Mode to create cookies with the `httpOnly` flag.
- To make cookies accessible only from HTTPS sites, prefix the cookie name with `__Secure-`. A forged insecure site cannot overwrite a secure cookie.
- To make cookies accessible only on the same host where they are set, prefix the cookie name with `__Host-`. A subdomain cannot overwrite the cookie value.

Keys and secrets

Web Agents Documentation uses cryptographic keys for encryption, signing, and securing network connections, and passwords. The following sections discuss how to secure keys and secrets in your deployment.

Use strong keys

Small keys are easily compromised. Use at least the recommended key size.

For more information about strong encryption, refer to the documentation for the web server where the agent runs. For NGINX, for example, refer to Security controls.

Rotate keys

Rotate keys regularly to:

- Limit the amount of data protected by a single key.
- Reduce dependence on specific keys, making it easier to migrate to stronger algorithms.
- Prepare for when a key is compromised. The first time you try key rotation shouldn't be during a real-time recovery.
- Conform to internal business compliance requirements.

Audits and logs

Audit trails

For security, troubleshooting, and regulatory compliance, agents are able to audit information for allowed and/or denied requests.

The agent audit logging service adheres to the log structure common across the ForgeRock Identity Platform. For information, refer to Auditing.

Web Agents Documentation supports propagation of the transaction ID across the ForgeRock Identity Platform, using the HTTP header `X-ForgeRock-TransactionId`. Consider configuring this header to prevent malicious actors from flooding the system with requests using the same transaction ID header to hide their tracks. For information, refer to Configuring the trust transaction header system property in AM's *Security guide*.

Log files

Agent logs contain informational, error, and warning events, to troubleshoot and debug transactions and events that take place within the agent instance.

Protect logs from unauthorized access, and make sure they contain a minimum of sensitive or personally identifiable information that could be used in attacks.

Make sure Agent Debug Level is the lowest level of logging necessary. For example, consider logging at the `ERROR` or `WARNING` level, instead of `TRACE` or `MESSAGE`. For more information, refer to logging configuration properties.

Properties reference

This reference describes agent configuration properties.

When you create an agent profile, you choose whether to store the agent configuration in AM's configuration store or locally to the agent installation. The local configuration file syntax is the same as that of a standard Java properties file.

Property files

The agent stores agent bootstrap and configuration properties in the file `agent.conf`.

For IIS Web Agent, the file is located by default at `C:\web_agents\iis_agent\instances\agent_1\config\agent.conf`

List of bootstrap properties

Property	Description	Function
com.forgerock.agents.config.plain.channels.insecure	Accept Secure Cookies From AM Over HTTP	Encryption
com.sun.identity.agents.config.debug.file.size	Agent Debug File Size	Debug
com.sun.identity.agents.config.debug.level	Agent Debug Level	Debug
com.sun.identity.agents.config.username	Agent Profile Name	Agent profile
com.sun.identity.agents.config.password	Agent Profile Password	Agent profile
com.sun.identity.agents.config.key	Agent Profile Password Encryption Key	Agent profile
com.sun.identity.agents.config.organization.name	Agent Profile Realm	Agent profile
com.sun.identity.agents.config.naming.url	AM Connection URL	Miscellaneous
com.forgerock.agents.config.cert.ca.file	CA Certificate File Name	Encryption
com.sun.identity.agents.config.connect.timeout	Connection Timeout	Miscellaneous

Property	Description	Function
com.sun.identity.agents.config.key.cache.disable	Disable Caching of Agent Profile Password Encryption Key	Encryption
org.forgerock.agents.config.connection.pool.enable	Enable Connection Pooling	Connection pooling
org.forgerock.agents.config.fragment.redirect.enable	Enable Fragment Redirect	Fragment redirect
org.forgerock.openam.agents.config.multivalue.pre.authn.cookies	Enable Multivalue for Pre-Authn Cookie	Cookies
com.sun.identity.agents.config.notification.enable	Enable Notifications	Profile
org.forgerock.agents.config.secure.channel.disable	Enable OpenSSL to Secure Internal Communications	Encryption
com.forgerock.agents.config.hostmap	Hostname to IP Address Map	General
org.forgerock.openam.agents.config.jwt.name	JWT Cookie Name	Profile
com.sun.identity.agents.config.local.audit.logfile	Local Agent Audit File Name	Audit
com.sun.identity.agents.config.local.logfile	Local Agent Debug File Name	Logs
com.sun.identity.agents.config.local.log.size	Local Audit Log Rotation Size	Logs
com.sun.identity.agents.config.repository.location	Location of Agent Configuration Repository	Profile
com.forgerock.agents.config.max.num.log.files	Maximum Number of Debug Log Files	Logs

Property	Description	Function
com.forgerock.agents.config.fallback.mode	Not-Enforced Fallback Mode	Not-enforced
org.forgerock.agents.config.cert.verify.depth	OpenSSL Certificate Verification Depth	Encryption
org.forgerock.openam.agents.config.policy.evaluation.realm	Policy Evaluation Realm	Policy client service
org.forgerock.openam.agents.config.policy.evaluation.application	Policy Set	Policy client service
org.forgerock.agents.config.postdata.preserve.dir	POST Data Storage Directory	POST data preservation
com.forgerock.agents.config.cert.key	Private Client Certificate File Name	Encryption
com.forgerock.agents.config.cert.key.password	Private Key Password	Encryption
com.sun.identity.agents.config.forward.proxy.host	Proxy Server Host Name	Forward proxy
com.sun.identity.agents.config.forward.proxy.password	Proxy Server Password	Forward proxy
com.sun.identity.agents.config.forward.proxy.port	Proxy Server Port	Forward proxy
com.sun.identity.agents.config.forward.proxy.user	Proxy Server User	Forward proxy
com.forgerock.agents.config.cert.file	Public Client Certificate File Name	Encryption
org.forgerock.agents.config.iis.headers.server.disable	Remove IIS HTTP Server Header	Microsoft IIS server
org.forgerock.agents.config.tls	Security Protocol List	Miscellaneous

Property	Description	Function
com.sun.identity.agents.config.trust.server.certs	Server Certificate Trust	Encryption
com.forgerock.agents.config.ciphers	Supported Cipher List	Encryption
com.sun.identity.agents.config.receive.timeout	TCP Receive Timeout	Miscellaneous
com.forgerock.agents.config.use.during.update	Use Cached Configuration After Update	Profile
org.forgerock.openam.agents.config.balance.websocket.connection.interval.in.minutes	Web Socket Connection Interval	Profile

List of all properties

Property	Description (UI name)	Function
com.forgerock.agents.config.plain.channels.insecure	Accept Secure Cookies From AM Over HTTP	Encryption
com.forgerock.agents.accept.sso.token	Accept SSO Token	Cookies
com.forgerock.agents.accept.ipdp.cookie	Accept SSO token cookie (deprecated)	Profile
com.forgerock.agents.cache_control_header.enable	Add Cache-Control Headers	Headers
com.sun.identity.agents.config.debug.file.size	Agent Debug File Size	Debug
com.sun.identity.agents.config.debug.level	Agent Debug Level	Debug
com.sun.identity.agents.config.agenturi.prefix	Agent Deployment URI Prefix	Profile

Property	Description (UI name)	Function
com.forgerock.agents.agent.logout.url.regex	Agent Logout URL Regular Expression (deprecated)	Logout redirect
com.forgerock.agents.jwt.aud.whitelist	Agent Profile ID Allow List	Profile
com.sun.identity.agents.config.username	Agent Profile Name	Agent profile
com.sun.identity.agents.config.password	Agent Profile Password	Agent profile
com.sun.identity.agents.config.key	Agent Profile Password Encryption Key	Agent profile
com.sun.identity.agents.config.organization.name	Agent Profile Realm	Agent profile
sunIdentityServerDeviceKeyValue	Agent Root URL for CDSSO	Profile
com.forgerock.agents.conditional.login.url	AM Conditional Login URL	Login redirect
com.sun.identity.agents.config.naming.url	AM Connection URL	Miscellaneous
com.sun.identity.agents.config.login.url	AM Login URL	Login redirect
com.sun.identity.agents.config.logout.url	AM Logout URL	Logout redirect
com.sun.identity.agents.config.anonymous.user.enable	Anonymous User	Client identification
com.sun.identity.agents.config.attribute.multi.value.separator	Attribute Multi-Value Separator	Attribute processing
com.sun.identity.agents.config.audit.accesstype	Audit Access Types	Audit
com.sun.identity.agents.config.log.disposition	Audit Log Location	Audit

Property	Description (UI name)	Function
com.forgerock.agents.config.auth.flow.callback	Authorization flow for applications using Javascript	Login redirect
com.forgerock.agents.config.cert.ca.file	CA Certificate File Name	Encryption
com.sun.identity.agents.config.cdsso.redirect.uri	CDSSO Redirect URI	Cross-domain single sign-on
com.sun.identity.agents.config.client.hostname.header	Client Hostname Header	Client identification
com.sun.identity.agents.config.client.ip.header	Client IP Address Header	Client identification
com.sun.identity.agents.config.client.ip.validation.enable	Client IP Validation	Not-enforced
com.forgerock.agents.advice.b64.url.encode	Composite Advice Encode	Advice handling
com.sun.am.use_redirect_for_advice	Composite Advice Handling	Advice handling
com.sun.identity.agents.config.polling.interval	Configuration Reload Interval	Profile
com.sun.identity.agents.config.connect.timeout	Connection Timeout	Miscellaneous
org.forgerock.openam.agents.config.continuous.security.cookies	Continuous Security Cookie Map	Continuous Security
org.forgerock.openam.agents.config.continuous.security.headers	Continuous Security Header Map	Continuous Security
com.sun.identity.agents.config.cdsso.cookie.domain	Cookie Domain List	Cross-domain single sign-on
com.sun.identity.agents.config.cookie.name	Cookie Name	Cookies
com.sun.identity.agents.config.cookie.reset	Cookie Reset List	Cookies

Property	Description (UI name)	Function
com.sun.identity.agents.config.freeformproperties	Custom Properties	Custom
com.forgerock.agents.jwt.aud.disable	Disable Audience Claim Validation	Profile
com.sun.identity.agents.config.key.cache.disable	Disable Caching of Agent Profile Password Encryption Key	Encryption
com.forgerock.agents.config.logout.redirect.disable	Disable Logout Redirection	Logout redirect
com.forgerock.agents.config.add.amlbcookie	Enable AM Load Balancer Cookie	Load balancing
org.forgerock.agents.config.connection.pool.enable	Enable Connection Pooling	Connection pooling
com.sun.identity.agents.config.cookie.reset.enable	Enable Cookie Reset	Cookies
com.sun.identity.agents.config.cookie.secure	Enable Cookie Security	Cookies
org.forgerock.openam.agents.config.allow.custom.login	Enable Custom Login Mode	Login redirect
com.sun.identity.agents.config.fqdn.check.enable	Enable FQDN Check	FQDN check
org.forgerock.agents.config.fragment.redirect.enable	Enable Fragment Redirect	Fragment redirect
com.sun.identity.cookie.httponly	Enable HTTP Only Mode	Cookies
org.forgerock.agents.config.logout.session.invalidate	Enable Invalidate Logout Session	Logout redirect
org.forgerock.openam.agents.config.multivalue.pre.authn.cookies	Enable Multivalue for Pre-Authn Cookie	Cookies

Property	Description (UI name)	Function
com.sun.identity.agents.config.notification.enable	Enable Notifications	Profile
com.sun.identity.agents.config.change.notification.enable	Enable Notifications of Agent Configuration Change	Profile
org.forgerock.agents.config.secure.channel.disable	Enable OpenSSL to Secure Internal Communications	Encryption
com.sun.identity.agents.config.override.host	Enable Override Request URL Host	Load balancing
com.sun.identity.agents.config.override.port	Enable Override Request URL Port	Load balancing
com.sun.identity.agents.config.override.protocol	Enable Override Request URL Protocol	Load balancing
com.sun.identity.agents.config.postdata.preserve.enable	Enable POST Data Preservation	POST data preservation
org.forgerock.agents.config.logout.regex.enable	Enable Regex for Logout URL List	Logout redirect
org.forgerock.agents.config.notenforced.ext.regex.enable	Enable Regular Expressions for Not-Enforced IPs	Not-enforced
com.sun.identity.agents.config.get.client.host.name	Enable Retrieve Client Hostname	Policy client service
org.forgerock.agents.config.cdsso.advice.cleanup.disable	Enable Session Cookie Reset After Authentication Redirect	Cross-domain single sign-on

Property	Description (UI name)	Function
com.sun.identity.agents.config.sso.only	Enable SSO Only Mode	General
com.sun.identity.agents.config.url.comparison.case.ignore	Enable URL Comparison Case Sensitivity Check	URL handling
com.sun.identity.agents.config.encode.cookie.special.chars.enable	Encode Special Characters in Cookies	Cookies
com.sun.identity.agents.config.encode.url.special.chars.enable	Encode Special Characters in URLs	URL handling
com.sun.identity.agents.config.notenforced.url.attributes.enable	Fetch Attributes for Not-Enforced URLs	Not-enforced
com.sun.identity.agents.config.fetch.from.root.resource	Fetch Policies From The Root Resource	Policy client service
com.sun.identity.agents.config.fqdn.default	FQDN Default	FQDN check
com.sun.identity.agents.config.fqdn.mapping	FQDN Virtual Host Map	FQDN check
com.sun.identity.agents.config.redirect.param	Goto Parameter Name	Goto parameter
group	Group	Profile
group	Group	Profile
org.forgerock.agents.config.json.header	Headers and Values to Receive JSON-Formatted Responses	JSON-formatted response
com.forgerock.agents.config.hostmap	Hostname to IP Address Map	General

Property	Description (UI name)	Function
org.forgerock.agents.config.json.response.code	HTTP Return Code for JSON-Formatted Responses	JSON-formatted response
com.sun.identity.agents.config.ignore.path.info.for.not.enforced.list	Ignore Path Info in Not-Enforced URLs	Not-enforced
com.sun.identity.agents.config.ignore.path.info	Ignore Path Info in Request URLs	Ignore path info
com.forgerock.agents.agent.invalid.url.regex	Invalid URL Regular Expression	URL handling
com.sun.identity.agents.config.notenforced.url.invert	Invert Not-Enforced URLs	Not-enforced
org.forgerock.agents.config.json.url.invert	Invert Properties That Receive JSON-Formatted Responses	JSON-formatted response
org.forgerock.openam.agents.config.jwt.name	JWT Cookie Name	Profile
org.forgerock.agents.config.json.url	List of URLs to Receive JSON-Formatted Responses	JSON-formatted response
com.sun.identity.agents.config.local.audit.logfile	Local Agent Audit File Name	Audit
com.sun.identity.agents.config.local.logfile	Local Agent Debug File Name	Logs
com.sun.identity.agents.config.local.log.size	Local Audit Log Rotation Size	Logs
com.sun.identity.agents.config.repository.location	Location of Agent Configuration Repository	Profile

Property	Description (UI name)	Function
com.sun.identity.agents.config.iis.logonuser	Logon and Impersonation	Microsoft IIS server
com.sun.identity.agents.config.logout.redirect.url	Logout Redirect URL	Logout redirect
com.sun.identity.agents.config.agent.logout.url	Logout URL List	Logout redirect
com.forgerock.agents.config.max.num.log.files	Maximum Number of Debug Log Files	Logs
com.forgerock.agents.header.mime.encode	MIME-Encode HTTP Header Values	Headers
com.forgerock.agents.config.fallback.mode	Not-Enforced Fallback Mode	Not-enforced
com.sun.identity.agents.config.notenforced.ip	Not-Enforced IP List	Not-enforced
org.forgerock.agents.config.notenforced.ipurl	Not-Enforced URL from IP Processing List	Not-enforced
com.sun.identity.agents.config.notenforced.url	Not-Enforced URL List	Not-enforced
org.forgerock.agents.config.cert.verify.depth	OpenSSL Certificate Verification Depth	Encryption
password	Password	Profile
org.forgerock.agents.config.cdsso.persistent.cookie.enable	Persist JWT Cookie	Cookies
com.sun.identity.agents.config.policy.cache.polling.interval	Policy Cache Polling Period	Policy client service
com.sun.identity.agents.config.policy.clock.skew	Policy Clock Skew	Policy client service

Property	Description (UI name)	Function
org.forgerock.openam.agents.config.policy.evaluation.realm	Policy Evaluation Realm	Policy client service
org.forgerock.openam.agents.config.policy.evaluation.application	Policy Set	Policy client service
com.sun.identity.agents.config.postcache.entry.lifetime	POST Data Entries Cache Period	POST data preservation
com.sun.identity.agents.config.postdata.preserve.stickysession.mode	POST Data Sticky Load Balancing Mode	Load balancing
com.sun.identity.agents.config.postdata.preserve.stickysession.value	POST Data Sticky Load Balancing Value	Load balancing
org.forgerock.agents.config.postdata.preserve.dir	POST Data Storage Directory	POST data preservation
com.forgerock.agents.config.cert.key	Private Client Certificate File Name	Encryption
com.forgerock.agents.config.cert.key.password	Private Key Password	Encryption
com.sun.identity.agents.config.profile.attribute.cookie.prefix	Profile Attribute Cookie Prefix	Cookies
com.sun.identity.agents.config.profile.attribute.fetch.mode	Profile Attribute Fetch Mode	Attribute processing
com.sun.identity.agents.config.profile.attribute.mapping	Profile Attribute Map	Attribute processing
com.sun.identity.agents.config.profile.attribute.cookie.maxage	Profile Attributes Cookie Maxage	Cookies
com.sun.identity.agents.config.forward.proxy.host	Proxy Server Host Name	Forward proxy
com.sun.identity.agents.config.forward.proxy.password	Proxy Server Password	Forward proxy

Property	Description (UI name)	Function
com.sun.identity.agents.config.forward.proxy.port	Proxy Server Port	Forward proxy
com.sun.identity.agents.config.forward.proxy.user	Proxy Server User	Forward proxy
com.forgerock.agents.public.am.url	Public AM URL	Login URL
com.forgerock.agents.config.cert.file	Public Client Certificate File Name	Encryption
org.forgerock.agents.config.conditional.login.pattern	Regular Expression Conditional Login Pattern	Login redirect
org.forgerock.agents.config.conditional.login.url	Regular Expression Conditional Login URL	Login redirect
com.forgerock.agents.notenforced.url.regex.enable	Regular Expressions for Not-Enforced URLs	Not-enforced
org.forgerock.agents.config.iis.headers.server.disable	Remove IIS HTTP Server Header	Microsoft IIS server
com.sun.identity.agents.config.replaypasswd.key	Replay Password Key	Microsoft IIS server
com.sun.identity.agents.config.logout.cookie.reset	Reset Cookies on Logout List	Logout redirect
com.forgerock.agents.call.session.refresh	Reset Idle Timeout	General
com.sun.identity.agents.config.access.denied.url	Resources Access Denied URL	General
com.sun.identity.agents.config.response.attribute.fetch.mode	Response Attribute Fetch Mode	Attribute processing

Property	Description (UI name)	Function
com.sun.identity.agents.config.response.attribute.mapping	Response Attribute Map	Attribute processing
com.forgerock.agents.session.cache.eventually.consistent	Retain Session Cache After Configuration Change	Profile
com.forgerock.agents.cdsso.cookie.samesite	SameSite Cookie Attribute	Cookies
org.forgerock.agents.config.tls	Security Protocol List	Miscellaneous
com.sun.identity.agents.config.trust.server.certs	Server Certificate Trust	Encryption
com.sun.identity.agents.config.session.attribute.fetch.mode	Session Attribute Fetch Mode	Attribute processing
com.sun.identity.agents.config.session.attribute.mapping	Session Attribute Map	Attribute processing
com.sun.identity.agents.config.iis.password.header	Show Password in HTTP Header	Microsoft IIS server
com.sun.identity.agents.config.sso.cache.polling.interval	SSO Cache Polling Period	Policy client service
org.forgerock.agents.pdp.javascript.repost	Submit POST Data using JavaScript	POST data preservation
com.forgerock.agents.config.ciphers	Supported Cipher List	Encryption
com.sun.identity.agents.config.receive.timeout	TCP Receive Timeout	Miscellaneous
org.forgerock.agents.config.skip.post.url	URLs Ignored by the POST Data Inspector	POST data preservation
com.forgerock.agents.no.remoteuser.module.compatibility	Use Built-in Apache HTTPD Authentication Directives	Miscellaneous

Property	Description (UI name)	Function
com.forgerock.agents.config.use.during.update	Use Cached Configuration After Update	Profile
com.sun.identity.agents.config.userid.param	User ID Parameter	Policy client service
com.sun.identity.agents.config.userid.param.type	User ID Parameter Type	Policy client service
org.forgerock.openam.agents.config.balance.websocket.connection.interval.in.minutes	Web Socket Connection Interval	Profile

Advice handling

Composite Advice Encode

A flag for whether to based64 URL-encode composite advices before sending them to custom login endpoints:

`true` : Advices are encoded to increase the security, and protect against cross-site scripting attacks.

`false` : Advices are not encoded

Default: `false`

Property name	`com.forgerock.agents.advice.b64.url.encode` Introduced in Web Agent 5.7
Function	Advice handling
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous (From AM 7)` Title: `Composite Advice Encode`

Composite Advice Handling

When `true`, the agent sends composite advice in the query (GET request) instead of sending it through a POST request.

Default: `false`

Property name	`com.sun.am.use_redirect_for_advice` Introduced in Web Agent 4.x
Function	Advice handling
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous (From AM 7)` Title: `Composite Advice Handling`

Agent profile

Agent Profile Password

The password required by the agent profile, encrypted with the key specified in Agent Profile Password Encryption Key.

To encrypt an agent profile password, run the `agentadmin` command with the `--p` option.

Default: Empty

Property name	`com.sun.identity.agents.config.password` Introduced in Web Agent 4.x
Function	Agent profile
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Agent Profile Realm

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 causes the agent to fail to recognize the realm.

Default: `/`

Property name	`com.sun.identity.agents.config.organization.name` Introduced in Web Agent 4.x
Function	Agent profile
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Agent Profile Password Encryption Key

The encryption key used to encrypt the agent profile password, which should be provided in Agent Profile Password.

To create a encryption key, run the `agentadmin` command with the `--k` option.

Default: Empty

Property name	`com.sun.identity.agents.config.key` Introduced in Web Agent 4.x
Function	Agent profile
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Agent Profile Name

The name of the agent profile in AM.

Property name	`com.sun.identity.agents.config.username` Introduced in Web Agent 4.x
Function	Agent profile
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Attribute processing

Attribute Multi-Value Separator

The separator between values in a multi-valued attribute. The separator applies to all attributes, such as profile, session, and response attributes.

In this example, the attribute `HTTP_CUSTOM_TEL` has two values separated by a pipe ('|'):

```
HTTP_CUSTOM_TEL = 45354345|1234
```

Note: If you use custom code to construct a multi-valued attribute, make sure that the attribute is a string containing the individual values separated by this parameter.

Default: `|`

Property name	`com.sun.identity.agents.config.attribute.multi.value.separator` Introduced in Web Agent 4.x
Function	Attribute processing
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Attribute Multi-Value Separator`

Profile Attribute Fetch Mode

Map profile attributes to HTTP headers or HTTP cookies:

`HTTP_COOKIE` : Map to HTTP cookies

`HTTP_HEADER` : Map to HTTP headers

Default: `NONE`

Property name	`com.sun.identity.agents.config.profile.attribute.fetch.mode` Introduced in Web Agent 4.x
Function	Attribute processing
Type	Constrained Values: "http_header", "http_cookie", "none"
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Profile Attribute Fetch Mode`

Session Attribute Map

Map the value of a specified session attribute to one or more HTTP headers or HTTP cookies, depending on the value of Session Attribute Fetch Mode.

- Map key: The name of an existing session attribute for the currently authenticated user.
- Map value: The name of one or more HTTP headers or HTTP cookies.

If the HTTP header or HTTP cookie name does not exist, the agent creates it. If the session attribute name (key) does not exist, the agent does not create the HTTP header or HTTP cookie.

> **Note**
>
> Underscores in header names can cause errors in some web containers. Either don't use underscores in header names, or see your web container documentation for information about how they are managed.

When an HTTP header name is used in a request header, it is prefixed by `HTTP_`. The agents automatically changes lower case letters to upper case, and hyphens (`-`) to underscores (`_`). For example, `CUSTOM-userid` becomes `HTTP_CUSTOM-USERID`.

Format:

```
com.sun.identity.agents.config.session.attribute.mapping[session_attribute]=ATTR1|ATTR2
```

Examples:

The following example maps the value of the session attribute `UserToken` to the HTTP header `CUSTOM-userid`:

`com.sun.identity.agents.config.session.attribute.mapping[UserToken]=CUSTOM-userid`.

The following example maps the value of the session attribute `UserId` to two HTTP headers:

`com.sun.identity.agents.config.session.attribute.mapping[UserId]=HEADER1|HEADER2`.

Default: Empty

Property name	`com.sun.identity.agents.config.session.attribute.mapping` Introduced in Web Agent 4.x
Function	Attribute processing
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Session Attribute Map`

Response Attribute Map

Map the value of a specified response attribute to one or more HTTP headers or HTTP cookies, depending on the value of Response Attribute Fetch Mode.

- Map key: The name of a response attribute returned by AM with a policy decision.
- Map value: The name of one or more HTTP headers or HTTP cookies.

Consider the following points for cookies:

- If an HTTP cookie with the mapped name does not exist, the agent creates it.
- If an HTTP cookie with the mapped name already exists, the agent recreates it.
- If an unauthenticated user attempts to access the protected page, the agent deletes HTTP cookies with the mapped name on the first request, and then creates them after login.
- If the profile attribute name (key) does not exist, the agent does not create the HTTP cookie.

Consider the following points for response headers:

- If a response header with the mapped name does not exist, the agent creates it.
- If an HTTP cookie with the mapped name already exists, the agent does not recreates it, it simply appends information to the header.

- If the profile attribute name (key) does not exist, the agent does not create the response header.
- Underscores in header names can cause errors in some web containers. Either don't use underscores in header names, or see your web container documentation for information about how they are managed.
- When an HTTP header name is used in a request header, it is prefixed by `HTTP_`. The agents automatically changes lower case letters to upper case, and hyphens (`-`) to underscores (`_`). For example, `CUSTOM-userid` becomes `HTTP_CUSTOM-USERID`.

Format: `response attribute = HEADER-NAME(S)`

Examples:

In the following example, the AM response attribute `uid` is mapped to `CUSTOM-User-Name`:
`com.sun.identity.agents.config.response.attribute.mapping[uid]=Custom-User-Name`

Default: Empty

Property name	`com.sun.identity.agents.config.response.attribute.mapping` Introduced in Web Agent 4.x
Function	Attribute processing
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Response Attribute Map`

Response Attribute Fetch Mode

Map response attributes to HTTP headers or HTTP cookies:

`HTTP_COOKIE`: Map to HTTP cookies

`HTTP_HEADER`: Map to HTTP headers

Default: `NONE`

Property name	`com.sun.identity.agents.config.response.attribute.fetch.mode` Introduced in Web Agent 4.x
Function	Attribute processing
Type	Constrained Values: "http_header", "http_cookie", "none"

Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Response Attribute Fetch Mode`

Session Attribute Fetch Mode

Map session attributes to HTTP headers or HTTP cookies:

`HTTP_COOKIE` : Map to HTTP cookies

`HTTP_HEADER` : Map to HTTP headers

Default: `NONE`

Property name	`com.sun.identity.agents.config.session.attribute.fetch.mode` Introduced in Web Agent 4.x
Function	Attribute processing
Type	Constrained Values: "http_header", "http_cookie", "none"
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Session Attribute Fetch Mode`

Profile Attribute Map

Map the value of a specified profile attribute to one or more HTTP headers or HTTP cookies, depending on the value of Profile Attribute Fetch Mode.

- Map key: The name of an existing profile attribute for the currently authenticated user.
- Map value: The name of one or more HTTP headers or HTTP cookies.

If the HTTP header or HTTP cookie name does not exist, the agent creates it. If the profile attribute name (key) does not exist, the agent does not create the HTTP header or HTTP cookie.

> **Note**
>
> Underscores in header names can cause errors in some web containers. Either don't use underscores in header names, or see your web container documentation for information about how they are managed.

To populate the value of profile attribute CN under `CUSTOM-Common-Name`, configure `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, `CUSTOM-userid` becomes `HTTP_CUSTOM-USERID`.

Format: `profile attribute = HEADER-NAME(S)`

Example: `[CN]=HEADER1|HEADER2`

Default: Empty

Property name	`com.sun.identity.agents.config.profile.attribute.mapping` Introduced in Web Agent 4.x
Function	Attribute processing
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Profile Attribute Map`

Audit

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`

Property name	`com.sun.identity.agents.config.audit.accesstype` Introduced in Web Agent 4.x
Function	Audit
Type	Constrained Values: "local", "remote", "both"
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Audit Access Types`

Audit Log Location

The location where the 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`

Property name	`com.sun.identity.agents.config.log.disposition` Introduced in Web Agent 4.x
Function	Audit
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Audit Log Location`

Local Agent Audit File Name

If Audit Log Location is `LOCAL` or `ALL`, this property gives the name of the local file that contains agent audit messages.

Default: `/web_agents/agent_type/instances/agent_nnn/logs/audit/audit.log`

Property name	`com.sun.identity.agents.config.local.audit.logfile` Introduced in Web Agent 4.x
Function	Audit
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Client identification

Anonymous User

Enable or disable REMOTE_USER processing for anonymous users.

Default: `false`

Property name	`com.sun.identity.agents.config.anonymous.user.enable` Introduced in Web Agent 4.x
Function	Client identification
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous` Title: `Anonymous User`

Client Hostname Header

Name of the HTTP header that holds the hostname of the client.

Default: Empty

Property name	`com.sun.identity.agents.config.client.hostname.header` Introduced in Web Agent 4.x
Function	Client identification
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Client Hostname Header`

Client IP Address Header

Name of the HTTP header that holds the IP address of the client.

Default: Empty

Property name	`com.sun.identity.agents.config.client.ip.header` Introduced in Web Agent 4.x
Function	Client identification
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Client IP Address Header`

Connection pooling

Enable Connection Pooling

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.

Default: `true`

Property name	`org.forgerock.agents.config.connection.pool.enable` Introduced in Web Agent 5.8
Function	Connection pooling
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	Yes
Required property	No
Restart required	No

Continuous Security

Continuous Security Cookie Map

Map of cookie name to entry in the environmental conditions map, used during policy evaluation:

- Map key: Cookie name in the inbound request
- Map value: Name of the entry in the environmental conditions map that contains the value of `cookie_name`

This property has the format `[cookie_name]=map_entry_name`, where:

Example:

`org.forgerock.openam.agents.config.continuous.security.cookies[trackingcookie1]=myCookieEntry`

Default: Empty

Property name	`org.forgerock.openam.agents.config.continuous.security.cookies` Introduced in Web Agent 4.x
Function	Continuous Security
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No

AM console	Tab: `Application` Title: `Continuous Security Cookie Map`

Continuous Security Header Map

Map of header name to entry in the environmental conditions map, used during policy evaluation:

- Map key: Header name in the inbound request
- Map value: Name of the entry in the environmental conditions map that contains the value of the header name

Example:

`org.forgerock.openam.agents.config.continuous.security.headers[User-Agent]=myUserAgentHeaderEntry`

Default: Empty

Property name	`org.forgerock.openam.agents.config.continuous.security.headers` Introduced in Web Agent 4.x
Function	Continuous Security
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Continuous Security Header Map`

Cookies

Accept SSO Token

A flag for whether the agent accepts SSO tokens and ID tokens as session cookies:

- `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. Set this property to `1` only for environments migrating from earlier versions of the agent, in the following scenarios:
  - Your custom login pages use SSO tokens as session tokens, and Enable Custom Login Mode is set to `2`.
  - Your applications, for example, REST or JavaScript clients, can only set SSO tokens.

The SSO token name is given by Cookie Name.

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.

The agent caches session information for SSO tokens.

Configure this property with Enable Custom Login Mode, as described in Login redirect configuration options in the *User Guide*.

Default: `0`

Property name	`com.forgerock.agents.accept.sso.token` Introduced in Web Agent 5.7
Function	Cookies
Type	Integer
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `SSO` Title: `Accept SSO Token`

Persist JWT Cookie

A flag to persist JWT cookies. If `true` the JWT cookie is not set as a Session Cookie.

Default: `false`

Property name	`org.forgerock.agents.config.cdsso.persistent.cookie.enable` Introduced in Web Agent 4.x
Function	Cookies
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `SSO (From AM 7)` Title: `Persist JWT Cookie`

Enable 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 name	`com.sun.identity.agents.config.cookie.secure` Introduced in Web Agent 4.x
Function	Cookies
Type	Integer
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `SSO` Title: `Enable Cookie Security`

SameSite Cookie Attribute

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`.

The attribute is not set for some browsers and circumstances, as described in https://www.chromium.org/updates/same-site/incompatible-clients.

Default: Empty

Property name	`com.forgerock.agents.cdsso.cookie.samesite` Introduced in Web Agent 5.7
Function	Cookies
Type	String
Bootstrap property	No
Required property	No
Restart required	No

AM console	Tab: `SSO` Title: `SameSite Cookie Attribute`

Cookie Reset 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: Empty

Property name	`com.sun.identity.agents.config.cookie.reset` Introduced in Web Agent 4.x
Function	Cookies
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `SSO` Title: `Cookie Reset List`

Encode Special Characters in Cookies

When `true`, 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 name	`com.sun.identity.agents.config.encode.cookie.special.chars.enable` Introduced in Web Agent 4.x
Function	Cookies
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No

Restart required	No
AM console	Tab: `Miscellaneous` Title: `Encode Special Characters in Cookies`

Enable Multivalue for Pre-Authn Cookie

Web agent uses the pre-authentication cookie `agent-authn-tx` to track the progress of authentication with AM and protect the request from replay attacks.

When this property is `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 `false`, the agent creates a pre-authentication cookie for each authentication request to AM, with the name of `agent-authn-tx-string`.

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 and 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 name	`org.forgerock.openam.agents.config.multivalue.pre.authn.cookies` Introduced in Web Agent 5.7
Function	Cookies
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `SSO (From AM 7)` Title: `Enable Multivalue for Pre-Authn Cookie`

Profile Attribute Cookie Prefix

A prefix for the cookie attributes headers.

Default: `HTTP_`

Property name	`com.sun.identity.agents.config.profile.attribute.cookie.prefix` Introduced in Web Agent 4.x
Function	Cookies
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous` Title: `Profile Attribute Cookie Prefix`

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`

Property name	`com.sun.identity.agents.config.cookie.name` Introduced in Web Agent 4.x
Function	Cookies
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `SSO` Title: `Cookie Name`

Profile Attributes Cookie Maxage

Number of seconds before expiration of custom cookies or the pre-authentication cookie, `agent-authn-tx`.

Pre-authentication cookies expire when the first of the following events occurs:

- Authentication completes successfully
- They reach the age configured by this property

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 name	`com.sun.identity.agents.config.profile.attribute.cookie.maxage` Introduced in Web Agent 4.x
Function	Cookies
Type	Integer
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous` Title: `Profile Attributes Cookie Maxage`

Enable HTTP Only Mode

When `true`, mark cookies as `HttpOnly` to prevent scripts and third-party software from accessing them.

Default: `true`

Property name	`com.sun.identity.cookie.httponly` Introduced in Web Agent 4.x
Function	Cookies
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `SSO (From AM 7)` Title: `Enable HTTP Only Mode`

Enable Cookie Reset

When `true`, the agent resets (blanks) cookies in the response before redirecting to authentication by issuing a Set-Cookie header to the client. An example header could be similar to this:

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

If FQDN Default is set, the agent sets the cookie domain to the domain specified by the property. Otherwise, the agent leaves the cookie domain blank.

Default: `false`

Property name	`com.sun.identity.agents.config.cookie.reset.enable` Introduced in Web Agent 4.x
Function	Cookies
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `SSO` Title: `Enable Cookie Reset`

Cross-domain single sign-on

Enable Session Cookie Reset After Authentication Redirect

Flag to reset the session cookie after an authentication redirect:

`true`: The agent does not reset the session cookie if a policy advice is present.

`false`: The agent resets the session cookie in all configured domains on every authentication redirect when a policy advice is present.

Default: `false`

Property name	`org.forgerock.agents.config.cdsso.advice.cleanup.disable` Introduced in Web Agent 5.6
Function	Cross-domain single sign-on
Type	Boolean: `true` returns true; all other strings return `false`.

Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `SSO (From AM 7)` Title: `Enable Session Cookie Reset After Authentication Redirect`

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.

Default: `agent/cdsso-oauth2`

Property name	`com.sun.identity.agents.config.cdsso.redirect.uri` Introduced in Web Agent 4.x
Function	Cross-domain single sign-on
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `SSO` Title: `CDSSO Redirect URI`

Cookie Domain List

List of domains, such as `.example.com`, in which cookies have to be set in CDSSO. If this property empty, 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 name	`com.sun.identity.agents.config.cdsso.cookie.domain` Introduced in Web Agent 4.x
Function	Cross-domain single sign-on
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `SSO` Title: `Cookie Domain List`

Custom

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 name	`com.sun.identity.agents.config.freeformproperties` Introduced in Web Agent 4.x
Function	Custom
Type	Unused
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Custom Properties`

Debug

Agent Debug Level

Debug level. Set to one of the constrained values.

Default: `Error`

Property name	`com.sun.identity.agents.config.debug.level` Introduced in Web Agent 4.x
Function	Debug
Type	Constrained Values: "info", "warning", "error", "debug", "all"
Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Agent Debug Level`

Agent Debug File Size

File size in bytes at which the debug log file is rotated.

Default: `0`, debug log file is never rotated

Property name	`com.sun.identity.agents.config.debug.file.size` Introduced in Web Agent 4.x
Function	Debug
Type	Integer
Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `Global (From AM 7.1)` Title: `Agent Debug File Size`

Encryption

Server Certificate Trust

A flag to 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`.

Default: `true`

Property name	`com.sun.identity.agents.config.trust.server.certs` Introduced in Web Agent 4.x
Function	Encryption
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	Yes
Required property	No
Restart required	No

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.

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`

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

Default: Empty

Property name	`com.forgerock.agents.config.cert.key` Introduced in Web Agent 4.x
Function	Encryption

Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Supported Cipher List

A colon separated list of one or more ciphers to support, as defined in http://www.openssl.org/docs/apps/ciphers.html.

Property name	`com.forgerock.agents.config.ciphers` Introduced in Web Agent 4.x
Function	Encryption
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Accept Secure Cookies From AM Over HTTP

A flag to accept secure cookies.

When `true`, the agent accepts secure cookies from AM over HTTP. When `false`, the agent rejects them.

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.

> **Note**
>
> It is best practice to use HTTPS for *all* connections to AM.

Default: `false`

Property name	`com.forgerock.agents.config.plain.channels.insecure` Introduced in Web Agent 5.7
Function	Encryption
Type	Integer

Bootstrap property	Yes
Required property	No
Restart required	No

Disable Caching of Agent Profile Password Encryption Key

By default, the encryption key for the agent profile password is cached. Set this property to `true` to disable caching and require the agent to read the encryption key every time it is needed.

Default: `false`

Property name	`com.sun.identity.agents.config.key.cache.disable` Introduced in Web Agent 2023.6
Function	Encryption
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	Yes
Required property	No
Restart required	No

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. You must also configure the Private 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.

Default: Empty

Property name	`com.forgerock.agents.config.cert.file` Introduced in Web Agent 4.x

Function	Encryption
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

CA Certificate File Name

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

The file should be PEM encoded. For example:

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

```
com.sun.identity.agents.config.trust.server.certs = false
```

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

Default: Empty

Property name	`com.forgerock.agents.config.cert.ca.file` Introduced in Web Agent 4.x
Function	Encryption
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

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. For example:

```
$ /web_agents/agent-type/bin> ./agentadmin --p "Encryption Key" "cat cert_password.file"
```

```
Encrypted password value: zck+6RKqjtc=
```

Where `Encryption Key` is the value of Agent Profile Password Encryption Key.

Default: Empty

Property name	`com.forgerock.agents.config.cert.key.password` Introduced in Web Agent 4.x
Function	Encryption
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Enable OpenSSL to Secure Internal Communications

Flag for whether Windows-based agents 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 name	`org.forgerock.agents.config.secure.channel.disable` Introduced in Web Agent 4.x
Function	Encryption
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	Yes
Required property	No
Restart required	No

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 are 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 (Server Certificate Trust is `false`).

Default: `9`

Property name	`org.forgerock.agents.config.cert.verify.depth` Introduced in Web Agent 4.x
Function	Encryption
Type	Integer
Bootstrap property	Yes
Required property	No
Restart required	No

FQDN check

FQDN Virtual Host Map

Key:Value maps of incoming hostname to outgoing domain. The map key must be set; the map value can contain one or more `*` wildcards. Map keys and values are case insensitive.

This property requires Enable FQDN Check to be `true`, and FQDN Default to be set to suitable default FQDN.

Examples:

`com.sun.identity.agents.config.fqdn.mapping[agent.localtest.me]=agent.example.com`

`com.sun.identity.agents.config.fqdn.mapping[agent.localtest.me]=agent-*`

`com.sun.identity.agents.config.fqdn.mapping[agent.localtest.me]=agent-**-other-**`

`com.sun.identity.agents.config.fqdn.mapping[agent.othertest.me]=other.example.com`

For more information, refer to FQDN checking in the *User Guide*.

Property name	`com.sun.identity.agents.config.fqdn.mapping` Introduced in Web Agent 4.x
Function	FQDN check

Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `FQDN Virtual Host Map`

Enable FQDN Check

When `true`, the agent checks whether request URLs match values in FQDN Default and FQDN Virtual Host Map.

Use this property to prevent the redirect of requests in the following scenarios:

- Where resource URLs differ from the FQDNs in AM policies, for example, in load balanced and virtual host environments.
- Where hostnames are virtual, allocated dynamically, or match a pattern, for example in a Kubernetes deployment.
- Where hostnames are partial.

If FQDN Default is not set, this property is automatically set to `false`.

Default: `false`

For more information, refer to FQDN checking in the *User Guide*.

Property name	`com.sun.identity.agents.config.fqdn.check.enable` Introduced in Web Agent 4.x
Function	FQDN check
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Enable FQDN Check`

FQDN Default

The default FQDN to access resources. Set this property during agent installation, and change it only if necessary. Without this value, the web server can fail to start.

This property requires Enable FQDN Check to be `true`.

> **Note**
>
> If you specify an FQDN in this property, also add it to the Agent Root URL for CDSSO.

For more information, refer to FQDN checking in the *User Guide*.

Property name	`com.sun.identity.agents.config.fqdn.default` Introduced in Web Agent 4.x
Function	FQDN check
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `FQDN Default`

Forward proxy

Proxy Server 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: Empty

Property name	`com.sun.identity.agents.config.forward.proxy.password` Introduced in Web Agent 4.x
Function	Forward proxy
Type	String
Bootstrap property	Yes
Required property	No

Restart required	No

Proxy Server 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: Empty

Property name	`com.sun.identity.agents.config.forward.proxy.port` Introduced in Web Agent 4.x
Function	Forward proxy
Type	Integer
Bootstrap property	Yes
Required property	No
Restart required	No

Proxy Server 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: Empty

Property name	`com.sun.identity.agents.config.forward.proxy.user` Introduced in Web Agent 4.x
Function	Forward proxy
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Proxy Server Host Name

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: Empty

Property name	`com.sun.identity.agents.config.forward.proxy.host` Introduced in Web Agent 4.x
Function	Forward proxy
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Fragment redirect

Enable Fragment Redirect

A flag to 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 for request URLs marked for JSON responses, usually for non-browser clients, such as JavaScript or other coded clients.

Default: `false`

Property name	`org.forgerock.agents.config.fragment.redirect.enable` Introduced in Web Agent 5.7
Function	Fragment redirect
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	Yes

Required property	No
Restart required	No
AM console	Tab: `Advanced (From AM 7)` Title: `Enable Fragment Redirect`

General

Enable SSO Only Mode

A flag to enable SSO only mode:

- `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.
- `false`: The agent manages user authorization, by using the policy engine in AM.

> **Tip**
>
> In SSO-only mode, consider configuring Reset Idle Timeout.

Default: `false`

Property name	`com.sun.identity.agents.config.sso.only` Introduced in Web Agent 4.x
Function	General
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global (From AM 7)` Title: `Enable SSO Only Mode`

Reset Idle Timeout

A flag for whether an agent configured in SSO-only mode should refresh the user's session idle time when the user accesses a protected resource.

AM sessions have an idle timeout after which they expire. 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: `false`

Property name	`com.forgerock.agents.call.session.refresh` Introduced in Web Agent 5.7
Function	General
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Reset Idle Timeout`

Hostname to IP Address Map

Map of hostname to an IP address. The mapped hostname is automatically resolved to the IP address.

- Map key: Hostname
- Map value: IP address

Configure this property in `agent.conf` or in the `Advanced` tab of the XUI.

Format: `com.forgerock.agents.config.hostmap[0]=<Hostname>|<IP>`

Example: `com.forgerock.agents.config.hostmap[0]=am.localtest.me|10.199.0.2`

Property name	`com.forgerock.agents.config.hostmap` Introduced in Web Agent 5.0
Function	General
Type	String List

Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `Advanced (From AM 7.2)` Title: `Hostname to IP Address Map`

Resources Access Denied URL

The URL of the customized access denied page. If empty, the agent returns an HTTP status of 403 (Forbidden). The URL can be absolute or relative.

The following values are not permitted:

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

Default: Empty

Property name	`com.sun.identity.agents.config.access.denied.url` Introduced in Web Agent 4.x
Function	General
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Resources Access Denied URL`

Goto parameter

Goto Parameter Name

Renames the `goto` parameter. The 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%www.example.com%3A8020%managers%2Findex.jsp`

The URL appended to the `goto2` parameter is the URL that the user tried to access when the agent redirected the request to the `accessDenied.html` page. Note that you configure the access denied page using Resources Access Denied URL.

This property also affects AM Logout URL.

Default: `goto`

Property name	`com.sun.identity.agents.config.redirect.param` Introduced in Web Agent 4.x
Function	Goto parameter
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous` Title: `Goto Parameter Name`

Headers

MIME-Encode HTTP Header Values

MIME-encoding of HTTP header values:

- Empty 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: Empty

Property name	`com.forgerock.agents.header.mime.encode` Introduced in Web Agent 5.7
Function	Headers
Type	Integer

Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous (From AM 7)` Title: `MIME-Encode HTTP Header Values`

Add Cache-Control Headers

When `true`, enables the use of Cache-Control headers to prevent proxies from caching resources accessed by unauthenticated users.

Default: `false`

Property name	`com.forgerock.agents.cache_control_header.enable` Introduced in Web Agent 4.x
Function	Headers
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous` Title: `Add Cache-Control Headers`

Ignore path info

Ignore Path Info in Request URLs

When `true`, while doing the not-enforced list check and URL policy evaluation, strip path info from the request URL. Use this property to match to the URL without PATHINFO, as defined by the apache or IIS servers.

Example:

- If Not-Enforced URL 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.

Example:

- If Not-Enforced URL List includes `http://host/webapp/servcontext/*` and the request URL is `http://host/webapp/servcontext/example.jsp`, the path info is `/servcontext/example.jsp`. When the path info stripped is, the resulting request URL is `http://host/webapp/`, which does not match the not-enforced list. Therefore, 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.

When this property is `true`, make sure that nothing follows a wildcard in the not-enforced list or policy.

> **Note**
>
> The NGINX Plus Web Agent does not support this setting.

Default: `false`

Property name	`com.sun.identity.agents.config.ignore.path.info` Introduced in Web Agent 4.x
Function	Ignore path info
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous` Title: `Ignore Path Info in Request URLs`

JSON-formatted response

List of URLs to Receive JSON-Formatted Responses

A list of resource URLs that trigger a JSON-formatted response from the agent, and, optionally, override the default HTTP status code.

Use this property for non-browser-based, or AJAX applications, that do not want to redirect users to the AM user interface for authentication.

> **Tip**
>
> 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.

Example:

`org.forgerock.agents.config.json.url[0]=http*://.example.com:/api/*`

`org.forgerock.agents.config.json.response.code=202`

Performing a GET operation that matches the example triggers an HTTP result code `202 Accepted`, and a JSON response containing `302 Found`.

Default: Empty

Property name	`org.forgerock.agents.config.json.url` Introduced in Web Agent 4.x
Function	JSON-formatted response
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous` Title: `List of URLs to Receive JSON-Formatted Responses`

Invert Properties That Receive JSON-Formatted Responses

When `true`, the values specified in the following properties do not trigger JSON-formatted responses:

- List of URLs to Receive JSON-Formatted Responses
- Headers and Values to Receive JSON-Formatted Responses

Only non-specified values trigger JSON-formatted responses.

Default: `false`

Property name	`org.forgerock.agents.config.json.url.invert` Introduced in Web Agent 4.x
Function	JSON-formatted response
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No

AM console	Tab: `Miscellaneous (From AM 7)` Title: `Invert Properties That Receive JSON-Formatted Responses`

Headers and Values to Receive JSON-Formatted Responses

Specify HTTP headers and associated values that trigger JSON-formatted errors to be returned.

Format:

`org.forgerock.agents.config.json.header[Header]=Value`

Use with HTTP Return Code for JSON-Formatted Responses, as follows:

`org.forgerock.agents.config.json.header[enableJsonResponse]=true`

`org.forgerock.agents.config.json.response.code=202`

Performing a GET operation that matches the example triggers an HTTP result code `202 Accepted`, and a JSON response containing `302 Found`.

Default: Empty

Property name	`org.forgerock.agents.config.json.header` Introduced in Web Agent 4.x
Function	JSON-formatted response
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous (From AM 7)` Title: `Headers and Values to Receive JSON-Formatted Responses`

HTTP Return Code for JSON-Formatted Responses

An HTTP response code to return when a JSON-formatted error is triggered.

> **Tip**
>
> To prevent user agents displaying their default error pages, set to a non-error HTTP code, for example `202`.

Example:

`org.forgerock.agents.config.json.url[0]=http*://.example.com:/api/*`

`org.forgerock.agents.config.json.response.code=202`

Performing a GET operation that matches the example triggers an HTTP result code `202 Accepted`, and a JSON response containing `302 Found`.

Default: Empty

Property name	`org.forgerock.agents.config.json.response.code` Introduced in Web Agent 4.x
Function	JSON-formatted response
Type	Integer
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous (From AM 7)` Title: `HTTP Return Code for JSON-Formatted Responses`

Load balancing

Enable Override Request URL Host

Enable if the agent is behind a SSL/TLS off-loader, load balancer, or proxy, where the users and the agent use a different host. When `true`, the host is overridden with the value from Agent Deployment URI Prefix.

When the following headers are defined on the proxy or load-balancer, they override the value of Agent Deployment URI Prefix:

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

If you are using these headers, do not configure the agent to override its hostname, port, or protocol.

Default: `false`

Property name	`com.sun.identity.agents.config.override.host` Introduced in Web Agent 4.x
Function	Load balancing
Type	Boolean: `true` returns true; all other strings return `false`.

Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Enable Override Request URL Host`

Enable Override Request URL Port

Enable if the agent is behind a SSL/TLS off-loader, load balancer, or proxy, where the users and the agent use a different port. When `true`, the port is overridden with the value from Agent Deployment URI Prefix.

When the following headers are defined on the proxy or load-balancer, they override the value of Agent Deployment URI Prefix:

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

If you are using these headers, do not configure the agent to override its hostname, port, or protocol.

Default: `false`

Property name	`com.sun.identity.agents.config.override.port` Introduced in Web Agent 4.x
Function	Load balancing
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Enable Override Request URL Port`

Enable Override Request URL Protocol

Enable if the agent is behind a SSL/TLS off-loader, load balancer, or proxy, where the users and the agent use a different protocol. When `true`, the protocol is overridden with the value from Agent Deployment URI Prefix.

When the following headers are defined on the proxy or load-balancer, they override the value of Agent Deployment URI Prefix:

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

If you are using these headers, do not configure the agent to override its hostname, port, or protocol.

Default: `false`

Property name	`com.sun.identity.agents.config.override.protocol` Introduced in Web Agent 4.x
Function	Load balancing
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Enable Override Request URL Protocol`

POST Data Sticky Load Balancing Value

A key-value pair separated by the equals (=) character that the agent creates when evaluating POST Data Sticky Load Balancing Mode.

For example, a setting of `lb=myserver` either sets an `lb` cookie with `myserver` value, or adds `lb=myserver` to the URL query string.

> **Note**
>
> If this property is defined in the bootstrap agent configuration file (agent.conf), it overrides the property in the AM configuration.

Default: Empty

Property name	`com.sun.identity.agents.config.postdata.preserve.stickysession.value` Introduced in Web Agent 4.x
Function	Load balancing
Type	String

Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `POST Data Sticky Load Balancing Value`

Enable AM Load Balancer Cookie

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`

Property name	`com.forgerock.agents.config.add.amlbcookie` Introduced in Web Agent 5.8
Function	Load balancing
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global (From AM 7.1)` Title: `Enable AM Load Balancer Cookie`

POST Data Sticky Load Balancing Mode

Whether to create a cookie, or to append a query string to the URL to assist with sticky load balancing:

- `COOKIE`: The agent creates a cookie with the value specified in POST Data Sticky Load Balancing Value.
- `URL`: The agent appends the value specified in POST Data Sticky Load Balancing Value to the URL query string.

> **Note**
>
> If this property is defined in the bootstrap agent configuration file (agent.conf), it overrides the property in the AM configuration.

Default: Empty

Property name	`com.sun.identity.agents.config.postdata.preserve.stickysession.mode` Introduced in Web Agent 4.x
Function	Load balancing
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `POST Data Sticky Load Balancing Mode`

Login URL

Public AM URL

The full URL of AM when it is behind a proxy during the custom login flow. For example, `protocol://public_am_fqdn:port/am`.

Use this property when 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/am as the public AM URL.

Default: Empty

Property name	`com.forgerock.agents.public.am.url` Introduced in Web Agent 5.7
Function	Login URL
Type	String
Bootstrap property	No
Required property	No

Restart required	No
AM console	Tab: `AM Services (From AM 7)` Title: `Public AM URL`

Login redirect

AM Conditional Login URL

Conditionally redirect users based on the incoming request URL.

If the incoming request URL matches a domain name in this list, the agent redirects the unauthenticated request to the specified URL for login. The URL can be an AM instance, site, or a different website.

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`: Agents match both the domain and its subdomains. For example, `example.com` matches `mydomain.example.com` and `www.example.com`. To combine domain and path, provide the port number: `www.example.com:8080/market`.
- `Subdomain`: For example, `example.com`. To combine subdomain and path, provide the port number: `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 pipe (`|`) character. Conditional rules that do not specify the incoming request's domain apply to every incoming request.

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 incoming login requests are redirected. The URL can be 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://am.example.com/am/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 agent redirects the request to the AM instance or site specified by AM Connection URL.

?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 sets the realm parameter, for example, because it lets the user chose it. In this case, 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 logs the user in to the realm whose DNS alias matches the incoming request URL. For example, an inbound request from `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.

If you 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`: The authentication module the user authenticates against. For example, `?module=myAuthModule`.
- `service`: 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 ampersand (`&`) character, for example, `realm=value&service=value`.

When configuring conditional login with multiple URLs, set up the parameters for each URL.

Examples:

`com.forgerock.agents.conditional.login.url[0]=example.com|https://am.example.com/am/oauth2/authorize`

`com.forgerock.agents.conditional.login.url[1]=myapp.domain.com|https://am2.example.com/am/oauth2/authorize?realm=/sales`

`com.forgerock.agents.conditional.login.url[3]=sales.example.com/marketplace|https://am1.example.com/am/oauth2/authorize?realm=/sales, https://am2.example.com/am/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://am3.example.com/am/oauth2/authorize?realm=/customers&module=myAuthModule`

For more information, refer to Login redirect in the *User Guide*.

Property name	`com.forgerock.agents.conditional.login.url` Introduced in Web Agent 4.x
Function	Login redirect
Type	String Map

Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `AM Conditional Login URL`

Authorization flow for applications using Javascript

> **Important**
>
> This feature is in Technology Preview, for use only with assistance from Forgerock.

This property provides support for single page applications (SPAs) that use embedded login or authorization dialogs within `iframe` or `embed` tags.

Provide a JavaScript reference to a callback function, relative to the `iframe` or `embed` used for authentication dialogs with AM.

Use this property to enable callbacks into JavaScript applications after an authentication or transactional authorization journey.

Default: Empty

Property name	`com.forgerock.agents.config.auth.flow.callback` Introduced in Web Agent 5.10
Function	Login redirect
Type	Unused
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Authorization flow for applications using Javascript`

Regular Expression Conditional Login URL

If the incoming request URL matches a pattern specified in Regular Expression Conditional Login Pattern, the agent redirects the request to the specified URL.

The specific URL can be an AM instance, site, or a different website.

Example:

In the following example, when the request matches the regular expression `.*shop`, the agent redirects it to the `sales` realm for authentication:

```
org.forgerock.agents.config.conditional.login.pattern[0] = .*shop
```

```
org.forgerock.agents.config.conditional.login.url[0] = http://am.example.com/am/oauth2/authorize?realm=sales
```

Default: Empty

Property name	`org.forgerock.agents.config.conditional.login.url` Introduced in Web Agent 4.x
Function	Login redirect
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services (From AM 7)` Title: `Regular Expression Conditional Login URL`

AM Login URL

The URL of a custom login page to which the agent redirects users for authentication.

> **Important**
>
> When redirecting incoming login requests to a custom login page, add the login page to Not-Enforced IP List or Not-Enforced URL List.

The login URL has the format `URL[?realm=realm_name&parameter1=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.

Specify as many parameters as your custom login pages require: `https://login.example.com/login.jsp?realm=marketplace&param1=value1`

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`.
- Users should always log in to the Top Level Realm.

Even if you 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.

Default: `AMURL/am/UI/Login`

Property name	`com.sun.identity.agents.config.login.url` Introduced in Web Agent 4.x
Function	Login redirect
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `AM Login URL`

Regular Expression Conditional Login Pattern

See Regular Expression Conditional Login URL.

Property name	`org.forgerock.agents.config.conditional.login.pattern` Introduced in Web Agent 4.x
Function	Login redirect
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services (From AM 7)` Title: `Regular Expression Conditional Login Pattern`

Enable Custom Login Mode

Sets the login redirection mode, as follows:

- `0`: Disabled; default login redirection mode enabled.
- `1`: Enabled; non-OIDC compliant login flow, **standard** flow, where the agent does the following:
  - Tracks user authentication.
  - Converts the SSO token into an ID token at the end of the authentication flow.
  - Redirects the user to the originally requested resource.
  - The SSO token name is given by Cookie Name.
- `2`: (For environments migrating from earlier versions of the agent) Enabled; non-OIDC compliant login flow, **non-standard** flow, where the agent does the following:
  - Does not track user authentication.
  - Redirects the user with a `goto` query parameter to the originally requested resource.
  - Configure this property with Accept SSO Token, as described in Login redirect configuration options in the *User Guide*.

Default: `0`

Property name	`org.forgerock.openam.agents.config.allow.custom.login` Introduced in Web Agent 5.5
Function	Login redirect
Type	Integer
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services (From AM 7)` Title: `Enable Custom Login Mode`

Logout redirect

Disable Logout Redirection

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.

When this property is `true`, consider setting Enable Invalidate Logout Session to `true`.

Default: `true`

Property name	`com.forgerock.agents.config.logout.redirect.disable` Introduced in Web Agent 4.x
Function	Logout redirect
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `Disable Logout Redirection`

Logout URL List

An expression that resolves to one or more application logout URLs. When the end user accesses a logout URL, the agent triggers a logout flow. Note that your web server must be able to handle the logout URLs.

Expressions can be wildcard expressions, Perl-compatible regular expressions, or ECMAScript-compatible (IIS) regular expressions.

For more information about the logout flow and properties to manage logout, refer to Trigger logout with a URL in the *User Guide*.

Examples:

`com.forgerock.agents.agent.logout.url=*/bank/log-me-out`

`com.forgerock.agents.agent.logout.url=/logout/`

`com.forgerock.agents.agent.logout.url=https:\/\/example.domain.com:443\/(protectedA|protectedB)\?(.&)*op=logout(\&.)*$`

Default: Empty

Property name	`com.sun.identity.agents.config.agent.logout.url` Introduced in Web Agent 4.x

Function	Logout redirect
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `Logout URL List`

Enable Regex for Logout URL List

A flag to evaluate expressions in Logout URL List:

- `true` : Evaluate expressions as Perl-compatible or ECMAScript-compatible (IIS) regular expressions.
- `false` : Evaluate expressions as wildcard expressions.

For more information about the logout flow and properties to manage logout, refer to Trigger logout with a URL in the *User Guide*.

Default: `false`

Property name	`org.forgerock.agents.config.logout.regex.enable` Introduced in Web Agent 4.x
Function	Logout redirect
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services (From AM 7)` Title: `Enable Regex for Logout URL List`

Enable Invalidate Logout Session

A flag for the agent to invalidate the end user session in AM when it redirects a request to the logout URL:

- `true`: Invalidate the user session. The agent deletes its own JWT cookie and invalidates the AM session. Use when the value of Logout URL List is a page in your application, and your application **does not** handle the session invalidation process.
- `false`: Do not invalidate the user session. The agent deletes its own JWT cookie but doesn't invalidate the AM session. Use when the value of Logout URL List is:
  - A single SAML v2.0 logout page in AM
  - A page of an AM end user
  - A page in your application, and your application **does** handle the session invalidation process

When Disable Logout Redirection is `true`, consider setting this property to `true`.

Default: `true`

Property name	`org.forgerock.agents.config.logout.session.invalidate` Introduced in Web Agent 5.6
Function	Logout redirect
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services (From AM 7)` Title: `Enable Invalidate Logout Session`

Agent Logout URL Regular Expression (deprecated)

d: 2023.9

A Perl-compatible or ECMAScript-compatible (IIS) regular expression that resolves to one or more application logout URLs.

This property is deprecated; use Agent Logout URL Regular Expression (deprecated) instead.

If this property is used, it is evaluated before Enable Regex for Logout URL List.

For more information about the logout flow and properties to manage logout, refer to Trigger logout with a URL in the *User Guide*.

Example:
`com.forgerock.agents.agent.logout.url.regex=https:\/\/example.domain.com:443\/(protectedA|protectedB)\?(.\&)*op=logout(\&.)*$`

Default: Empty

Property name	`com.forgerock.agents.agent.logout.url.regex` Introduced in Web Agent 4.x
Function	Logout redirect
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services (From AM 7)` Title: `Agent Logout URL Regular Expression (deprecated)`

AM Logout URL

The page to which the agent redirects the end user on log out. It can be a page in AM, such as `https://am.example.com:8443/am/UI/Logout?realm=/alpha`, or a page in the application.

The AM logout page invalidates the user session in AM, but pages in an application might not invalidate the user session in AM. See Enable Invalidate Logout Session for configuration options.

Default: `AM_URL/am/UI/Logout`

> **Important**
>
> By default, a realm is not included in the logout URL, and the user is redirected to the root realm on logout. Take care to include a realm if required.

Property name	`com.sun.identity.agents.config.logout.url` Introduced in Web Agent 4.x
Function	Logout redirect
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No

AM console	Tab: `AM Services` Title: `AM Logout URL`

Reset Cookies on Logout List

A list of cookies to be reset upon logout in the format: `name[=value][;Domain=value]`.

For example, `Cookie2=value;Domain=subdomain.domain.com` equates to:
`com.sun.identity.agents.config.logout.cookie.reset[0]=Cookie2=value;Domain=subdomain.domain.com`

Default: Empty

Property name	`com.sun.identity.agents.config.logout.cookie.reset` Introduced in Web Agent 4.x
Function	Logout redirect
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `Reset Cookies on Logout List`

Logout Redirect URL

The page to which the agent redirects the end user on log out if Disable Logout Redirection is `false`. Configure with Logout URL List.

> **Important**
>
> This URL must be available in your web server.

Depending on the redirect URL, perform this additional configuration:

- Add the URL to the Not-Enforced URL List.
- If the URL doesn't perform a REST logout to AM, set Enable Invalidate Logout Session to `true`.
- If the URL isn't relative to AM, or in the same scheme, FQDN, and port, add it to the AM validation service.

For more information, refer to Logout in the *User Guide*.

Default: Empty

Property name	`com.sun.identity.agents.config.logout.redirect.url` Introduced in Web Agent 4.x
Function	Logout redirect
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `Logout Redirect URL`

Logs

Local Audit Log Rotation Size

The maximum size in bytes of the local audit log files. The agent rotates audit log files when they reach this size, and stores rotated files with a timestamp.

Default: `52428800`

Property name	`com.sun.identity.agents.config.local.log.size` Introduced in Web Agent 4.x
Function	Logs
Type	Integer
Bootstrap property	Yes
Required property	No
Restart required	No

Local Agent Debug File Name

The local file in which the agent writes debug log messages after startup.

During agent startup the location of the logs can be based on the container which is being used, or defined in the site configuration file for the server. For example, bootstrap logs for NGINX Plus Web Agent can be written to `/var/log/nginx/error.log`.

Default: `/web_agents/agent_type/instances/agent_nnn/logs/debug/debug.log`

Property name	`com.sun.identity.agents.config.local.logfile` Introduced in Web Agent 4.x
Function	Logs
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Maximum Number of Debug Log Files

The maximum number of debug log files to store after log rotation.

If this property is `10`, the agent stores the current debug log file and up to 9 rotated log files. When logs are rotated again and a new log file is generated, the agent deletes the oldest of the stored log files and keeps the new log file.

To store no rotated logfiles, set Agent Debug File Size to zero.

Default: `0`, store all rotated log files

Property name	`com.forgerock.agents.config.max.num.log.files` Introduced in Web Agent 5.10.1
Function	Logs
Type	Integer
Bootstrap property	Yes
Required property	No
Restart required	No

Microsoft IIS server

Show Password in HTTP Header

Set to `true` if encrypted password should be set in HTTP header `AUTH_PASSWORD`.

Default: `false`

Property name	`com.sun.identity.agents.config.iis.password.header` Introduced in Web Agent 4.x
Function	Microsoft IIS server
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Show Password in HTTP Header`

Logon and Impersonation

When `true`, the agent does Windows Logon and User Impersonation.

Default: `false`

Property name	`com.sun.identity.agents.config.iis.logonuser` Introduced in Web Agent 4.x
Function	Microsoft IIS server
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Logon and Impersonation`

Remove IIS HTTP Server Header

When `true`, the IIS agent will remove the Server header

Default: `false`

Property name	`org.forgerock.agents.config.iis.headers.server.disable` Introduced in Web Agent 2023.2

Function	Microsoft IIS server
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	Yes
Required property	No
Restart required	No

Replay Password Key

DES key for decrypting the basic authentication password in the session.

Default: Empty

Property name	`com.sun.identity.agents.config.replaypasswd.key` Introduced in Web Agent 4.x
Function	Microsoft IIS server
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Replay Password Key`

Miscellaneous

TCP 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`

Property name	`com.sun.identity.agents.config.receive.timeout` Introduced in Web Agent 5.5
Function	Miscellaneous

Type	Integer
Bootstrap property	Yes
Required property	No
Restart required	No

AM Connection URL

A space-delimited 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 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/am/`

Property name	`com.sun.identity.agents.config.naming.url` Introduced in Web Agent 4.x
Function	Miscellaneous
Type	String List
Bootstrap property	Yes
Required property	No
Restart required	No

Connection 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`

Property name	`com.sun.identity.agents.config.connect.timeout` Introduced in Web Agent 5.5
Function	Miscellaneous
Type	Integer
Bootstrap property	Yes

Required property	No
Restart required	No

Security Protocol List

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)

SSLv2 is always disabled, regardless of the setting.

This property is relevant to all Web Agents using OpenSSL libraries.

To change the default value, set an environment variable, `AM_SSL_OPTIONS`.

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

Property name	`org.forgerock.agents.config.tls` Introduced in Web Agent 4.x
Function	Miscellaneous
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No

Use Built-in Apache HTTPD Authentication Directives

A regular expression pattern to specify which not-enforced URLs can use built-in Apache authentication directives, such as `AuthName`, `FilesMatch`, and `Require`, for basic authentication.

Requests with not-enforced URLs that match the expression can use built-in Apache authentication directives.

Default: No requests can use built-in Apache authentication directives.

Property name	`com.forgerock.agents.no.remoteuser.module.compatibility` Introduced in Web Agent 5.9
Function	Miscellaneous
Type	Unused
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced (From AM 7.2)` Title: `Use Built-in Apache HTTPD Authentication Directives`

Not-enforced

Ignore Path Info in Not-Enforced URLs

When `true`, strip path info and query from the request URL before comparing it with the URLs in Not-Enforced URL 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`.

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

Default: `true`

Property name	`com.sun.identity.agents.config.ignore.path.info.for.not.enforced.list` Introduced in Web Agent 4.x
Function	Not-enforced
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Ignore Path Info in Not-Enforced URLs`

Regular Expressions for Not-Enforced URLs

> **Important**
>
> Regular expressions are evaluated differently by different engines. When you use regular expressions in not-enforced lists, make sure that the expressions are evaluated in the way you expect. Double check that the correct URLs are enforced and not enforced.

When `true`, allow the use of Perl-compatible or ECMAScript-compatible (IIS) regular expressions in Not-Enforced URL List settings.

Default: `false`

Property name	`com.forgerock.agents.notenforced.url.regex.enable` Introduced in Web Agent 4.x
Function	Not-enforced
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application (From AM 7)` Title: `Regular Expressions for Not-Enforced URLs`

Enable Regular Expressions for Not-Enforced IPs

> **Important**
>
> Regular expressions are evaluated differently by different engines. When you use regular expressions in not-enforced lists, make sure that the expressions are evaluated in the way you expect. Double check that the correct URLs are enforced and not enforced.

A flag to enable Perl-compatible regular expressions or ECMAScript-compatible (IIS) in Not-Enforced URL from IP Processing List.

Default: `false`

Property name	`org.forgerock.agents.config.notenforced.ext.regex.enable` Introduced in Web Agent 4.x
Function	Not-enforced
Type	Boolean: `true` returns true; all other strings return `false`.

Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application (From AM 7)` Title: `Enable Regular Expressions for Not-Enforced IPs`

Not-Enforced Fallback Mode

A flag to specify whether the agent allows traffic to resources specified in the not-enforced lists when AM is not available:

- `true`: While AM is unavailable, the agent reads the cached agent profile configuration until it expires. After the cache expires, reads the local configuration file (`agent.conf`). If not-enforced properties are configured in `agent.conf`, the 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.

Configure the following properties in `agent.conf`, even if the agent profile is in centralized configuration:

- `com.forgerock.agents.config.fallback.mode = true`
- `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`

Default: `false`

Property name	`com.forgerock.agents.config.fallback.mode` Introduced in Web Agent 4.x
Function	Not-enforced
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Not-Enforced Fallback Mode`

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 name	`com.sun.identity.agents.config.client.ip.validation.enable` Introduced in Web Agent 4.x
Function	Not-enforced
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Client IP Validation`

Invert Not-Enforced URLs

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

- If Not-Enforced URL List is empty, all URLs are enforced
- At least one URL must be enforced. To allow access to any URL without authentication, consider disabling the agent.

Default: `false`

Property name	`com.sun.identity.agents.config.notenforced.url.invert` Introduced in Web Agent 4.x
Function	Not-enforced
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No

AM console	Tab: `Application` Title: `Invert Not-Enforced URLs`

Not-Enforced URL List

A space-delimited list of URIs for which the agent does not enforce authentication or request policy evaluations.

For information about configuring not-enforced lists, refer to Not-enforced rules in the *User Guide*.

Default: Empty

Property name	`com.sun.identity.agents.config.notenforced.url` Introduced in Web Agent 4.x
Function	Not-enforced
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Not-Enforced URL List`

Fetch Attributes for Not-Enforced URLs

When `true`, the agent fetches profile, response, and session attributes that are mapped by policy evaluations, and forwards these attributes to not-enforced URLs.

Default: `false`

Property name	`com.sun.identity.agents.config.notenforced.url.attributes.enable` Introduced in Web Agent 4.x
Function	Not-enforced
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No

AM console	Tab: `Application` Title: `Fetch Attributes for Not-Enforced URLs`

Not-Enforced IP List

A space-delimited list of IP addresses or network CIDR notation addresses for which the agent does not enforce authentication or request policy evaluations.

Supported values are IPV4 and IPV6 addresses, IPV4 and IPV6 ranges of addresses delimited by the - character, and network ranges specified in CIDR notation.

This property can apply to methods. The following example does not enforce GET requests in the specified IP range: `com.sun.identity.agents.config.notenforced.ip[1,GET]=iprange`

For information about configuring not-enforced lists, refer to Not-enforced rules in the *User Guide*.

Default: Empty

Property name	`com.sun.identity.agents.config.notenforced.ip` Introduced in Web Agent 4.x
Function	Not-enforced
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application` Title: `Not-Enforced IP List`

Not-Enforced URL from IP Processing List

A space-delimited list of IP addresses or network CIDR notation addresses for which the agent does not enforce authentication or request policy evaluations.

When `|` and a list of IP addresses is used, the request is not enforced if it comes from one of the IP ranges AND it is asking for one of the URLs.

In the following example, the agent does not enforce HTTP requests from the IP addresses 192.6.8.0/24 to any file in /public, or any files or directories that start with the string login in the directory /free_access URI:

`org.forgerock.agents.config.notenforced.ipurl[1]=192.6.8.0/24|http://www.example.com:8080/public/* /free_access/login`

Default: Empty

Property name	`org.forgerock.agents.config.notenforced.ipurl` Introduced in Web Agent 4.x
Function	Not-enforced
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Application (From AM 7)` Title: `Not-Enforced URL from IP Processing List`

POST data preservation

Enable POST Data Preservation

A flag to enable HTTP POST data preservation.

Default: `false`

Property name	`com.sun.identity.agents.config.postdata.preserve.enable` Introduced in Web Agent 4.x
Function	POST data preservation
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Enable POST Data Preservation`

POST Data Entries Cache Period

Number of minutes before expiry of the Post Data Preservation cache.

Consider setting Profile Attributes Cookie Maxage to at least the value of this property.

Default: `10`

Property name	`com.sun.identity.agents.config.postcache.entry.lifetime` Introduced in Web Agent 4.x
Function	POST data preservation
Type	Integer
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `POST Data Entries Cache Period`

POST Data Storage Directory

The local directory where the agent writes preserved POST data while requesting authorization from AM.

If you change this directory, make sure that the new directory has the correct read/write permissions for the ID that the server uses.

Default: `/web_agents/agent_type/pdp-cache`

Property name	`org.forgerock.agents.config.postdata.preserve.dir` Introduced in Web Agent 4.x
Function	POST data preservation
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `POST Data Storage Directory`

URLs Ignored by the POST Data Inspector

A list of URLs that are not be processed by the agent POST data inspector. Other modules on the same server can 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**
>
> URLs added to this property should also be added to Not-Enforced URL List.

Default: Empty

Property name	`org.forgerock.agents.config.skip.post.url` Introduced in Web Agent 4.x
Function	POST data preservation
Type	String Map
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced (From AM 7)` Title: `URLs Ignored by the POST Data Inspector`

Submit POST Data using JavaScript

When `true`, preserved POST data is resubmitted to the destination server after authentication by using JavaScript.

Default: `false`

Property name	`org.forgerock.agents.pdp.javascript.repost` Introduced in Web Agent 4.x
Function	POST data preservation
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No

Restart required	No
AM console	Tab: `Advanced (From AM 7)` Title: `Submit POST Data using JavaScript`

Policy client service

User ID Parameter

The User ID passed in the session from AM to the `REMOTE_USER` server variable.

Default: `UserToken`

Property name	`com.sun.identity.agents.config.userid.param` Introduced in Web Agent 4.x
Function	Policy client service
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `User ID Parameter`

Policy Cache Polling Period

Polling interval in minutes during which an entry remains valid after being added to the agent cache.

Default: `3`

Property name	`com.sun.identity.agents.config.policy.cache.polling.interval` Introduced in Web Agent 4.x
Function	Policy client service
Type	Integer
Bootstrap property	No
Required property	No

Restart required	No
AM console	Tab: `AM Services` Title: `Policy Cache Polling Period`

Policy Clock Skew

Number of seconds to allow for time difference between agent system and AM. Clock skew in seconds = AgentTime - AMServerTime.

Use this property to adjust for small time differences encountered despite use of a time-synchronization service. When this property is not set and agent time is greater than AM server time, the agent can make policy calls to the AM server before the policy subject cache has expired, or you can see infinite redirection occur.

Default: `0`

Property name	`com.sun.identity.agents.config.policy.clock.skew` Introduced in Web Agent 4.x
Function	Policy client service
Type	Integer
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `Policy Clock Skew`

Policy Evaluation Realm

In AM 6.5, this property was named Realm.

The realm where AM evaluates polices for policy decision requests from the agent. This property is recognized by AM, not the agent.

The policy set configured by Policy Set must exist in the realm configured by this property. 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

Property name	`org.forgerock.openam.agents.config.policy.evaluation.realm` Introduced in Web Agent 4.x
Function	Policy client service
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `Policy Evaluation Realm`

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/`.

Use this property when a client is expected to access multiple resources on the same path. However, caching can be expensive if very many policies are defined for the root resource.

Default: `false`

Property name	`com.sun.identity.agents.config.fetch.from.root.resource` Introduced in Web Agent 4.x
Function	Policy client service
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `Fetch Policies From The Root Resource`

Enable Retrieve Client Hostname

When `true`, get the client hostname through DNS reverse lookup for use in policy evaluation. This setting can impact performance.

Default: `false`

Property name	`com.sun.identity.agents.config.get.client.host.name` Introduced in Web Agent 4.x
Function	Policy client service
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `Enable Retrieve Client Hostname`

SSO Cache Polling Period

Polling interval in minutes during which an SSO entry remains valid after being added to the agent cache.

Default: `3`

Property name	`com.sun.identity.agents.config.sso.cache.polling.interval` Introduced in Web Agent 4.x
Function	Policy client service
Type	Integer
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `SSO Cache Polling Period`

User ID Parameter Type

Fetch user ID from `SESSION` or `LDAP` attributes.

Default: `SESSION`

Property name	`com.sun.identity.agents.config.userid.param.type` Introduced in Web Agent 4.x
Function	Policy client service
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `User ID Parameter Type`

Policy Set

In AM 6.5, this property was named Application.

The name of the policy set where AM evaluates polices 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.

The policy set configured by this property 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 this property to use it.

Default: `iPlanetAMWebAgentService`

Property name	`org.forgerock.openam.agents.config.policy.evaluation.application` Introduced in Web Agent 4.x
Function	Policy client service
Type	String

Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `AM Services` Title: `Policy Set`

Profile

Accept SSO token cookie (deprecated)

Use Accept SSO Token instead of this property.

Property name	`com.forgerock.agents.accept.ipdp.cookie` Introduced in Web Agent 5.7
Function	Profile
Type	Integer
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Accept SSO token cookie (deprecated)`

Agent Profile ID Allow List

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

Property name	`com.forgerock.agents.jwt.aud.whitelist` Introduced in Web Agent 5.7

Function	Profile
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Agent Profile ID Allow List`

Web Socket Connection Interval

The time in minutes before WebSockets to AM are killed and reopened. Use this property to balance the distribution of connections across the AM servers on the site.

Default: `30`

Property name	`org.forgerock.openam.agents.config.balance.websocket.connection.interval.in.minutes` Introduced in Web Agent 4.x
Function	Profile
Type	Integer
Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Web Socket Connection Interval`

Enable Notifications of Agent Configuration Change

A flag to specify whether AM sends a notification to the 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 name	`com.sun.identity.agents.config.change.notification.enable` Introduced in Web Agent 4.x

Function	Profile
Type	Unused
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Enable Notifications of Agent Configuration Change`

Configuration Reload Interval

Time in minutes after which the agent fetches the configuration from AM. Used if notifications are disabled.

Default: `60`

Property name	`com.sun.identity.agents.config.polling.interval` Introduced in Web Agent 4.x
Function	Profile
Type	Integer
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Configuration Reload 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 has virtual host names, add URLs with the virtual host names to this list. AM checks that the `goto` URLs match one of the agent root URLs for CDSSO.

Default: `agent-root-URL`

Property name	`sunIdentityServerDeviceKeyValue` Introduced in Web Agent 4.x
Function	Profile
Type	Unused
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Agent Root URL for CDSSO`

Disable Audience Claim Validation

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`

Property name	`com.forgerock.agents.jwt.aud.disable` Introduced in Web Agent 5.7
Function	Profile
Type	Integer

Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Disable Audience Claim Validation`

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 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 name	`org.forgerock.openam.agents.config.jwt.name` Introduced in Web Agent 4.x
Function	Profile
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `JWT Cookie Name`

Password

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

If you change the password, manually update the password in Agent Profile Password.

Property name	`password` Introduced in Web Agent 4.x

Function	Profile
Type	Unused
Bootstrap property	No
Required property	No
Restart required	No

Location of Agent Configuration Repository

The management mode for the agent configuration:

- `centralized`: The configuration is managed through AM.
- `local`: The configuration is managed locally in the agent configuration file. In local configuration, you cannot manage the agent configuration through the AM console.

Default: `centralized`

Property name	`com.sun.identity.agents.config.repository.location` Introduced in Web Agent 4.x
Function	Profile
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Location of Agent Configuration Repository`

Retain Session Cache After Configuration Change

Use this property to manage how the session cache is used after a change to the agent configuration:

- `0`: Purge the session cache, and re-read the user session data.
- `1`: Do not purge the session cache, and do not re-read the user session data. Use this value to prevent the agent from flooding AM instances with requests, when the agent configuration changes regularly, and the changes do not affect the agent authorisation decisions.

Default: `0`

Property name	`com.forgerock.agents.session.cache.eventually.consistent` Introduced in Web Agent 5.9
Function	Profile
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Advanced (From AM 7.2)` Title: `Retain Session Cache After Configuration Change`

Agent Deployment URI Prefix

Overrides the request URL given by the agent, when the agent 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 agent.

At least one of the following properties must be enabled:

- Enable Override Request URL Port
- Enable Override Request URL Protocol
- Enable 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.

Default: `agent-root-URL`

Property name	`com.sun.identity.agents.config.agenturi.prefix` Introduced in Web Agent 4.x
Function	Profile
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Agent Deployment URI Prefix`

Use Cached Configuration After Update

> **Important**
>
> Use this property only on recommendation from ForgeRock support, to reduce unnecessary concurrent authentication requests to AM.

When the agent receives requests after the agent config has been updated in AM, a single request thread calls AM to download the new configuration. This property manages the response for the concurrent request threads, as follows:

- `false`: Concurrent request threads wait for the time specified by TCP Receive Timeout for the retrieving request thread to complete, and then they use the new configuration.
- `true`: Concurrent request threads that can use the out-of-date, cached configuration do so, without waiting for the new configuration.

Set this property according to the value of Location of Agent Configuration Repository:

- `local`: In the local bootstrap file.
- `central`: In the AM console. The value in the AM console takes precedence over the local bootstrap file.

Default: `false`

Property name	`com.forgerock.agents.config.use.during.update` Introduced in Web Agent 4.x
Function	Profile
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `Advanced` Title: `Use Cached Configuration After Update`

Enable Notifications

When `true`, AM can sent 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`

Property name	`com.sun.identity.agents.config.notification.enable` Introduced in Web Agent 4.x
Function	Profile
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	Yes
Required property	No
Restart required	No
AM console	Tab: `Global` Title: `Enable Notifications`

Group

Status of the agent configuration.

Property name	`group` Introduced in Web Agent 4.x
Function	Profile
Type	Unused
Bootstrap property	No
Required property	No
Restart required	No

URL handling

Encode Special Characters in URLs

When `true`, encode the URL a URL with special characters before doing policy evaluation.

Default: `false`

Property name	`com.sun.identity.agents.config.encode.url.special.chars.enable` Introduced in Web Agent 4.x
Function	URL handling

Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous` Title: `Encode Special Characters in URLs`

Enable URL Comparison Case Sensitivity Check

When `true`, enforce case insensitivity in policy evaluation and not-enforced URL evaluation.

Default: `true`

Property name	`com.sun.identity.agents.config.url.comparison.case.ignore` Introduced in Web Agent 4.x
Function	URL handling
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous` Title: `Enable URL Comparison Case Sensitivity Check`

Invalid URL Regular Expression

A Perl-compatible or ECMAScript-compatible (IIS) regular expression to parse valid request URLs. The 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`, configure the following regular expression:

```
com.forgerock.agents.agent.invalid.url.regex=http[s]?:\/\/[^\/]+\/(?i)(?!\\|[?]\/\/|\.\/|[?]\/\.|\/\|\\.|~|[?]%2d|%20|[?]%[0-1][0-9a-f]|%[7-9a-f][0-9a-f]|[?]%25)[?]?.
```

Default: Empty

Property name	`com.forgerock.agents.agent.invalid.url.regex` Introduced in Web Agent 4.x
Function	URL handling
Type	String
Bootstrap property	No
Required property	No
Restart required	No
AM console	Tab: `Miscellaneous (From AM 7)` Title: `Invalid URL Regular Expression`